Rerfrigerator eye

Smart Fridge System for the Internet of Things era with Image Processing, Cloud Computing and Data Analytics

Presented by

Ahmed Al-Badrawy
Ahmed El-Morsy
Aliaa El-Zohairy
Mohamed Al-Kout
Nora Basha

Supervised by

Assoc. Prof. Sherif Kishk

July, 2016

Mansoura Faculty of Engineering
Electronics & Communication Department

B.Sc. Final Year Project

Rerfrigerator eye

Smart Fridge System for the Internet of Things era with Image Processing, Cloud Computing and Data Analytics

Presented by

Ahmed Al-Badrawy
Ahmed El-Morsy
Aliaa El-Zohairy
Mohamed Al-Kot
Nora Basha

Supervised by

Assoc. Prof. Sherif Kishk

Mansoura, Egypt
July, 2016

Acknowledgment

Praise Allah for his blessings which without, we wouldn't have been able to complete and finalize this project.

We are honored that our work has been supervised by Dr. Sherif Kishk. We would like to express our deepest gratitude and sincere appreciation to him for his help, valuable advice, continuing support, endless patience, and guidance throughout the preparation of this work.

We would like to thank our families; Mothers, fathers, brothers and sisters who have helped us with all they could do by assisting us mentally, financially where they haven't slept the nights just to serve us in our gatherings and give us a great motivation.

We would like to thank our friends who have helped us with all their possibilities to search, buy the components needed and to make a full problem research on our society.

Abstract

In this book, we’ll be introducing Refrigerator eye project, as a Smart Fridge System for the Internet of Things era with Image Processing, Cloud Computing and Data Analytics to make our users able to perform some activities in less time, and use this time on more productive things.

Moreover, we’ll discuss the problems of refrigerators, some purposed solutions to these problems, also we’ll review how this project facilitates the user's lifestyle and help people not to waste time in a world where time is precious for everyone.

This document contains full details about how the hardware and software of this project have developed from scratch till they came to life, as a reliable prototype that proves our concept, and finally our future plans to improve its efficiency, and ensure that we provide a simple, safe and fun learning environment to our users.

Contents

Chapter 1
Introduction

1.1 Problem

Everything is going to be connected and so our refrigerators. We have many food products that vary in their type, expiration date and quantity. Expired food is harmful to our health especially if we eat or use it and it may waste time if we figured, that it is expired, late. We eat food without knowing how many calories are in it, it may affect our health specially if someone have a disease, health complaint …etc. While shopping for food products, we may forget what we want, what we have and don't have, so we are not able to determine what to buy.

Refrigerators breakdown for many different reasons and sometimes we don't know what is wrong or what to do, so we waste lots of money and time.

1.2 Solution

A standalone device which is outside the refrigerator or it can be built in the refrigerator while manufacturing. It recognizes each food type that enters it, stores its weight and expiring date and display its nutrition information. It suggests recipes according to the food products stored. When a food product is expired or insufficient, the user receives a mobile application notification of it and a suggested shopping list of the food products that are about to run out.

The smart system predicts probable errors and notifies the user to change its parameter (temperature, distance from the wall, …etc.) to prevent performance problems. This improves the Refrigerator performance and saves time and money spent on maintenance.

1.3 Novelty and Features

Our product has some features that are not offered by any smart fridge manufacturers today, as they offer a complete fridge with smart screen, various applications and monitoring system for the fridge parameters, but the food is not recognized and no shopping list is suggested, also the fridge does not predict errors. That makes our product unique and innovative.

With our product customer does not have to buy a whole new fridge to get the benefit of smart fridge features, as our product works beside any ordinary fridge and that is an innovative advantage in our product.

Feature Name	**Brief Description**
Performance Monitoring	Sets of sensors deployed in the Refrigerator to monitor its parameters (temperature, distance from the wall, …etc.), readings are displayed on the device screen.
Error Prediction	Sensors readings from different Refrigerators are analyzed on the cloud and gives a notification for the probable or existing errors.
Food Recognition	The camera outside the Refrigerator captures an image of each food product the user inserts. Image processing is used to recognize the food type. Weight sensors deployed determine the food product weight while capturing its image. Image processing is done using Python image processing and computer vision libraries (OpenCV, NumPy, SciPy, ...etc.) The code used in image processing runs remotely over a web application.
Shopping List	Once a food product reaches 10% of its original weight, it will be added to the suggested shopping list that is send to the mobile application.
Expiration date	It is stored by the barcode scanner in case of a canned food product or manually by the user from the screen or the mobile application
Nutrition Information	When the camera recognizes a product, its nutrition information is displayed on the screen. Nutrition information is also accessible by the user through the screen or the mobile app.
Recipes	According to the food products stored, the system suggests recipes to be cooked.
Screen	The user interface to our system through the screen. The screen will have other apps that allow him to play music and more.

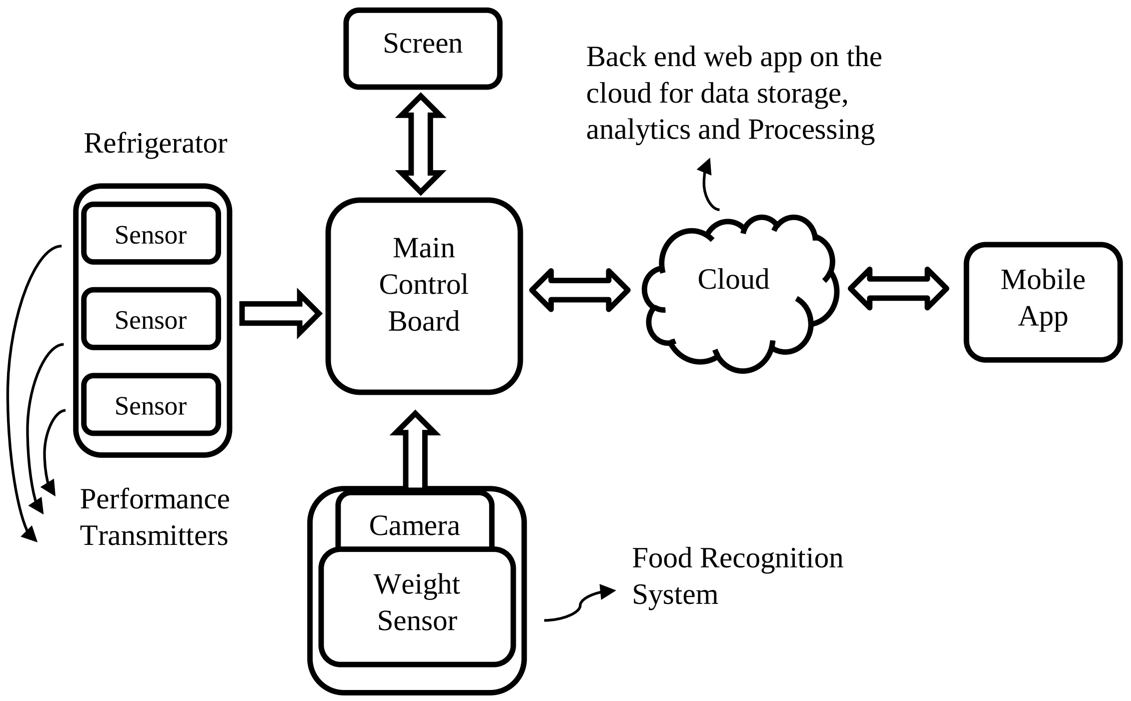
1.4 Approach

Before starting the project, we need to well understand the image processing fundamentals, OpenCV-python tutorials, Java, android programming concepts and Data analytics & Mining.

After that we can start connecting the hardware we bought and make the interfaces ourselves and code the raspberry pi from scratch with the ready-made libraries to take readings from the sensors and take images by the camera. The images will be processed on the raspberry pi by image processing libraries and the help of similar online projects.

Collected data will be send to the cloud for food recognition, and data analysis and storage so some networking will be done on the raspberry pi. The food recognition is really a challenge as it is under-research and projects done in this area are rare. Food recognition will be done on the cloud. working on the cloud will need some effort while working with big data, data analysis and different cloud APIs. Our front end will be on the mobile and screen app. The screen app is a challenge as we have two options whether to customize on the OS or make a launcher app. The mobile app will be built from scratch but using samples of code from different sources is probable.

1.5 System modules



1.6 Impact

While searching for the project idea this summer vacation, we wanted an idea that solves a problem and helps us learn something new. This idea solves the previously indicated problems by the implementation of many growing technological trends nowadays. The device uses computer vision to recognize food, suggest the shopping list and use the internet to get the food nutrition information to help people with diabetes control their diet. A small part of machine learning is involved in suggesting recipes.

The process of fridge monitoring and predicting error is the first step of using the huge experience and data gathered from many refrigerators to forecast performance and that involve data analytics.

Information and Data are uploaded to the cloud, processed and received by the device without any human interaction, so the device is a pure IoT application.

This idea goes with today's trend and we can build our company after this project success. Even if we couldn't start our own company, we will have learnt many things from our project like image processing, cloud computing, big data, Embedded systems and many personal and development skills and so on that will support us in our career.

The customers today are always connected to the internet, so the device adds its value without the need of huge changes in the customers’ lifestyle.

This project is to facilitate the user's lifestyle. The design is intended that people do not waste time in a world where time is precious for everyone.

Make grocery shopping is something we all do and takes away much of our time, this project wants the user to perform this activity in less time, and use this time on more productive things.

Chapter 2
Embedded System Elements

2 Overview

2.1 Main Control Board.

We don't need high computational power to control the Sensors and the camera. So we`ll use Embedded Linux Board like (raspberry Pi, beaglebone, Intel Galileo, and so on ….). it wasn't easy to make the decision. This guide will compare each board with an emphasis on their features and capabilities for maker and electronics projects. All of these boards run a version of Linux and are great for putting sensors, gadgets, and other hardware on the internet. However, you'll see there are quite a few differences in the hardware and capabilities that might make you prefer one board over the other.

	Beaglebone Black Rev C	**Intel Galileo**	**Raspberry Pi 2**
Picture			
CPU	ARM Cortex-A8	Intel X1000	quad-core ARM Cortex-A7
Speed	1ghz	400mhz	900Mhz
RAM	512MB DDR3	256MB	1 Gb
GPU	PowerVR SGX530	None	Broadcom VideoCore IV
Internal Storage	4GB	8MB	None
External Storage	MicroSD	MicroSD	SD card
Networking	10/100Mbit Ethernet	10/100Mbit Ethernet	10/100Mbit Ethernet
Approximate Price	$55	$80	$35

The following table compares the I/O capabilities of each board:

	BeagleBone Black Rev C	**Intel Galileo**	**Raspberry Pi 2**
Digital I/O Pins	65 Pins	14	40 Pins
Digital I/O Power	3.3V	3.3V or 5V (switched with jumper)	3.3V
Analog Input	7 with 12-bit ADC, 0-1.8V (no external reference input)	6 with 12-bit ADC, 0-5V (no external reference input)	None
PWM Output	8	6 (limited speeds prevent fine servo control)	1
UART	4	2 (1 exposed through 3.5mm jack)	1
SPI	2	1	2
I2C	2	1	1
USB port	1 standard A connector	1 micro AB connector	4
Video Output	Micro HDMI	None	HDMI, Composite RCA, DSI
Video Input	None	None	CSI (camera)
Audio Output	Micro HDMI	None	HDMI, 3.5mm jack
Power Output	3.3V up to 250mA, 5V up to 1A	3.3V up to 800mA, 5V up to 800mA	3.3V up to 50mA, 5V up to 300-500mA

Looking at the results, the Beaglebone Black has the strongest memory and integer performance. However, the Beaglebone Black's floating point performance is slightly behind the Raspberry Pi. This can be explained because the ARM Cortex-A8 processor on the Beaglebone Black has a 'VFPLite' floating point unit which isn't as fast as other ARM FPUs. If you only care about performance and don't have a floating point heavy workload, the Beaglebone Black is a good board to consider.

2.1.1 Raspberry pi 2

The big reason to upgrade to a Pi 2 Model B over a classic Raspberry Pi Model B+ is the big boost in performance

The Pi 2 has 4 processors in one chip (the B+ has only one), an ARMv7 core vs an ARMv6, and 1 Gig of RAM vs 512 MB for the model B and B+

Those 3 improvements translate to pretty big performance increases!

2.1.1.1 *Hardware specs.*

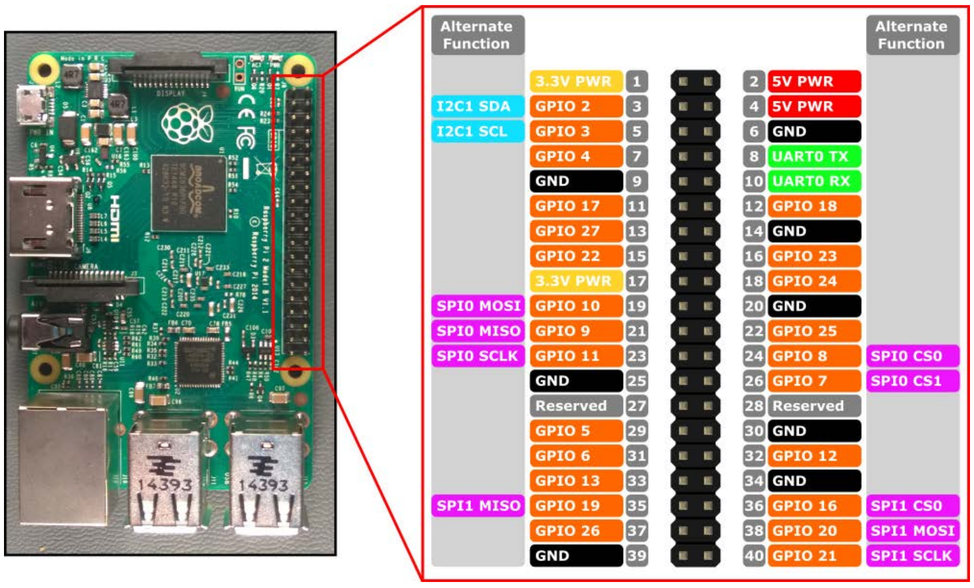
The Raspberry Pi 2 Model B is the second generation Raspberry Pi. It replaced the original Raspberry Pi 1 Model B+ in February 2015. Compared to the Raspberry Pi 1 it has:

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM

Like the (Pi 1) Model B+, it also has:

- USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core

Because it has an ARMv7 processor, it can run the full range of ARM GNU/Linux distributions, including Snappy Ubuntu Core, as well as Microsoft Windows 10



2.1.1.2 The Operating System

The Raspberry Pi primarily uses Linux for its operating system (OS) rather than Microsoft Windows or OS X (for Apple). An operating system is a program that makes it easier for the end user to use the underlying hardware. For example, although the processor (the chip at the centre of the Raspberry Pi that does the work) can do only one thing at a time, the operating system gives the impression the computer is doing lots of things by rapidly switching between different tasks. Furthermore, the operating system controls the hardware and hides the complexity that allows the Raspberry Pi to talk to networks or SD cards.

2.1.1.3 Raspbian OS.

Raspbian is the first distro every new Raspberry Pi owner should ideally use. It’s based on Debian Linux and, as we mentioned, is fully optimised for the Raspberry Pi’s hardware. It’s also an excellent starting point as it contains a plethora of pre-installed programs that will help get you up and running, and into the wonderful world of the Raspberry Pi.

The Raspbian operating system itself is very well constructed and developed. Running LXDE (Lightweight X11 Desktop Environment) as the desktop environment makes this a considerably streamlined operating system, ideally suited to the limited system resources of the Raspberry Pi. Raspbian leans heavily toward the educational aspects of the Raspberry Pi, as per the ethos of the Raspberry Pi Foundation. Packaged within you’ll find such programs as Scratch – the childfriendly graphical beginner programming language whereby games and live story books can be created. Python is also included, a programming language that’s ideal for beginners.

2.1.1.3.1 Installing Raspbian

1. **Download NOOBS**
   On your desktop or laptop, go to www.raspberrypi.org/downloads. Click the NOOBS thumbnail and then, on the next page, look for the red buttons beneath NOOBS on the left (not NOOBS LITE on the right). Click 'Download ZIP', or 'Download Torrent' if you use BitTorrent or similar
2. **Download SD Formatter**
   Now head to www.sdcard.org, click the Downloads link at the top of the page and then choose 'SD Formatter for Windows Download'. Accept the terms and conditions to download the program, which we’ll use to copy our Raspbian image onto the SD card.
3. **Prepare the files**
   Once NOOBS and SD Formatter have finished downloading, find them inside your Downloads folder. First, unzip the NOOBS file. Next, unzip SD Formatter and run the setup file within. Follow the simple instructions to install the software to your computer.
4. **Format your SD card**
   Run the SD Formatter program once it has finished installing. Select the SD card that you've plugged into your SD card reader using the Drive drop-down. In the Format Options, select FULL(Erase). Click Format when you're ready to wipe your SD card clean.
5. **Copy NOOBS across**

Now, navigate to your SD card using your File Explorer and open it up – it should now be blank, ready for some fresh files to be written. Select all of the files inside the unzipped NOOBS folder, then drag them across to the SD card to begin copying them.

6. **Install Raspbian**
   Now you have a fresh NOOBS SD card, simply eject it from your Windows 10 machine and then insert it into your Raspberry Pi's SD card slot. Power up your Pi and you'll be greeted with an easy-to-follow installer – select Raspbian from the menu to start the installation.

2.1.1.3.2 Installing software in Raspbian

The easiest way to manage installing, upgrading, and removing software is using APT (Advanced Packaging Tool) which comes from Debian. If a piece of software is packaged in Debian and works on the Raspberry Pi's ARM architecture, it should also be available in Raspbian.

To install or remove packages you need root user permissions, so your user needs to be in sudoers or you must be logged in as root. Read more about users and root.

To install new packages, or update existing ones, you will need an internet connection

Note that installing software uses up disk space on your SD card, so you should keep an eye on disk usage and use an appropriately sized SD card.

Also note that a lock is performed while software is installing, so you cannot install multiple packages at the same time.

SOFTWARE SOURCES

APT keeps a list of software sources on your Pi in a file at /etc/apt/sources.list. Before installing software, you should update your package list with apt-get update:

```
sudo apt-get update
```

INSTALLING A PACKAGE WITH APT

```
sudo apt-get install tree
```

Typing this command should inform the user how much disk space the package will take up and asks for confirmation of the package installation. Entering Y (or just hitting Enter, as yes is the default action) will allow the installation to occur. This can be bypassed by adding the -y flag to the command:

```
sudo apt-get install tree -y
```

Installing this package makes tree available for the user.

USING AN INSTALLED PACKAGE

tree is a command line tool which provides a visualisation of the directory structure of the current directory, and all it contains.

Typing tree runs the tree command. For example:

```
tree
..
├── hello.py
├── games
│   ├── asteroids.py
│   ├── pacman.py
│   ├── README.txt
│   └── tetris.py
```

Typing man tree gives the manual entry for the package tree

Typing whereis tree shows where tree lives:

```
tree: /usr/bin/tree
```

UNINSTALLING A PACKAGE WITH APT

REMOVE

You can uninstall a package with apt-get remove:

```
sudo apt-get remove tree
```

The user is prompted to confirm the removal. Again, the -y flag will auto-confirm.

PURGE

You can also choose to completely remove the package and its associated configuration files with apt-get purge:

```
sudo apt-get purge tree
```

UPGRADING EXISTING SOFTWARE

If software updates are available, you can get the updates with sudo apt-get update and install the updates with sudo apt-get upgrade, which will upgrade all of your packages. To upgrade a specific package, without upgrading all the other out-of-date packages at the same time, you can use sudo apt-get install somepackage (which may be useful if you're low on disk space or you have limited download bandwidth).

2.1.1.3.3 Windows IOT Core.

Windows 10 IoT Core is a version of Windows 10 that is optimized for smaller devices with or without a display, and that runs on the Raspberry Pi 2 and 3, Arrow DragonBoard 410c & MinnowBoard MAX. Windows 10 IoT Core utilizes the rich, extensible Universal Windows Platform (UWP) API for building great solutions.

Hardware Requirements

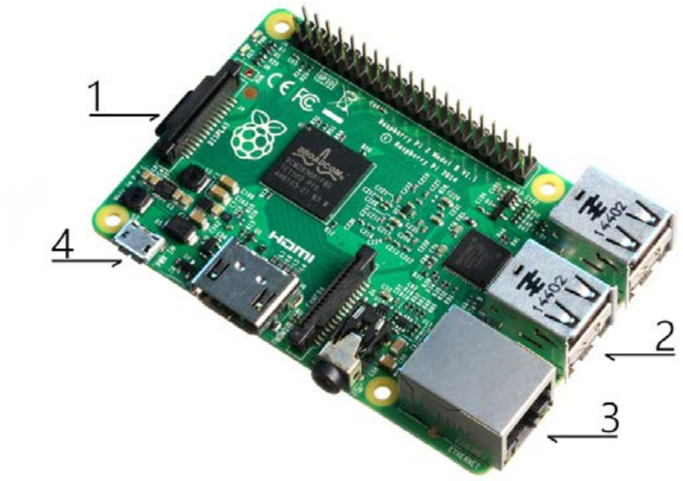
- Memory
  - ✓ Headless
    256 MB RAM (128 MB free to OS) / 2 GB Storage
  - ✓ Headed
    512 MB RAM (256 MB free to OS) / 2 GB Storage
- Processor

400 MHz or faster (x86 requires PAE, NX and SSE2 support)

2.1.1.3.4 Set up on Raspberry Pi 2

Hook up your board

1. Insert the MicroSD card included in your kit (the slot is on the opposite side of the board shown below).
2. Insert the WiFi Dongle included in your kit into one of the USB ports on your Raspberry Pi 2.
3. Insert the Ethernet Cable and connect it to your local network.
4. Connect the power supply to the micro USB port on the board.

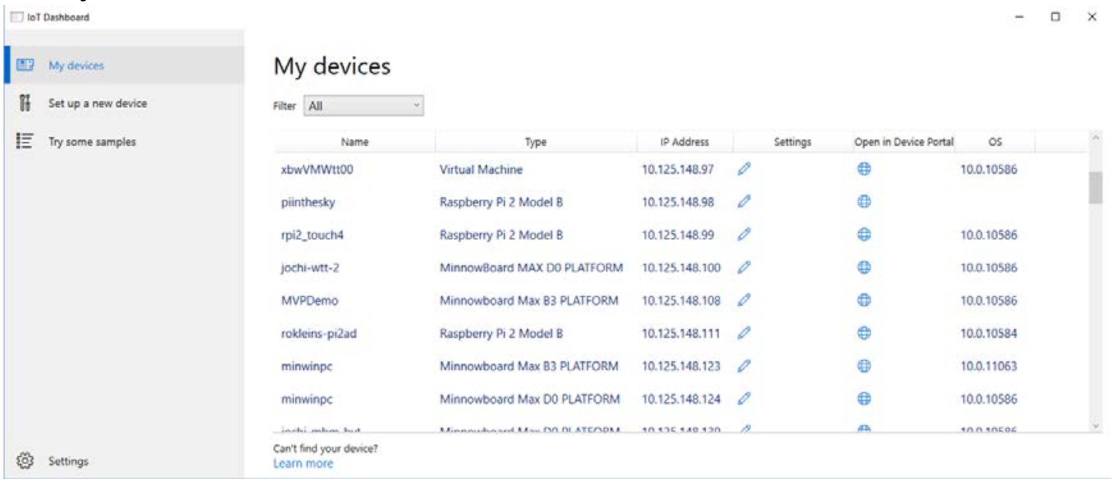


Download and Install the IoT Dashboard tool

The IoT Dashboard tool displays all the Windows 10 IoT Core devices on your network.

Boot Windows 10 IoT Core:

1. Windows 10 IoT Core will boot automatically after connecting the power supply. Allow the Pi about five minutes for the first boot.
2. Find your device on the IoT Dashboard application. When run, the application automatically finds all Windows IoT Core devices on the local network and displays device information such as the name, device type, IP address, and more. Select the My Devices tab to view the current devices on the network.



My devices

Filter: All

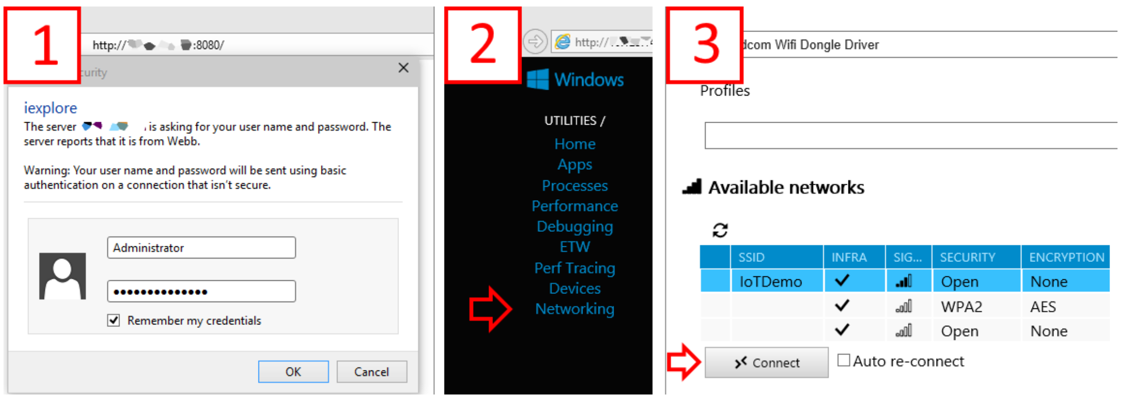
Name	Type	IP Address	Settings	Open in Device Portal	OS
xbwVMWtt00	Virtual Machine	10.125.148.97	✎	🌐	10.0.10586
piinthesky	Raspberry Pi 2 Model B	10.125.148.98	✎	🌐	
rpi2_touch4	Raspberry Pi 2 Model B	10.125.148.99	✎	🌐	10.0.10586
jochi-wtt-2	MinnowBoard MAX D0 PLATFORM	10.125.148.100	✎	🌐	10.0.10586
MVPDemo	Minnowboard Max B3 PLATFORM	10.125.148.108	✎	🌐	10.0.10586
rokleins-pi2ad	Raspberry Pi 2 Model B	10.125.148.111	✎	🌐	10.0.10584
minwinpc	Minnowboard Max B3 PLATFORM	10.125.148.123	✎	🌐	10.0.11063
minwinpc	Minnowboard Max D0 PLATFORM	10.125.148.124	✎	🌐	10.0.10586

Can't find your device?
Learn more

Configure your Raspberry Pi 2

Finally, you'll need to configure your Raspberry Pi 2 for WiFi connection using the Web-Based Management Tool. In Windows IoT Core Dashboard, Click on the Open in Device Portal icon.

1. Enter Administrator for the username, and supply your password (p@ssw0rd by default)
2. Click on Networking in the left-hand pane
3. Under Available networks, select network you would like to connect to and supply the connection credentials. Click Connect to initiate the connection



http://[redacted]:8080/

iexplore

The server [redacted] is asking for your user name and password. The server reports that it is from Webb.

Warning: Your user name and password will be sent using basic authentication on a connection that isn't secure.

Administrator

••••••••••••••

[✓] Remember my credentials

dcom Wifi Dongle Driver

Profiles

Available networks

	SSID	INFRA	SIG...	SECURITY	ENCRYPTION
	IoTDemo	✓		Open	None
		✓		WPA2	AES
		✓		Open	None

☐ Auto re-connect

2.1.1.4 REMOTE-ACCESS

Sometimes you need access to a Raspberry Pi without connecting a monitor to it: for example, if the Pi is embedded in something like a robot; if you want to view some information from it from elsewhere; or maybe if you just don't have a monitor spare!

- ✓ IP address.
  - o How to find your Raspberry Pi's IP address in order to connect to it
- ✓ Access over Internet
  - o Remote access to the Pi over the Internet using Weaved
- ✓ VNC
  - o Remote access to the Pi's graphical interface, viewed in a window on another computer
- ✓ SSH
  - o Access the command line of the Pi from another computer
- ✓ SFTP
  - o Copy files between your Pi and another computer using SFTP (Secure FTP)
- ✓ SCP
  - o Copy files between your Pi and another computer using SCP (Secure Copy)
- ✓ SSHFS
  - o Copy files between your Pi and another computer using SSHFS (SSH Filesystem)
- ✓ rsync
  - o Synchronize folders between the Pi and another computer using rsync over SSH
- ✓ FTP
  - o Copy files between your Pi and another computer using FTP
- ✓ Web server

  - Set up a website or a web page to display some information about the Pi, using a web browser on another machine, on the network, or on the internet

2.1.1.4.1 VNC (VIRTUAL NETWORK COMPUTING)

Sometimes it is not convenient to work directly on the Raspberry Pi. Maybe you would like to work on it from another computer by remote control.

VNC is a graphical desktop sharing system that allows you to remotely control the desktop interface of one computer from another. It transmits the keyboard and mouse events from the controller, and receives updates to the screen over the network from the remote host.

You will see the desktop of the Raspberry Pi inside a window on your computer. You'll be able to control it as though you were working on the Raspberry Pi itself.

- On your Pi (using a monitor or via SSH), install the TightVNC package:

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

- Next, run TightVNC Server which will prompt you to enter a password and an optional view-only password:

```
tightvncserver
```

- Start a VNC server from the terminal: This example starts a session on VNC display one (:1) with full HD resolution:

```
vncserver :1 -geometry 1920x1080 -depth 24
```

Note that since by default an X session is started on display zero, you will get an error in case you use :0.

- Since there are now two X sessions running, which would normally be a waste of resources, it is suggested to stop the displaymanager running on :0 using

```
service lightdm stop
```

- Now, on your computer, install and run the VNC client:
  On a Linux machine install the package xtightvncviewer

```
sudo apt-get install xtightvncviewer
```

- Otherwise, TightVNC is downloadable from tightvnc.com

2.1.1.4.2 Running VNC at Startup

There are several different methods of arranging for some code to be run as the Pi starts. The method described below is probably the easiest to use. You can adapt it to run other commands instead of starting the VNC server.

Step 1.

Open a Terminal session on the Pi, or connect using SSH. A new terminal or SSH session will automatically start you off in your home directory of /home/pi. If you are not in this directory, change to it by typing:

```
$ cd /home/pi
```

Then cd to the .config directory by typing:

```
$ cd .config
```

Note the '.' at the start of the folder name. This makes it a hidden folder that will not show up when you type 'ls'

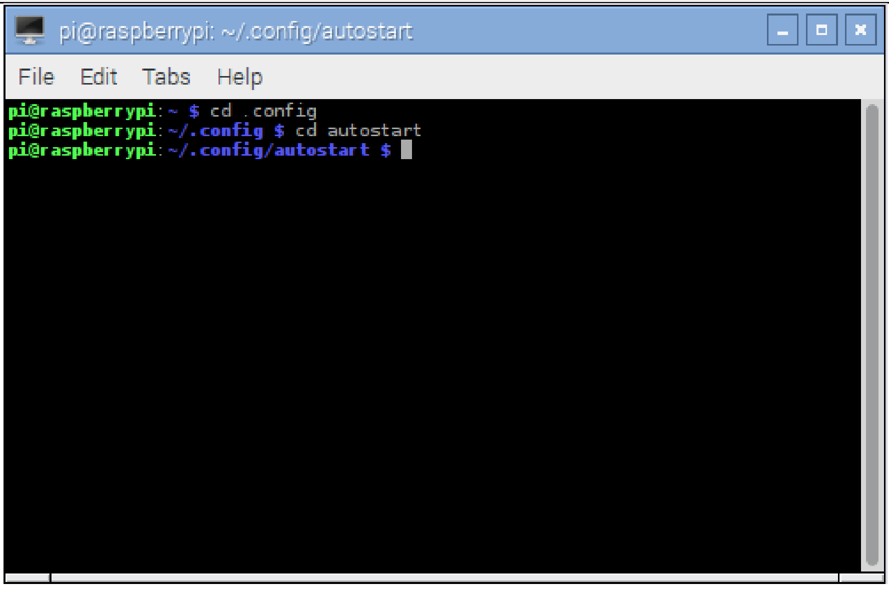
Step 2.

Issue the command below to create a new directory inside .config called 'autostart'.

```
$ mkdir autostart
```

cd into that new directory by typing:

```
$ cd autostart
```



```
pi@raspberrypi:~ $ cd .config
pi@raspberrypi:~/.config $ cd autostart
pi@raspberrypi:~/.config/autostart $
```

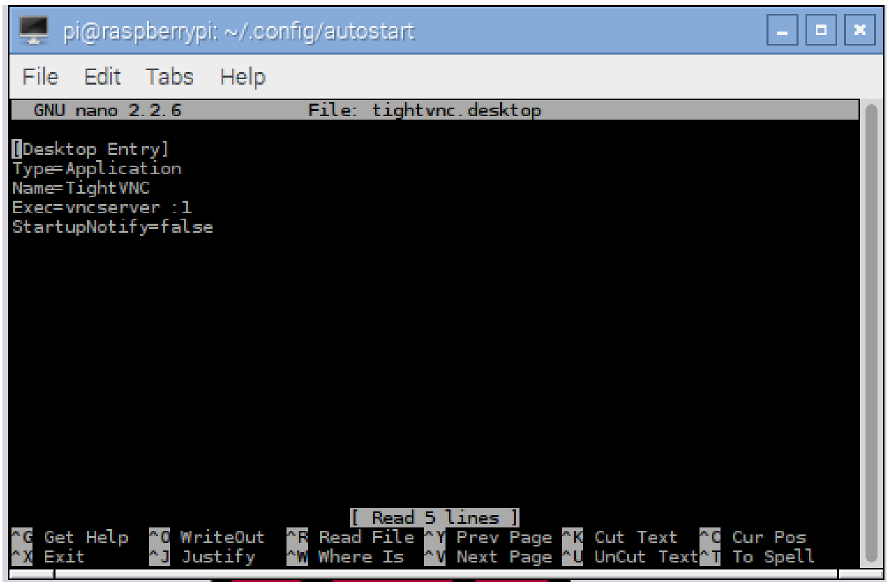
Step 3.

All that remains is to edit a new configuration file. So type the following command to open the nano editor on the new file:

```
$ nano tightvnc.desktop
```

Edit the contents of the file with the following text.

```
[Desktop Entry]
Type=Application
Name=TightVNC
Exec=vncserver :1
StartupNotify=false
```



```
  GNU nano 2.2.6              File: tightvnc.desktop

[Desktop Entry]
Type=Application
Name=TightVNC
Exec=vncserver :1
StartupNotify=false

                                [ Read 5 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Type Ctrl-X and then Y to save the changes to the file.
That's all there is to it. The next time you reboot the VNC server will restart automatically.

2.2 Food Recognition System.

2.2.1 Camera

USING A STANDARD USB WEBCAM

Rather than using the Raspberry Pi camera module, you can use a standard USB webcam to take pictures and video on the Raspberry Pi.

- INSTALL FSWEBCAM
  First, install the fswebcam package:

```
sudo apt-get install fswebcam
```

- BASIC USAGE
  Enter the command fswebcam followed by a filename and a picture will be taken using the webcam, and saved to the filename specified:

```
fswebcam image.jpg
```

This command will show the following information:

```
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Corrupt JPEG data: 2 extraneous bytes before marker 0xd4
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to 'image.jpg'.
```

- SPECIFY RESOLUTION
  The webcam used in this example has a resolution of 1280 x 720 so to specify the resolution I want the image to be taken at, use the -r flag:

```
fswebcam -r 1280x720 image2.jpg
```

This command will show the following information:

```
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
--- Capturing frame...
Corrupt JPEG data: 1 extraneous bytes before marker 0xd5
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to 'image2.jpg'.
```

- BASH SCRIPT
  You can write a Bash script which takes a picture with the webcam. The script below saves the images in the /home/pi/webcam directory, so create the webcam subdirectory first with:

```
mkdir webcam
```

- To create a script, open up your editor of choice and write the following example code:

```
#!/bin/bash
DATE=$(date +"%Y-%m-%d_%H%M")
fswebcam -r 1280x720 --no-banner /home/pi/webcam/$DATE.jpg
```

This script will take a picture and name the file with a timestamp. Say we saved it as webcam.sh, we would first make the file executable:

```
chmod +x webcam.sh
```

- Then run with:

```
./webcam.sh
```

2.2.2 Wight sensor

DYMO Digital Postal Scale / Shipping Scale, 25-pound, The DYMO M25 Digital Postal Scale weighs envelopes and packages up to 25 lbs (11 kg) – without a trip to the Post Office or mailroom. Connect via USB to your PC or Mac®* to use with popular mailing/shipping software, including Endicia® InstaRate** and DYMO Stamps®*** for calculating and purchasing USPS®-approved postage online with no monthly fee. Conveniences include a hold feature that locks weight display for ten seconds, a tare function for accurately weighing items in a container, an automatic shut-off, and an LCD screen that displays the weight in easy-to-read digital format – in lbs/oz or kg/g.

READING A DYMO USB SCALE USING PYTHON.

The first step is to install PyUSB 1.0. Unzip the contents of the archive from the PyUSB sourceforge page and run python setup.py install.

Using PyUSB on Windows requires installing a suitable backend such as libusb-win32. After installing libusb-win32, it’s necessary to install a device filter for the scale. Connect the USB scale to the computer and press the power button. Windows should detect the scale and install the device driver automatically. The DYMO M25 shows up in Device Manager as a “USB Input Device” with vendor ID 0922 and product ID 8003.

Using the libusb-win32 Filter Wizard, install a device filter for the scale. It will show up in the list as “vid:0922 pid:8003 rev:0100 USB Input Device”.

Now it’s possible to read data from the scale using the usb module in Python:

```
import usb.core
import usb.util
VENDOR_ID = 0x0922
PRODUCT_ID = 0x8003
# find the USB device
device = usb.core.find(idVendor=VENDOR_ID,
                       idProduct=PRODUCT_ID)
# use the first/default configuration
device.set_configuration()
# first endpoint
```

```
endpoint = device[0][(0,0)][0]
# read a data packet
attempts = 10
data = None
while data is None and attempts > 0:
    try:
        data = device.read(endpoint.bEndpointAddress,
                           endpoint.wMaxPacketSize)
    except usb.core.USBError as e:
        data = None
        if e.args == ('Operation timed out',):
            attempts -= 1
            continue
print data
```

The data packet is a 6-element array; for example:

```
array('B', [3, 2, 11, 255, 0, 0])
```

The 1st element has always had the value 3 in my tests, so I’m not sure what it does.

The 2nd element indicates whether the value is stable.

The 3rd element (value 11 in the example above) is probably a flag byte. The only values I have observed so far are 2 and 11. A value of 2 indicates that the scale is in kg mode, and a value of 11 indicates that the scale is in lbs/ounces mode. Curiously, whenever the scale reads 0, it seems to indicate lbs/oz mode.

Thanks to Daniel Rice (see his comment below), we now know what the 4th element (index 3) is for; it’s for calculating the scaling factor when reading in ounces. In the above example it has a value of 255. This is in fact the signed value -1, which indicates that the raw value is in tenths, due to a scaling factor of 10^-1 or 0.1. For a value of 254, the scaling factor is 10^-2, or 0.01. I’ll refer to this value as scaling_factor below.

The elements at indices 4 and 5 are used to calculate the weight.

In kg mode:

```
grams = data[4] + (256 * data[5])
```

In pounds/ounces mode:

```
ounces = scaling_factor * (data[4] + (256 * data[5]))
```

If you check the mode, you can just convert to whatever unit you need no matter what mode the scale is in.

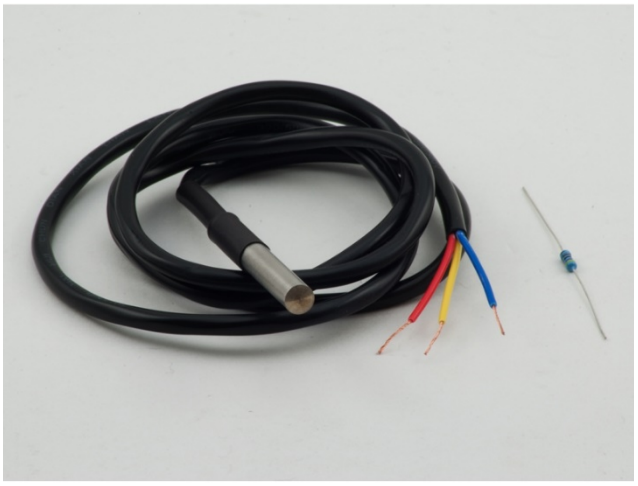
```
DATA_MODE_GRAMS = 2
DATA_MODE_OUNCES = 11
raw_weight = data[4] + data[5] * 256
if data[2] == DATA_MODE_OUNCES:
    ounces = raw_weight * scaling_factor
    weight = "%s oz" % ounces
elif data[2] == DATA_MODE_GRAMS:
    grams = raw_weight
    weight = "%s g" % grams
print weight
```

The scale is capable of reading negative values if you zero it and then remove the weight, but I haven't yet figured out how to get negative weight values yet. The scale itself displays the correct value, but the data packet reads as if it were zero, except the second array element has a value of 5 instead of 2.

2.3 Performance Transmitters

2.3.1 Temperature sensor

While searching for a simple way to measure temperature using my Raspberry Pi I came across the DS18B20 1-wire digital temperature sensor. This promised an accurate way of measuring temperature with a few wires and almost no external components.



This is a pre-wired and waterproofed version of the DS18B20 sensor. Handy for when you need to measure something far away, or in wet conditions. While the sensor is good up to 125°C the cable is jacketed in PVC so we suggest keeping it under 100°C. Because they are digital, you don't get any signal degradation even over long distances! These 1-wire digital temperature sensors are fairly precise (±0.5°C over much of the range) and can give up to 12 bits of precision from the onboard digital-to-analog converter. They work great with any microcontroller using a single digital pin, and you can even connect multiple ones to the same pin, each one has a unique 64-bit ID burned in at the factory to differentiate them. Usable with 3.0-5.0V systems.

2.3.1.1 Cable specs

- Stainless steel tube 6mm diameter by 30mm long
- Cable is 36" long / 91cm, 4mm diameter
- Contains DS18B20 temperature sensor
- If your sensor has four wires - Red connects to 3-5V, Black connects to ground and White is data. The copper wire is soldered to the wire shielding
- If your sensor has three wires - Red connects to 3-5V, Blue/Black connects to ground and Yellow/White is data

2.3.1.2 DS18B20 Technical specs

- Usable temperature ranges: -55 to 125°C (-67°F to +257°F)

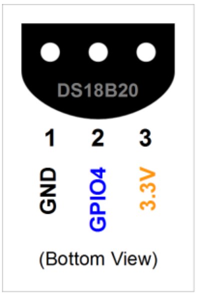
- 9 to 12-bit selectable resolution
- Uses 1-Wire interface- requires only one digital pin for communication
- Unique 64-bit ID burned into chip
- Multiple sensors can share one pin
- ±0.5°C Accuracy from -10°C to +85°C
- Temperature-limit alarm system
- Query time is less than 750ms
- Usable with 3.0V to 5.5V power/data

1.1.1.1 Sensing temperature with Python

The diagram on the right shows the DS18B20 device. It has three pins and comes in a TO-92 package which means it looks similar to other devices you may have used such as transistors.

Pin 1 is Ground. Pin 2 is the data pin and Pin 3 is the power pin. The only external component required is a single 4.7Kohm resistor.

In my testing I didn’t have one of these so I used 2 x 2.2Kohm resistors in series. This worked fine. I used a small piece of breadboard and some jumper cables to connect it to the GPIO header on my Raspberry Pi.



Pin 1 was connected to P1-06 (Ground)

Pin 2 was connected to P1-07 (GPIO4)

Pin 3 was connected to P1-01 (3.3V)

A 4.7Kohm resistor was placed between Pin 2 and Pin 3.

It is important to double check that you don’t confuse Pin 1 and Pin 3 on the device otherwise the power will be applied the wrong way round! Once you have connected everything together you can power up your Raspberry Pi.

```
sudo apt-get update
sudo apt-get upgrade
```

In order to configure the sensor, you just need to make a small change to the config.txt file using:

```
sudo nano /boot/config.txt
```

add the following line to the bottom :

```
dtoverlay=w1-gpio,gpiopin=4
```

You can save the file using CTRL-X, Y then RETURN. The device is setup to report its temperature via GPIO4.

For the changes to take effect you will need to reboot using:

```
sudo reboot
```

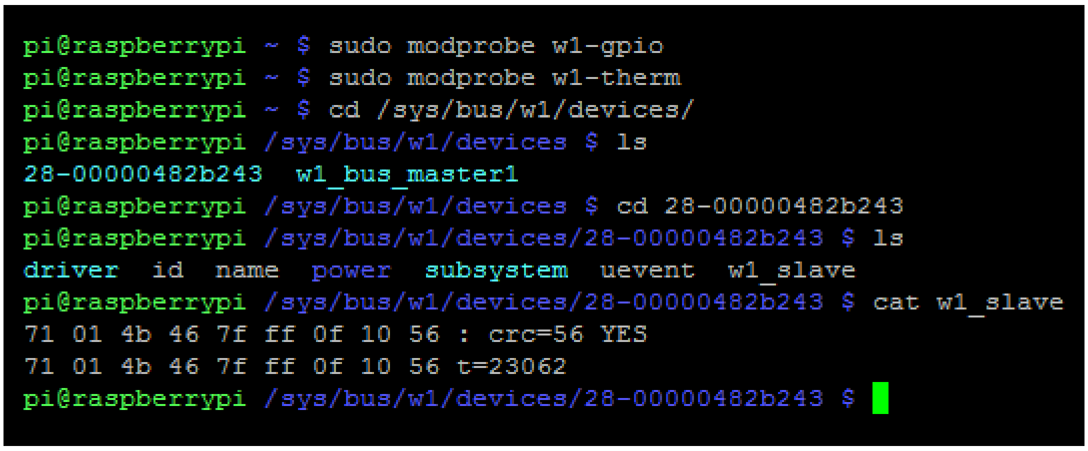
Use the commands below to go to the directory that contains the detected 1-wire devices:

```
cd /sys/bus/w1/devices
ls
```

This will list the directories associated with your 1-wire devices. Each one has a unique ID and in my case it is 28-00000482b243. Your ID will be different so be sure to use that in the example code below. Using “cd” we can change to the temperature sensor directory, list the contents and then view the “w1_slave” file:

```
cd 28-00000482b243
ls
cat w1_slave
```

The complete command line setup process looks like this:



```
pi@raspberrypi ~ $ sudo modprobe w1-gpio
pi@raspberrypi ~ $ sudo modprobe w1-therm
pi@raspberrypi ~ $ cd /sys/bus/w1/devices/
pi@raspberrypi /sys/bus/w1/devices $ ls
28-00000482b243  w1_bus_master1
pi@raspberrypi /sys/bus/w1/devices $ cd 28-00000482b243
pi@raspberrypi /sys/bus/w1/devices/28-00000482b243 $ ls
driver  id  name  power  subsystem  uevent  w1_slave
pi@raspberrypi /sys/bus/w1/devices/28-00000482b243 $ cat w1_slave
71 01 4b 46 7f ff 0f 10 56 : crc=56 YES
71 01 4b 46 7f ff 0f 10 56 t=23062
pi@raspberrypi /sys/bus/w1/devices/28-00000482b243 $
```

The “w1_slave” file contains a bunch of data but the “t=23062” at the end is the temperature reading. In this example the temperature is 23.062 degrees Celsius (centigrade).
Try touching the sensor with your finger and then using “cat w1_slave” to take another reading
In order to read the temperature via Python here is a basic script :

```
#!/usr/bin/python
def gettemp(id):
  try:
    mytemp = ''
    filename = 'w1_slave'
    f = open('/sys/bus/w1/devices/' + id + '/' + filename, 'r')
    line = f.readline() # read 1st line
    crc = line.rsplit(' ',1)
    crc = crc[1].replace('\n', '')
    if crc=='YES':
      line = f.readline() # read 2nd line
      mytemp = line.rsplit('t=',1)
    else:
      mytemp = 99999
    f.close()
    return int(mytemp[1])
  except:
    return 99999
```

```
if __name__ == '__main__':
  # Script has been called directly
  id = '28-00000482b243'
  print "Temp : " + '{:.3f}'.format(gettemp(id)/float(1000))
```

2.3.2 Distance Sensor

In previous tutorials we've outlined temperature sensing, all of which can plug directly into the Raspberry Pi's GPIO ports. The HC-SR04 ultrasonic range finder is very simple to use, however the signal it outputs needs to be converted from 5V to 3.3V so as not to damage our Raspberry Pi! We'll introduce some Physics along with Electronics in this tutorial in order to explain each step!

What you'll need:

- HC-SR04
- 1kΩ Resistor
- 2kΩ Resistor
- Jumper Wires

Ultrasonic Distance Sensors:
Sound consists of oscillating waves through a medium (such as air) with the pitch being determined by the closeness of those waves to each other, defined as the frequency. Only some of the sound spectrum (the range of sound wave frequencies) is audible to the human ear, defined as the "Acoustic" range. Very low frequency sound below Acoustic is defined as "Infrasound", with high frequency sounds above, called "Ultrasound". Ultrasonic sensors are designed to sense object proximity or range using ultrasound reflection, similar to radar, to calculate the time it takes to reflect ultrasound waves between the sensor and a solid object. Ultrasound is mainly used because it's inaudible to the human ear and is relatively accurate within short distances. You could of course use Acoustic sound for this purpose, but you would have a noisy robot, beeping every few seconds.

A basic ultrasonic sensor consists of one or more ultrasonic transmitters (basically speakers), a receiver, and a control circuit. The transmitters emit a high frequency ultrasonic sound, which bounce off any nearby solid objects. Some of that ultrasonic noise is reflected and detected by the receiver on the sensor. That return signal is then processed by the control circuit to calculate the time difference between the signal being transmitted and received. This time can subsequently be used, along with some clever math, to calculate the distance between the sensor and the reflecting object.

The HC-SR04 Ultrasonic sensor we'll be using in this tutorial for the Raspberry Pi has four pins: ground (GND), Echo Pulse Output (ECHO), Trigger Pulse Input (TRIG), and 5V Supply (Vcc). We power the module using Vcc, ground it using GND, and use our Raspberry Pi to send an input signal to TRIG, which triggers the sensor to send an ultrasonic pulse. The pulse waves bounce off any nearby objects and some are reflected back to the sensor. The sensor detects these return waves and measures the time between the trigger and returned pulse, and then sends a 5V signal on the ECHO pin.

ECHO will be "low" (0V) until the sensor is triggered when it receives the echo pulse. Once a return pulse has been located ECHO is set "high" (5V) for the duration of that pulse. Pulse duration is the full time between the sensor outputting an ultrasonic pulse, and the return pulse being detected by the sensor receiver. Our Python script must therefore measure the pulse duration and then calculate distance from this.

IMPORTANT. The sensor output signal (ECHO) on the HC-SR04 is rated at 5V. However, the input pin on the Raspberry Pi GPIO is rated at 3.3V. Sending a 5V signal into that unprotected 3.3V input port could damage your GPIO pins, which is something we want to avoid! We'll need to use a small voltage divider circuit, consisting of two resistors, to lower the sensor output voltage to something our Raspberry Pi can handle.

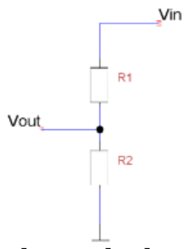
Voltage Dividers

A voltage divider consists of two resistors (R1 and R2) in series connected to an input voltage (Vin), which needs to be reduced to our output voltage (Vout). In our circuit, Vin will be ECHO, which needs to be decreased from 5V to our Vout of 3.3V.

The following circuit and simple equation can be applied to many applications where a voltage needs to be reduced. If you don't want to learn the techy bit, just grab 1 x 1kΩ and 1 x 2kΩ resistor.

$$Vout = \frac{\mathrm{Vin} * \mathrm{R2}}{\mathrm{R1} + \mathrm{R2}}$$

$$\frac{\mathrm{Vout}}{\mathrm{Vin}} = \frac{\mathrm{R2}}{\mathrm{R1} + \mathrm{R2}}$$



Without getting too deep into the math side, we only actually need to calculate one resistor value, as it's the dividing ratio that's important. We know our input voltage (5V), and our required output voltage (3.3V), and we can use any combination of resistors to achieve the reduction. I happen to have a bunch of extra 1kΩ resistors, so I decided to use one of these in the circuit as R1.

Plugging our values in, this would be the following:

$$\frac{3.3}{5} = \frac{R2}{1000 + R2}$$

$$.66 = \frac{R2}{1000 + R2}$$

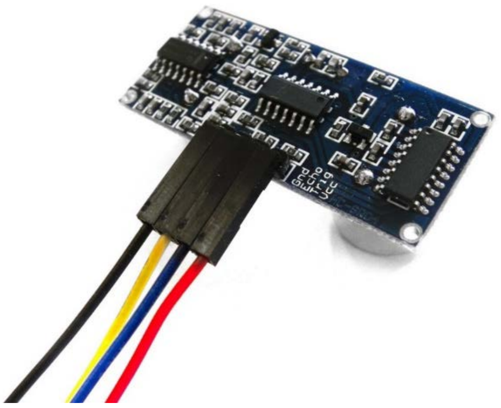
$$R2 = 1941$$

So, we'll use a 1kΩ for R1 and a 2kΩ resistor as R2!

Assemble the Circuit

We’ll be using four pins on the Raspberry Pi for this project: GPIO 5V [Pin 2]; Vcc (5V Power), GPIO GND [Pin 6]; GND (0V Ground), GPIO 23 [Pin 16]; TRIG (GPIO Output) and GPIO 24 [Pin 18]; ECHO (GPIO Input)

1. Plug four of your male to female jumper wires into the pins on the HC-SR04 as follows: Red; Vcc, Blue; TRIG, Yellow; ECHO and Black; GND.



2. Plug Vcc into the positive rail of your breadboard, and plug GND into your negative rail.
3. Plug GPIO 5V [Pin 2] into the positive rail, and GPIO GND [Pin 6] into the negative rail.
4. Plug TRIG into a blank rail, and plug that rail into GPIO 23 [Pin 16]. (You can plug TRIG directly into GPIO 23 if you want). I personally just like to do everything on a breadboard!
5. Plug ECHO into a blank rail, link another blank rail using R1 (1kΩ resistor)
6. Link your R1 rail with the GND rail using R2 (2kΩ resistor). Leave a space between the two resistors.
7. Add GPIO 24 [Pin 18] to the rail with your R1 (1kΩ resistor). This GPIO pin needs to sit between R1 and R2

That's it! Our HC-SR04 sensor is connected to our Raspberry Pi!

Sensing with Python

Now that we’ve hooked our Ultrasonic Sensor up to our Pi, we need to program a Python script to detect distance!

The Ultrasonic sensor output (ECHO) will always output low (0V) unless it’s been triggered in which case it will output 5V (3.3V with our voltage divider!). We therefore need to set one GPIO pin as an output, to trigger the sensor, and one as an input to detect the ECHO voltage change.

First, import the Python GPIO library, import our time library (so we make our Pi wait between steps) and set our GPIO pin numbering.

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
```

Next, we need to name our input and output pins, so that we can refer to it later in our Python code. We’ll name our output pin (which triggers the sensor) GPIO 23 [Pin 16] as TRIG, and our input pin (which reads the return signal from the sensor) GPIO 24 [Pin 18] as ECHO.

```
TRIG = 23
ECHO = 24
```

We’ll then print a message to let the user know that distance measurement is in progress.

```
print "Distance Measurement In Progress"
```

Next, set your two GPIO ports as either inputs or outputs as defined previously.

```
GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
```

Then, ensure that the Trigger pin is set low, and give the sensor a second to settle.

```
GPIO.output(TRIG, False)
print "Waiting For Sensor To Settle"
time.sleep(2)
```

The HC-SR04 sensor requires a short 10uS pulse to trigger the module, which will cause the sensor to start the ranging program (8 ultrasound bursts at 40 kHz) in order to obtain an echo response. So, to create our trigger pulse, we set out trigger pin high for 10uS then set it low again.

```
GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)
```

Now that we’ve sent our pulse signal we need to listen to our input pin, which is connected to ECHO. The sensor sets ECHO to high for the amount of time it takes for the pulse to go and come back, so our code therefore needs to measure the amount of time that the ECHO pin stays high. We use the “while” string to ensure that each signal timestamp is recorded in the correct order.

The time.time() function will record the latest timestamp for a given condition. For example, if a pin goes from low to high, and we’re recording the low condition using the time.time() function, the recorded timestamp will be the latest time at which that pin was low.

Our first step must therefore be to record the last low timestamp for ECHO (pulse_start) e.g. just before the return signal is received and the pin goes high.

```
while GPIO.input(ECHO)==0:
  pulse_start = time.time()
```

Once a signal is received, the value changes from low (0) to high (1), and the signal will remain high for the duration of the echo pulse. We therefore also need the last high timestamp for ECHO (pulse_end).

```
while GPIO.input(ECHO)==1:
  pulse_end = time.time()
```

We can now calculate the difference between the two recorded timestamps, and hence the duration of pulse (pulse_duration).

```
pulse_duration = pulse_end - pulse_start
```

With the time it takes for the signal to travel to an object and back again, we can calculate the distance using the following formula.

$$Speed = \frac{Distance}{Time}$$

The speed of sound is variable, depending on what medium it's travelling through, in addition to the temperature of that medium. However, some clever physicists have calculated the speed of sound at sea level so we'll take our baseline as the 343m/s. If you're trying to measure distance through water, this is where you're falling down – make sure you're using the right speed of sound!

We also need to divide our time by two because what we've calculated above is actually the time it takes for the ultrasonic pulse to travel the distance to the object and back again. We simply want the distance to the object! We can simplify the calculation to be completed in our Python script as follows:

$$34300 = \frac{Distance}{.5 * Time}$$

$$17150 = \frac{Distance}{Time}$$

$$17150 * Time = Distance$$

We can plug this calculation into our Python script:

```
distance = pulse_duration x 17150
```

distance = pulse_duration x 17150
Now we need to round our distance to 2 decimal places (for neatness!)

distance = round(distance, 2)

Then, we print the distance. The below command will print the word "Distance:" followed by the distance variable, followed by the unit "cm"

```
print "Distance:",distance,"cm"
```

Finally, we clean our GPIO pins to ensure that all inputs/outputs are reset

```
GPIO.cleanup()
```

2.4 Screen

7-inch Capacitive Touch Screen LCD, HDMI interface, supports various systems.

2.4.1 Features

- 800×480 high resolutions, touch control
- Supports Raspberry Pi, and driver is provided (works with custom Raspbian directly)
- Supports BB Black, comes with related images like: Angstrom
- Supports Banana Pi / Banana Pro, comes with related images like : Lubuntu, Raspbian
- Not only for mini-PCs, it can work as a computer monitor just like any other general HDMI screen
- (touch function is unavailable in this case)
- HDMI interface for displaying, USB interface for touch control

2.4.2 Working with Raspberry Pi

2.4.2.1 How to program Raspbian image file

In order to use with Raspberry Pi, you should configure the original system first. Of course, you can program a ready-to-use system image file to your Raspberry Pi board as well. In this section, we will illustrate how to program the image file by taking the ready-to-use system image file, 7inch HDMI LCD (B) Raspberry Pi 2 module B Raspbian image, as an example. This image file supports Raspberry Pi 2 Model B. Instead, for Raspberry Pi Model B/A+/B+, you can use 7inch HDMI LCD (B) Raspberry Pi B / B+ Raspbian image.

1) Download the zip file to your PC, unzip it and get an .img file.
2) Connect a TF card to your PC, and format your TF card with the SDFormatter.exe

Notices: The capability of TF card in used here should be more than 4GB. In this operation, a TF card reader is also required, which has to be purchased separately.

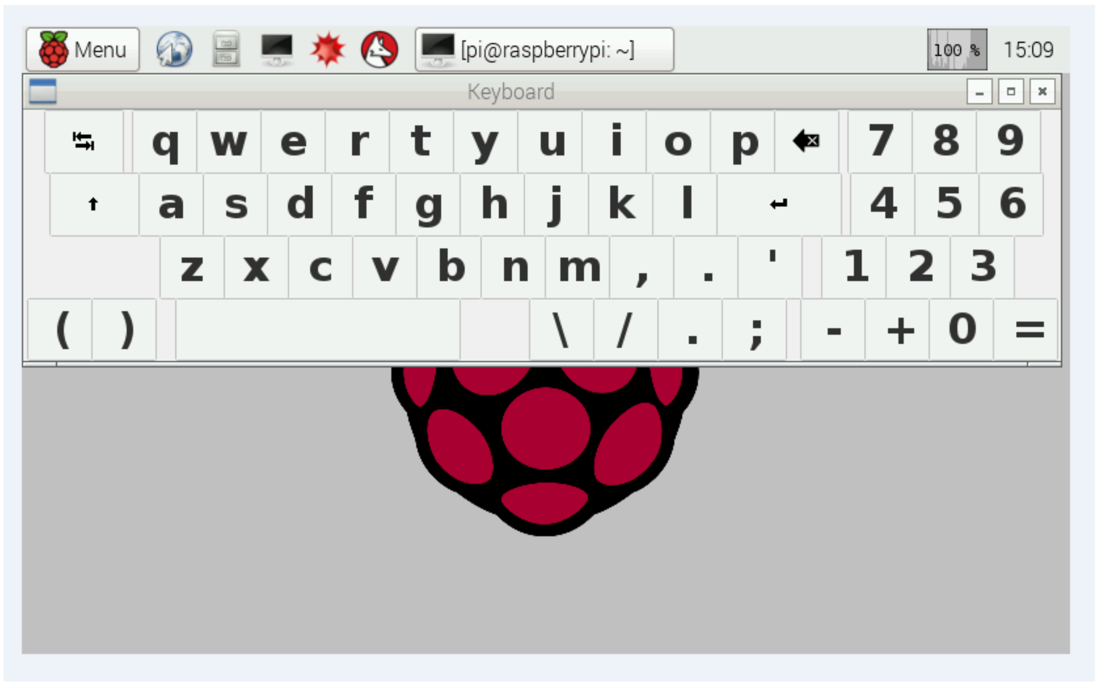
3) Start the Win32DiskImager.exe, and select the system image file copied into your PC, then, click the button Write to program the system image file.

2.4.2.2 Hardware connection

1. Connect the LCD to the HDMI on the Raspberry Pi board with a HDMI cable;
2. Connect the USB Touch interface on the LCD to the USB interface on the Raspberry Pi board with a USB type-A male to micro-B cable.

2.4.2.3 Virtual keyboard of Raspberry Pi

The Virtual keyboard of Raspbian system enables you to save the USB resource, providing easy system operations. After the LCD is working properly, this function can be invoked by the following command: DISPLAY=:0.0 matchbox-keyboard -s 100 extended Now, the virtual keyboard is ready to use, as Figure 2 shows.



2.4.2.4 Source code and protocol

1. Copy the source code to your Pi.
2. Execute the following command: cd wavesahre-7inch-touchscreen-driver chmod +x install.sh sudo apt-get update sudo ./install.sh
3. Shut down your Pi then power up again and usually you can use the display and touch functions.
4. Note: You may need for help about the source code but please consult the related website. We don't provide any supports of development environment building and source code modification.

Chapter 3
Fruit and Vegetable Recognition

3 Overview

This chapter includes the detailed explanation of the fruit and vegetables recognition technique, starting with some different image acquisition systems, different image processing software tools available, a comparison between them and why we used open cv with its python interface, passing by the image processing topics we used mentioning necessary theoretical background and ending at an explanation of the methodology we used to recognize fruits and vegetable.

3.1 Image acquisition system

3.1.1 How cameras work

A camera may work with the light of the visible spectrum or with other portions of the electromagnetic spectrum. A still camera is an optical device which creates a single image of an object or scene, and records it on an electronic sensor or photographic film. All cameras use the same basic design: light enters an enclosed box through a converging lens and an image is recorded on a light-sensitive medium.

A shutter mechanism controls the length of time that light can enter the camera. Most photographic cameras have functions that allow a person to view the scene to be recorded, allow for a desired part of the scene to be in focus, and to control the exposure so that it is not too bright or too dim. Digital cameras capture light onto an electronic image sensor. The lens of a camera captures the light from the subject and brings it to a focus on the sensor.

Due to the optical properties of photographic lenses, only objects within a limited range of distances from the camera will be reproduced clearly. The process of adjusting this range is known as changing the camera's focus.

The size of the aperture and the brightness of the scene controls the amount of light that enters the camera during a period of time, and the shutter controls the length of time that the light hits the recording surface. Equivalent exposures can be made using a large aperture size with a fast shutter speed and a small aperture with a slow shutter.

1.1.1 Camera types

3.1.1.1 Webcam

A webcam is a video camera that feeds or streams its image in real time to or through a computer to computer network. When "captured" by the computer, the video stream may be saved, viewed or sent on to other networks via systems such as the internet, and email as an attachment.

A webcam is generally connected by a USB cable, or similar cable, or built into computer hardware, such as laptops. We used in our project HP HD-4110 - 1080P widescreen webcam. Its autofocus capability reduced out of focus blurring and reduced the required preprocessing.

3.1.1.2 Camera module

A camera module is an image sensor integrated with control electronics and an interface. Buying a camera module for a specific development board is better than using a webcam, but if the development board doesn't have a specific camera module, buying a webcam will be best solution. An example for camera module is raspberry pi camera module. The limitation of using this type of camera was its low resolution.

3.1.1.3 Hidden camera

The camera is "hidden" because it is either not visible to the subject being filmed, or is disguised as another object like Spy Camera for Raspberry Pi.

3.1.1.4 Document camera

Document cameras, also known as visual presenters or docucams, are real-time image capture devices for displaying an object to a large audience. Document camera is able to magnify and project the images of actual, three-dimensional objects, as well as transparencies. They are, in essence, high resolution web cams, mounted on arms so as to facilitate their placement over a page.

3.1.1.5 Stereo camera

A stereo camera is a type of camera with two or more lenses with a separate image sensor or film frame for each lens. This allows the camera to simulate human binocular vision, and therefore gives it the ability to capture three-dimensional images, a process known as stereo photography.

3.1.1.6 CCTV camera

Closed-circuit television (CCTV) cameras can produce images or recordings for surveillance purposes, and can be either video cameras, or digital stills cameras. Video cameras are either analogue or digital, which means that they work on the basis of sending analogue or digital signals to a storage device such as a video tape recorder (analog) or desktop computer or laptop computer (Digital).

3.1.1.7 Line scan camera

Line scan cameras contain a single row of pixels used to capture data very quickly. As the object moves past the camera, a complete image can be reconstructed in software line by line. Line scan systems are best employed in high-speed processing or fast-moving conveyor line applications. With a single row of pixels, line scan cameras can build continuous images not limited to a specific vertical resolution. This allows for much higher resolutions than with area scan cameras.

Unlike area scan cameras, a line scan camera can expose a new image while the previous image is still transferring its data. This is because the pixel readout is faster than the camera exposure. When building a composite image, the line scan camera can either move over an object or have moving objects presented to it. Coordination of production/camera motion and image acquisition timing are critical for line scan cameras, but they require only simple illumination.

3.1.1.8 Area scan camera

Area scan cameras contain a matrix of pixels that capture an image of a given scene. They are more general purpose than line scan cameras, and offer easier setup and alignment. Area scan cameras are best suited towards applications where the object is stationary, even if only momentarily. Area scan cameras provide a fixed resolution, allowing for simplified installation where moving cameras or objects are not desired or practical.

Area scan cameras can image a defined area quickly, whereas a line scan camera must be moved over the area to produce a similar image. Greater flexibility can be had with area scan cameras, as a single image can be segmented into multiple regions-of-interest to look for specific objects rather than having to process the entire image.

3.2 Image processing software tools

There are many software tools that are used in image processing. This tools includes different image processing libraries provided by different programing languages. This tools also includes programs such as MATLAB and LABVIEW. In the following sections we will explore the difference between some of the image processing software tools, the usage of each tool and why we selected openCV with its python interface.

3.2.1 Programming languages

In the next sections different programming languages libraries for image processing and their basic features are discussed.

3.2.1.1 JAVA

- **JAITools and Jiffle**
  - Open-source raster image processing library for Java
  - Provides a range of image operators and utilities compatible with the JAI (Java Advanced Imaging) framework.
  - Also includes Jiffle: a scripting language for image creation and raster algebra.
  - More information about JAITools can be found at the official website (1).
- **ImageJ (Program and API)**
  - ImageJ is written in Java, which allows it to run on Linux, Mac OS X and Windows, in both 32-bit and 64-bit modes.
  - More information about ImageJ can be found in (2)
- **OpenCV**
  This implementation is not a complete port of OpenCV. Currently, this library supports real-time capture, video file import, basic image treatment such as brightness, contrast and threshold and object detection.

3.2.1.2 Python

- **NumPy and SciPY**: is a library for the Python programming language that (among other things) provides support for large, multi-dimensional arrays. These libraries are necessary for using Open CV python.
- **OpenCV**
- **PIL and PILLOW**: PIL and Pillow have their place if you need to do some quick and dirty image manipulations, but if you're serious about learning about image processing, computer vision, and image search engines Open CV and Simple CV are recommended.
- **Scikit-learn**: Scikit-learn isn't an image processing or computer vision library — it's a machine learning library. That said, you can't have advanced computer vision techniques without some sort of machine learning, whether it be clustering, vector quantization, classification models, etc. Scikit-learn also includes a handful of image feature extraction functions as well.
- **Ilastik**: It is mainly for image segmentation and classification.
- **scikit-image**

From above, python provide powerful image processing libraries like PIL and Open CV. Python code is implemented in much less time than C/C++. Open CV Python codes approximately run at the same speed as C/C++ open CV using Open CV Python bindings.

3.2.1.3 C/C++

- **OpenCV**
- **EmguCV**: is a cross platform .Net wrapper to the OpenCV image processing library (If C# or VB or VC++ is used)
- **Magick++**
- **CImg**:
  (Open source) which is used to load/save various file formats, access pixel values, display/transform/filter images, draw primitives (text, faces, curves, 3d

objects, ...), compute statistics, manage user interactions on images. It fully works on different operating systems (Unix, Windows, Mac OS X, BSD) and is compatible with various C++ compilers.

- **DIPlib**:
  is a platform “independent” scientific image processing library written in C. It consists of a large number of functions for processing and analyzing multi-dimensional image data. The library provides functions for performing transforms, filter operations, object generation, local structure analysis, object measurements and statistical analysis of images. Key design features include ample support for different data types (binary, integer, floating point, complex) and dimensionalities.

3.2.2 Programs

- **MATLAB**: MATLAB provides DIP image is a MATLAB toolbox for scientific image processing and analysis. It supports image visualization, image manipulation, image processing and image analysis
  Computer vision toolbox, image processing toolbox, and image acquisition toolbox are also provided by MATLAB.
- **LABVIEW**: LABVIEW provides a computer vision toolbox. Although, using LABVIEW will depend on using an NI kit in the processing.
  Programs are known for their large size and low speed, also they are not suitable for real time application or embedded applications.

3.3 Image processing theoretical background

3.3.1 What is image processing

Image processing involves changing the nature of an image in order to either improve its pictorial information for human interpretation, or render it more suitable for autonomous machine perception which is our case. A procedure which makes an image look better may be the very worst procedure for machine perception. Humans like their images to be sharp, clear and detailed; machines prefer their images to be simple, uncluttered and with less details.

3.3.2 Aspects of image processing

It is convenient to subdivide different image processing algorithms into broad subclasses. There are different algorithms for different tasks and problems, and often we would like to distinguish the nature of the task at hand.

Image enhancement. This refers to processing an image so that the result is more suitable for a particular application. Example include:

- sharpening or de-blurring an out of focus image,
- highlighting edges,
- improving image contrast, or brightening an image
- removing noise.

Image restoration. This may be considered as reversing the damage done to an image by a known cause, for example:

- removing of blur caused by linear motion,

- removal of optical distortions,
- removing periodic interference.

Image segmentation. This involves subdividing an image into constituent parts, or isolating certain aspects of an image:

- Finding lines, circles, or particular shapes in an image,
- in an aerial photograph, identifying cars, trees, buildings, or roads.

These classes are not disjoint; a given algorithm may be used for both image enhancement or for image restoration. However, we should be able to decide what it is that we are trying to do with our image: simply make it look better (enhancement), or removing damage (restoration).

3.3.3 Image processing task steps

Acquiring the image. First we need to produce a digital image from a paper envelope. This can be done using either a CCD camera, or a scanner.

Preprocessing. This is the step taken before the major image processing task. The problem here is to perform some basic tasks in order to render the resulting image more suitable for the job to follow. In this case it may involve enhancing the contrast, removing noise, or identifying regions likely to contain the postcode.

Segmentation. Here is where we actually get the postcode; in other words, we extract from the image that part of it which contains just the postcode.

Representation and description. These terms refer to extracting the particular features which allow us to differentiate between objects. Here we will be looking for curves, holes and corners which allow us to distinguish the different digits which constitute a postcode.

Recognition and interpretation. This means assigning labels to objects based on their descriptors (from the previous step), and assigning meanings to those labels. So we identify particular digits, and we interpret a string of four digits at the end of the address as the postcode.

3.3.4 Types of images

Binary. Each pixel is just black or white. Since there are only two possible values for each pixel, we only need one bit per pixel. Such images can therefore be very efficient in terms of storage. Images for which a binary representation may be suitable include text (printed or handwriting), fingerprints, or architectural plans.

Greyscale. Each pixel is a shade of grey, normally from 0 (black) to 255 (white). This range means that each pixel can be represented by eight bits, or exactly one byte. This is a very natural range for image file handling. Other greyscale ranges are used, but generally they are a power of 2. Such images arise in medicine (X-rays), images of printed works, and indeed 256 different grey levels is sufficient for the recognition of most natural objects. An example is the street scene shown in figure 1.17 below.

True color, or RGB. Here each pixel has a particular color; that color being described by the amount of red, green and blue in it. If each of these components has a range of 0-255, this gives a total of different possible colors in the image of 255^3. This is enough colors for any image. Since the total number of bits required for each pixel is 24, such images are also called 24-bit color images. Such an image may be considered as consisting of a stack of three matrices; representing the red, green and blue values for each pixel. This means that for every pixel there correspond three values. An example is shown in figure 1.18.

Indexed. Most color images only have a small subset of the more than sixteen million possible colors. For convenience of storage and file handling, the image has an associated color map or color palette, which is simply a list of all the colors used in that image. Each pixel has a value which does not give its color (as for an RGB image), but an index to the color in the map.
It is convenient if an image has 256 colors or less, for then the index values will only require one byte each to store. Some image file formats (for example, Compuserve GIF), allow only 256 colors or fewer in each image, for precisely this reason. Figure 1.19 shows an example. In this image the indices, rather than being the grey values of the pixels, are simply indices into the color map. Without the color map, the image would be very dark and colorless. In the figure, for example, pixels labelled 5 correspond to 0.2627, 0.2588, 0.2549, which is a dark greyish color.

3.3.5 Image processing operations classes

image processing operations may be divided into three classes based on the information required to perform the transformation. These classes are point processing, neighbors processing and transforms.

3.3.5.1 Point processing

A pixel's grey value is changed without any knowledge of its surrounds. Although point operations are the simplest, they contain some of the most powerful and widely used of all image processing operations. They are especially useful in image pre-processing, where an image is required to be modified before the main job is attempted.

3.3.5.2 Neighbors processing

3.3.5.2.1 Neighbors meaning

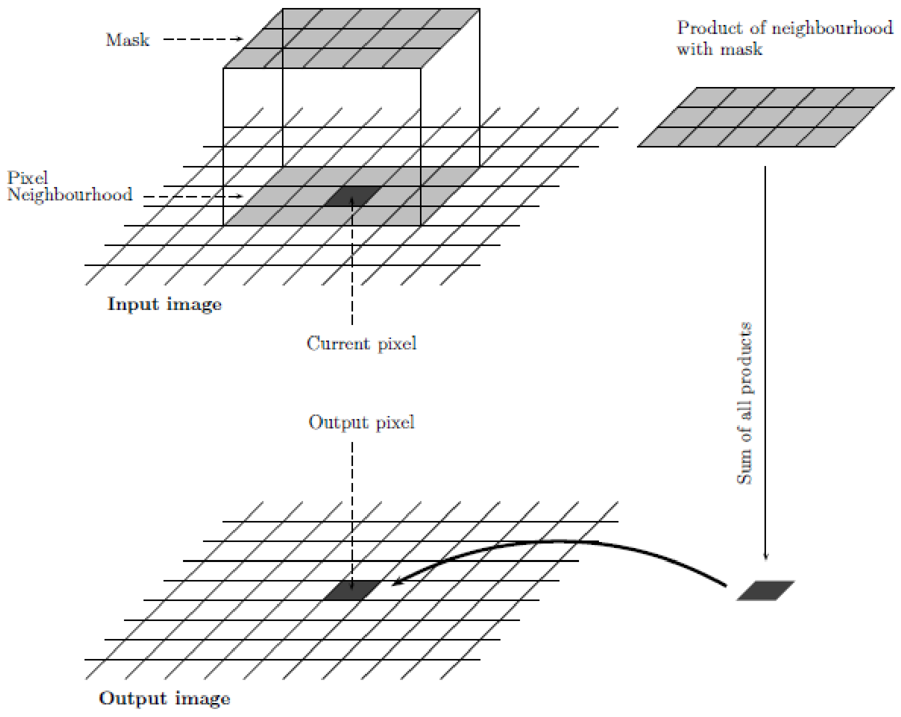
A pixel P at coordinates (x, y) has four horizontal and vertical neighbors whose coordinates are given by (x + 1, y), (x 1, y), (x, y + 1), (x, y - 1) This set of pixels, called the 4-neighbors of p, is denoted by N4(p). Each pixel is a unit distance from (x, y), and some of the neighbor locations of p lie outside the digital image if (x, y) is on the border of the image. The four diagonal neighbors of p have coordinates (x + 1, y + 1), (x + 1, y-1), (x - 1, y + 1), (x 1, y -1) and are denoted by N0(p), These points, together with the 4-neighbors, are called the 8-neighbors of p, denoted by N8(p). As before, some of the neighbor locations in N0(p) and N8(p) fall outside the image if (x, y) is on the border of the image.

3.3.5.2.2 Masks and filtering

An image can be modified by applying a particular function to each pixel value which is the case of point processing. Neighborhood processing may be considered as an extension of this, where a function is applied to a neighborhood of each pixel. The idea is to move a mask: a rectangle (usually with sides of odd length) or other shape over the given image. As we do this, we create a new image whose pixels have grey values calculated from the grey values under the mask, as shown in figure. The combination of mask and function is called a filter. If the function by which the new grey value is calculated is a linear function of all the grey values in the mask, then the filter is called a linear filter. A linear filter can be implemented by multiplying all elements in the mask by corresponding elements in the neighborhood, and adding up all these products. So, to change the grey level of a given pixel we need only know the value of the grey levels in a small neighborhood of pixels around the given pixel.

Here is the code to apply this low pass filter to an image in Python with different Kernel size:

```
import cv2
import numpy as np
img = cv2.imread('input.jpg')
rows, cols = img.shape[:2]
kernel_identity = np.array([[0,0,0], [0,1,0], [0,0,0]])
kernel_3x3 = np.ones((3,3), np.float32) / 9.0
kernel_5x5 = np.ones((5,5), np.float32) / 25.0
cv2.imshow('Original', img)
output = cv2.filter2D(img, -1, kernel_identity)
cv2.imshow('Identity filter', output)
output = cv2.filter2D(img, -1, kernel_3x3)
cv2.imshow('3x3 filter', output)
output = cv2.filter2D(img, -1, kernel_5x5)
cv2.imshow('5x5 filter', output)
cv2.waitKey(0)
```



3.3.5.3 Transforms

A transform represents the pixel values in some other, but equivalent form. Transforms allow for some very efficient and powerful algorithms, as we shall see later on. We may consider that in using a transform, the entire image is processed as a single large block. Discrete Fourier transform and discrete Wavelet transform are examples of transforms.

The Fourier Transform is of fundamental importance to image processing. It allows us to perform tasks which would be impossible to perform any other way; its efficiency allows us to perform other tasks more quickly. The Fourier Transform provides, among other things, a powerful alternative to linear spatial filtering; it is more efficient to use the Fourier transform than a spatial filter for a large filter. The Fourier Transform also allows us to isolate and process particular image frequencies, and so perform low-pass and high-pass filtering with a great degree of precision.

Since our project did not involve large noisy images, spacial domain techniques were efficient and did not need high computational power required for the transform.

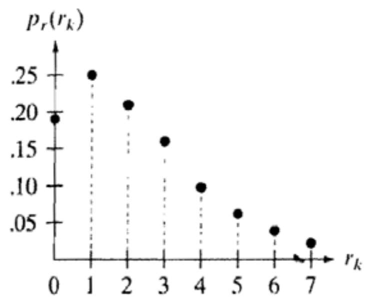
3.3.6 Preprocessing and image restoration

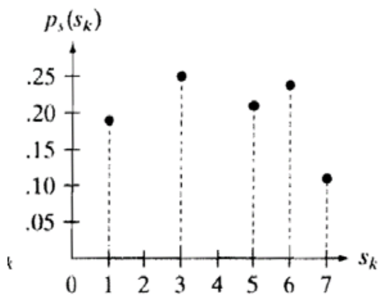
Image restoration concerns the removal or reduction of degradations which have occurred during the acquisition of the image. Such degradations may include noise, which are errors in the pixel values, or optical effects such as out of focus blurring, or blurring due to camera motion. We shall see that some restoration techniques can be performed very successfully using neighborhood operations, while others require the use of frequency domain processes. Image restoration remains one of the most important areas of image processing, but in this chapter the emphasis will be on the techniques for dealing with restoration, rather than with the degradations themselves, or the properties of electronic equipment which give rise to image degradation.

3.3.6.1 Histogram equalization

Given a greyscale image, its histogram consists of the histogram of its grey levels; that is, a graph indicating the number of times each grey level occurs in the image. We can infer a great deal about the appearance of an image from its histogram. In a dark image, the grey levels (and hence the histogram) would be clustered at the lower end, in a uniformly bright image, the grey levels would be clustered at the upper end and in a well contrasted image, the grey levels would be well spread out over much of the range.

Given a poorly contrasted image, we would like to enhance its contrast, by spreading out its histogram using histogram equalization.





3.3.6.2 Noise reduction

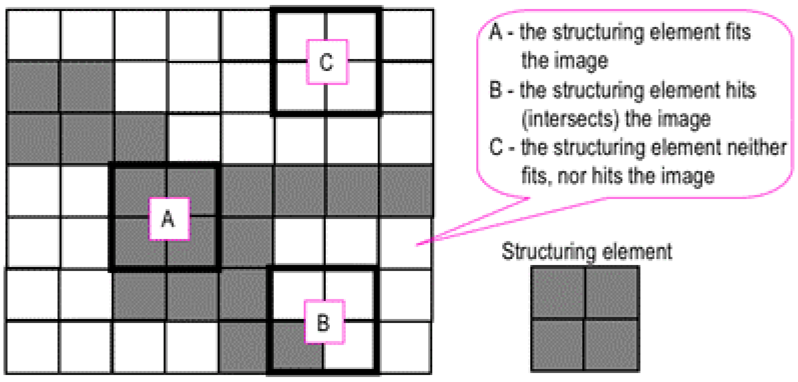
We may define noise to be any degradation in the image signal, caused by external disturbance. If an image is being sent electronically from one place to another, via satellite or wireless transmission, or through networked cable, we may expect errors to occur in the image signal. These errors will appear on the image output in different ways depending on the type of disturbance in the signal. Usually we know what type of errors to expect, and hence the type of noise on the image; hence we can choose the most appropriate method for reducing the effects. Cleaning an image corrupted by noise is thus an important area of image restoration.

Out of focus noise and motion blurring can be reduced by nonlinear spacial filters such as Weiner filter. Noise with global effects such as periodic noise is reduced by frequency domain filtering using Fourier transform.

3.3.7 Morphological filters

Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image. Morphological operations rely only on the relative ordering of pixel values, not on their numerical values, and therefore are especially suited to the processing of binary images.

Morphological techniques probe an image with a small shape or template called a structuring element. The structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighborhood of pixels. Some operations test whether the element "fits" within the neighborhood, while others test whether it "hits" or intersects the neighborhood.


C
A
B
A - the structuring element fits the image
B - the structuring element hits (intersects) the image
C - the structuring element neither fits, nor hits the image
Structuring element

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called structuring element or kernel which decides the nature of operation. Two basic morphological operators are Erosion and Dilation. Then its variant forms like Opening, Closing, Gradient also comes into play. In the following section morphological topics are discussed with python examples.

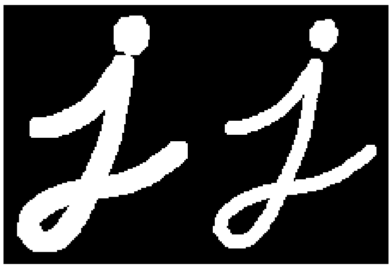
3.3.7.1 Dilation

Dilation is also known as Minkowski addition; information. As you see in figure, dilation has the effect of increasing the size of an object.

It is just opposite of erosion. Here, a pixel element is ‘1’ if at least one pixel under the kernel is ‘1’. So it increases the white region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won’t come back, but our object area increases. It is also useful in joining broken parts of an object.

Example of dilation with Open CV:

```
dilation = cv2.dilate(img, kernel, iterations = 1)
```

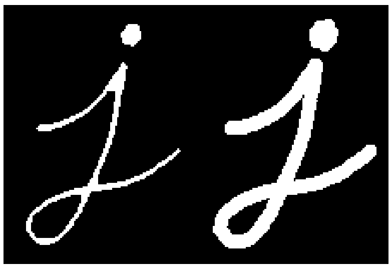


3.3.7.2 Erosion

Erosion has the effect of making an image thinner. The basic idea of erosion is just like soil erosion only it erodes away the boundaries of foreground object (Always try to keep foreground in white). So what does it do? The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero). So what happened is that, all the pixels near boundary will be discarded depending upon the size of kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white.

Example of erosion with Open CV:

```
kernel = np.ones((5,5), np.uint8)
erosion = cv2.erode(img, kernel, iterations = 1)
```

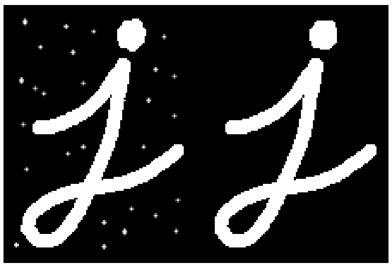


3.3.7.3 Opening

Opening is just another name of erosion followed by dilation. It is useful in removing noise, as we explained above.

Here we use the function, cv2.morphologyEx()

Example of opening with Open CV:



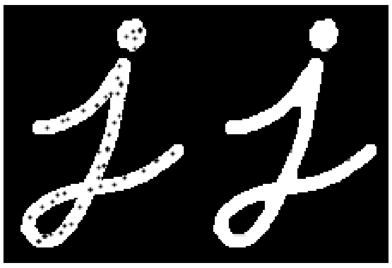
```
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
```

3.3.7.4 *Closing*

Closing is reverse of Opening, Dilation followed by Erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object.

Example of closing with Open CV:

```
closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
```



3.3.7.5 *Structuring element*

OpenCV has a function, cv2.getStructuringElement(). You just pass the shape and size of the kernel.

Example of getting structuring element with Open CV:

```
cv2.getStructuringElement(cv2.MORPH_RECT,(5,5))
```

3.3.8 Segmentation

Image segmentation is the process of partitioning an image into meaningful regions. Regions can be foreground versus background or individual objects in the image. The regions are constructed using some feature such as color, edges, or neighbor similarity.

3.3.8.1 *Edge detection*

Edges contain some of the most useful information in an image. We may use edges to measure the size of objects in an image; to isolate particular objects from their background; to recognize or classify objects. An edge may be loosely defined as a local discontinuity in the pixel values which exceeds a given threshold. Here we discuss the canny edge detection algorithm.

Canny edge detection

Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in 1986. It is a multi-stage algorithm and we will go through each stages.

1- Noise reduction: Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter.

2- Finding Intensity Gradient of the Image: Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (Gx) and vertical direction (Gy). From these two images, we can find edge gradient and direction for each pixel as follows:

$$\text{Edge Gradient} = \sqrt{a^2 + b^2}$$

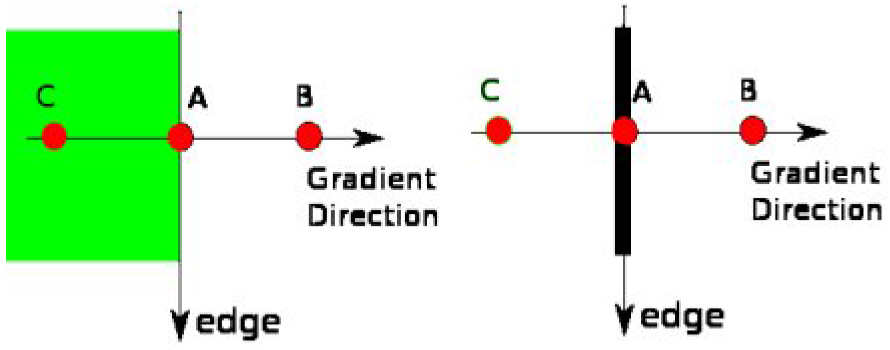
Where $a = G_X$

$b = G_y$

$$\text{Angle}(\Theta) = \tan^{-1}(b/a)$$

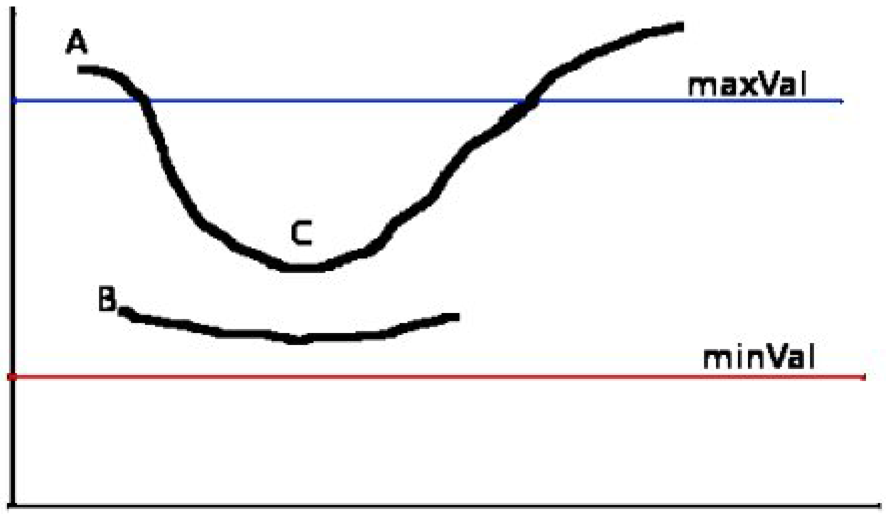
Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

3- Non-maximum suppression: After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. Check the image below:



Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero). In short, the result you get is a binary image with “thin edges”.

4- Hysteresis Thresholding: This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to “sure-edge” pixels, they are considered to be part of edges. Otherwise, they are also discarded. See the image below:



The edge A is above the maxVal, so considered as “sure-edge”. Although edge C is below maxVal, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above minVal and is in same region as that of edge C, it is not connected to any “sure-edge”, so that is discarded. So it is very important that we have to select minVal and maxVal accordingly to get the correct result. This stage also removes small pixels' noises on the assumption that edges are long lines. So what we finally get is strong edges in the image.

Canny's approach is based on three basic objectives:

1. **Low error rate.**
   All edges should be found, and there should be no spurious responses. That is, the edges detected must be as close as possible to the true edges.
2. **Edge points should be well localized.**
   The edges located must be as close as possible to the true edges. That is, the distance between a point marked as an edge by the detector and the center of the true edge should be minimum.
3. **Single edge point response.**
   The detector should return only one point for each true edge point. That is, the number of local maxima around the true edge should be minimum. 'This means that the detector should not identify multiple edge pixels where only a single edge point exists.

The essence of Canny's work was in expressing the preceding three criteria mathematically and then attempting to find optimal solutions to these formulations. In general, it is difficult (or impossible) to find a closed-form solution that satisfies all the preceding objectives.

Canny edge detection with open CV

OpenCV puts all the above in single function, cv2.Canny(). We will see how to use it. First argument is our input image. Second and third arguments are our minVal and maxVal respectively. Third argument is aperture_size. It is the size of Sobel kernel used for find image gradients. By default, it is 3. Last argument is L2gradient which specifies the equation for finding gradient magnitude. If it is True, it uses the equation mentioned above which is more accurate, otherwise it uses this function: Edge Gradient = |a| + |b|. By default, it is False.

```
img = cv2.imread('messi5.jpg',0)
edges = cv2.Canny(img, 100, 200)
```

3.3.9 Contours

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition. For better accuracy, use binary images. So before finding contours, apply threshold or canny edge detection.

3.3.9.1 Chain codes

Chain codes are used to represent a boundary by a connected sequence of straight-line segments of specified length and direction. Typically, this representation is based on 4- or 8-connectivity of the segments. The direction of each segment is coded by using a numbering scheme. A boundary code formed as a sequence of such directional numbers is referred to as a Freeman chain code.

Digital images usually are acquired and processed in a grid format with equal spacing in the x- and y-directions, so a chain code can be generated by following a boundary in, say, a clockwise direction and assigning a direction to the segments connecting every pair of pixels. This method generally is unacceptable for two principal reasons: (1) The resulting chain tends to be quite long and (2) any small disturbances along the boundary due to noise or imperfect segmentation cause changes in the code that may not be related to the principal shape features of the boundary.

An approach frequently used to circumvent these problems is to resample the boundary by selecting a larger grid spacing. Then, as the boundary is traversed, a boundary point is assigned to each node of the large grid, depending on the proximity of the original boundary to that node. The resampled boundary obtained in this way then can be represented by a 4- or 8-code, the coarser boundary points represented by an 8-directional chain code: It is a simple matter to convert from an 8-code to a 4-code, and vice versa The starting point is (arbitrarily) at the topmost, leftmost point of the boundary, which gives the chain code 0766 ... 12. As might be expected, the accuracy of the resulting code representation depends on the spacing of the sampling grid.

3.3.9.2 Contours in Open CV

In OpenCV, finding contours is like finding white object from black background. So remember, object to be found should be white and background should be black.

there are three arguments in cv2.findContours() function, first one is source image, second is contour retrieval mode, third is contour approximation method. And it outputs the image, contours and hierarchy. contours are a Python list of all the contours in the image. Each individual contour is a Numpy array of (x, y) coordinates of boundary points of the object.

To draw the contours, cv2.drawContours function is used. It can also be used to draw any shape provided you have its boundary points. Its first argument is source image, second argument is the contours which should be passed as a Python list, third argument is index of contours (useful when drawing individual contour. To draw all contours, pass (-1) and remaining arguments are color, thickness etc.

To draw all the contours in an image:

```
img = cv2.drawContours(img, contours, -1, (0,255,0), 3)
```

To draw an individual contour, say 4th contour:

```
img = cv2.drawContours(img, contours, 3, (0,255,0), 3)
```

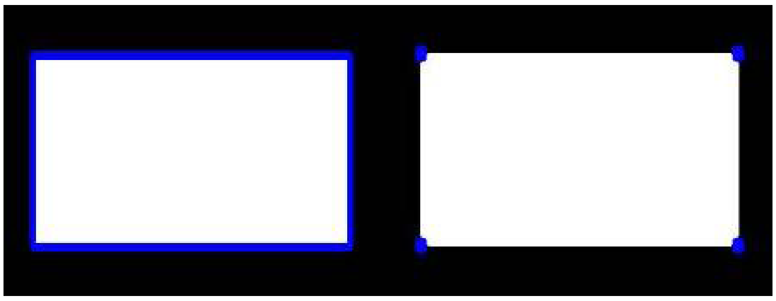
the third argument in cv2.findContours function is called the contour approximation method.

Above, we told that contours are the boundaries of a shape with same intensity. It stores the (x, y) coordinates of the boundary of a shape. But does it store all the coordinates? That is specified by this contour approximation method.

If you pass cv2.CHAIN_APPROX_NONE, all the boundary points are stored. But we do not need all these points.

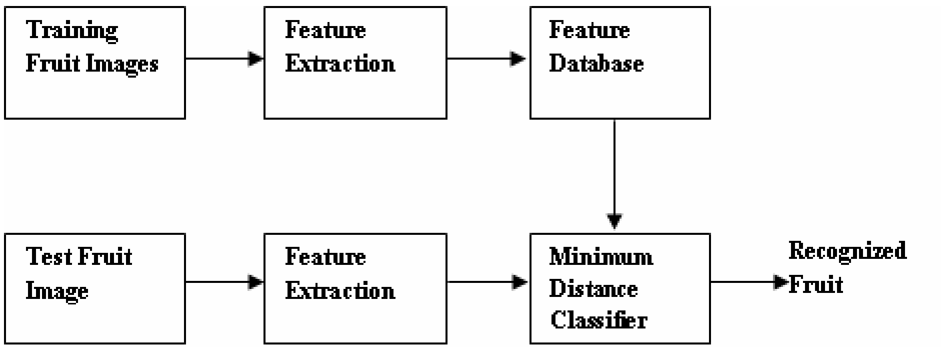
For example, you found the contour of a straight line. Do you need all the points on the line to represent that line? The answer is No, we need just two end points of that line. This is what cv2.CHAIN_APPROX_SIMPLE does. It removes all redundant points and compresses the contour, thereby saving memory.

Below image of a rectangle demonstrate this technique. Just draw a circle on all the coordinates in the contour array (drawn in blue color). First image shows points I got with cv2.CHAIN_APPROX_NONE (734 points) and second image shows the one with cv2.CHAIN_APPROX_SIMPLE (only 4 points). See, how much memory it saves.



3.4 Recognition Methodology

In the previous sections we discussed the image processing topics involved with the recognition technique we used. Based on (3, 2010) the computer vision strategies used to recognize a fruit rely on four basic features which characterize the object: intensity, color, shape and texture. Different features of color, size, shape, and texture are combined together for each fruit type as vector of 42 feature. Classification is based on the minimum Euclidean distance between the unknown type vector and a stored known fruits and vegetables vectors.



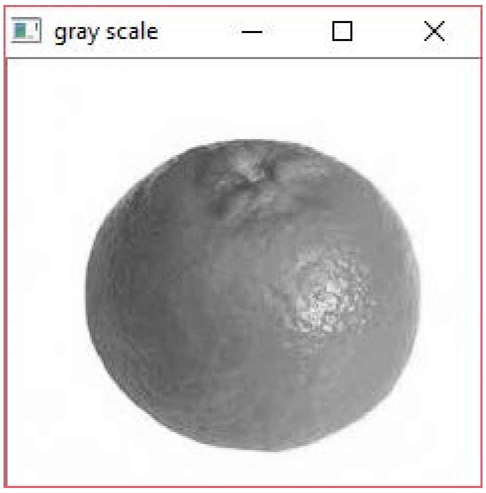
3.4.1 Algorithm for Extracting Region of Interest

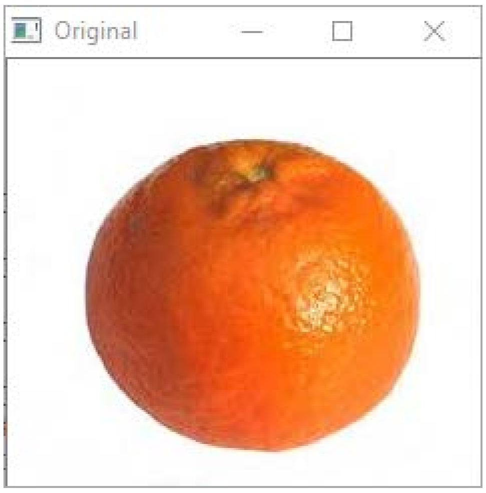
1- The input image is converted to gray scale.

```
im = cv2.imread('C:/Users/Nora Basha/Desktop/tan.jpg')
imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
```

After the image is acquired, it is read using cv2.imread() function, then the image is converted using cv2.cvtColor into gray scale.

Note that the image is read without passing the color argument to cv2.imread() as we will use the gray and the colored version of the image later in the code.



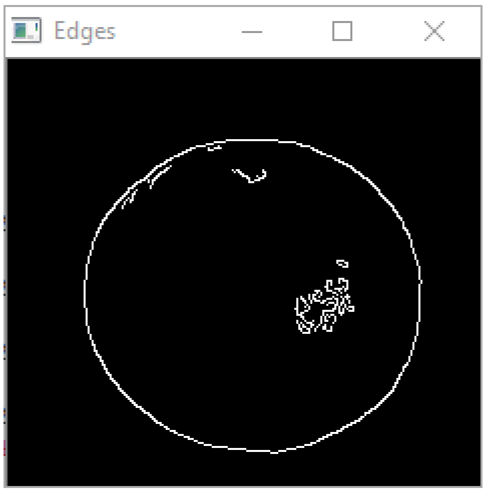


2- **Perform edge detection operation on the gray scale image using Canny algorithm to produce a binary image.**

```
edges = cv2.Canny(imgray, 127, 255)
```

Edges have various detection Algorithm as mentioned in the above sections. We selected Canny Algorithm as it provides noise reduction as well as acceptable edge detection compared to using Sobel filter.

The threshold value of cv2.Canny() is selected by trying different values till finding the most acceptable edge detection results.

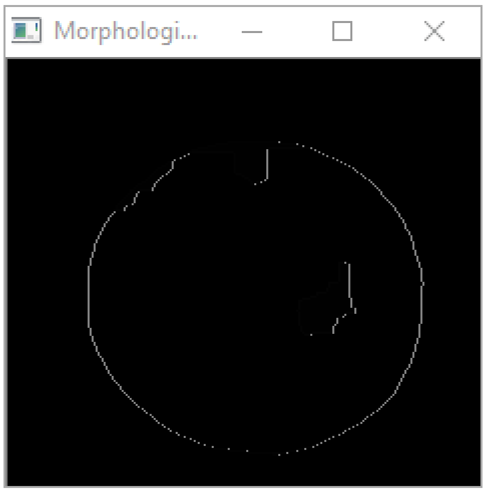


3- **Close small holes using the Closing morphological operator with a rectangular structuring element.**

```
k=cv2.getStructuringElement(cv2.MORPH_RECT,(20,20))
closed=cv2.morphologyEx(edges, cv2.MORPH_CLOSE, k)
```

Closing is performed on the edges to close small holes so that continuous contours are detected easily in the next step.

Structuring element shape is selected according to trial.



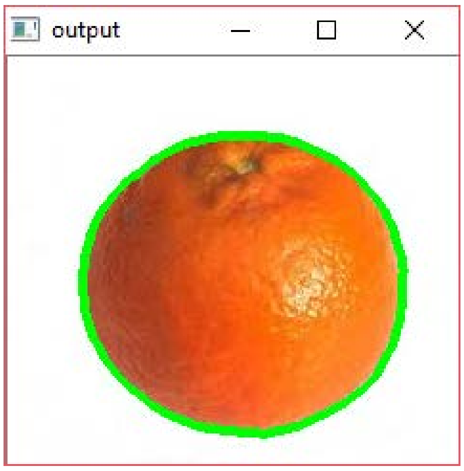
4- **Find the area of the Region of Interest from the binary image, calculate the contour length and Area.**

```
image, contours, hierarchy = cv2.findContours(closed, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
cnt=contours[0]
```

Our region of interest is the fruit or vegetable outer shape which can be now easily found using chain approximation technique of finding contours.

All contours are detected using cv2.findContours(), then they are stored in a numpy array called ‘contours’.

The desired contour is easily isolated from the rest of contours in the third line.



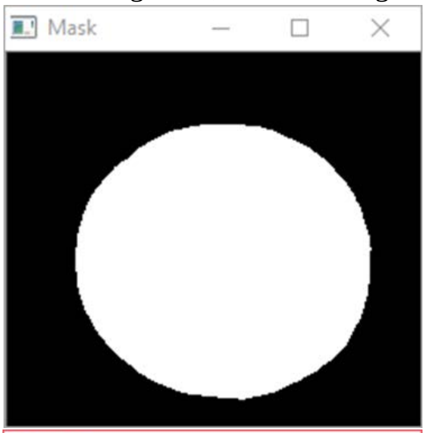
5- **Split the input image into its RGB components, then crop the colored region of interest.**

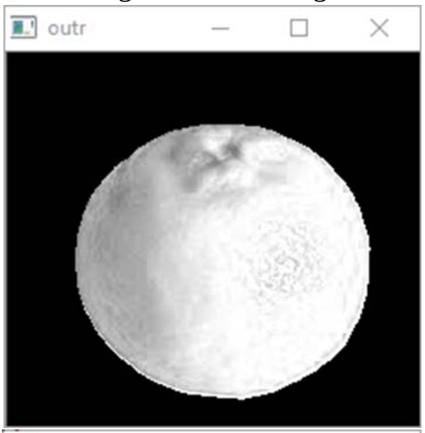
```
b,g,r = cv2.split(im)
mask = np.zeros_like(imgray)
cv2.drawContours(mask, [cnt], 0, 255, -1)
outb = np.zeros_like(imgray)
outb[mask == 255] = b[mask == 255]
outg = np.zeros_like(imgray)
outg[mask == 255] = g[mask == 255]
```

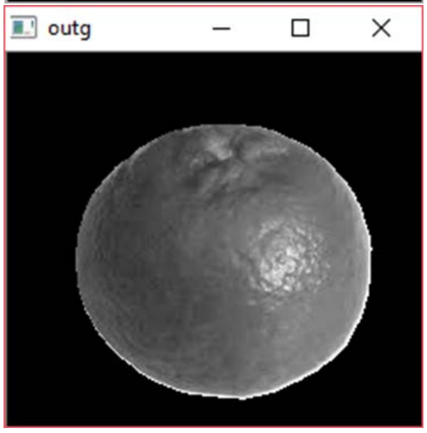
```
outr = np.zeros_like(imgray)
outr[mask == 255] = r[mask == 255]
out= cv2.merge([outb, outg, outr])
```

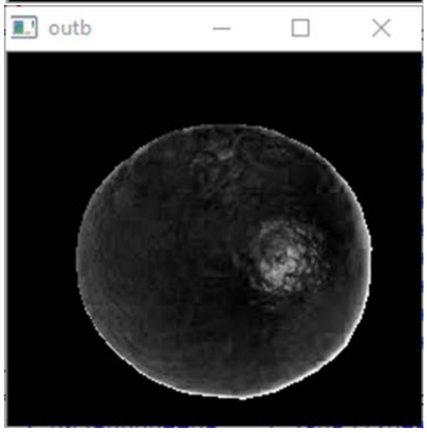
To crop the colored region of interest, we crop the detected contour from the R, G and B components then merge them again. The cropping is done using the following steps:

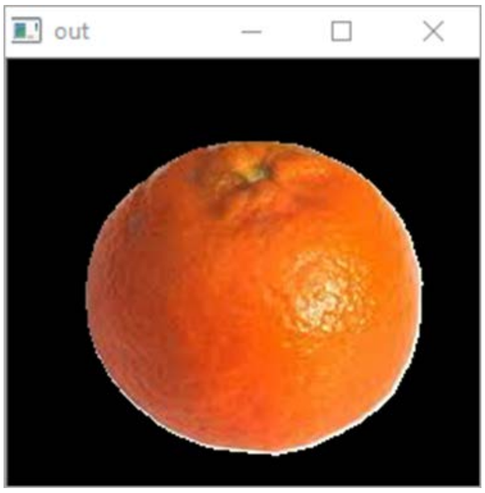
a. A mask of zeros is generated at the same size of the gray scale image using a numpy array as shown in figure 2.21.
b. The detected contour stored in 'cnt' is drawn on the mask as a white contour by cv2.drawContours(), notice that the mask now is a binary image.
c. Generate three numpy array of zeros at the same size of the gray image. These arrays are used to store the output of filtering the mask in step a and the R, G and B component.
d. For the R component, replace the values of 'outr' by the values of 'r' corresponding to the indices of white pixels in the mask as indicated in line 9 and 10. 'outr', 'outg' and 'outb' are shown in figures 2.20, 2.23, 2.24.
e. Repeat step d for G and B.
f. Merge outr, outb and outg. The cropped color image is shown in figure 2.22.











6- Divide the colored region of interest into 4 parts, split each part into its RGB components and get statistical features for each components.

```
# Numbers of rows
nRows = 2
n=3
# Number of columns
mCols = 2
m=3
# Tam of Y region
regionY=hight/nRows
# Tam of X region
regionX = width/mCols
# Total of regions
totalRegions = nRows* mCols
t=totalRegions+1
l=1
for i in range(1,3):
        for j in range(1,3):
                  part=out[(i-1)*regionX:i*regionX,
                       (j-1)*regionY:j*regionY]

                  cv2.imwrite(str(i)+str(j)+'.jpg',part)
                  l+=1
#Extracting features
x[0]=cv2.contourArea(cnt)
x[1]=cv2.arcLength(cnt,True)
piece11=cv2.imread('F:/python codes/11.jpg')
blue,green,red=cv2.split(piece11)
x[2]=np.average(blue)
x[3]=np.average(green)
x[4]=np.average(red)
x[5]=np.var(blue)
x[6]=np.var(green)
x[7]=np.var(red)
(a1,b1)=piece11.shape[:2]
```

```
rgb111 = piece11[a1/2+20,b1/2+10]
x[8]= rgb111[0]+rgb111[1]*256+rgb111[2]*256*256
rgb112 = piece11[a1/2+6,b1/2+2]
x[9]=rgb112[0]+rgb112[1]*256+rgb112[2]*256*256
rgb113 = piece11[a1/2+11,b1/2+1]
x[10]=rgb113[0]+rgb113[1]*256+rgb113[2]*256*256
rgb114 = piece11[a1/2+20,b1/2+2]
x[11]=rgb114[0]+rgb114[1]*256+rgb114[2]*256*256
```

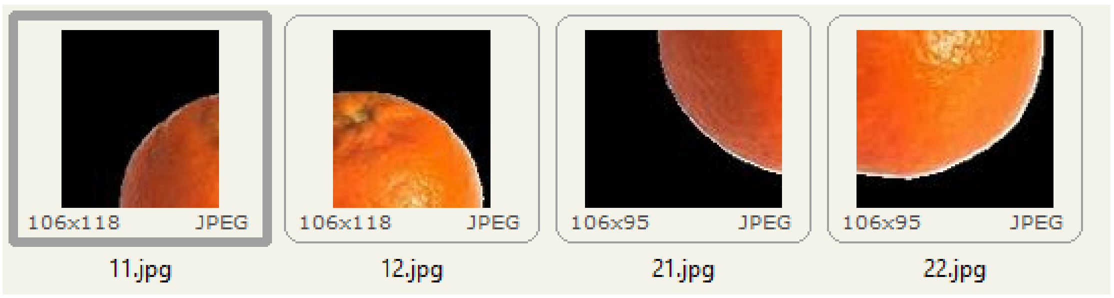
For each part the average and variance of R, G and B are calculated and stored in the features vector. Shape features (contour length and area) are stored also in the features vector.

For some pixels, the R, G and B values are integrated in a single integer N using the equation:

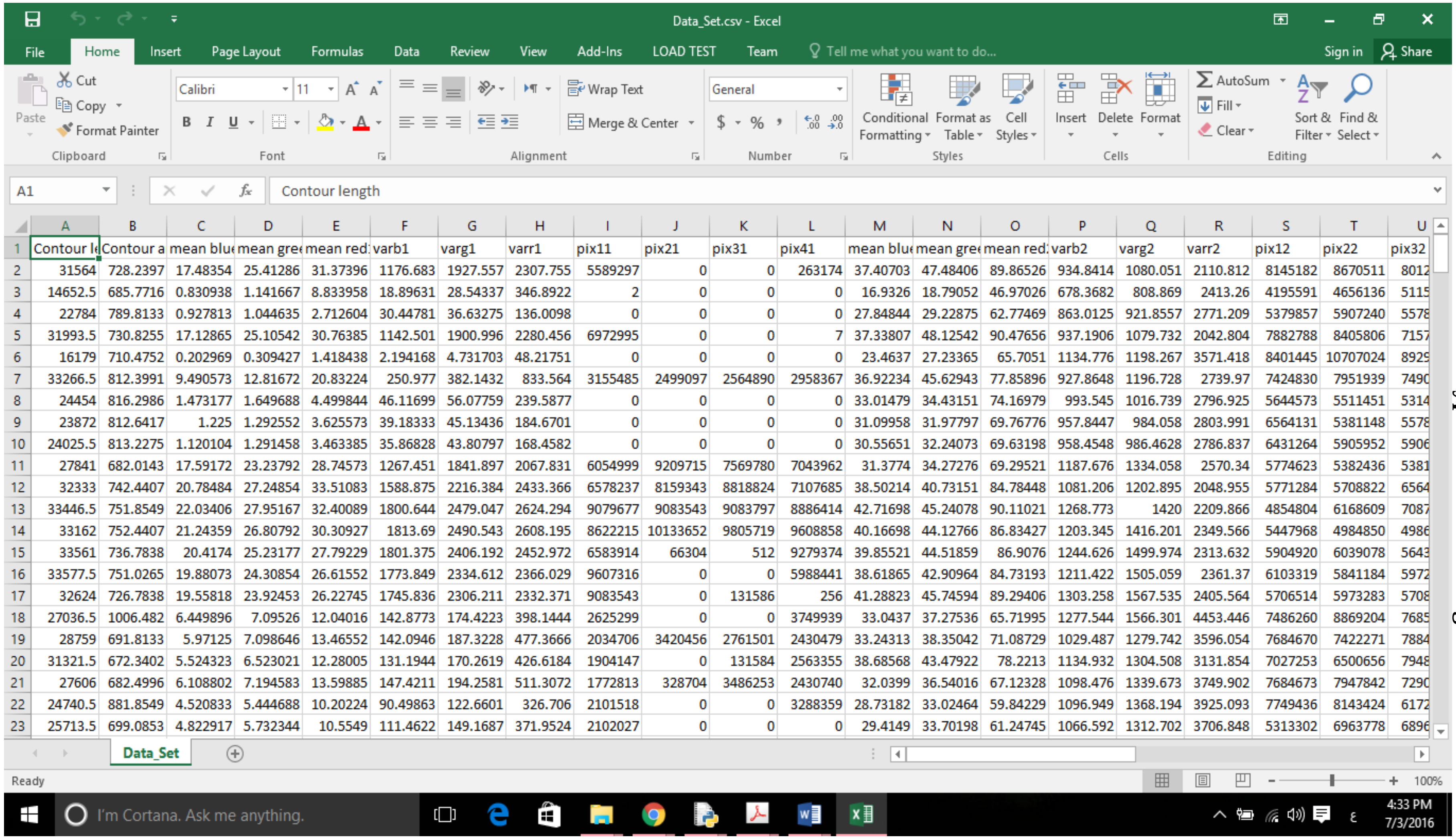
N= blue value + green value * 256 + red value * 256*256.

Feature vector example for orange is shown:

```
[ 2.06340000e+04  5.39043720e+02  1.11164854e+01  2.23211545e+01
   5.33029261e+01  3.85068379e+02  1.19221274e+03  6.63028535e+03
   9.71780800e+06  5.12000000e+02  5.00000000e+00  1.03068610e+07
   1.86186441e+01  5.98014071e+01  1.22046130e+02  8.09762786e+02
   3.70557729e+03  1.40853838e+04  1.55553590e+07  1.67380660e+07
   1.60804070e+07  1.54240270e+07  1.77387289e+01  3.04134062e+01
   7.31728898e+01  8.57218629e+02  1.69795869e+03  8.79567766e+03
   2.00000000e+00  2.56000000e+02  6.55410000e+04  2.00000000e+00
   3.20151936e+01  8.30810328e+01  1.65410030e+02  1.84229798e+03
   4.19901846e+03  1.35209446e+04  1.67465370e+07  1.65478800e+07
   1.64805570e+07  1.62207230e+07]
```



7- **Store features for different types of fruits and vegetables in a .csv file**



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Contour l	Contour a	mean blu	mean gre	mean red	varb1	varg1	varr1	pix11	pix21	pix31	pix41	mean blu	mean gre	mean red	varb2	varg2	varr2	pix12	pix22	pix32
2	31564	728.2397	17.48354	25.41286	31.37396	1176.683	1927.557	2307.755	5589297	0	0	263174	37.40703	47.48406	89.86526	934.8414	1080.051	2110.812	8145182	8670511	8012
3	14652.5	685.7716	0.830938	1.141667	8.833958	18.89631	28.54337	346.8922	2	0	0	0	16.9326	18.79052	46.97026	678.3682	808.869	2413.26	4195591	4656136	5115
4	22784	789.8133	0.927813	1.044635	2.712604	30.44781	36.63275	136.0098	0	0	0	0	27.84844	29.22875	62.77469	863.0125	921.8557	2771.209	5379857	5907240	5578
5	31993.5	730.8255	17.12865	25.10542	30.76385	1142.501	1900.996	2280.456	6972995	0	0	7	37.33807	48.12542	90.47656	937.1906	1079.732	2042.804	7882788	8405806	7157
6	16179	710.4752	0.202969	0.309427	1.418438	2.194168	4.731703	48.21751	0	0	0	0	23.4637	27.23365	65.7051	1134.776	1198.267	3571.418	8401445	10707024	8929
7	33266.5	812.3991	9.490573	12.81672	20.83224	250.977	382.1432	833.564	3155485	2499097	2564890	2958367	36.92234	45.62943	77.85896	927.8648	1196.728	2739.97	7424830	7951939	7490
8	24454	816.2986	1.473177	1.649688	4.499844	46.11699	56.07759	239.5877	0	0	0	0	33.01479	34.43151	74.16979	993.545	1016.739	2796.925	5644573	5511451	5314
9	23872	812.6417	1.225	1.292552	3.625573	39.18333	45.13436	184.6701	0	0	0	0	31.09958	31.97797	69.76776	957.8447	984.058	2803.991	6564131	5381148	5578
10	24025.5	813.2275	1.120104	1.291458	3.463385	35.86828	43.80797	168.4582	0	0	0	0	30.55651	32.24073	69.63198	958.4548	986.4628	2786.837	6431264	5905952	5906
11	27841	682.0143	17.59172	23.23792	28.74573	1267.451	1841.897	2067.831	6054999	9209715	7569780	7043962	31.3774	34.27276	69.29521	1187.676	1334.058	2570.34	5774623	5382436	5381
12	32333	742.4407	20.78484	27.24854	33.51083	1588.875	2216.384	2433.366	6578237	8159343	8818824	7107685	38.50214	40.73151	84.78448	1081.206	1202.895	2048.955	5771284	5708822	6564
13	33446.5	751.8549	22.03406	27.95167	32.40089	1800.644	2479.047	2624.294	9079677	9083543	9083797	8886414	42.71698	45.24078	90.11021	1268.773	1420	2209.866	4854804	6168609	7087
14	33162	752.4407	21.24359	26.80792	30.30927	1813.69	2490.543	2608.195	8622215	10133652	9805719	9608858	40.16698	44.12766	86.83427	1203.345	1416.201	2349.566	5447968	4984850	4986
15	33561	736.7838	20.4174	25.23177	27.79229	1801.375	2406.192	2452.972	6583914	66304	512	9279374	39.85521	44.51859	86.9076	1244.626	1499.974	2313.632	5904920	6039078	5643
16	33577.5	751.0265	19.88073	24.30854	26.61552	1773.849	2334.612	2366.029	9607316	0	0	5988441	38.61865	42.90964	84.73193	1211.422	1505.059	2361.37	6103319	5841184	5972
17	32624	726.7838	19.55818	23.92453	26.22745	1745.836	2306.211	2332.371	9083543	0	131586	256	41.28823	45.74594	89.29406	1303.258	1567.535	2405.564	5706514	5973283	5708
18	27036.5	1006.482	6.449896	7.09526	12.04016	142.8773	174.4223	398.1444	2625299	0	0	3749939	33.0437	37.27536	65.71995	1277.544	1566.301	4453.446	7486260	8869204	7685
19	28759	691.8133	5.97125	7.098646	13.46552	142.0946	187.3228	477.3666	2034706	3420456	2761501	2430479	33.24313	38.35042	71.08729	1029.487	1279.742	3596.054	7684670	7422271	7884
20	31321.5	672.3402	5.524323	6.523021	12.28005	131.1944	170.2619	426.6184	1904147	0	131584	2563355	38.68568	43.47922	78.2213	1134.932	1304.508	3131.854	7027253	6500656	7948
21	27606	682.4996	6.108802	7.194583	13.59885	147.4211	194.2581	511.3072	1772813	328704	3486253	2430740	32.0399	36.54016	67.12328	1098.476	1339.673	3749.902	7684673	7947842	7290
22	24740.5	881.8549	4.520833	5.444688	10.20224	90.49863	122.6601	326.706	2101518	0	0	3288359	28.73182	33.02464	59.84229	1096.949	1368.194	3925.093	7749436	8143424	6172
23	25713.5	699.0853	4.822917	5.732344	10.5549	111.4622	149.1687	371.9524	2102027	0	0	0	29.4149	33.70198	61.24745	1066.592	1312.702	3706.848	5313302	6963778	6896

3.4.2 Classification

The classifier in our project is a multiclass neural network classifier designed using Azure machine learning studio, trained by 70% of the data set obtained from step 7 and tested by the rest of the data set. In the next section the classifier configuration is discussed.

1- Multiclass neural network

Multiclass Neural Network module was used to create a neural network model that can be used to predict a target that has multiple values. For example, neural networks are frequently used in complex computer vision tasks, such as digit or letter recognition, document classification, and pattern recognition.

Classification using neural networks is a supervised learning method. A neural network is a set of interconnected layers, in which the inputs lead to outputs by a series of weighted edges and nodes. The weights on the edges are learned when training the neural network on the input data. The direction of the graph proceeds from the inputs through the hidden layer, with all nodes of the graph connected by the weighted edges to nodes in the next layer. To compute the output of the network for any given input, a value is calculated for each node in the hidden layers and in the output layer. For each node, the value is set by calculating the weighted sum of the values of the nodes in the previous layer and applying an activation function to that weighted sum.

2- Tune Model Hyperparameters

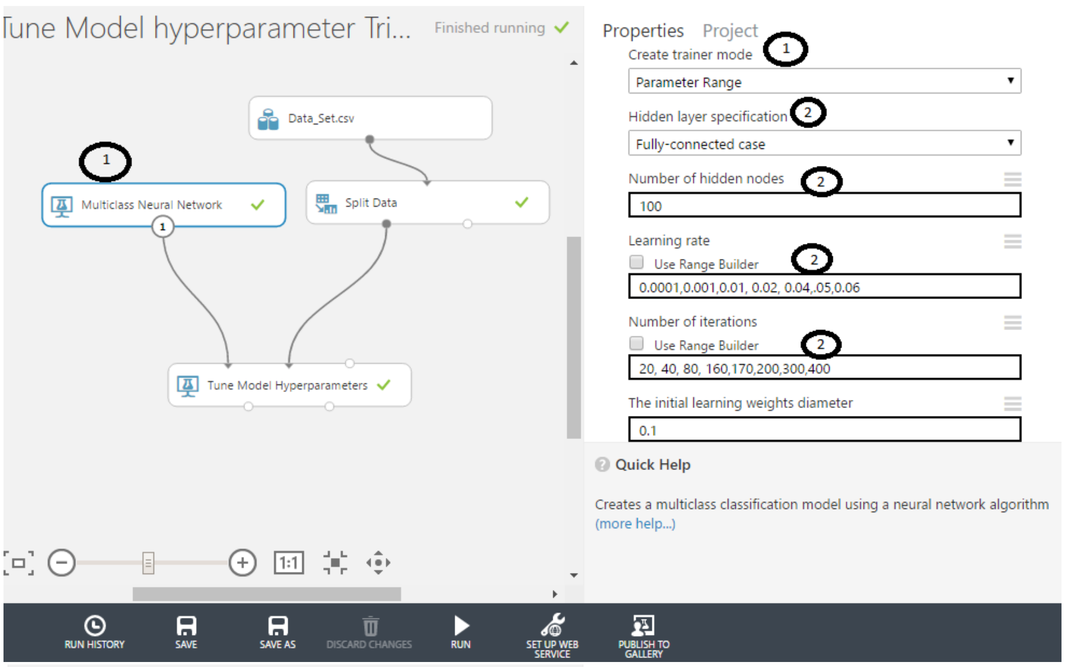
Tune Model Hyperparameters module performed a parameter sweep on a model to determine the optimum parameter settings such as the learning rate and number of iterations that produces the minimum error or max accuracy.

3.4.3 Classifier Configuration

3.4.3.1 Selecting Optimum Parameter for the neural network

Optimum parameter settings for the neural network such as the learning rate and number of iterations that produces the maximum accuracy is determined using the shown configuration.

1- Set the multiclass neural network trainer mode to parameter range.
2- Select the number of hidden layer nodes and determine the range of parameters that need to be tested.
3- Add the dataset you want to use for training and connect it to the middle input of Tune Model Hyperparameters which is 70% of the data set of features.
4- In the Properties pane of Tune Model Hyperparameters set options that define the number of parameters used and the number of iterations and choose Entire grid.
5- For **Label column**, launch the column selector to choose a single label column which is named **Type** in our case.
6- Run the experiment.
7- The Tune Model Hyperparameters module outputs a set of evaluation results, indicating the parameters that produced the best models, and the accuracy of all models on the available metrics.
8- Visualize the output of the model and select the parameters that guarantee maximum accuracy.



Properties Project

Create trainer mode: Parameter Range

Hidden layer specification: Fully-connected case

Number of hidden nodes: 100

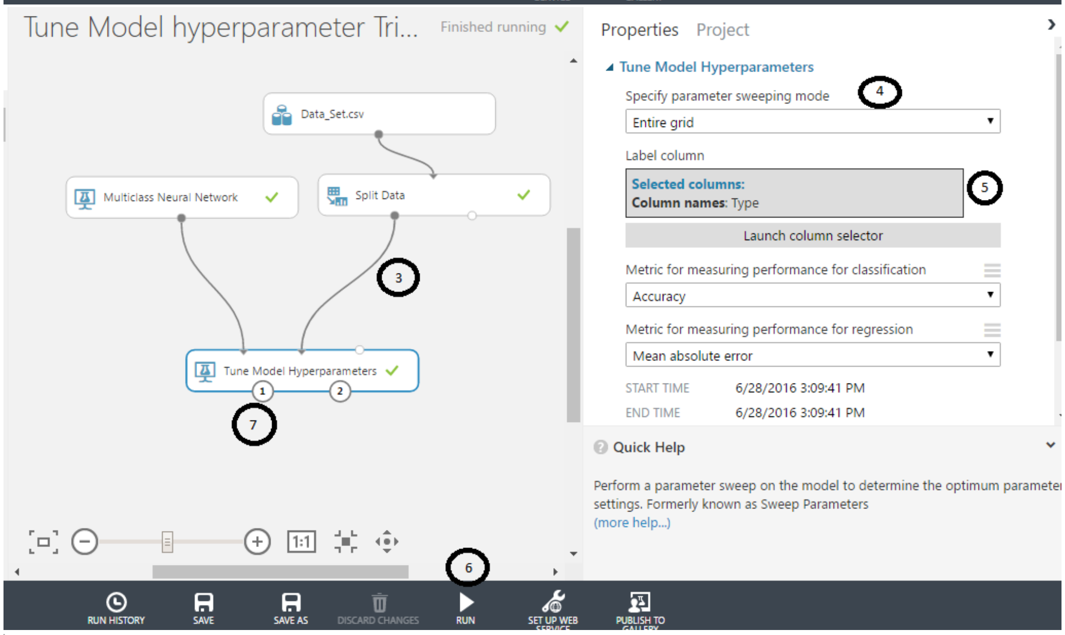
Learning rate
Use Range Builder
0.0001,0.001,0.01, 0.02, 0.04,.05,0.06

Number of iterations
Use Range Builder
20, 40, 80, 160,170,200,300,400

The initial learning weights diameter: 0.1

Quick Help

Creates a multiclass classification model using a neural network algorithm (more help...)



Properties Project

Tune Model Hyperparameters

Specify parameter sweeping mode: Entire grid

Label column
Selected columns:
Column names: Type
Launch column selector

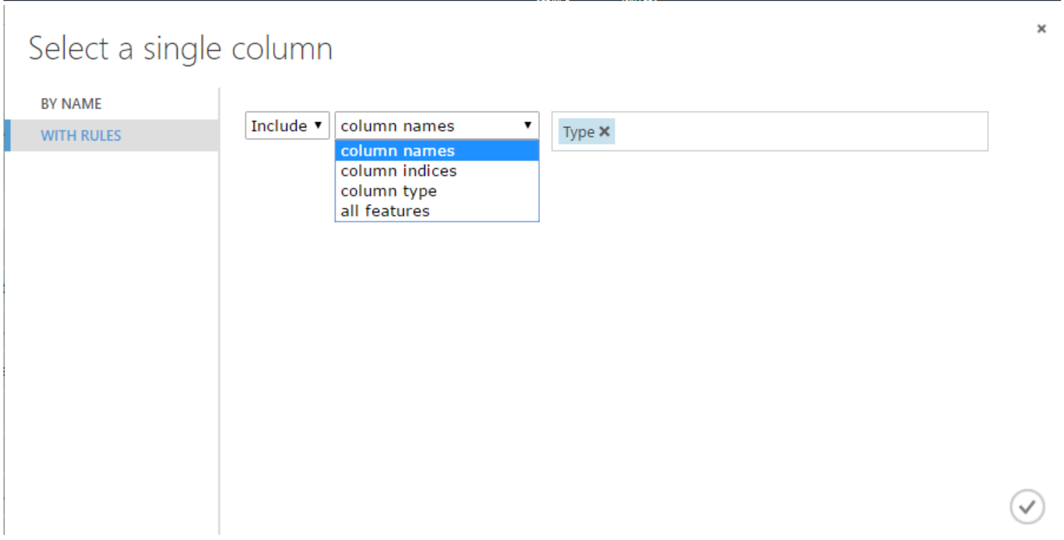
Metric for measuring performance for classification: Accuracy

Metric for measuring performance for regression: Mean absolute error

START TIME 6/28/2016 3:09:41 PM
END TIME 6/28/2016 3:09:41 PM

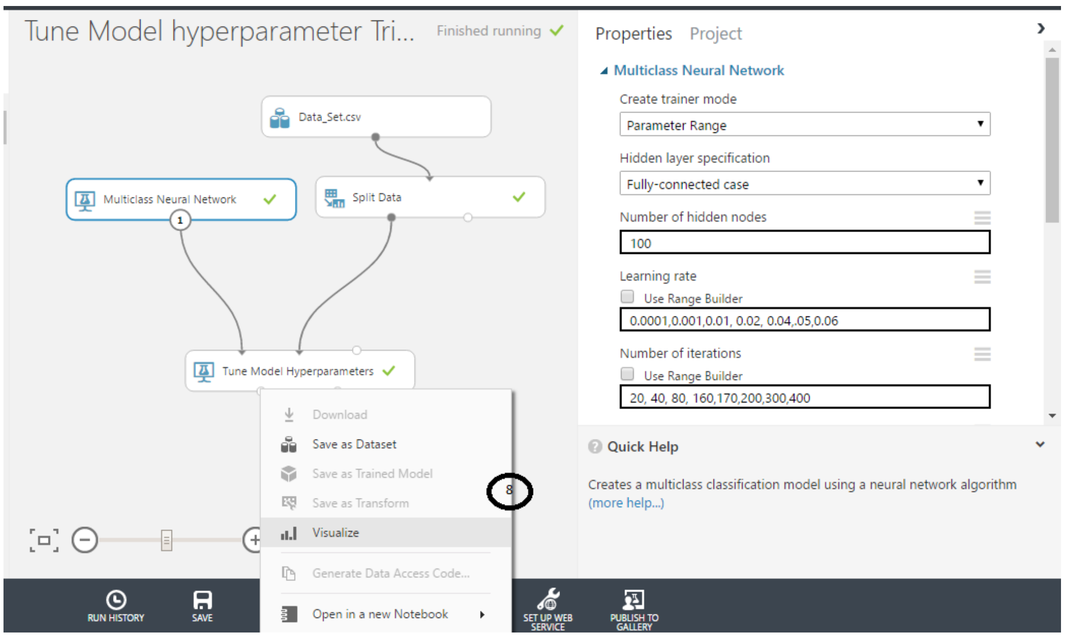
Quick Help

Perform a parameter sweep on the model to determine the optimum paramete settings. Formerly known as Sweep Parameters
(more help...)



Select a single column

Include column names Type



Multiclass Neural Network

Create trainer mode: Parameter Range

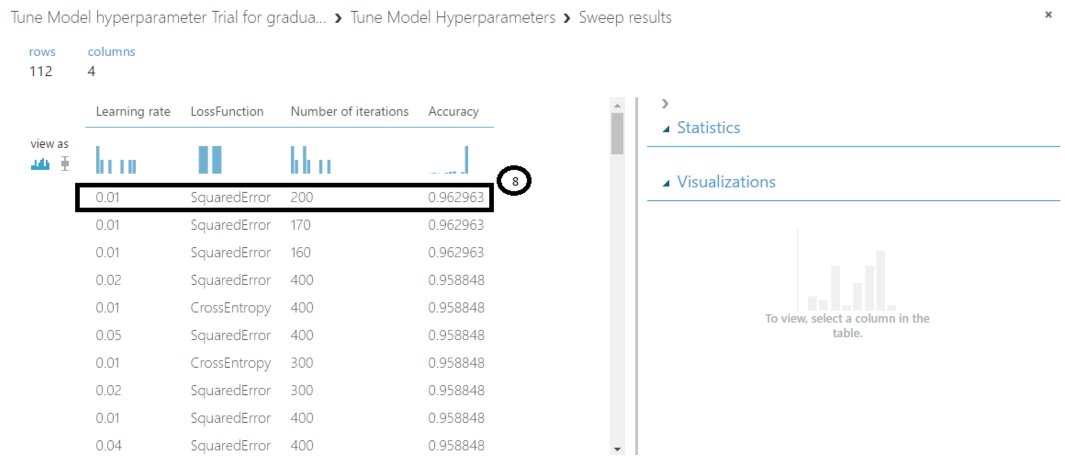
Hidden layer specification: Fully-connected case

Number of hidden nodes: 100

Learning rate: Use Range Builder — 0.0001,0.001,0.01, 0.02, 0.04,.05,0.06

Number of iterations: Use Range Builder — 20, 40, 80, 160,170,200,300,400

Creates a multiclass classification model using a neural network algorithm (more help...)

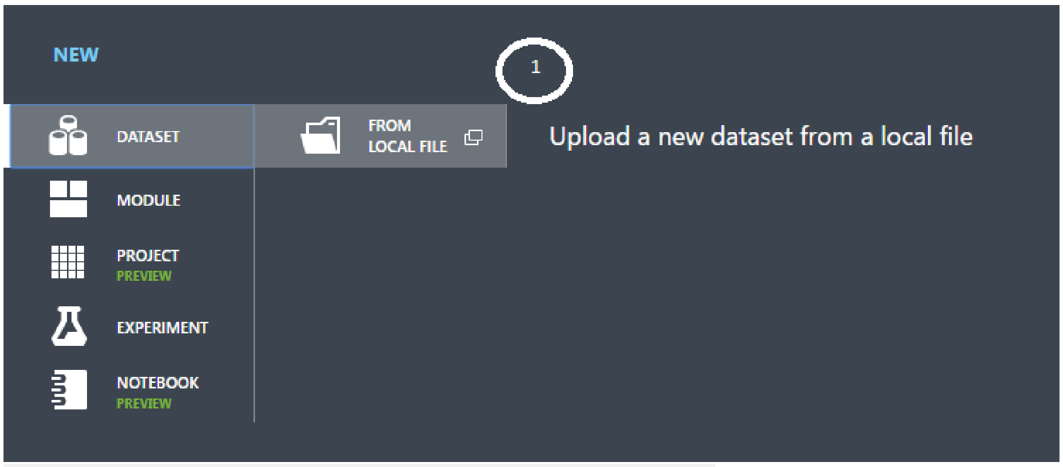

Tune Model hyperparameter Trial for gradua... › Tune Model Hyperparameters › Sweep results
view as
8
Statistics
Visualizations
To view, select a column in the table.

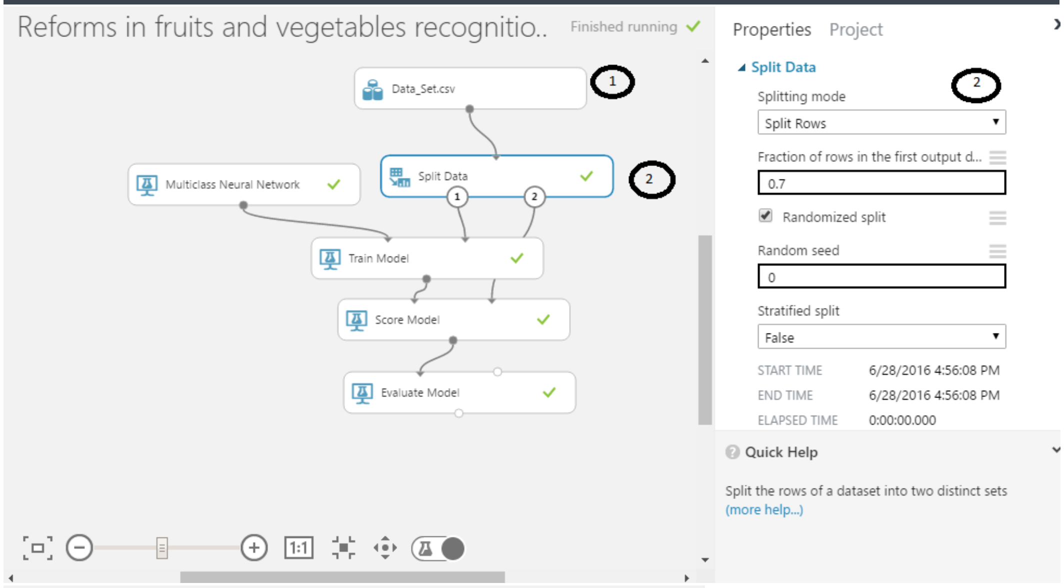
rows 112, columns 4

Learning rate	LossFunction	Number of iterations	Accuracy
0.01	SquaredError	200	0.962963
0.01	SquaredError	170	0.962963
0.01	SquaredError	160	0.962963
0.02	SquaredError	400	0.958848
0.01	CrossEntropy	400	0.958848
0.05	SquaredError	400	0.958848
0.01	CrossEntropy	300	0.958848
0.02	SquaredError	300	0.958848
0.01	SquaredError	400	0.958848
0.04	SquaredError	400	0.958848

3.4.3.2 Building Training Experiment

1- Upload the data set and insert it to the working space.
2- Use **Data split** module to split data to training set and testing set.
3- Insert **Multiclass neural network** module and set its parameter to the values obtained from the previous section.
4- Insert **Train Model** and connect it to the classification module and the training data set. For Label column, launch the column selector to choose a single label column which is named Type in our case
5- Insert **Score Model** and connect it to the output of the training model and the testing data set.
6- Insert **Evaluate Model** to determine evaluate the efficiency of training.
7- Run the experiment
8- Visualize the output of the evaluate model and check the confusion matrix.





Properties Project

Split Data

Splitting mode: Split Rows

Fraction of rows in the first output d...: 0.7

☑ Randomized split

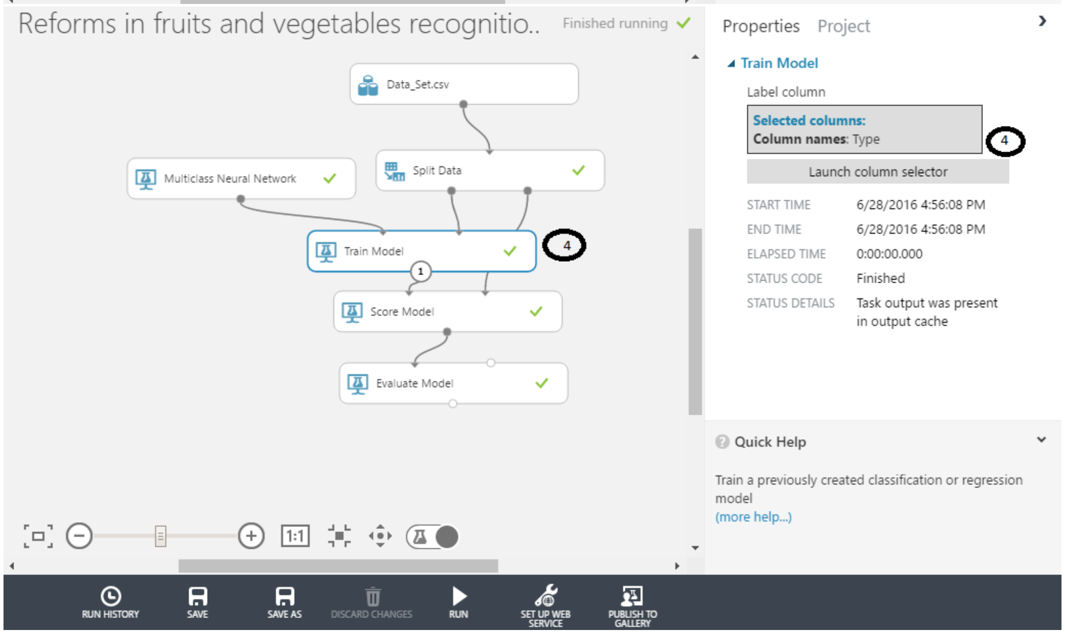
Random seed: 0

Stratified split: False

START TIME	6/28/2016 4:56:08 PM
END TIME	6/28/2016 4:56:08 PM
ELAPSED TIME	0:00:00.000

Quick Help

Split the rows of a dataset into two distinct sets
(more help...)



Properties Project

Train Model

Label column

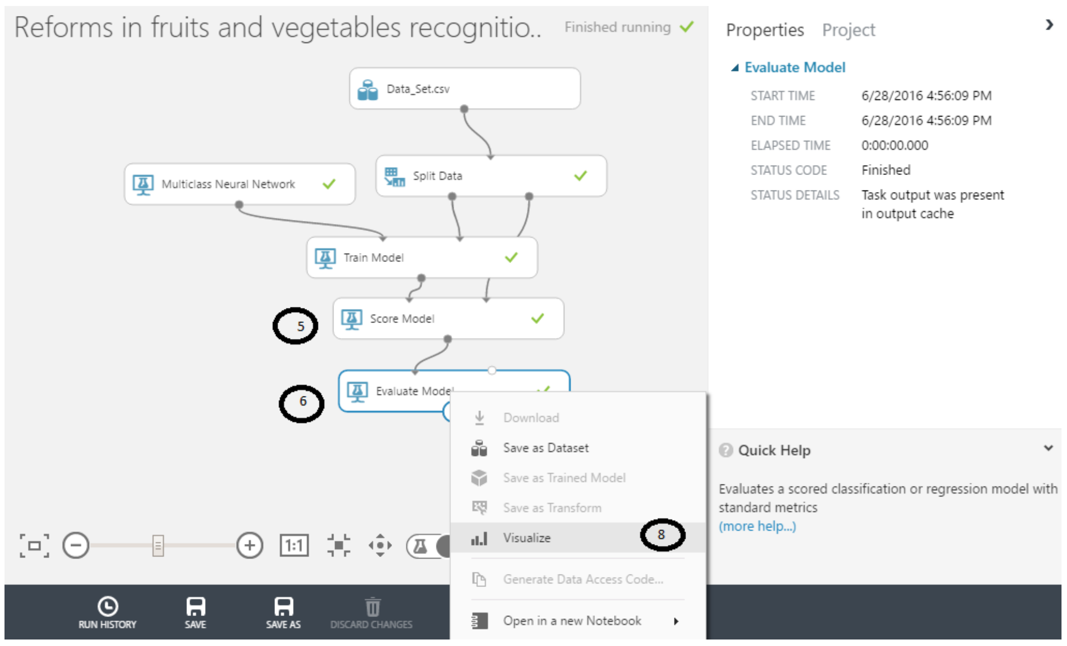
Selected columns:
Column names: Type

Launch column selector

START TIME	6/28/2016 4:56:08 PM
END TIME	6/28/2016 4:56:08 PM
ELAPSED TIME	0:00:00.000
STATUS CODE	Finished
STATUS DETAILS	Task output was present in output cache

Quick Help

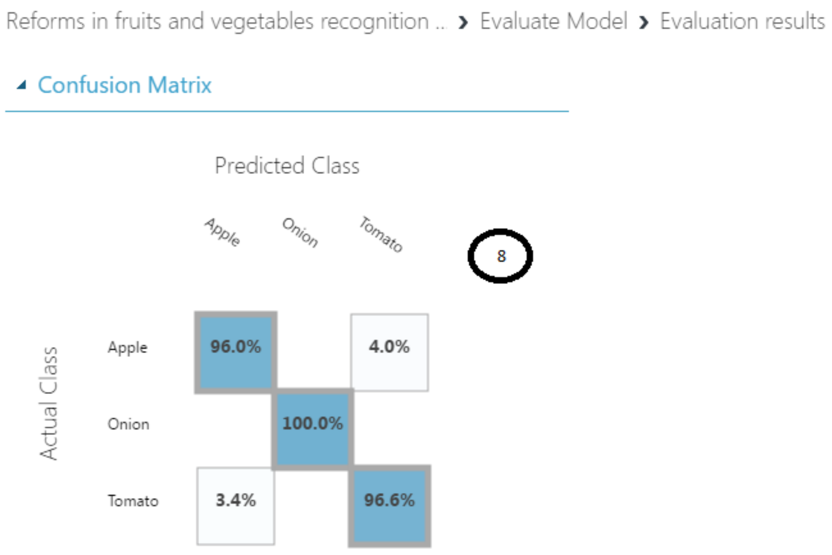
Train a previously created classification or regression model
(more help...)



Evaluate Model

START TIME	6/28/2016 4:56:09 PM
END TIME	6/28/2016 4:56:09 PM
ELAPSED TIME	0:00:00.000
STATUS CODE	Finished
STATUS DETAILS	Task output was present in output cache

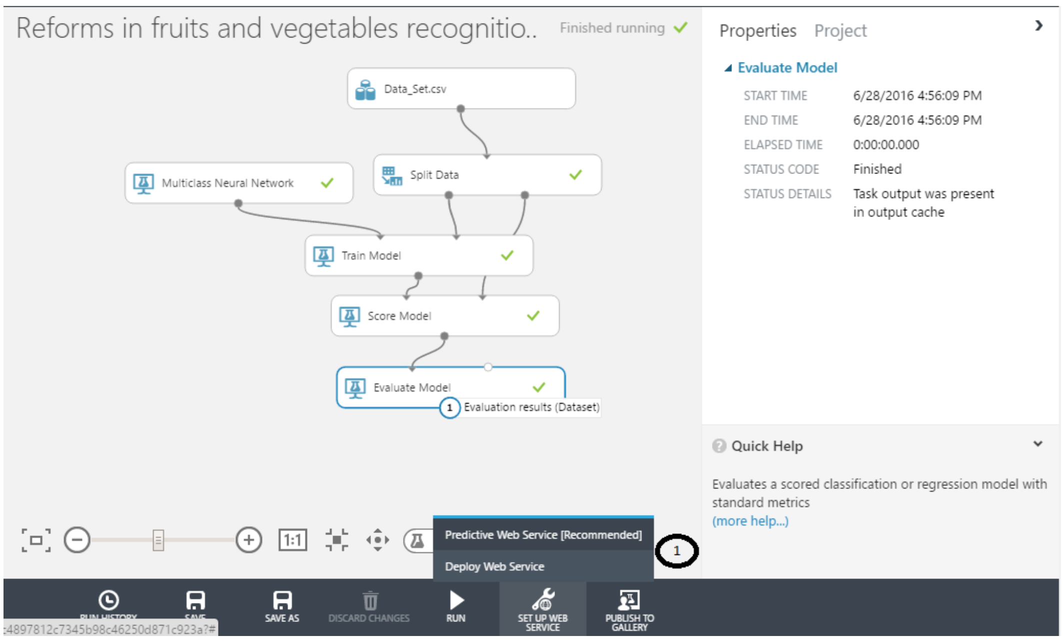
Evaluates a scored classification or regression model with standard metrics
(more help...)


Reforms in fruits and vegetables recognition .. › Evaluate Model › Evaluation results
Confusion Matrix
8

Actual Class \ Predicted Class	Apple	Onion	Tomato
Apple	96.0%		4.0%
Onion		100.0%	
Tomato	3.4%		96.6%

3.4.3.3 Deploying Predictive Web Service

1- Select deploy web service.
2- Choose project columns to modify the input and the output of the web service.
3- Run the experiment.
4- Choose deploy web service again.
5- Save the web service API.
6- Use Request/Response to get the python code representing the classifier.



Properties Project

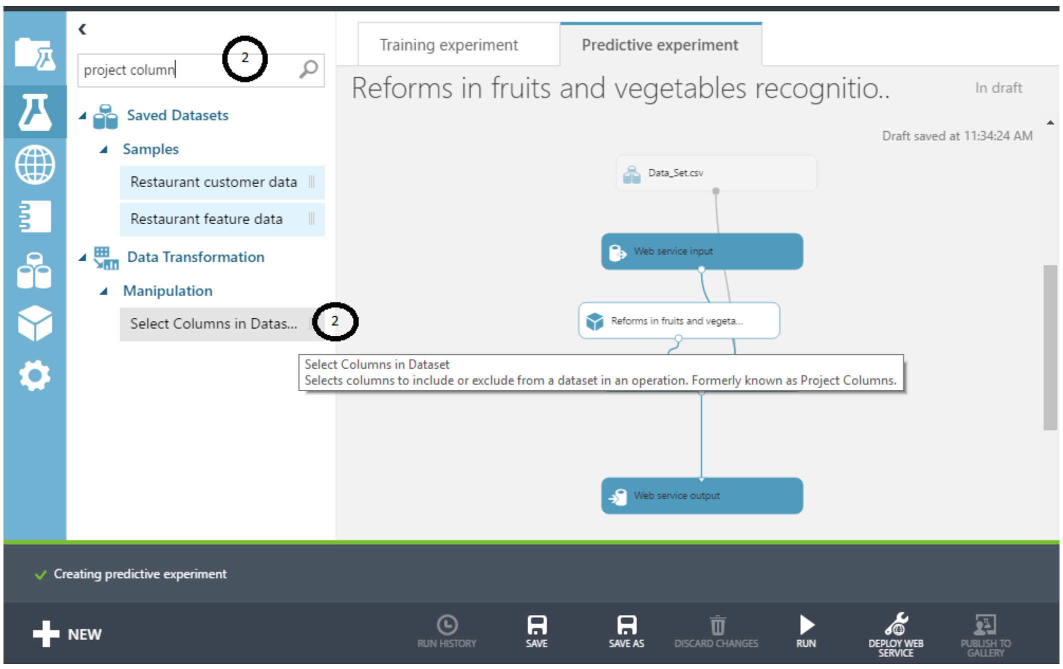
Evaluate Model

START TIME	6/28/2016 4:56:09 PM
END TIME	6/28/2016 4:56:09 PM
ELAPSED TIME	0:00:00.000
STATUS CODE	Finished
STATUS DETAILS	Task output was present in output cache

Quick Help

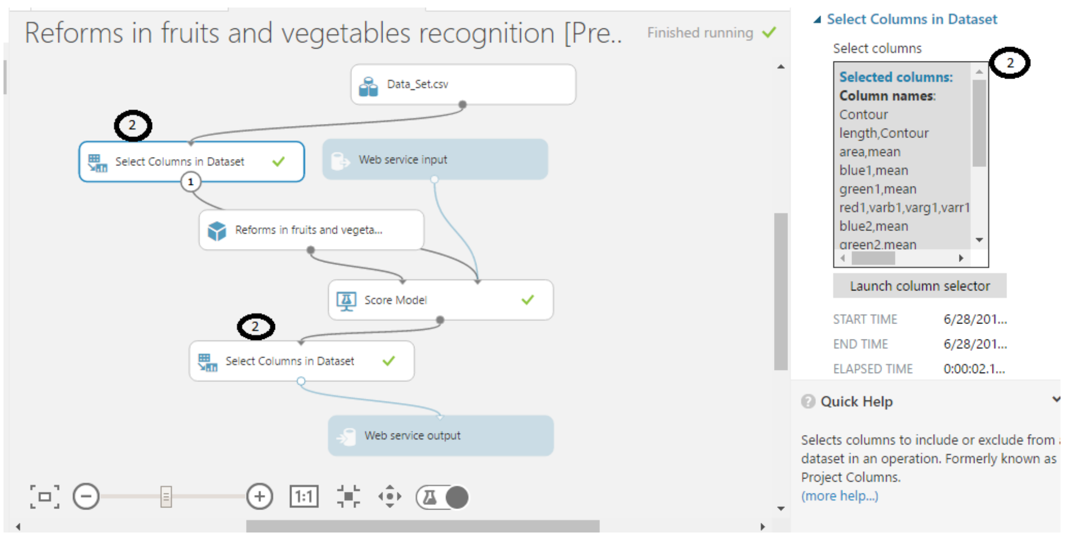
Evaluates a scored classification or regression model with standard metrics
(more help...)

:4897812c7345b98c46250d871c923a?#



Select Columns in Dataset
Selects columns to include or exclude from a dataset in an operation. Formerly known as Project Columns.

Creating predictive experiment


Reforms in fruits and vegetables recognition [Pre..
Finished running
Data_Set.csv
2
Select Columns in Dataset
Web service input
1
Reforms in fruits and vegeta...
Score Model
Select Columns in Dataset
Web service output
Launch column selector

Select Columns in Dataset

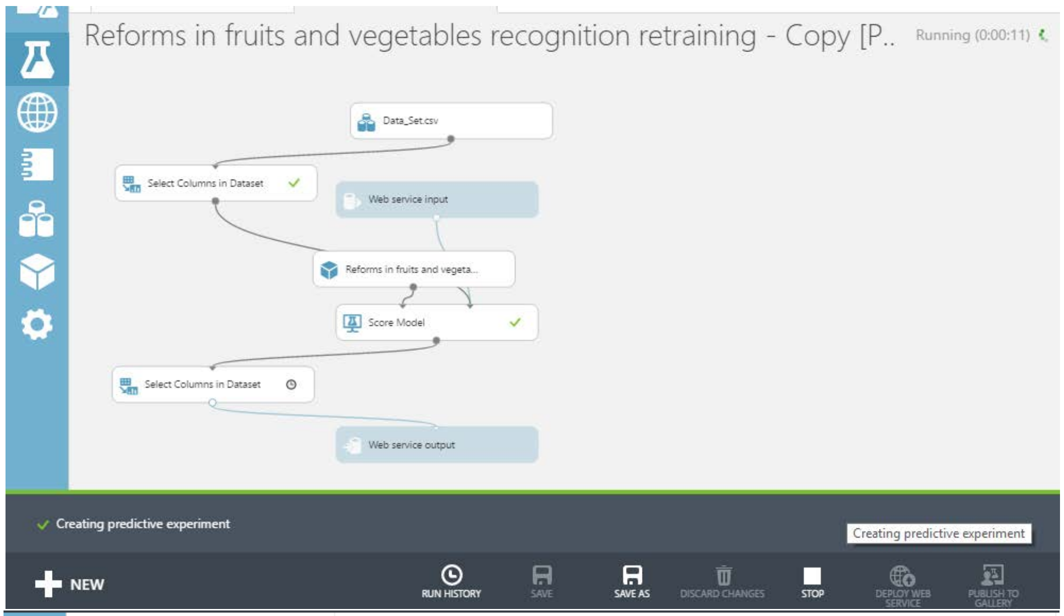
Select columns

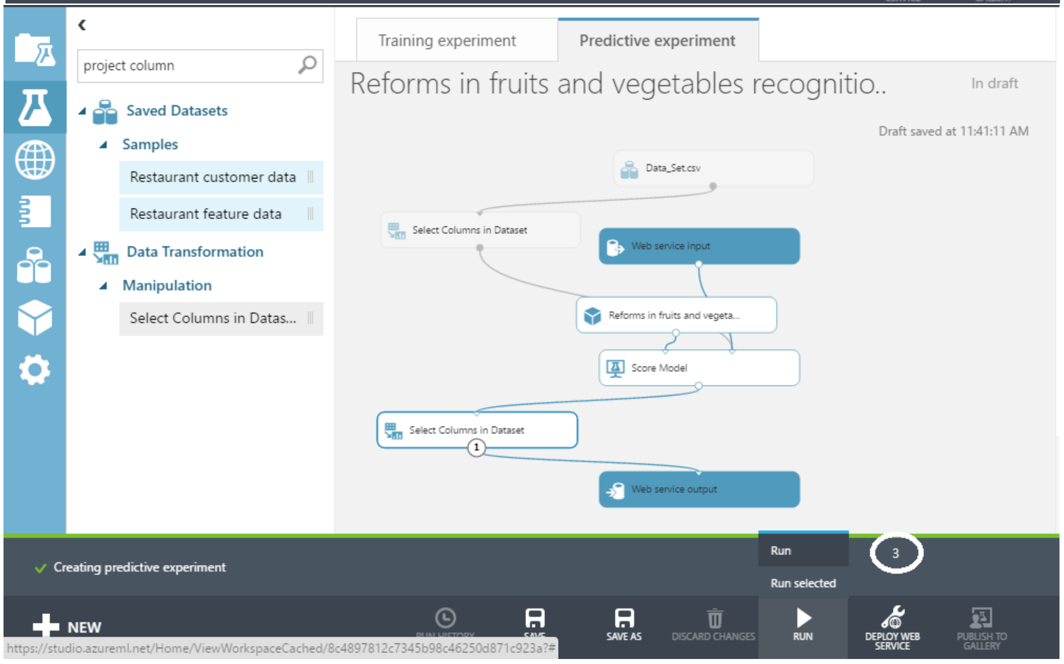
Selected columns:
Column names: Contour length,Contour area,mean blue1,mean green1,mean red1,varb1,varg1,varr1 blue2,mean green2,mean

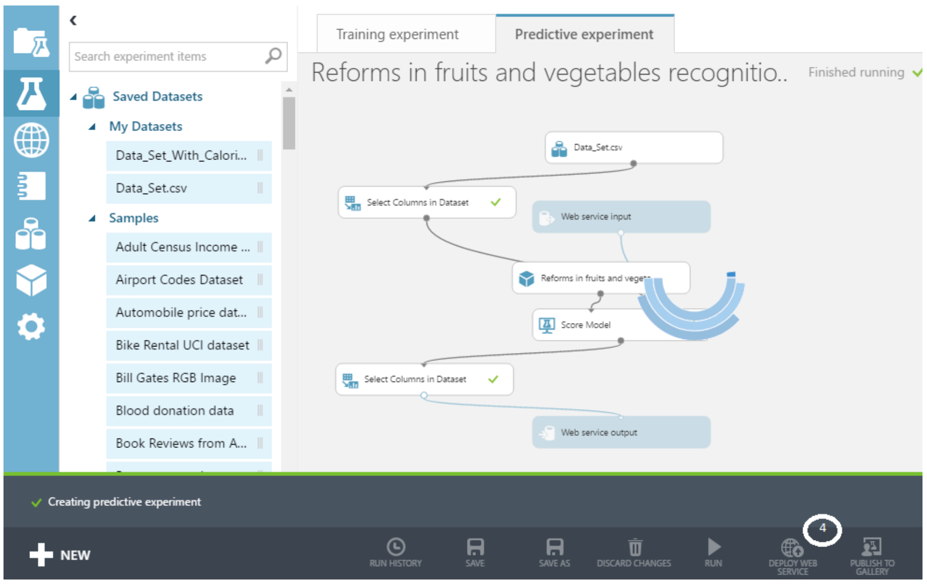
START TIME	6/28/201...
END TIME	6/28/201...
ELAPSED TIME	0:00:02.1...

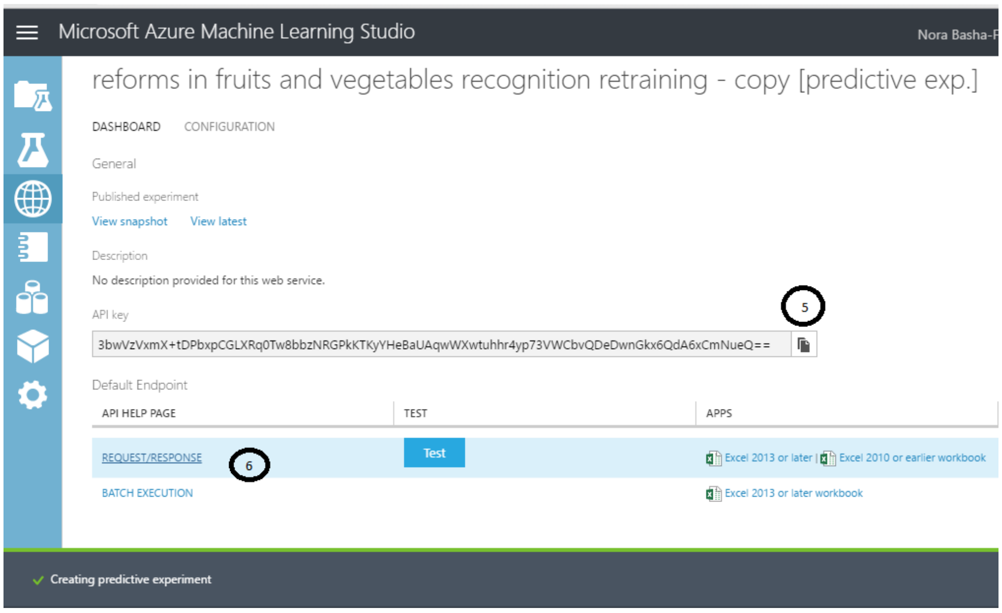
Quick Help

Selects columns to include or exclude from a dataset in an operation. Formerly known as Project Columns.
(more help...)


Reforms in fruits and vegetables recognition retraining - Copy [P..
Running (0:00:11)
Data_Set.csv
Select Columns in Dataset
Web service input
Reforms in fruits and vegeta...
Score Model
Select Columns in Dataset
Web service output
Creating predictive experiment
Creating predictive experiment
NEW
RUN HISTORY
SAVE
SAVE AS
DISCARD CHANGES
STOP
DEPLOY WEB SERVICE
PUBLISH TO GALLERY


project column
Saved Datasets
Samples
Restaurant customer data
Restaurant feature data
Data Transformation
Manipulation
Select Columns in Datas...
Training experiment
Predictive experiment
Reforms in fruits and vegetables recognitio..
In draft
Draft saved at 11:41:11 AM
Data_Set.csv
Select Columns in Dataset
Web service input
Reforms in fruits and vegeta...
Score Model
Select Columns in Dataset
Web service output
Run
Run selected
3
Creating predictive experiment
NEW
RUN HISTORY
SAVE
SAVE AS
DISCARD CHANGES
RUN
DEPLOY WEB SERVICE
PUBLISH TO GALLERY
https://studio.azureml.net/Home/ViewWorkspaceCached/8c4897812c7345b98c46250d871c923a?#


Search experiment items
Saved Datasets
My Datasets
Data_Set_With_Calori...
Data_Set.csv
Samples
Adult Census Income ...
Airport Codes Dataset
Automobile price dat...
Bike Rental UCI dataset
Bill Gates RGB Image
Blood donation data
Book Reviews from A...
Training experiment
Predictive experiment
Reforms in fruits and vegetables recognitio..
Finished running
Data_Set.csv
Select Columns in Dataset
Web service input
Reforms in fruits and vegeta...
Score Model
Select Columns in Dataset
Web service output
Creating predictive experiment
4
NEW
RUN HISTORY
SAVE
SAVE AS
DISCARD CHANGES
RUN
DEPLOY WEB SERVICE
PUBLISH TO GALLERY


Microsoft Azure Machine Learning Studio
Nora Basha-F
DASHBOARD
CONFIGURATION
5
6
Test
Excel 2013 or later
Excel 2010 or earlier workbook
Excel 2013 or later workbook

reforms in fruits and vegetables recognition retraining - copy [predictive exp.]

General

Published experiment

View snapshot View latest

Description

No description provided for this web service.

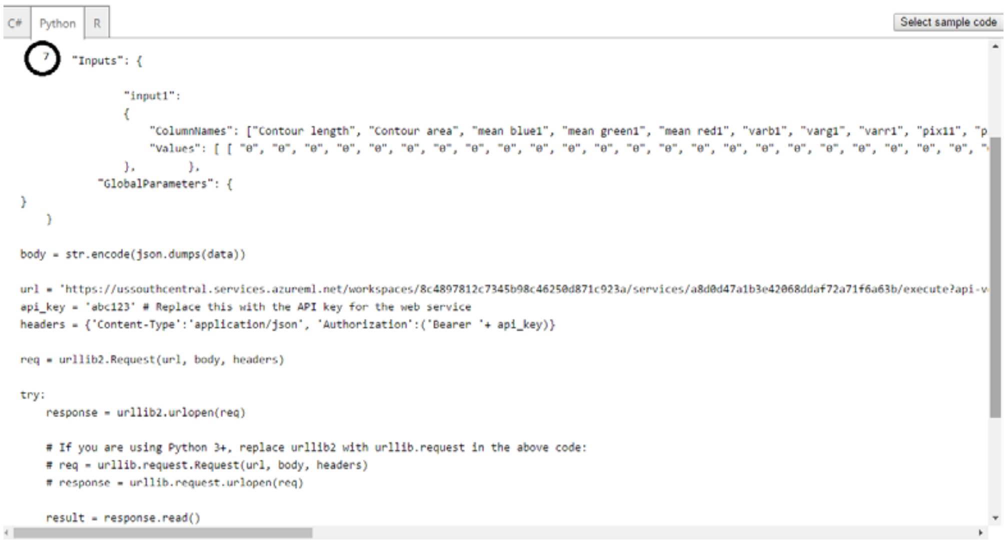
API key

3bwVzVxmX+tDPbxpCGLXRq0Tw8bbzNRGPkKTKyYHeBaUAqwWXwtuhhr4yp73VWCbvQDeDwnGkx6QdA6xCmNueQ==

Default Endpoint

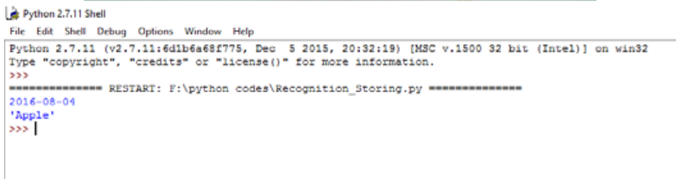
API HELP PAGE	TEST	APPS
REQUEST/RESPONSE	Test	Excel 2013 or later | Excel 2010 or earlier workbook
BATCH EXECUTION		Excel 2013 or later workbook

✓ Creating predictive experiment



```
"Inputs": {

        "input1":
        {
            "ColumnNames": ["Contour length", "Contour area", "mean blue1", "mean green1", "mean red1", "varb1", "varg1", "varr1", "pix11", "p
            "Values": [ [ "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "
        },        },
    "GlobalParameters": {
}
    }

body = str.encode(json.dumps(data))

url = 'https://ussouthcentral.services.azureml.net/workspaces/8c4897812c7345b98c46250d871c923a/services/a8d0d47a1b3e42068ddaf72a71f6a63b/execute?api-v
api_key = 'abc123' # Replace this with the API key for the web service
headers = {'Content-Type':'application/json', 'Authorization':('Bearer '+ api_key)}

req = urllib2.Request(url, body, headers)

try:
    response = urllib2.urlopen(req)

    # If you are using Python 3+, replace urllib2 with urllib.request in the above code:
    # req = urllib.request.Request(url, body, headers)
    # response = urllib.request.urlopen(req)

    result = response.read()
```

After little adjustment in the code the recognition is done resulting the following output when entering an apple image:



```
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec  5 2015, 20:32:19) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: F:\python codes\Recognition_Storing.py ==============
2016-08-04
'Apple'
>>> |
```

Chapter 4
Cloud System

4 Overview

4.1 Why Cloud System

Let us review the scenario of computing prior to the announcement and availability of cloud computing: The users who are in need of computing are expected to invest money on computing resources such as hardware, software, networking, and storage; this investment naturally costs a bulk currency to the users as they have to buy these computing resources, keep these in their premises, and maintain and make it operational—all these tasks would add cost. And, this is a particularly true and huge expenditure to the enterprises that require enormous computing power and resources, compared with classical academics and individuals.

On the other hand, it is easy and handy to get the required computing power and resources from some provider (or supplier) as and when it is needed and pay only for that usage. This would cost only a reasonable investment or spending, compared to the huge investment when buying the entire computing infrastructure. This phenomenon can be viewed as capital expenditure versus operational expenditure. As one can easily assess the huge lump sum required for capital expenditure (whole investment and maintenance for computing infrastructure) and compare it with the moderate or smaller lump sum required for the hiring or getting the computing infrastructure only to the tune of required time, and rest of the time free from that. Therefore, cloud computing is a mechanism of bringing–hiring or getting the services of the computing power or infrastructure to an organizational or individual level to the extent required and paying only for the consumed services.

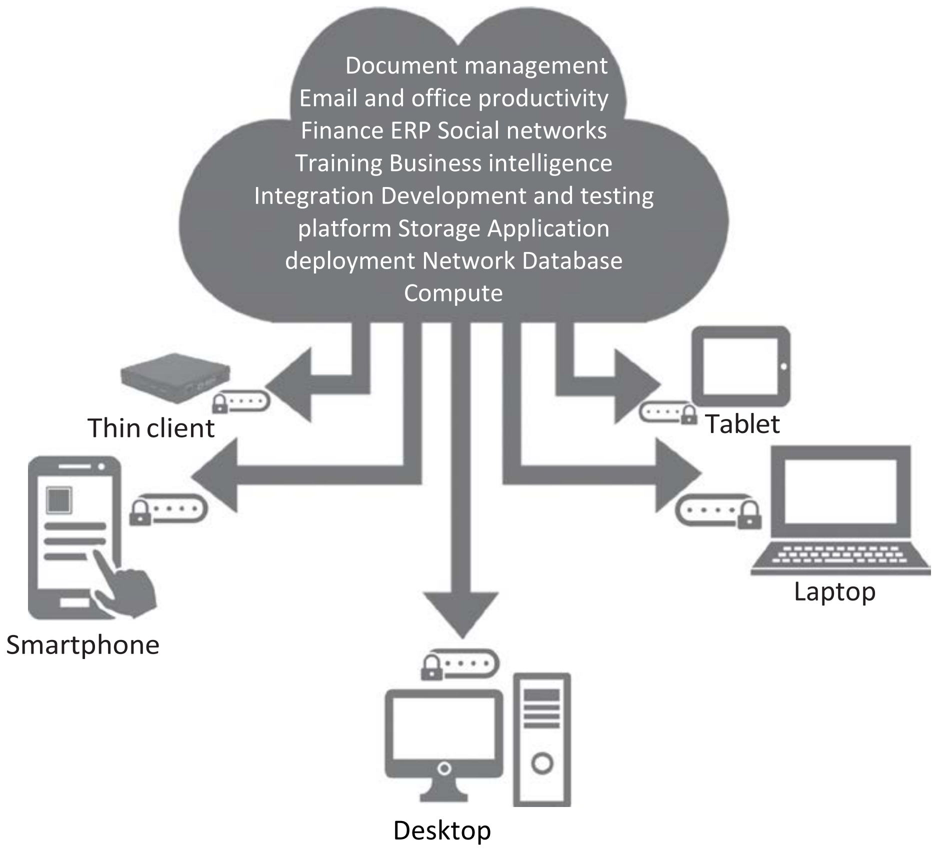
One can compare this situation with the usage of electricity (its services) from its producer-cum-distributor (in India, it is the state-/government-owned electricity boards that give electricity supply to all residences and organizations) to houses or organizations; here, we do not generate electricity (comparable with electricity production–related tasks); rather, we use it only to tune up our requirements in our premises, such as for our lighting and usage of other electrical appliances, and pay as per the electricity meter reading value.

Therefore, cloud computing is needed in getting the services of computing resources. Thus, one can say as a one-line answer to the need for cloud computing that it eliminates a large computing investment without compromising the use of computing at the user level at an operational cost. Cloud computing is very economical and saves a lot of money. A blind benefit of this computing is that even if we lose our laptop or due to some crisis our personal computer—and the desktop system—gets damaged, still our data and files will stay safe and secured as these are not in our local machine (but remotely located at the provider's place—machine).

In addition, one can think to add security while accessing these remote computing resources as depicted in Figure 2.1.

Figure 2.1 shows several cloud computing applications. The cloud represents the Internet-based computing resources, and the accessibility is through some secure support of connectivity. It is a computing solution growing in popularity, especially among individuals and small- and medium-sized companies (SMEs). In the cloud computing model, an organization's core computer power resides offsite and is essentially subscribed to rather than owned.

Thus, cloud computing comes into focus and much needed only when we think about what computing resources and information technology (IT) solutions are required. This need caters to a way to increase capacity or add capabilities on the fly without investing in new infrastructure, training new personnel, or licensing new software. Cloud computing encompasses the subscription-based or pay-per-use service model of offering computing to end users or customers over the Internet and thereby extending the IT's existing capabilities.


Document management
Email and office productivity
Finance ERP Social networks
Training Business intelligence
Integration Development and testing
platform Storage Application
deployment Network Database
Compute
Thin client
Tablet
Smartphone
Laptop
Desktop

4.1.1 5-4-3 Principles of Cloud computing

The 5-4-3 principles put forth by NIST describe (a) the five essential characteristic features that promote cloud computing, (b) the four deployment models that are used to narrate the cloud computing opportunities for customers while looking at architectural models, and (c) the three important and basic service offering models of cloud computing.

- **Five Essential Characteristics**

Cloud computing has five essential characteristics, which are shown in Figure 2.2. Readers can note the word essential, which means that if any of these characteristics is missing, then it is not cloud computing:

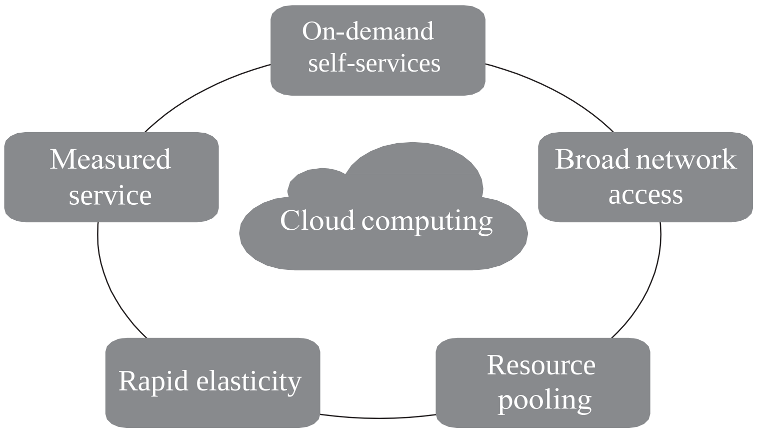
1. On-demand self-service: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.

2. Broad network access: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and personal digital assistants [PDAs]).

3. Elastic resource pooling: The provider's computing resources are pooled to serve multiple consumers using a multitenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify the location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, and network bandwidth.

4. Rapid elasticity: Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

5. Measured service: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service.



- **Four Cloud Deployment Models**

Deployment models describe the ways with which the cloud services can be deployed or made available to its customers, depending on the organizational structure and the provisioning location. One can understand it in this manner too: cloud (Internet)-based computing resources—that is, the locations where data and services are acquired and provisioned to its customers—can take various forms. Four deployment models are usually distinguished, namely, public, private, community, and hybrid cloud service usage:

1. Private cloud: The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

2. Public cloud: The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

3. Community cloud: The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise.

4. Hybrid cloud: The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

- **Three Service Offering Models**

The three kinds of services with which the cloud-based computing resources are available to end customers are as follows: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). It is also known as the service–platform–infrastructure (SPI) model of the cloud and is shown in Figure 2.3. SaaS is a software distribution model in which applications (software, which is one of the most important computing resources) are hosted by a vendor or service provider and made available to customers over a network, typically the Internet. PaaS is a paradigm for delivering operating systems and associated services (e.g., computer aided software engineering [CASE] tools, integrated development environments [IDEs] for developing software solutions) over the Internet without downloads or installation. IaaS involves outsourcing the equipment used to support operations, including storage, hardware, servers, and networking components.

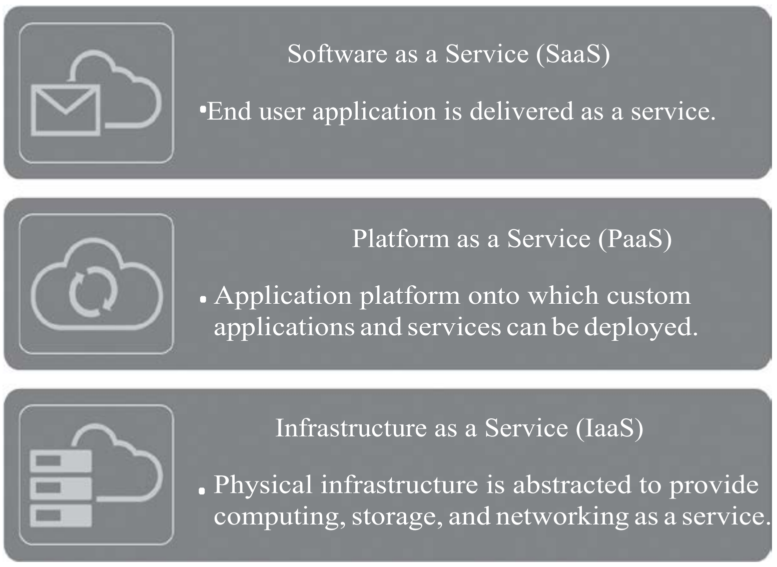
1. Cloud SaaS: The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure, including network, servers, operating systems, storage, and even individual application capabilities, with the possible exception of limited user-specific application configuration settings. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based e-mail), or a program interface. The consumer does not

manage or control the underlying cloud infrastructure. Typical applications offered as a service include customer relationship management (CRM), business intelligence analytics, and online accounting software.

2. Cloud PaaS: The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure but has control over the deployed applications and possibly configuration settings for the application-hosting environment. In other words, it is a packaged and ready-to-run development or operating framework. The PaaS vendor provides the networks, servers, and storage and manages the levels of scalability and maintenance. The client typically pays for services used. Examples of PaaS providers include Google App Engine and Microsoft Azure Services.

3. Cloud IaaS: The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources on a pay-per-use basis where he or she is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over the operating systems, storage, and deployed applications and possibly limited control of select networking components (e.g., host firewalls). The service provider owns the equipment and is responsible for housing, cooling operation, and maintenance. Amazon Web Services (AWS) is a popular example of a large IaaS provider.

The major difference between PaaS and IaaS is the amount of control that users have. In essence, PaaS allows vendors to manage everything, while IaaS requires more management from the customer side. Generally speaking, organizations that already have a software package or application for a specific purpose and want to install and run it in the cloud should opt to use IaaS instead of PaaS.


Software as a Service (SaaS)
•End user application is delivered as a service.
Platform as a Service (PaaS)
• Application platform onto which custom applications and services can be deployed.
Infrastructure as a Service (IaaS)
• Physical infrastructure is abstracted to provide computing, storage, and networking as a service.

4.1.2 Cloud Ecosystem

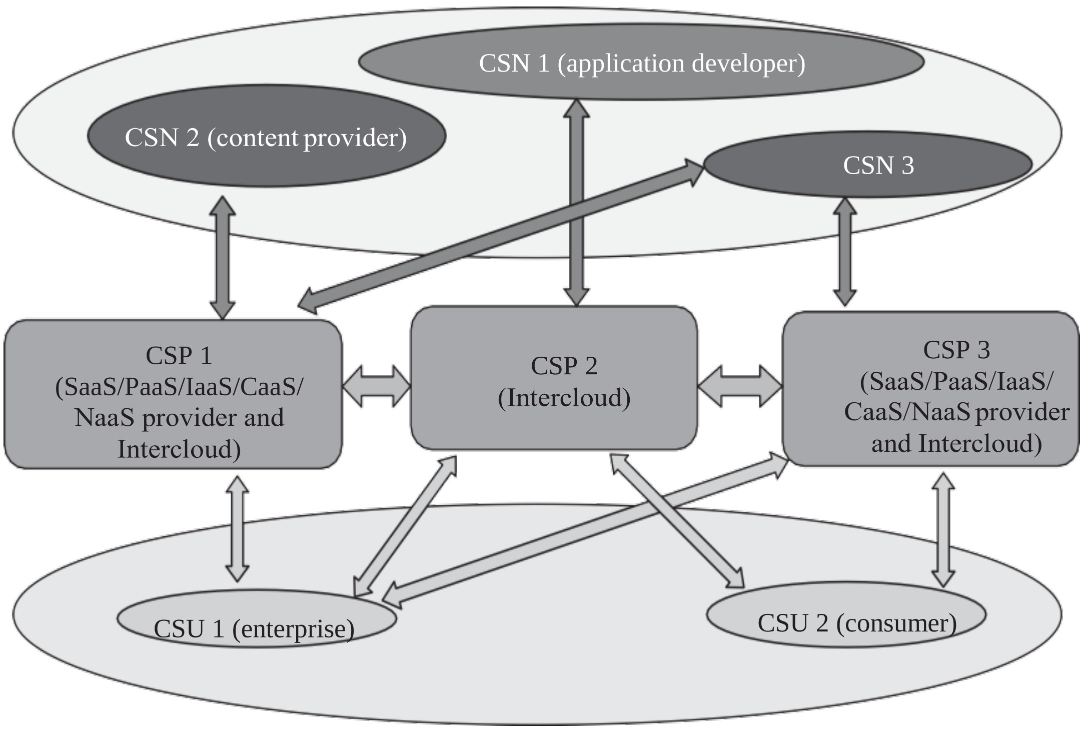
Cloud ecosystem is a term used to describe the complete environment or system of interdependent components or entities that work together to enable and support the cloud services. To be more precise, the cloud computing's ecosystem is a complex environment that includes the description of every item or entity along with their interaction; the complex entities include the traditional elements of cloud computing such as software (SaaS), hardware (PaaS and/or IaaS), other infrastructure (e.g., network, storage), and also stakeholders like consultants, integrators, partners, third parties, and anything in their environments that has a bearing on the other components of the cloud.

The cloud ecosystem of interacting components and organizations with individuals, together known as the actors who could be responsible for either providing or consuming cloud services, can be categorized in the following manner:

1. Cloud service users (CSUs): A consumer (an individual/person), enterprise (including enterprise administrator), and/or government/public institution or organization that consumes delivered cloud services; a CSU can include intermediate users that will deliver cloud services provided by a cloud service provider (CSP) to actual users of the cloud service, that is, end users. End users can be persons, machines, or applications.

2. CSPs: An organization that provides or delivers and maintains or manages cloud services, that is, provider of SaaS, PaaS, IaaS, or any allied computing infrastructure.

3. Cloud service partners (CSNs): A person or organization (e.g., application developer; content, software, hardware, and/or equipment provider; system integrator; and/or auditor) that provides support to the building of a service offered by a CSP (e.g., service integration).


CSN 1 (application developer)
CSN 2 (content provider)
CSN 3
CSP 1 (SaaS/PaaS/IaaS/CaaS/ NaaS provider and Intercloud)
CSP 2 (Intercloud)
CSP 3 (SaaS/PaaS/IaaS/ CaaS/NaaS provider and Intercloud)
CSU 1 (enterprise)
CSU 2 (consumer)

In layman's terms, the cloud ecosystem describes the usage and value of each entity in the ecosystem, and when all the entities in the ecosystem are put together, users are now able to have an integrated suite made up of the best-of-breed solutions. An example of this ecosystem can be a cloud accounting solution such as Tally; while this SaaS vendor focuses on their support for accounting and integrated payroll solutions, they can engage (collaborate) with any other third-party CSPs who could support additional features in the accounting software like reporting tools, dashboards, work papers, workflow, project management, and CRM, covering the majority of a client or customer firm's software needs. And, any other additional requirement that may be essential will likely be added by a partner joining the ecosystem in the near future. Figure 2.4 illustrates the idea of a cloud ecosystem.

4.1.3 Requirements for Cloud Services

From the concepts illustrated in the earlier sections, one can understand that the cloud services or service offering models require certain features to be exhibited in order to be considered as services. The following are the basic requirements for anything that can be considered as a service by the actors of the cloud computing ecosystem, which can be offered or provisioned through the cloud:

1. Multitenancy: Multitenancy is an essential characteristic of cloud systems aiming to provide isolation of the different users of the cloud system (tenants) while maximizing resource sharing. It is expected that multitenancy be supported at various levels of a cloud infrastructure. As an example, at the application level, multitenancy is a feature that allows a single instance of an application (say, database system) and leverages the economy of scale to satisfy several users at the same time.

2. Service life cycle management: Cloud services are paid as per usage and can be started and ended at any time. Therefore, it is required that a cloud service support automatic service provisioning. In addition, metering and charging or billing settlement needs to be provided for services that are dynamically created, modified, and then released in virtual environments.

3. Security: The security of each individual service needs to be protected in the multitenant cloud environment; the users (tenants) also support the needed secured services, meaning that a cloud provides strict control for tenants' service access to different resources to avoid the abuse of cloud resources and to facilitate the management of CSUs by CSPs.

4. Responsiveness: The cloud ecosystem is expected to enable early detection, diagnosis, and fixing of service-related problems in order to help the customers use the services faithfully.

5. Intelligent service deployment: It is expected that the cloud enables efficient use of resources in service deployment, that is, maximizing the number of deployed services while minimizing the usage of resources and still respecting the SLAs. For example, the specific application characteristics (e.g., central processing unit [CPU]-intensive, input/output [IO]-intensive) that can be provided by developers or via application monitoring may help CSPs in making efficient use of resources.

6. Portability: It is expected that a cloud service supports the portability of its features over various underlying resources and that CSPs should be able to accommodate cloud workload portability (e.g., VM portability) with limited service disruption.

7. Interoperability: It is expected to have available well-documented and well-tested specifications that allow heterogeneous systems in cloud environments to work together.

8. Regulatory aspects: All applicable regulations shall be respected, including privacy protection.

9. Environmental sustainability: A key characteristic of cloud computing is the capability to access, through a broad network and thin clients, on-demand shared pools of configurable resources that can be rapidly provisioned and released. Cloud computing can then be considered in its essence as an ICT energy consumption consolidation model, supporting mainstream technologies aiming to optimize energy consumption (e.g., in data centers) and application performance. Examples of such technologies include virtualization and multitenancy.

10. Service reliability, service availability, and quality assurance: CSUs demand for their services end-to-end quality of service (QoS) assurance, high levels of reliability, and continued availability to their CSPs.

11. Service access: A cloud infrastructure is expected to provide CSUs with access to cloud services from any user device. It is expected that CSUs have a consistent experience when accessing cloud services.

12. Flexibility: It is expected that the cloud service be capable of supporting multiple cloud deployment models and cloud service categories.

13. Accounting and charging: It is expected that a cloud service be capable to support various accounting and charging models and policies.

14. Massive data processing: It is expected that a cloud supports mechanisms for massive data processing (e.g., extracting, transforming, and loading data). It is worth to note in this context that distributed and/ or parallel processing systems will be used in cloud infrastructure deployments to provide large-scale integrated data storage and processing capabilities that scale with software-based fault tolerance.

The expected requirements for services in the IaaS category include the following:

- Computing hardware requirements (including processing, memory, disk, network interfaces, and virtual machines)
- Computing software requirements (including OS and other preinstalled software)
- Storage requirements (including storage capacity)
- Network requirements (including QoS specifications, such as bandwidth and traffic volumes)
- Availability requirements (including protection/backup plan for computing, storage, and network resources)

The expected service requirements for services in the PaaS category include the following:

- Requirements similar to those of the IaaS category
- Deployment options of user-created applications (e.g., scale-out options)

The expected service requirements for services in the SaaS category include the following:

- Application-specific requirements (including licensing options)
- Network requirements (including QoS specifications such as bandwidth and traffic volumes)

4.1.4 Cloud Application

A cloud application is an application program that functions or executes in the cloud; the application can exhibit some characteristics of a pure desktop application and some characteristics of a pure web-based application. A desktop application resides entirely on a single device at the user's location (it does not necessarily have to be a desktop computer), and on the other hand, a web application is stored entirely on a remote server and is delivered over the Internet through a browser interface.

Like desktop applications, cloud applications can provide fast responsiveness and can work offline. Like web applications, cloud applications need not permanently reside on the local device, but they can be easily updated online. Cloud applications are, therefore, under the user's constant control, yet they need not always consume storage space on the user's computer or communications device. Assuming that the user has a reasonably fast Internet connection, a well-written cloud application offers all the interactivity of a desktop application along with the portability of a web application.

A cloud application can be used with a web browser connected to the Internet. Now, it is possible for the user interface portion of the application to exist on the local device and for the user to cache data locally, enabling full offline mode when desired. Also, a cloud application, unlike a web app, can be used in any sensitive situation where wireless devices—connectivity—are not allowed (i.e., even when no Internet connection is available for some period).

An example of cloud application is a web-based e-mail (e.g., Gmail, Yahoo mail); in this application, the user of the e-mail uses the cloud—all of the emails in their inbox are stored on servers at remote locations at the e-mail service provider.

However, there are many other services that use the cloud in different ways. Here is yet another example: Dropbox is a cloud storage service that lets us easily store and share files with other people and access files from a mobile device as well.

4.1.5 Benefits and Drawbacks

One of the attractions of cloud computing is accessibility. If our applications and documents are in the cloud and are not saved on an office server, then we can access and use them at anytime, anywhere for our working, whether we are at work, at home, or even at a friend's house. Cloud computing also enables precisely the right amount of computing power and resources to be used for applications. Cloud computing vendors provide computing-related services as a bundle of computing power and parcel it out on demand. Customers can draw and make use as much or as little computing power as they need, being charged only for the usage time/computing power; accordingly, this scheme can save money. This also implies that scalability is one of the cloud computing's big benefits. When we need more computing power, cloud computing can give instant access to exactly what we need. In the cloud model, an organization's core computer power resides offsite and is essentially subscribed to rather than owned. There is no capital expenditure, only operational expenditure. It also relieves us from the responsibility and costs of maintenance of the entire computing infrastructure and pushes all these to the cloud vendor or provider. The cloud also offers a new level of reliability. The virtualization technology enables a vendor's cloud software to automatically move data from a piece of hardware that goes bad or is pulled offline to a section of the system or hardware that is functioning or operational. Therefore, the client gets seamless access to the data. Separate backup systems, with cloud disaster recovery strategies, provide another layer of dependability and reliability. Finally, cloud computing also promotes a green alternative to paper-intensive office functions. It is because it needs less computing hardware on premise, and all computing-related tasks take place remotely with minimal computing hardware requirement with the help of technological innovations such as virtualization and multitenancy. Another viewpoint on the green aspect is that cloud computing can reduce the environmental impact of building, shipping, housing, and ultimately destroying (or recycling) computer equipment as no one is going to own many such systems in their premises and managing the offices with fewer computers that consume less energy comparatively. A consolidated set of points briefing the benefits of cloud computing can be as follows:

1. Achieve economies of scale: We can increase the volume output or productivity with fewer systems and thereby reduce the cost per unit of a project or product.

2. Reduce spending on technology infrastructure: It is easy to access data and information with minimal upfront spending in a pay-as-you-go approach, in the sense that the usage and payment are similar to an electricity meter reading in the house, which is based on demand.

3. Globalize the workforce: People worldwide can access the cloud with Internet connection.

4. Streamline business processes: It is possible to get more work done in less time with less resource.

5. Reduce capital costs: There is no need to spend huge money on hardware, software, or licensing fees.

6. Pervasive accessibility: Data and applications can be accessed anytime, anywhere, using any smart computing device, making our life so much easier.

7. Monitor projects more effectively: It is possible to confine within budgetary allocations and can be ahead of completion cycle times.

8. Less personnel training is needed: It takes fewer people to do more work on a cloud, with a minimal learning curve on hardware and software issues.

9. Minimize maintenance and licensing software: As there is no too much of on-premise computing resources, maintenance becomes simple and updates and renewals of software systems rely on the cloud vendor or provider.

10. Improved flexibility: It is possible to make fast changes in our work environment without serious issues at stake.

Drawbacks to cloud computing are obvious. The main point in this context is that if we lose our Internet connection, we have lost the link to the cloud and thereby to the data and applications. There is also a concern about security as our entire working with data and applications depend on other's (cloud vendor or providers) computing power. Also, while cloud computing supports scalability (i.e., quickly scaling up and down computing resources depending on the need), it does not permit the control on these resources as these are not owned by the user or customer. Depending on the cloud vendor or provider, customers may face restrictions on the availability of applications, operating systems, and infrastructure options. And, sometimes, all development platforms may not be available in the cloud due to the fact that the cloud vendor may not aware of such solutions. A major barrier to cloud computing is the interoperabebility of applications, which is the ability of two or more applications that are required to support a business need to work together by sharing data and other business-related resources. Normally, this does not happen in the cloud as these applications may not be available with a single cloud vendor and two different vendors having these applications do not cooperate with each other.

4.2 Cloud Service Providers Comparison

Cloud computing is one of the most popular buzzwords used these days. It is the upcoming technology provisioning resources to the consumers in the form of different services like software, infrastructure, platform, and security. Services are made available to users on demand via the Internet from a cloud computing provider's servers as opposed to being provided from a company's own on-premise servers. Cloud services are designed to provide easy, scalable access to applications, resources, and services and are fully managed by a cloud service provider. A cloud service can dynamically scale to meet the needs of its users, and because the service provider supplies the hardware and software necessary for the service, there is no need for a company to provision or deploy its own resources or allocate information technology (IT) staff to manage the service. Examples of cloud services include online data storage and backup solutions, web-based e-mail services, hosted office suites and document collaboration services, database processing, and managed technical support services.

4.2.1 Computation

- **Amazon Web Services**

Amazon Elastic Compute Cloud (Amazon EC2) provides basic computation service in AWS. It presents a virtual computing environment and enables resizable compute capacity. Users can simply use a pre-configured Amazon Machine Image (AMI) (pre-configured operating system and application software) or create their own AMIs. Users can then choose between different instance types with different virtual CPU cores and amount of memory. There are also special types of instance to meet various application needs, e.g. High-CPU/ Memory/ I/O Instances, Cluster Compute Instances (for HPC and network-bound applications), Cluster GPU Instances and EBS-Optimized Instances (enable Amazon EC2 instances to fully utilize the IOPS provisioned on an EBS volume). EC2 instances can be launched in multiple locations (Regions and Availability Zones). Failures are insulated among different Availability Zones. Regions are geographically distributed and consist of one or more Availability Zones. Amazon Elastic IP Addresses are static IP addresses and can be used to remap public IP addresses to any instance in an account in case of instance or Availability Zone failures. Apart from launching instances on demand, users can also reserve certain instances or bid on unused EC2 capacity to run Spot Instances. EC2 also offers Auto Scaling and Elastic Load Balancing services. Auto Scaling allows users to scale up/down their EC2 capacity automatically when pre-define events are triggered. Elastic Load Balancing automatically distributes incoming traffic across multiple Amazon EC2 instances, which brings improved responsiveness as well as fault tolerance.

- **Windows Azure**

Windows Azure Virtual Machines provides IaaS similar to EC2. To create a VM, users need to choose a virtual hard disk (VHD) for the VM's image. Users can either use VHDs provided by Microsoft (Windows Server) and its partners (Linux images), or upload their own VHDs. Then users need to specify the size of new VM (different number of cores and amount of memory).

- **Google**

Google Cloud Platform contains a suite of products that allows users to build applications and websites, store and analyze data on Google's infrastructure. Google Compute Engine is the IaaS cloud platform that offers flexible VMs hosted on Google. Currently it is still in limited preview stage and is open by invitation only and preferably to those with large computational workloads. It only supports Linux based virtual machines running on KVM hypervisor right now. Users can choose the specific location (called zone) to launch instances. A zone is defined by (region, availability group) tuple which is similar to (region, availability zone) in AWS.

4.2.2 Storage

- **Amazon Web Services**

EC2 instances come with a virtual local disk, but data in this disk may be lost if the instance fails. AWS provides Elastic Block Store (EBS) offers persistent storage to EC2 instances and is independent from instance life. EBS provides block level storage volumes and can be mounted as devices by running EC2 instances. EBS behaves like a raw/unformatted block device and users can create a file system on it.

There are two types of EBS volume: Standard volume and Provisioned IOPS volume. Users can choose Provisioned IOPS volumes if predictable and high I/O performance is desired. An EBS volume is placed in a specific availability zone and automatically replicated within the same availability zone. Users can create consistent snapshots of EBS volumes which will be stored in Amazon S3 and automatically replicated across multiple availability zones.

Amazon Simple Storage Service (S3) is a fully redundant data storage for the Internet. Amazon Glacier provides extremely low-cost storage specifically for data archiving and backup. It is optimized for data that is infrequently accessed and retrieval of data may take several hours. AWS Storage Gateway service allows users to back up of on-premises application data to Amazon S3 for future recovery. AWS Import/Export service offers faster data transfer into and out of AWS by using portable storage devices rather than transferring data via the Internet.

Amazon offers Relational Database Service (Amazon RDS) to give users access to the capabilities of MySQL, Oracle or Microsoft SQL Server database engine. Amazon SimpleDB provides NoSQL database service for smaller datasets and Amazon DynamoDB provides fully-managed, high performance, NoSQL database service.

- **Windows Azure**

Windows Azure Blob provides storage to store large amounts of unstructured data. A blob is a file of any type and size. There are two types of blobs in Windows Azure Storage: block and page blobs. Block blobs consists of blocks (each block up to 4MB) and are efficient when uploading large blobs. Most files are block blobs. Page blobs are a collection of 512-byte pages optimized for random read and write operations. Page blobs are more efficient when ranges of bytes in a file are modified frequently. Each VM is associated with an OS disk (if a provided VHD is used to create a VM, that VHD is copied to VM's OS disk) and one or more data disks. Each disk is stored in a blob which is replicated both within a single datacenter and across datacenters.

Microsoft provides Windows Azure SQL Database as a relational database option. Windows Azure Table is a NoSQL datastore which is ideal for storing structured, non-relational data. Windows Azure Queue is a service for storing large numbers of messages that can be accessed from anywhere.

- **Google**

Each VM in Google Compute Engine has an ephemeral disk (by default 10GB) tied to the lifetime of VM instance. Users can request persistent disks which are independent disks which could outlive an instance's lifespan. Data written to persistent disks is automatically replicated across multiple disks in data centers. During current Limited Preview period, specific zones may be taken down for maintenance and upgrades. Data on ephemeral disks will be lost during the maintenance window period. Data on persistent disks will still be there, but users need to migrate their persistent disks ahead of time manually with FTP or rsync.

Google Cloud Storage is a service for developers to store and access data in Google's cloud and is similar to Amazon S3. Developers can store objects and files up to terabytes and manage access to the data. All data is replicated to multiple data centers for high availability.

Google Cloud SQL is a relational SQL database service based on MySQL and is good for medium or small data sets.

4.2.3 Networking

By default, a VM instance in AWS or Windows Azure is standalone and with its own public IP address. Amazon Virtual Private Cloud (VPC) and Windows Azure Virtual Network (VNET) allow users to group VMs into a private and isolated network in the cloud. In a VPC/VNET, users can define the virtual network topology and have complete control over private IP address range (all VMs in the same VPC/VNET can be accessed through a single public IP address), creation of subnets, and configuration of route tables and network gateways. Amazon VPC also allows assigning an Elastic IP address (a static, public address) to any VM in VPC to make it addressable from the Internet. Users can also create and attach additional network interfaces (elastic network interface, or ENI) to any VM in VPC.

To extend on-premises datacenter into the public cloud, Amazon and Windows Azure both provide solutions for hybrid cloud. In VPC/VNET, users can create an encrypted IPsec hardware VPN connection between corporate VPN gateway and VPC/VNET.

- **Amazon Web Services**

Compared to a VPN connection over the Internet, Amazon offers another network service called AWS Direct Connect which is suitable for high-bandwidth and latency-sensitive applications. Direct Connect uses industry standard 802.1Q VLANs to establish a dedicated and private connection between premise to AWS. Since it is hard for VPN hardware to support data transfer rates above 4 Gbps, users can easily get more network capacity with multiple Direct Connect connections, each with 1 Gbps or 10 Gbps. By transferring data directly to and from AWS, users also get a more consistent network experience. One nice thing is this dedicated connection can be partitioned into multiple virtual interfaces. So the same connection could be used to access both public resources (e.g. objects in Amazon S3) and private resources (e.g. EC2 instances in VPC), or access multiple VPCs.

Amazon also provides Route53, a DNS web service with which users could create and manage public DNS records. By using a global network of DNS servers, DNS queries will be routed to the nearest DNS server and answered with low latency.

- **Windows Azure**

Windows Azure Connect provides agent-based, machine-to-machine connections between Windows Azure services and on-premises resources. With Windows Azure Connect, VMs in Windows Azure can join the domain on premises. So VMs in Windows Azure have IP addresses that look like other networked resources in the same domain rather than use external virtual IP addresses. This greatly helps domain management (e.g. authentication, name resolution, domain-wide maintenance, remote debug) and distributed application development (e.g. a web application hosted in Windows Azure can securely access an on-premise SQL Server database server).

Windows Azure also offers Traffic Manager to load balance incoming traffic across multiple Windows Azure services, ensure high availability and improve the responsiveness by serving end-users with the closest service.

- **Google**

Each VM instance in Google Compute Engine belongs to a single network, which defines the address range and gateway address of all instances connected to it. Users can specify firewall rules for an instance. An instance can get an external IP address when it is started. Traffic between the instance and the Internet or other instances in different networks will use this public IP address. An instance without an external IP address can only access instances in the same network.

4.2.4 Other Features

- **PaaS technology**

To allow users to focus on their applications rather than infrastructure, AWS, Windows Azure and Google all provide PaaS technology to simplify application deployment and management. PaaS will handle all deployment details such as capacity provisioning, load balancing, auto-scaling and health monitoring. All users have to do is just to upload their applications.

AWS Elastic Beanstalk is such a PaaS technology provided by Amazon. In addition, AWS Cloud Formation enables users to create a collection of related AWS resources. For example, users could quickly launch multi-tier web applications with a Cloud Formation template.

Windows Azure Cloud Services is a similar PaaS platform from Microsoft. Moreover, if all users want is a web site or web application, they could get web hosting service directly from Windows Azure Web Sites. For application development, Windows Azure provides SDKs for .net, php, java and python.

Google App Engine is a Google's PaaS service to host web applications. It supports applications written in three programming languages: Java, Python and Go (an open source programming environment developed by Google). Users can choose between three options for storing data: Google Cloud SQL, Google Cloud Storage or App Engine Datastore. App Engine Datastore is a distributed, schema-less NoSQL object datastore which features a query engine and ACID transactions. Two options are provided with different availability and consistency guarantees in App Engine Datastore. The primary data repository is called High Replication Datastore (HRD), in which data is replicated across multiple data centers based on Paxos.

- **Caching**

Both Amazon ElastiCache (beta) and Windows Azure Caching (Preview) support creating a Cache Cluster consisting of a collection of cache nodes to store information in-memory from backend sources (e.g. SQL Database, session state and output caching for ASP applications). The Cache Cluster can scale up/down by adding/deleting cache nodes.

Each cache node in Amazon ElastiCache runs Memcached (a memory object caching system) software. Windows Azure Caching (Preview) also supports Memcache now. In addition, it has two options -- collocated caches which just use part of memory on the virtual machines, or dedicated caches which use all available memory on the virtual machines for caching.

- **Big Data Support**

Amazon provides Elastic MapReduce (Amazon EMR) to instantly provision as much or as little capacity as users wish for data-intensive applications. Users can focus on data analysis rather than time-consuming set-up, management or tuning of Hadoop clusters. Users can modify the number of instances while the job flow is running. MapR distribution is provided in EMR and HBase can run on EMR now.

Microsoft also provides a Hadoop service on Windows Azure. On Windows Azure, the data a MapReduce job works on is typically kept in blob storage. In contrast, for Amazon EMR, input data needs to be loaded from S3 to EC2 instances first, and the results will be written back to S3 again at the end. Windows Azure also supports Pig and Hive. It also provides a HiveQL (a SQL-like language Hive offers) driver for Excel, with which HiveQL queries can be created directly from Excel.

As the birthplace of MapReduce, it is interesting Google does not provide any Hadoop service at the moment. But it is said a private beta of MapR distribution is running on the Google Compute Engine and may serve as Hadoop platform in the future. Google BigQuery service allows users to do interactive analysis against very large datasets (up to billions of rows) with great speed. It is not a database but uses SQL-like queries. Usually it uses a small number of very large, append-only tables. Google Prediction API is a cloud-based machine learning tool with Google's machine learning algorithms.

- **Messaging**

Amazon Simple Queue Service (SQS) offers a reliable way for messages to travel between applications. It stores in-flight messages and does not require applications to be always available. Amazon Simple Notification Service (SNS) provides another way for messaging. Messages published from an application will be delivered to subscribers immediately.

Windows Azure offers "Service Bus" for communications between applications or services. Service Bus provides three options to meet different communication requirements. Like Amazon SQS, queues provide FIFO guaranteed message delivery communication. But it is just one-directional and serves as a broker that stores sent messages until they are received. Topics offer the same way as Amazon SNS but it is again one-directional communication. It delivers messages that match specific criteria to corresponding downstream subscriptions. Unlike queues and topics, relays provide bi-directional communication. But it just passes messages on to the destination application and does not store any messages. So it is suitable for communications between on-premises applications (e.g. web services) and public endpoints projected in the cloud.

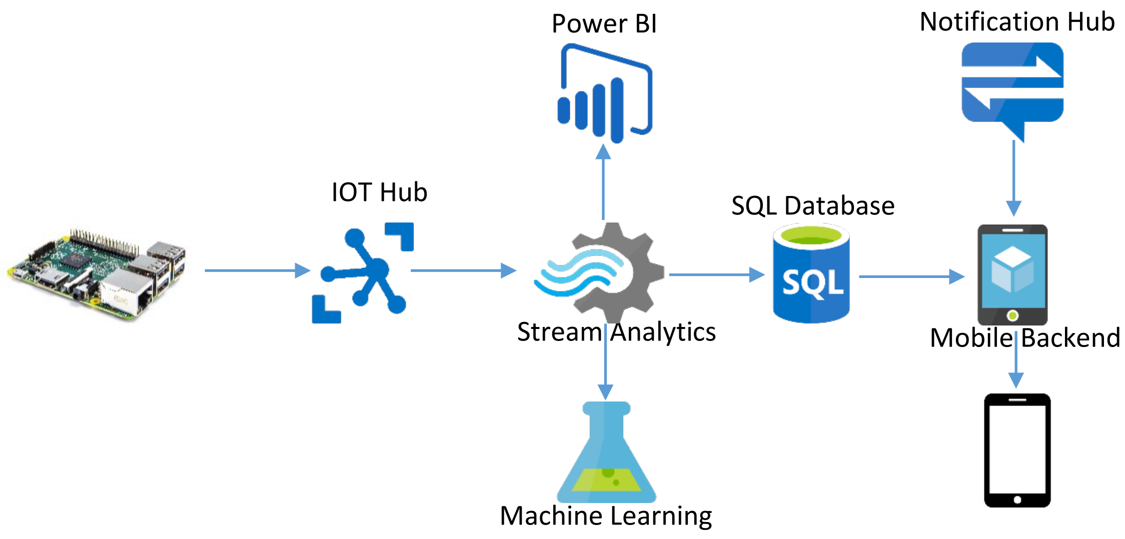
- **Content Delivery Network**

Amazon offers Cloud-Front to deliver dynamic, static and streaming content using a global network of edge locations. Windows Azure Content Delivery Network (CDN) also provides a solution for delivering high-bandwidth content by caching blobs and static content of compute instances at physical nodes distributed globally.

- **Metric Monitoring**

Amazon provides Cloud-Watch to enable users to monitor AWS cloud resources including EC2 instances, EBS volumes, RDS DB instances, etc. Users could track basic metrics such as CPU utilization, disk and network activity of each EC2 instance at a five-minute frequency. Users could also specify other metrics and update metrics at one-minute intervals. The monitoring functionality is integrated into Management Portal for Windows Azure, which allows minimal and verbose monitoring similar to Amazon Cloud-Watch.

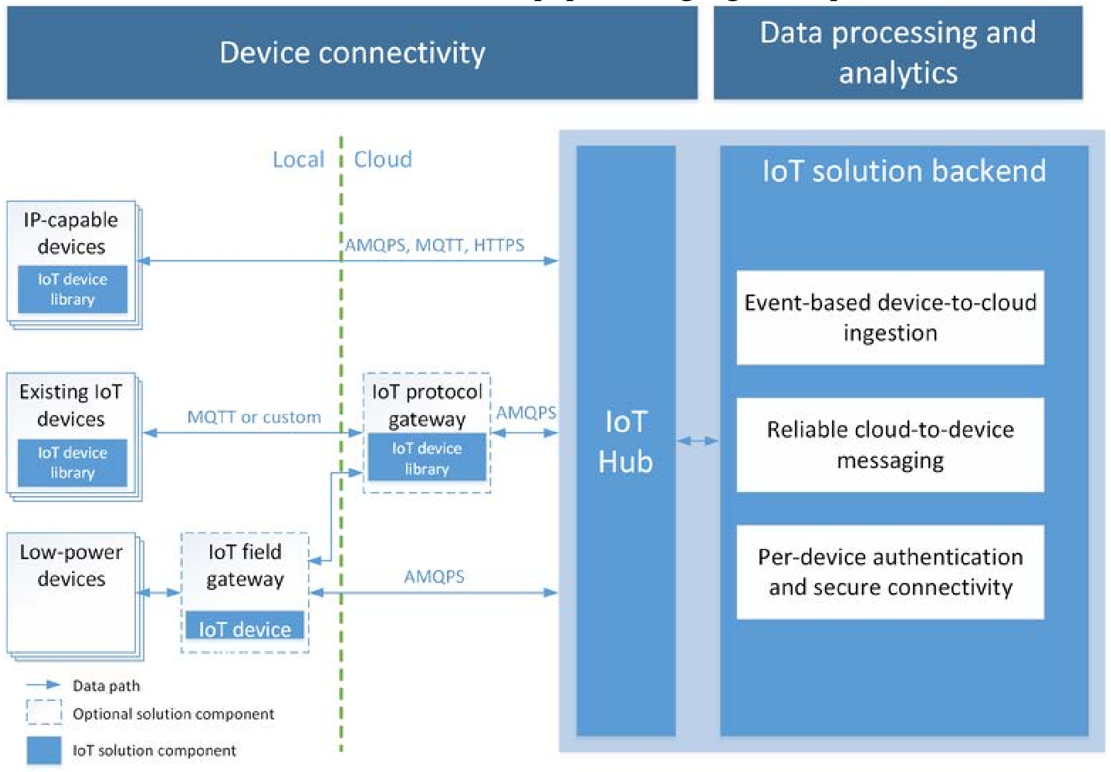
4.3 Microsoft Azure



4.3.1 IOT Hub

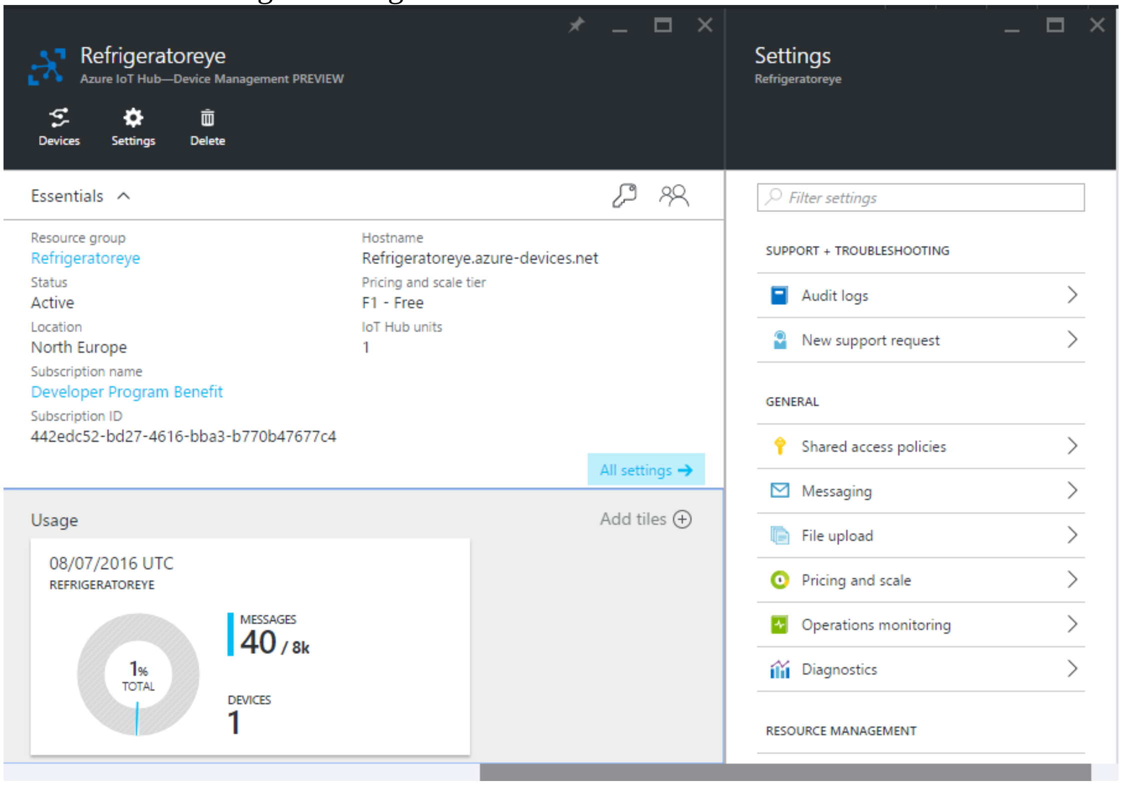
Azure IoT Hub is a fully managed service that enables reliable and secure bidirectional communications between millions of IoT devices and a solution back end. Azure IoT Hub:

- Provides reliable device-to-cloud and cloud-to-device messaging at scale.
- Enables secure communications using per-device security credentials and access control.
- Provides extensive monitoring for device connectivity and device identity management events.
- Includes device libraries for the most popular languages and platforms.



Create an IoT hub

1. Log on to the Azure Preview Portal.
2. In the jumpbar, click **New**, then click **Internet of Things**, and then click **Azure IoT Hub**.
3. In the **New IoT Hub** blade, specify the desired configuration for the IoT Hub.
4. Once the new IoT hub options are configured, click **Create**. It can take a few minutes for the IoT hub to be created. To check the status, you can monitor the progress on the Startboard. Or, you can monitor your progress from the Notifications section.
5. After the IoT hub has been created successfully, open the blade of the new IoT hub, take note of the URI, and select the **Key** icon on the top.
6. Select the Shared access policy called **iothubowner**, then copy and take note of the connection string on the right blade.


Refrigeratoreye
Azure IoT Hub—Device Management PREVIEW

Essentials

Resource group: Refrigeratoreye
Status: Active
Location: North Europe
Subscription name: Developer Program Benefit
Subscription ID: 442edc52-bd27-4616-bba3-b770b47677c4
Hostname: Refrigeratoreye.azure-devices.net
Pricing and scale tier: F1 - Free
IoT Hub units: 1

Usage
08/07/2016 UTC
REFRIGERATOREYE
1% TOTAL
MESSAGES 40 / 8k
DEVICES 1

Settings
Refrigeratoreye

Manage IoT Hub

Before a device can communicate with IoT Hub, you must add details of that device to the IoT Hub device identity registry. When you add a device to your IoT Hub device identity registry, the hub generates the connection string that the device must use when it establishes its secure connection to your hub. You can also use the device identity registry to disable a device and prevent it from connecting to your hub.

To add devices to your IoT hub and manage those devices, you can use either of:

- The cross-platform, command-line iothub-explorer tool
- The Windows-only, graphical Device Explorer tool

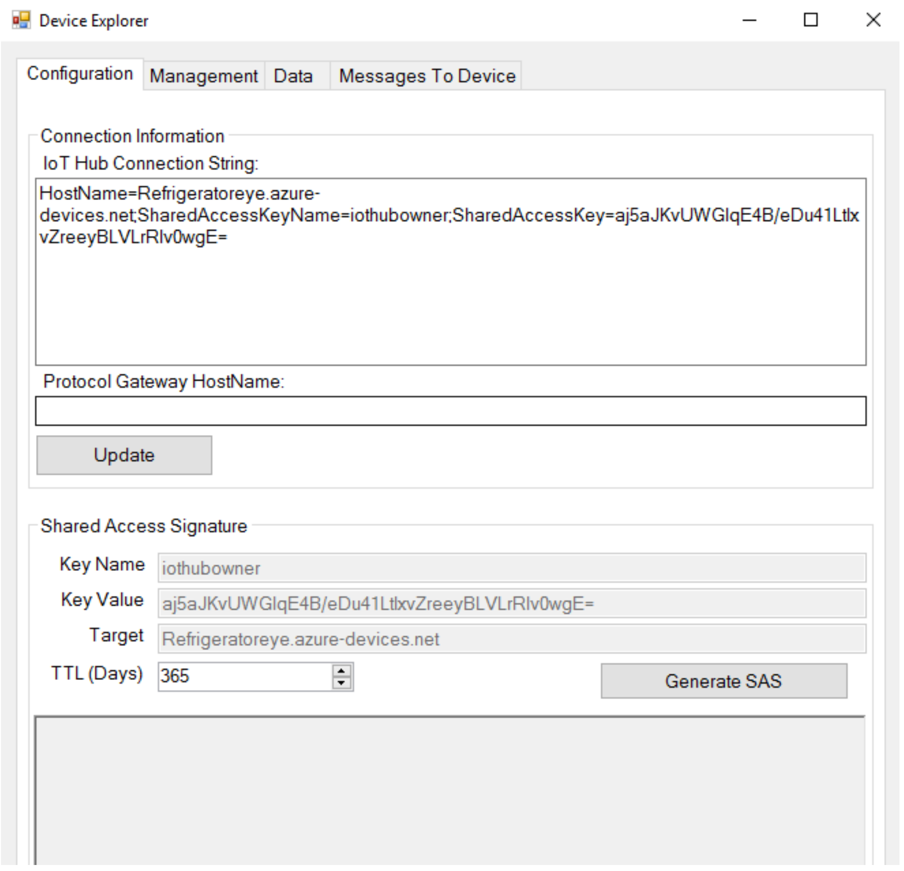
Use either of these tools to generate a device-specific connection string that you can copy and paste in the source code of the application running on your device. Both tools are available in this repository.

Note: While IoT Hub supports multiple authentication schemes for devices, both these tools generate a pre-shared key to use for authentication.

You can also use both of these tools to monitor the messages that your device sends to an IoT hub and send commands to you your devices from IoT Hub.

Use the Device Explorer tool to provision a device

1. Setup the tool.
2. Use the **Connection String** you took earlier
3. On the **Configuration** tab, paste the IoT Hub connection-string for your IoT hub into **IoT Hub connection string** and click **Update**.
4. Click the **Management** tab to manage the devices connected to the IoT hub.
5. On the **Management** tab, click **Create** to register a new device with your IoT hub. The **Create Device** dialog appears. In the **Device ID** field, type a unique name for your device such as **mydevice**, or select **Auto Generate ID** to generate a unique ID. Then click **Create**.
6. **Copy connection string** to copy the device connection string to the clipboard. You can now paste this connection-string into the source code of the device application you are working with. The samples in this repository use connection strings in the format HostName=<*iothub-name*>*.azure-devices.net;DeviceId=*<*device-name*>*;SharedAccessKey=*<*device-key*>



IoT Hub Connection String:

```
HostName=Refrigeratoreye.azure-
devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=aj5aJKvUWGlqE4B/eDu41Ltlx
vZreeyBLVLrRlv0wgE=
```

Protocol Gateway HostName:

Key Name: iothubowner
Key Value: aj5aJKvUWGlqE4B/eDu41LtlxvZreeyBLVLrRlv0wgE=
Target: Refrigeratoreye.azure-devices.net
TTL (Days): 365

Prepare your development environment

1. On the **Raspberry Pi** python is already installed.
2. Download the Azure IoT device SDK to your Raspberry Pi

```
git clone --recursive https://github.com/Azure/azure-iot-sdks.git
```

3. Confirm that you now have a copy of the SDK under the directory ./azure-iot-sdks. Then cd to the directory:

```
cd azure-iot-sdks
```

4. Prepare your environment by running. Answer **y** when you are prompted to install the additional components needed to run the samples:

```
sudo c/build_all/linux/setup.sh
sudo c/build_all/linux/build.sh
```

5. After a successful build, the iothub_client.so Python extension module is copied to the **python/device/samples** folder. Please follow instructions in Sample applications to run the Python samples.

Note: On some small footprint Linux devices, like a *Raspberry Pi* using Raspbian OS, the following build error may occur: virtual memory exhausted: Cannot allocate memory. In such a case please try to increase the swap file size on your platform and retry the build.

Choosing a protocol for use with IoT Hub

As noted before, IoT Hub supports HTTP, MQTT, and AMQPS protocols. There are a number of key considerations in choosing the one you'll use:

- **Library support:** The Microsoft Azure IoT SDK GitHub repository contains C, .NET, Java, and Node.js libraries to enable devices to work with IoT Hub, but not all libraries support all protocols.
- **Cloud-to-device pattern:** HTTP does not have an efficient way to implement server push. As such, when using HTTP, devices must poll IoT Hub for cloud-to-device messages. This is very inefficient for both the device and IoT Hub, and introduces latency in command delivery to a device. Although in our case cloud does not send commands to the device, a possible extension is to send a command to switch on a dashcam if the back end detects sudden breaking. To minimize latency in delivering commands, use the AMQPS or MQTT protocol.
- **Payload size**: AMQPS and MQTT are binary protocols, which are significantly more compact than HTTP. Using AMQPS or MQTT helps minimize any charges that arise from the phone's cellular data connection to IoT Hub.
- **Field gateways**: When using HTTP or MQTT, you cannot connect multiple devices (each with its own per-device credentials) using the same transport layer security (TLS) connection. It follows that these protocols are suboptimal when implementing a field gateway because they require one TLS connection between the field gateway and IoT Hub for each device that's connected to the gateway. However, in the current solution there is only one Raspberry Pi connected to all the sensors, so there is only one TLS connection to IoT Hub.
- **Low resource devices**: The MQTT and HTTP libraries have a smaller footprint than the AMQP libraries. As such, if the device has few resources (for example, less than 1 MB of RAM), these protocols might be the only protocol implementation available. However, modern smart phones typically have sufficient RAM to use the

AMQPS protocol. (Note that the C library that supports the AMQPS protocol is now much more compact than earlier versions.)

- **Network traversal**: MQTT uses port 8883. This can cause problems in networks that are closed to non-HTTP protocols. You can use both HTTPS and AMQPS over WebSockets in this scenario. However, this is unlikely to be an issue over the public data network that's used by the Raspberry Pi.

We Used the AMQP Protocol.

4.3.2 SQL Database

SQL Database is a relational database service in the cloud based on the market-leading Microsoft SQL Server engine, with mission-critical capabilities. SQL Database delivers predictable performance, scalability with no downtime, business continuity and data protection—all with near-zero administration. You can focus on rapid app development and accelerating your time to market, rather than managing virtual machines and infrastructure. Because it's based on the SQL Server engine, SQL Database supports existing SQL Server tools, libraries and APIs, which makes it easier for you to move and extend to the cloud.

Azure SQL Database manages billions of transactions and millions of databases per day. And one of its key features is that SQL Database is always learning and adapting with your app. That way you can dynamically maximize performance, reliability, and data security—with little effort on your part.

Threat detection and alerts

With threat and anomaly detection, SQL Database has built-in behavioral analysis, real-time alerts, a configurable threat policy, an audit log, and intelligent ways to detect and fix unusual patterns.

Automatic tuning

SQL Database is equipped to make your app run at its best performance. By continuously learning your app's patterns, adaptively self-tuning its performance, and automatically refining without you doing anything—because we know you don't want to anyway.

Insights when you need them

SQL Database is able to track each query and its duration, frequency, and resource utilization. Based on this telemetry, automatic algorithms optimally tune your databases exactly to your queries. Additionally, SQL Database provides insights to help minimize time tuning queries and troubleshooting performance issues. That means you gain direct insight into resource consumption, top performing queries, and the ability to drill down for more details.

No administration required

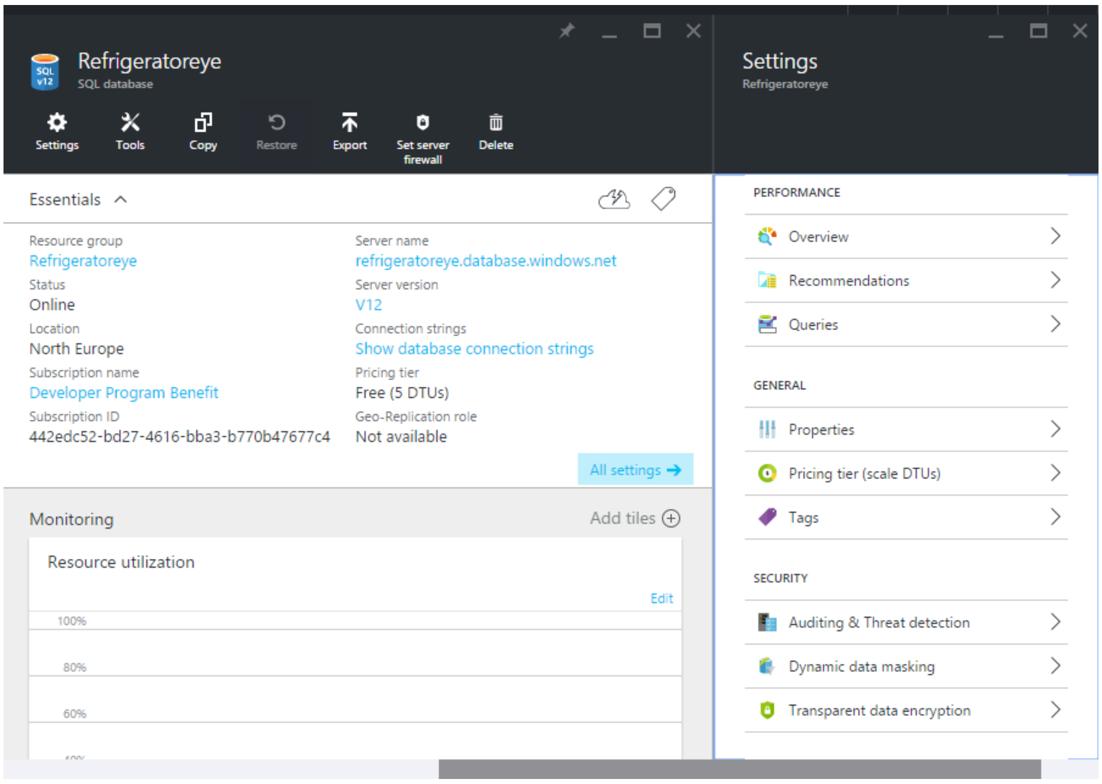
SQL Database provides the automatic administration and data protection your app needs so you can go back to doing what you do best—coding. You automatically get back-ups, disaster recovery failover, infrastructure maintenance, security and software patches, and feature updates. You'll not only save money and time as SQL Database works behind the scenes, but you'll also never have to worry about that dreaded app downtime.

Scales on the fly

One of the advantages of running SQL Database on Microsoft Azure is being able to scale performance up or down, and on the fly to quickly adapt to changing workload demands. SQL Database offers a broad spectrum of performance levels to meet the specific needs of your application. And each level guarantees performance, so your app users have a predictable performance experience.

Keep your app and business running

Azure's industry leading 99.99% availability service level agreement (SLA), powered by a global network of Microsoft-managed datacenters, helps keep your app running 24/7. With every SQL database, you take advantage of built-in data protection, fault tolerance, and data protection that you would otherwise have to design, buy, build, and manage. Even so, depending on the demands of your business, you may demand additional layers of protection to ensure your app and your business can recover quickly in the event of a disaster, an error, or something else. With SQL Database, each service tier offers a different menu of features you can use to get up and running and stay that way. You can use point-in-time restore to return a database to an earlier state, as far back as 35 days. In addition, if the datacenter hosting your databases experiences an outage, you can failover to database replicas in a different region. Or you can use replicas for upgrades or relocation to different regions.


Refrigeratoreye
SQL database

Resource group: Refrigeratoreye
Status: Online
Location: North Europe
Subscription name: Developer Program Benefit
Subscription ID: 442edc52-bd27-4616-bba3-b770b47677c4
Server name: refrigeratoreye.database.windows.net
Server version: V12
Connection strings: Show database connection strings
Pricing tier: Free (5 DTUs)
Geo-Replication role: Not available

Create a SQL Database logical server to host SQL databases

1. connect to the Azure portal.

2. Click **New**, type **SQL Database** and then click **SQL Database (new logical server)**
3. Click **Create** to open a template to create an empty logical server that can host single databases and elastic database pools.
4. Provide the values for the server **Properties**.
5. Click **Create** and in the notification area, you can see that deployment has started.

Create a new Azure SQL database

1. connect to the Azure portal.
2. Click **New**, type **SQL Database** and then click **SQL Database (new database)**
3. Click **Create** to create a new database in the SQL Database service.
4. Provide the values for the server **Properties**.
5. Click **Create** and in the notification area, you can see that deployment has started.

4.3.3 Mobile App Backend

Azure App Service is a fully managed Platform as a Service (PaaS) offering for professional developers that brings a rich set of capabilities to web, mobile and integration scenarios. *Mobile Apps* in *Azure App Service* offer a highly scalable, globally available mobile application development platform for Enterprise Developers and System Integrators that brings a rich set of capabilities to mobile developers.

Why Mobile Apps?

Mobile Apps in *Azure App Service* offers a highly scalable, globally available mobile application development platform for Enterprise Developers and System Integrators that brings a rich set of capabilities to mobile developers. With Mobile Apps you can:

- **Build native and cross platform apps** - whether you're building native iOS, Android, and Windows apps or cross-platform Xamarin or Cordova (Phonegap) apps, you can take advantage of App Service using native SDKs.
- **Connect to your enterprise systems** - with Mobile Apps you can add corporate sign on in minutes, and connect to your enterprise on-premises or cloud resources.
- **Build offline-ready apps with data sync** - make your mobile workforce productive by building apps that work offline and use Mobile Apps to sync data in the background when connectivity is present with any of your enterprise data sources or SaaS APIs.
- **Push Notifications to millions in seconds** - engage your customers with instant push notifications on any device, personalized to their needs, sent when the time is right.

Mobile App Features

The following features are important to cloud-enabled mobile development:

- **Authentication and Authorization** - Select from an ever-growing list of identity providers, including Azure Active Directory for enterprise authentication, plus social providers like Facebook, Google, Twitter and

Microsoft Account. Azure Mobile Apps provides an OAuth 2.0 service for each provider. You can also integrate the SDK for the identity provider for provider specific functionality.

- **Data Access** - Azure Mobile Apps provides a mobile-friendly OData v3 data source linked to SQL Azure or an on-premise SQL Server. This service can be based on Entity Framework, allowing you to easily integrate with other NoSQL and SQL data providers, including Azure Table Storage, MongoDB, DocumentDB and SaaS API providers like Office 365 and Salesforce.com.
- **Offline Sync** - Our Client SDKs make it easy for you to build robust and responsive mobile applications that operate with an offline data set that can be automatically synchronized with the backend data, including conflict resolution support.
- **Push Notifications** - Our Client SDKS seamlessly integrate with the registration capabilities of Azure Notification Hubs, allowing you to send push notifications to millions of users simultaneously.
- **Client SDKs** - We provide a complete set of Client SDKs that cover native development (iOS, Android and Windows), cross-platform development (Xamarin for iOS and Android, Xamarin Forms) and hybrid application development (Apache Cordova). Each client SDK is available with an MIT license and is open-source.

Azure App Service Features.

The following platform features are generally useful for mobile production sites.

- **Auto Scaling** - App Service enables you to quickly scale-up or out to handle any incoming customer load. Manually select the number and size of VMs or set up auto-scaling to scale your mobile app backend based on load or schedule.
- **Staging Environments** - App Service can run multiple versions of your site, allowing you to perform A/B testing, test in production as part of a larger DevOps plan and do in-place staging of a new backend.
- **Continuous Deployment** - App Service can integrate with common SCM systems, allowing you to automatically deploy a new version of your backend by pushing to a branch of your SCM system.
- **Virtual Networking** - App Service can connect to on-premise resources using virtual network, ExpressRoute or hybrid connections.
- **Isolated / Dedicated Environments** - App Service can be run in a fully isolated and dedicated enviroment for securely running Azure App Service apps at high scale. This is ideal for application workloads requiring very high scale, isolation or secure network access.

Create a new Azure mobile app backend

1. Log in at the Azure Portal.
2. Click **+NEW** > **Web + Mobile** > **Mobile App**, then provide a name for your Mobile App backend.
3. For the **Resource Group**, select an existing resource group, or create a new one (using the same name as your app.)
4. For the **App Service plan**, the default plan (in the Standard tier) is selected. You can also select a different plan, or create a new one. The App Service plan's settings determine thelocation, features, cost and compute resources associated with your app.

After you decide on the plan, click **Create**. This creates the Mobile App backend.

5. In the **Settings** blade for the new Mobile App backend, click **Quick start** > your client app platform > **Connect a database**.
6. In the **Add data connection** blade, click **SQL Database** > **Create a new database**, type the database **Name**, choose a pricing tier, then click **Server**. You can reuse this new database. If you already have a database in the same location, you can instead choose **Use an existing database**. The use of a database in a different location isn't recommended due to bandwidth costs and higher latency.
7. In the **New server** blade, type a unique server name in the **Server name** field, provide a login and password, check **Allow azure services to access server**, and click **OK**. This creates the new database.
8. Back in the **Add data connection** blade, click **Connection string**, type the login and password values for your database, and click **OK**. Wait a few minutes for the database to be deployed successfully before proceeding.

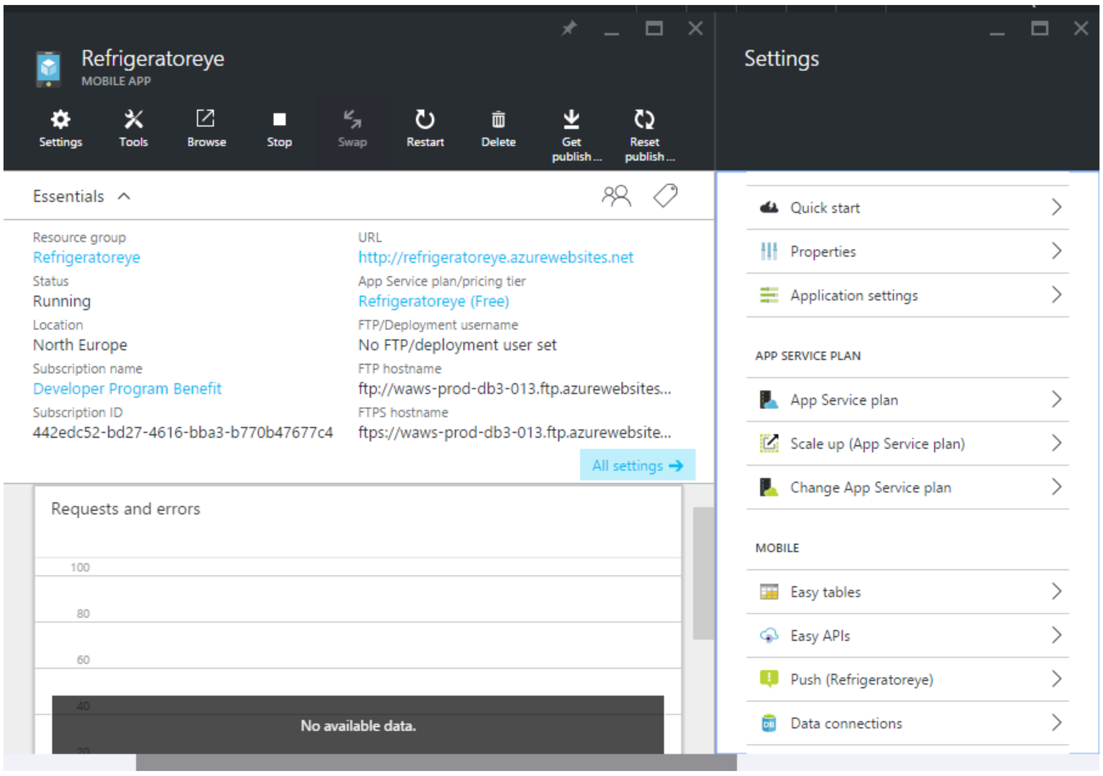
Apps, and App Service plans

An app in App Service can be associated with only one App Service plan at any given time.

Both apps and plans are contained in a resource group. A resource group serves as the life-cycle boundary for every resource contained within it. Resource groups enable you to manage all the pieces of an application together.

The ability to have multiple App Service plans in a single resource group allows you to allocate different apps to different physical resources. For example, this allows separation of resources between dev, test and production environments. A scenario for this is when you might want to allocate one plan with its own dedicated set of resources for your production apps, and a second plan for your dev and test environments. In this way, load testing against a new version of your apps will not compete for the same resources as your production apps, which are serving real customers.

Having multiple plans in a single resource group also enables you to define an application that spans across geographical regions. For example, a highly available app running in two regions will include two plans, one for each region, and one app associated with each plan. In such a situation, all the copies of the app will be associated with a single resource group. Having a single view of a resource group with multiple plans and multiple apps makes it easy to manage, control and view the health of the application.

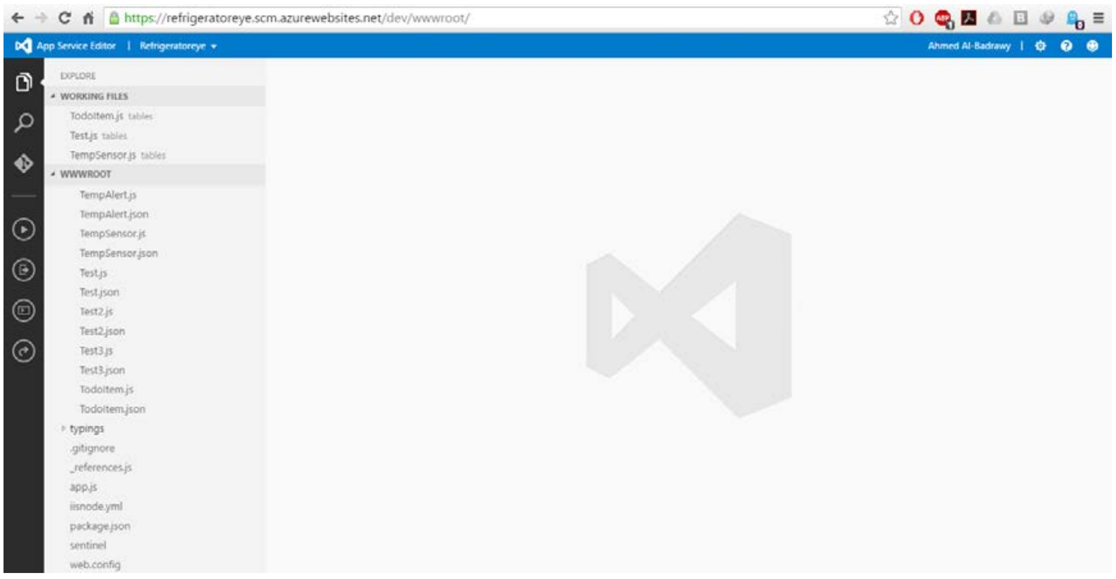


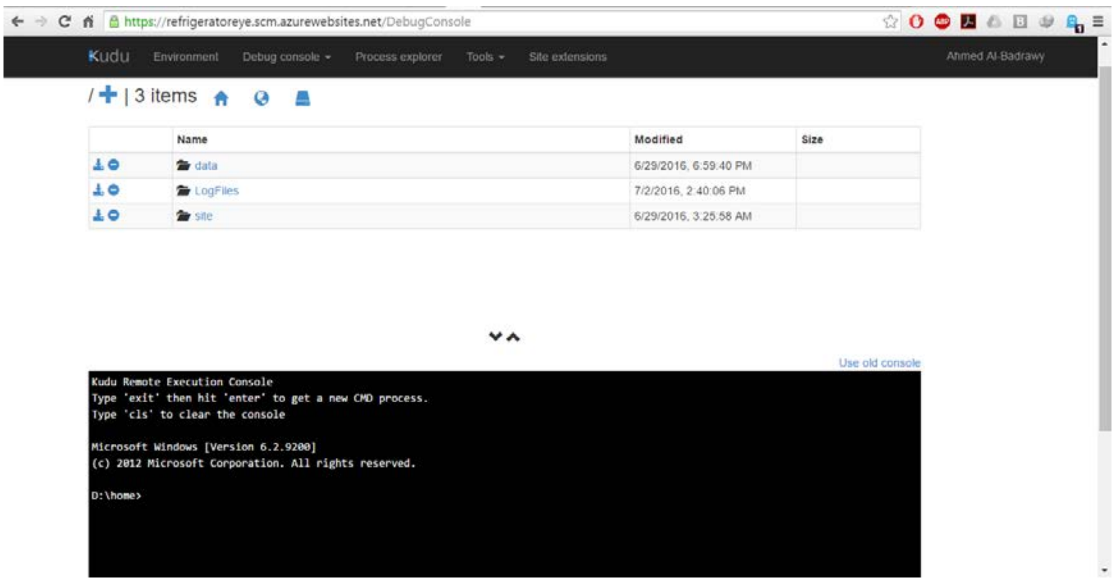
Resource group
Refrigeratoreye
Status
Running
Location
North Europe
Subscription name
Developer Program Benefit
Subscription ID
442edc52-bd27-4616-bba3-b770b47677c4

URL
http://refrigeratoreye.azurewebsites.net
App Service plan/pricing tier
Refrigeratoreye (Free)
FTP/Deployment username
No FTP/deployment user set
FTP hostname
ftp://waws-prod-db3-013.ftp.azurewebsites...
FTPS hostname
ftps://waws-prod-db3-013.ftp.azurewebsite...

To Create backend tables for data storage: Go to the **mobile service backend**, Select **All settings**, **Easy tables**, and add your tables.

Working online on the backend





	Name	Modified	Size
	data	6/29/2016, 6:59:40 PM	
	LogFiles	7/2/2016, 2:40:06 PM	
	site	6/29/2016, 3:25:58 AM	

```
Kudu Remote Execution Console
Type 'exit' then hit 'enter' to get a new CMD process.
Type 'cls' to clear the console

Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

D:\home>
```

Backend Code

```
// ----------------------------------------------------------------------------
// Refrigerator eye.
// ----------------------------------------------------------------------------
// This is a base-level Azure Mobile App SDK.
var express = require('express'),
    azureMobileApps = require('azure-mobile-apps');
// Set up a standard Express app
var app = express();
// If you are producing a combined Web + Mobile app, then you should handle
// anything like logging, registering middleware, etc. here
// Configuration of the Azure Mobile Apps can be done via an object, the
// environment or an auxiliary file. For more information, see
// http://azure.github.io/azure-mobile-apps-node/global.html#configuration
var mobileApp = azureMobileApps({
// Explicitly enable the Azure Mobile Apps home page
        homePage: true,
// Explicitly enable swagger support. UI support is enabled by
// installing the swagger-ui npm module.
        swagger: true
});
// Import the files from the tables directory to configure the /tables endpoint
mobileApp.tables.import('./tables');
// Import the files from the api directory to configure the /api endpoint
mobileApp.api.import('./api');
// Initialize the database before listening for incoming requests
// The tables.initialize() method does the initialization asynchronously
// and returns a Promise.
mobileApp.tables.initialize()
    .then(function () {
    app.use(mobileApp); // Register the Azure Mobile Apps middleware
    app.listen(process.env.PORT || 3000); // Listen for requests
});
```

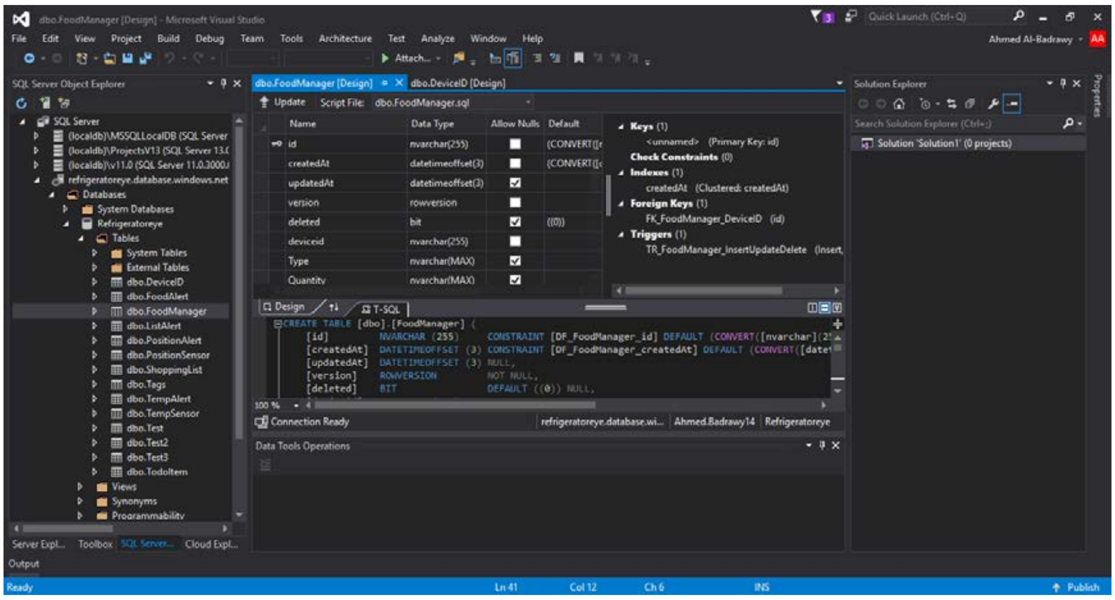
Table Sample Code

```
var azureMobileApps = require('azure-mobile-apps'),
promises = require('azure-mobile-apps/src/utilities/promises'),
logger = require('azure-mobile-apps/src/logger');
var table = azureMobileApps.table();
table.insert(function (context) {
// For more information about the Notification Hubs JavaScript SDK,
// see http://aka.ms/nodejshubs
logger.info('Running TodoItem.insert');
// Define the GCM payload.
var payload = {
      "data": {
          "message": context.item.text
       }
};
// Execute the insert. The insert returns the results as a Promise,
// Do the push as a post-execute action within the promise flow.
return context.execute()
          .then(function (results) {
                 // Only do the push if configured
                 if (context.push) {
                     // Send a GCM native notification.
                     context.push.gcm.send(null, payload, function (error) {
                          if (error) {
                               logger.error('Error while sending push notification: ', error);
                          } else {
                               logger.info('Push notification sent successfully!');
                           }
                     });
              }
             // Don't forget to return the results from the context.execute()
              return results;
        })
        .catch(function (error) {
             logger.error('Error while running context.execute: ', error);
        });
});
module.exports = table;
```

SQL Tables

```
CREATE TABLE [dbo].[DeviceID] (
    [id]         NVARCHAR (255)     CONSTRAINT [DF_DeviceID_id] DEFAULT
(newid()) NOT NULL,
    [createdAt] DATETIMEOFFSET (3) CONSTRAINT [DF_DeviceID_createdAt] DEFAULT
(CONVERT([datetimeoffset](3),sysutcdatetime(),(0))) NOT NULL,
    [updatedAt] DATETIMEOFFSET (3) NULL,
    [version]   ROWVERSION         NOT NULL,
    [deleted]   BIT                DEFAULT ((0)) NULL,
    PRIMARY KEY NONCLUSTERED ([id] ASC)
);
```

```
GO
CREATE CLUSTERED INDEX [createdAt]
    ON [dbo].[DeviceID]([createdAt] ASC);

GO
CREATE TRIGGER [TR_DeviceID_InsertUpdateDelete] ON [dbo].[DeviceID]
                 AFTER INSERT, UPDATE, DELETE
              AS
              BEGIN
                     SET NOCOUNT ON;
                     IF TRIGGER_NESTLEVEL() > 3 RETURN;

                     UPDATE [dbo].[DeviceID] SET [dbo].[DeviceID].[updatedAt]
= CONVERT (DATETIMEOFFSET(3), SYSUTCDATETIME())
                     FROM INSERTED
                     WHERE INSERTED.id = [dbo].[DeviceID].[id]
              END
-----------------------------------------------------
CREATE TABLE [dbo].[FoodManager] (
    [id]          NVARCHAR (255)     CONSTRAINT [DF_FoodManager_id] DEFAULT
(CONVERT([nvarchar](255),newid(),(0))) NOT NULL,
    [createdAt]  DATETIMEOFFSET (3) CONSTRAINT [DF_FoodManager_createdAt]
DEFAULT (CONVERT([datetimeoffset](3),sysutcdatetime(),(0))) NOT NULL,
    [updatedAt]  DATETIMEOFFSET (3) NULL,
    [version]    ROWVERSION         NOT NULL,
    [deleted]    BIT                DEFAULT ((0)) NULL,
    [deviceid]   NVARCHAR (255)     NOT NULL,
    [Type]       NVARCHAR (MAX)     NULL,
    [Quantity]   NVARCHAR (MAX)     NULL,
    [Calories]   NVARCHAR (MAX)     NULL,
    [Protein]    NVARCHAR (MAX)     NULL,
    [Fat]        NVARCHAR (MAX)     NULL,
    [Carbs]      NVARCHAR (MAX)     NULL,
    [Sugar]      NVARCHAR (MAX)     NULL,
    [Sodium]     NVARCHAR (MAX)     NULL,
    [Vitamin_C]  NVARCHAR (MAX)     NULL,
    [Strat_Date] NVARCHAR (MAX)     NULL,
    [End_Date]   NVARCHAR (MAX)     NULL,
    [Start_Date] NVARCHAR (MAX)     NULL,
    PRIMARY KEY NONCLUSTERED ([id] ASC),
    CONSTRAINT [FK_FoodManager_DeviceID] FOREIGN KEY ([deviceid]) REFERENCES
[dbo].[DeviceID] ([id])
);
GO
CREATE CLUSTERED INDEX [createdAt]
    ON [dbo].[FoodManager]([createdAt] ASC);

GO
CREATE TRIGGER [TR_FoodManager_InsertUpdateDelete] ON [dbo].[FoodManager]
                 AFTER INSERT, UPDATE, DELETE
              AS
              BEGIN
                     SET NOCOUNT ON;
                     IF TRIGGER_NESTLEVEL() > 3 RETURN;

                     UPDATE [dbo].[FoodManager] SET
[dbo].[FoodManager].[updatedAt] = CONVERT (DATETIMEOFFSET(3),
SYSUTCDATETIME())
                     FROM INSERTED
                     WHERE INSERTED.id = [dbo].[FoodManager].[id]
              END
```



Name	Data Type	Allow Nulls	Default
id	nvarchar(255)	☐	(CONVERT([r
createdAt	datetimeoffset(3)	☐	(CONVERT([c
updatedAt	datetimeoffset(3)	✓	
version	rowversion	☐	
deleted	bit	✓	((0))
deviceid	nvarchar(255)	☐	
Type	nvarchar(MAX)	✓	
Quantity	nvarchar(MAX)	✓	

Keys (1)
<unnamed> (Primary Key: id)
Check Constraints (0)
Indexes (1)
createdAt (Clustered: createdAt)
Foreign Keys (1)
FK_FoodManager_DeviceID (id)
Triggers (1)
TR_FoodManager_InsertUpdateDelete (Insert,

```
CREATE TABLE [dbo].[FoodManager] (
    [id]        NVARCHAR (255)     CONSTRAINT [DF_FoodManager_id] DEFAULT (CONVERT([nvarchar](2
    [createdAt] DATETIMEOFFSET (3) CONSTRAINT [DF_FoodManager_createdAt] DEFAULT (CONVERT([date
    [updatedAt] DATETIMEOFFSET (3) NULL,
    [version]   ROWVERSION         NOT NULL,
    [deleted]   BIT                DEFAULT ((0)) NULL,
```

4.3.4 Stream Analytics

Azure Stream Analytics (ASA) is a fully managed, cost effective real-time event processing engine that helps to unlock deep insights from data. Stream Analytics makes it easy to set up real-time analytic computations on data streaming from devices, sensors, web sites, social media, applications, infrastructure systems, and more.

With a few clicks in the Azure portal, you can author a Stream Analytics job specifying the input source of the streaming data, the output sink for the results of your job, and a data transformation expressed in a SQL-like language. You can monitor and adjust the scale/speed of your job in the Azure portal to scale from a few kilobytes to a gigabyte or more of events processed per second.

Stream Analytics leverages years of Microsoft Research work in developing highly tuned streaming engines for time-sensitive processing, as well as language integrations for intuitive specifications of such.

What can I use Stream Analytics for?

Vast amounts of data are flowing at high velocity over the wire today. Organizations that can process and act on this streaming data in real time can dramatically improve efficiencies and differentiate themselves in the market. Scenarios of real-time streaming analytics can be found across all industries: personalized, real-time stock-trading analysis and alerts offered by financial services companies; real-time fraud detection; data and identity protection services; reliable ingestion and analysis of data generated by sensors and actuators embedded in physical objects (Internet of Things, or IoT); web clickstream analytics; and customer relationship management (CRM) applications issuing alerts when customer experience within a time frame is degraded. Businesses are looking for the most flexible, reliable and cost-effective way to do such real-time event-stream data analysis to succeed in the highly competitive modern business world.

Create a Stream Analytics job

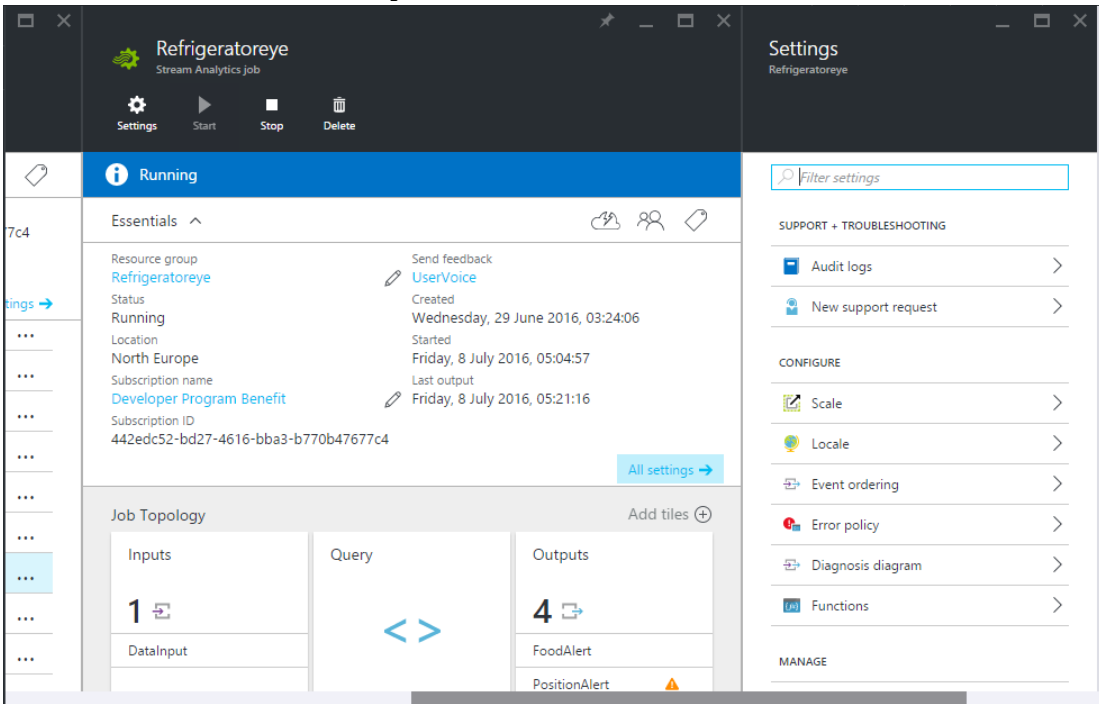
- Log on to the Microsoft Azure Portal

- In the menu, click **New**, then click **Internet of Things**, and then click **Stream Analytics Job**
- Enter a name for the job (We chose “PiStorageJob”), a preferred region, then choose your subscription. At this stage you are also offered to create a new resource group or to use an existing resource group. Choose the resource group you created earlier (In our case, Pi2Suite).
- Once the job is created, open your **Job’s blade** or click on the **pinned tile**, and find the section titled “*Job Topology*” and click the **Inputs** tile. In the Inputs blade, click on **Add**
- Enter the following settings:
    - ✓ Input Alias = *DataInput*
    - ✓ Source Type = *Data Stream*
    - ✓ Source = *IoT Hub*
    - ✓ IoT Hub = *raspPiIoT* (use the name for the IoT Hub you create before)
    - ✓ Shared Access Policy Name = *iothubowner*
    - ✓ Shared Access Policy Key = *The iothubowner primary key can be found in your IoT Hub Settings -> Shared access policies*
    - ✓ IoT Hub Consumer Group = "" (leave it to the default empty value)
    - ✓ Event serialization format = *JSON*
    - ✓ Encoding = *UTF-8*
- Back to the **Stream Analytics Job blade**, click on the **Query tile** (next to the Inputs tile). In the Query settings blade, type in the below query and click **Save**:

```
SELECT
   DeviceID,Temp
INTO
   [TempAlert]
FROM
   [DataInput]
WHERE
   Temp > 6

SELECT
   DeviceID,Position
INTO
   [PositionAlert]
FROM
   [DataInput]
WHERE
   Position < 10

SELECT
   DeviceID,Type,Quantity,Start_Date,End_Date
INTO
   [FoodAlert]
FROM
   [DataInput]
WHERE
    Quantity < 0
```

```
SELECT
   Temp,Position
INTO
   [PowerBI]
FROM
   [DataInput]
```

- Back to the **Stream Analytics Job blade**, click on the **Outputs** tile and in the **Outputs blade**, click on **Add**
- Enter the following settings then click on **Create**:
  - ✓ Output Alias = PositionAlert
  - ✓ Sink = *SQL Databse*
  - ✓ Subscription = *choose subscription name where the SQL Database exists*
  - ✓ Database = *choose data base*
  - ✓ Server name = *baste database server url*
  - ✓ Username = Username
  - ✓ Password = Password
  - ✓ Table = Existing Table Name
- Back to the **Stream Analytics Job blade**, click on the **Outputs tile**, and in the **Outputs blade**, click on **Add**
- Enter the following settings then click on **Create**:
  - ✓ Output Alias = PowerBI
  - ✓ Source = Power BI
  - ✓ Authorization = *Username & Password*
  - ✓ Dataset Name = *Create new dataset*
  - ✓ Table Name = *Create new table*
- Back in the** Stream Analytics blade**, start the job by clicking on the** Start **button at the top


Refrigeratoreye
Stream Analytics job
Settings Start Stop Delete
Running
Essentials
Resource group: Refrigeratoreye
Status: Running
Location: North Europe
Subscription name: Developer Program Benefit
Subscription ID: 442edc52-bd27-4616-bba3-b770b47677c4
Send feedback: UserVoice
Created: Wednesday, 29 June 2016, 03:24:06
Started: Friday, 8 July 2016, 05:04:57
Last output: Friday, 8 July 2016, 05:21:16
All settings
Job Topology — Add tiles
Inputs: 1 — DataInput
Query
Outputs: 4 — FoodAlert, PositionAlert

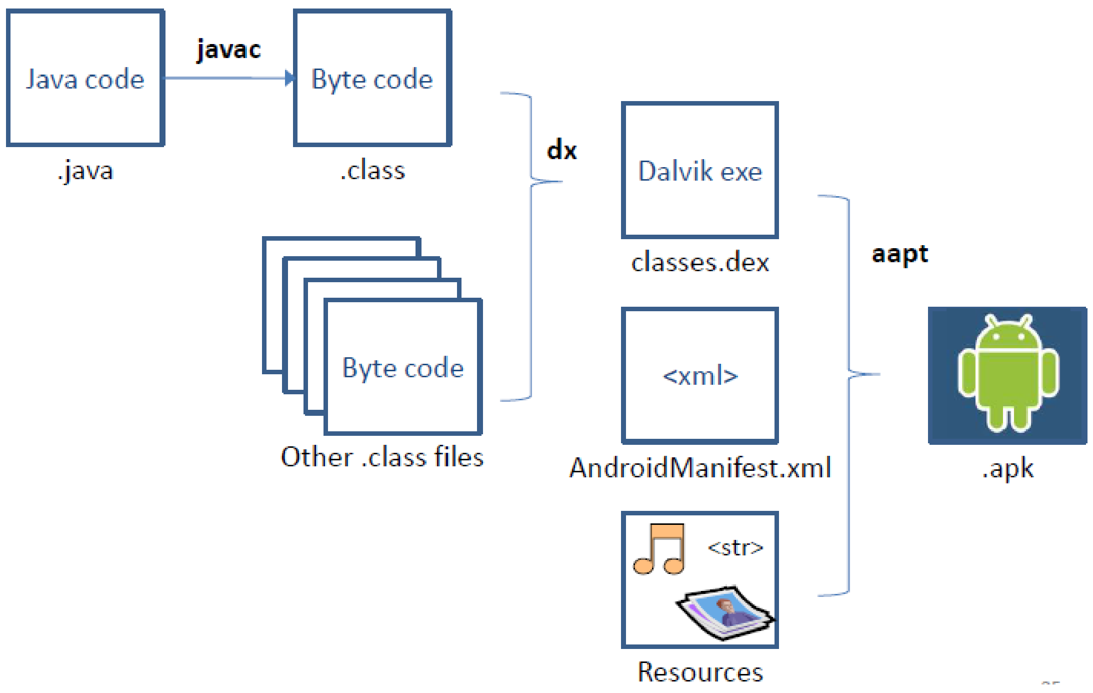
Settings
Refrigeratoreye
Filter settings
SUPPORT + TROUBLESHOOTING: Audit logs, New support request
CONFIGURE: Scale, Locale, Event ordering, Error policy, Diagnosis diagram, Functions
MANAGE

Chapter 5
Mobile App

5 Overview

5.1 Introduction

If you've got a burning idea for an application that you're dying to share, or if you recognize the power and possibilities of the Android platform, you've come to the right place. In exchange for its difficulty, however, Google's Android offers unprecedented power, control, and—yes—responsibility to those who are brave enough to develop for it. This is where our job comes in. We're here to make the process of learning to write amazing Android software as simple as possible. We all know that people with ample facial hair appear to be more authoritative on all subjects. Kevin Grant has been developing for Android since its inception and has worked on a breadth of user-facing products, developing beautiful and intuitive interfaces for millions of users. While he doesn't have a beard, we all know that people with a perpetual five o'clock shadow know how to get things done. From here on out, we're going to take this conversation into the first person. We banter enough amongst ourselves—it's not necessary to confuse you in the process. So without further ado, in return for making this learning process as easy as possible, the goal of this chapter is to teach the skills necessary to develop Android based applications using the Android Studio Integrated Development Environment (IDE) and the Android 5 Software Development Kit (SDK). Beginning with the basics, this chapter provides an outline of the steps necessary to set up an Android development and testing environment. An overview of Android Studio is included covering areas such as tool windows, the code editor and the Designer tool. An introduction to the architecture of Android is followed by an in-depth look at the design of Android applications and user interfaces using the Android Studio environment. More advanced topics such as database management, content providers and intents are also covered, as are touch screen handling, gesture recognition, camera access and the playback and recording of both video and audio. Before anything you must have the full picture of how android system works as shown in Fig (1.0)



5.2 Programming language

In this section, we provide brief comments on several popular programming languages in the next section we introduce Java: as shown in table 1.1

Programming language	Description
C++	C++, an extension of C, was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories. C++ provides a number of features that “spruce up” the C language, but more important, it provides capabilities for object-oriented programming.
Objective-C	Objective-C is an object-oriented language based on C. It was developed in the early 1980s and later acquired by Next, which in turn was acquired by Apple. It has become the key programming language for the Mac OS X oper-ating system and all iOS-powered devices (such as iPods, iPhones and iPads).
Visual Basic	Microsoft’s Visual Basic language was introduced in the early 1990s to sim-plify the development of Microsoft Windows applications. Its latest versions support object-oriented programming.
Visual C#	Microsoft’s three primary object-oriented programming languages are Visual Basic, Visual C++ (based on C++) and C# (based on C++ and Java, and devel-oped for integrating the Internet and the web into computer applications).
PHP	PHP is an object-oriented, “open-source” (see Section 1.7) “scripting” language supported by a community of users and developers and is used by numerous websites including Wikipedia

	and Facebook. PHP is platform independent—implementations exist for all major UNIX, Linux, Mac and Windows operat-ing systems. PHP also supports many databases, including MySQL.
Python	Python, another object-oriented scripting language, was released publicly in 1991. Developed by Guido van Rossum of the National Research Institute for Mathematics and Computer Science in Amsterdam (CWI), Python draws heavily from Modula-3—a systems programming language. Python is “exten-sible”—it can be extended through classes and programming interfaces.
JavaScript	JavaScript is the most widely used scripting language. It’s primarily used to add programmability to web pages—for example, animations and interactiv-ity with the user. It’s provided with all major web browsers.
Fortan	Fortran (Formula Translator) was developed by IBM Corporation in the mid-1950s to be used for scientific and engineering applications that require complex mathematical computations. It’s still widely used and its latest versions are object oriented.
C	C was implemented in 1972 by Dennis Ritchie at Bell Laboratories.Itini-tially became widely known as the UNIX operating system’s development language. Today, most of the code for general-purpose operating systems is written in C or C++.

5.3 Meeting java

Welcome to Java—the world’s most widely used computer programming language. You’re already familiar with the powerful tasks computers perform. Java has also become the language of choice for implementing Internet-based applications and software for devices that communicate over a network. In use today are more than a billion general-purpose computers and billions more ava-enabled cell phones, smartphones and handheld devices (such as tablet computers). there are some reasons for using java as a software language:

- "simple, object-oriented, and familiar".
- "robust and secure".
- "architecture-neutral and portable “execute with "high performance".
- "interpreted, threaded, and dynamic".

5.3.1 Java platform

One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate runtime support. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to architecture-specific machine code. Java bytecode instructions are analogous to machine code, but they are intended to be executed by a virtual machine (VM) written specifically for the host hardware. End users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a web browser for Java applets. Standard libraries provide a generic way to access host-specific features such as graphics, threading, and networking. The use of universal bytecode makes porting simple. However, the overhead of interpreting bytecode into machine instructions makes interpreted programs almost always run more slowly than native executables. However, just-in-time (JIT) compilers that compile bytecodes to machine code during runtime were introduced from an early stage. Java itself is platform-independent, and is adapted to the particular platform it is to run on by a Java virtual machine for it, which translates the Java bytecode into the platform's machine language.

5.3.2 Implementations

Oracle Corporation is the current owner of the official implementation of the Java SE platform, following their acquisition of Sun Microsystems on January 27, 2010. This implementation is based on the original implementation of Java by Sun. The Oracle implementation is available for Microsoft Windows (still works for XP, while only later versions currently "publicly" supported), Mac OS X, Linux and Solaris. Because Java lacks any formal standardization recognized by Ecma International, ISO/IEC, ANSI, or other third-party standards organization, the Oracle implementation is the de facto standard. The Oracle implementation is packaged into two different distributions: The Java Runtime Environment (JRE) which contains the parts of the Java SE platform required to run Java programs and is intended for end users, and the Java Development Kit (JDK), which is intended for software developers and includes development tools such as the Java compiler, Javadoc, Jar, and a debugger. OpenJDK is another notable Java SE implementation that is licensed under the GNU GPL. The implementation started when Sun began releasing the Java source code under the GPL. As of Java SE 7, Open JDK is the official Java reference implementation. The goal of Java is to make all implementations of Java compatible.

5.3.3 Performance

Programs written in Java have a reputation for being slower and requiring more memory than those written in C++.the addition of language features supporting better code analysis (such as inner classes, the String Builder class. With Java 1.5, the performance was improved with the addition of the java.util.concurrent package, including Lock free implementations of the ConcurrentMaps and other multi-core collections, and it was improved further Java 1.6.Some platforms offer direct hardware support for Java; there are microcontrollers that can run Java in hardware instead of a software Java virtual machine, and ARM based processors can have hardware support for executing Java bytecode through their Jazelle option (while its support is mostly dropped in current implementations of ARM).

5.4 How java and android work together

After we write a program in Java for Android, we click on a button to change our code into another form that is understood by Android. This other form is called **Dalvik Executable** (**DEX**) code, and the transformation process is called **compiling**. Compiling takes place on the development machine after we click on that button. We will see this work right after we set up our development environment. Android is a fairly complex system, but you do not need to understand it in depth to be able to make amazing apps. The part of the Android system that executes (runs) our compiled DEX code is called the Dalvik Virtual Machine (DVM). The DVM itself is a piece of software written in another language that runs on a specially adapted version of the Linux operating system. So what the user sees of Android, is itself just an app running on another operating system. The purpose of the DVM is to hide the complexity and diversity of the hardware and software that Android runs on but, at the same time, its purpose is to expose all of its useful features. This exposing of features generally works in two ways. The DVM itself must have access to the hardware, which it does, but this access must be programmer friendly and easy to use. The way the DVM allows us access is indeed easy to use because of the Android Application Programming Interface (API). (1).

5.4.1 The Android API

The Android API is the code that makes it really easy to do exceptional things. A simple analogy could be drawn with a machine, perhaps a car. When you step on the accelerator, a whole bunch of things happen under the hood. We don't need to understand about combustion or fuel pumps because a smart engineer has provided an interface for us. In this case, a mechanical interface—the accelerator pedal. Take the following line of Java code as an example; it will probably look a little intimidating if you are completely new to Android: locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);However, once you learn that this single line of code searches for the available satellites and then communicates with them in orbit around the Earth while retrieving your precise latitude and longitude on the planet, it is easy to begin to glimpse the power and depth of the Android API in conjunction with the DVM . Even if that code does look a little challenging at the moment, imagine talking to satellite in some other way!

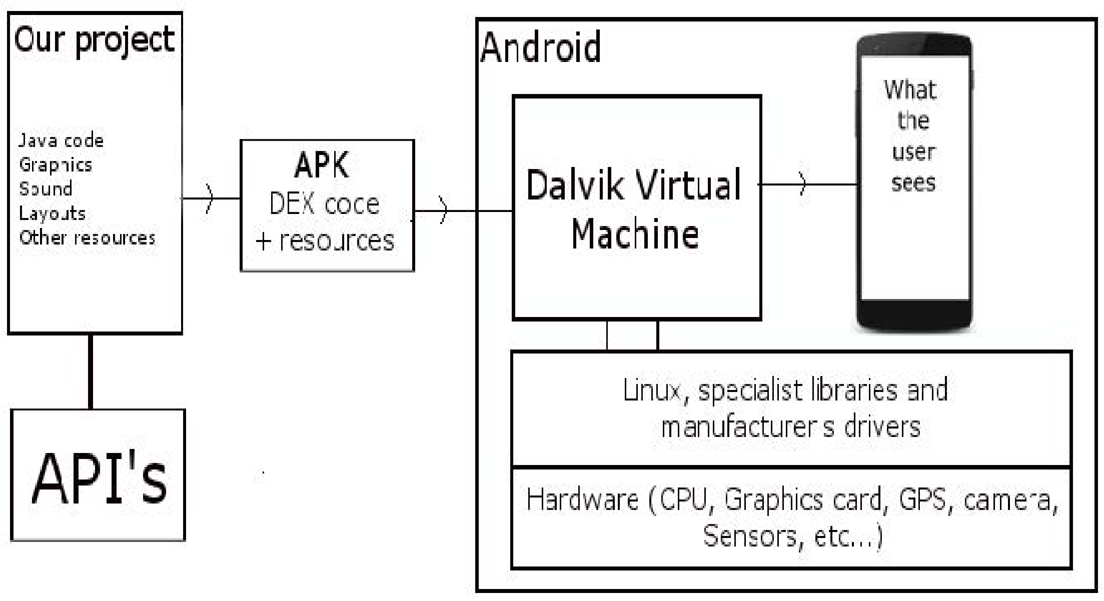
5.4.2 Java is object-oriented

Java is a programming language that has been around a lot longer than Android. It is an object-oriented language. This means that it uses the concept of reusable programming objects. If this sounds like technical jargon, another analogy will help. Java enables us and others (such as the Android development team) to write Java code that can be structured based on real-world "things" and, here is the important part, it can be reused. So, using the car analogy, we could ask the question: if a manufacturer makes more than one car in a day, do they redesign each and every part for each and every car? The answer, of course, is no. They get highly skilled engineers to develop exactly the right components that are honed, refined, and improved over years. Then, that same component is reused again and again, as well as occasionally improved. Now, if you are going to be picky about my analogy, then you can argue that each of the car's components still have to be built from raw materials using real-life engineers, or robots, and so on. This is true. What the software engineers actually do when they write their code is build a blueprint for an object. We then create an object from their blueprint using Java code and, once we have that object, we can configure it, use it, combine it with other objects, and more. Furthermore, we can design blueprints and make objects from them as well. The compiler then translates (manufactures) our custom-built creation into DEX code. In Java, a blueprint is called a class. When a class is transformed into a real working thing, we call it an object.

5.4.3 What exactly is android?

Android is the name of the mobile operating system owned by American company; Google. It most commonly comes installed on a variety of smartphones and tablets from a host of manufacturers offering users access to Google's own services like Search, YouTube, Maps, Gmail and more. This means you can easily look for information on the web, watch videos, search for directions and write emails on your phone, just as you would on your computer, but there's more to Android than these simple examples. We know that to get things done on Android, we write Java code of our own, which also uses the Java code of the Android API. This is then compiled into DEX code and run by the DVM, which in turn has connections to an underlying operating system called Linux. Then the manufacturers of the Android devices and individual hardware components write advanced software called drivers, which ensure that their hardware (CPU, GPU, GPS receivers, and so on) can run on the underlying Linux operating system. Our compiled Java code, along with some other resources, is placed in a bundle of files called an Android application package (APK), and this is what the DVM needs

to run our app. this process is explained in the following figure



5.4.4 The development environment

Java Development Kit (JDK) is a program development environment for writing Java applets and applications. It consists of a runtime environment that "sits on top" of the operating system layer as well as the tools and programming that developers need to compile, debug, and run applets and applications written in the Java language. A development environment is a term that refers to having very thing you need in order to develop, set up, and be ready to go in one place. We need the following two things to get started:

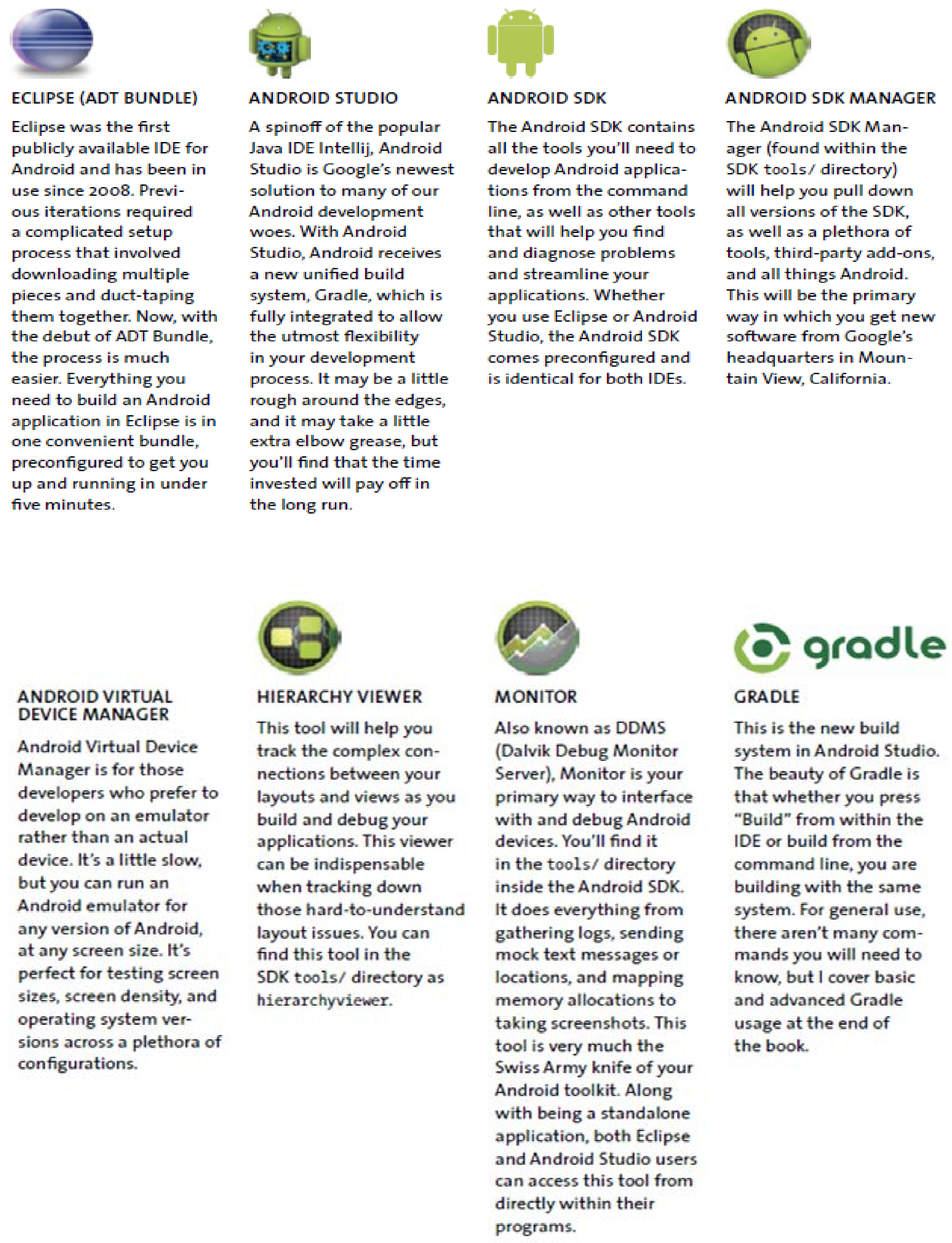
- ❖ We talked a fair bit about compiling our Java code, as well as other people's Java code, into DEX code that will run on the DVM, on people's Android devices. In order to use Java code, we need a free software called the JDK. The JDK also includes other people's code, which is separate from the Android API.
- ❖ There is a whole range of tools that are required to develop for Android, and we also need the Android API, of course. This whole suite of requirements is collectively known as the Android software development kit (SDK). Fortunately, downloading and installing a single application will give us these things all bundled together. This single application is called Android Studio.

Installing the JDK

As a little bit of preparation before we install the JDK, you need to know which operating system you have and whether it is 32 or 64 bit. If you are unsure, use this little tip to find out. Now we are ready to install the JDK. This fairly simple set of steps will set up the JDK quickly. The only slight delay is the download itself, which could take a while on slower Internet connections. The actual installation process should be fast and trouble free by some steps in this source (4)

5.5 Welcome to android

Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. It has been the best-selling OS on tablets and on smartphones since 2013, and has the largest installed base. Android’s user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping. In addition to touchscreen devices, Google has further developed Android for television, cars, and wristwatches, each with a specialized yet similar interface. Applications are usually developed in Java programming language using the Android software development kit (SDK), but other development environments are also available. Eclipse and Android Studio are the two supported integrated development environments (IDEs) for Android development. Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing. There are, however, a few other tools you should be aware of that will be very useful now and in your future work with Android. While you may not use all these tools until you’re getting ready to ship an application, it will be helpful to know about them when the need arises Fig (1.2) & Fig (1.3)



ECLIPSE (ADT BUNDLE)

Eclipse was the first publicly available IDE for Android and has been in use since 2008. Previous iterations required a complicated setup process that involved downloading multiple pieces and duct-taping them together. Now, with the debut of ADT Bundle, the process is much easier. Everything you need to build an Android application in Eclipse is in one convenient bundle, preconfigured to get you up and running in under five minutes.

ANDROID STUDIO

A spinoff of the popular Java IDE Intellij, Android Studio is Google's newest solution to many of our Android development woes. With Android Studio, Android receives a new unified build system, Gradle, which is fully integrated to allow the utmost flexibility in your development process. It may be a little rough around the edges, and it may take a little extra elbow grease, but you'll find that the time invested will pay off in the long run.

ANDROID SDK

The Android SDK contains all the tools you'll need to develop Android applications from the command line, as well as other tools that will help you find and diagnose problems and streamline your applications. Whether you use Eclipse or Android Studio, the Android SDK comes preconfigured and is identical for both IDEs.

ANDROID SDK MANAGER

The Android SDK Manager (found within the SDK `tools/` directory) will help you pull down all versions of the SDK, as well as a plethora of tools, third-party add-ons, and all things Android. This will be the primary way in which you get new software from Google's headquarters in Mountain View, California.

ANDROID VIRTUAL DEVICE MANAGER

Android Virtual Device Manager is for those developers who prefer to develop on an emulator rather than an actual device. It's a little slow, but you can run an Android emulator for any version of Android, at any screen size. It's perfect for testing screen sizes, screen density, and operating system versions across a plethora of configurations.

HIERARCHY VIEWER

This tool will help you track the complex connections between your layouts and views as you build and debug your applications. This viewer can be indispensable when tracking down those hard-to-understand layout issues. You can find this tool in the SDK `tools/` directory as `hierarchyviewer`.

MONITOR

Also known as DDMS (Dalvik Debug Monitor Server), Monitor is your primary way to interface with and debug Android devices. You'll find it in the `tools/` directory inside the Android SDK. It does everything from gathering logs, sending mock text messages or locations, and mapping memory allocations to taking screenshots. This tool is very much the Swiss Army knife of your Android toolkit. Along with being a standalone application, both Eclipse and Android Studio users can access this tool from directly within their programs.

GRADLE

This is the new build system in Android Studio. The beauty of Gradle is that whether you press "Build" from within the IDE or build from the command line, you are building with the same system. For general use, there aren't many commands you will need to know, but I cover basic and advanced Gradle usage at the end of the book.

5.5.1 Getting started Setting up Android Studio

Now that the JDK is installed and ready to go, we are only one step away from building our first Android app. Installing Android Studio can take a bit longer than the JDK and is a little more nuanced, but it is nothing a determined, aspiring, developer won't be able to handle with ease. Now that we know what to expect, we can get on with the installation of Android Studio. Follow the given steps from this source book (2)

5.6 What makes an Android app

We already know that we will write Java code that will itself use other people's Java code and will be compiled into DEX code that runs on the DVM. In addition to this, we will also be adding and editing other files as well. These files are known as Android resources.

5.6.1 Android resources

Our app will include resources such as images, sounds, and user interface layouts that are kept in separate files from the Java code. They will also include files that contain the textual content of our app. It is a convention to refer to the text in our app through separate files because it makes them easy to change, and this makes it easy to create apps that work for multiple different languages. Furthermore, the actual UI layouts of our apps, despite the option to implement them with a visual designer, are actually read from text-based files by Android. Android (or any computer), of course, cannot read and recognize text in the same way that a human can. Therefore, we must present our resources in a highly organized and predefined manner. To do so, we will use Extensible Markup Language (XML). XML is a huge topic but, fortunately, its whole purpose is to be both human and machine readable. We do not need to learn this language, we just need to observe (and then conform to) a few rules. Furthermore, most of the time when we interact with XML, we will do so through a neat visual editor provided by Android Studio. We can tell when we are dealing with an XML resource because the filename will end with the .xml extension.

5.6.2 The structure of Android's Java code

In addition to these resources, it is worth noting that Java, as used in Android, has a structure to its code. There are many millions of lines of code that we can take advantage of. This code will obviously need to be organized in a way that makes it easy to find and refer to. It is organized under predefined packages that are specific to Android.

5.6.3 Android packages

Whenever we create a new Android app, we will choose a unique name known as a package. We will see how to do this in the Our first Android app section. Packages are often separated into sub packages, so they can be grouped together with other similar packages. We can simply think of these as folders and subfolders. We can also think of all the packages that the Android API makes available to us as books that contain code, from a library. Some common Android packages we will use include the following:

• android. Graphics

• android. Database

• android. view. Animation

As you can see, they are arranged and named to make what is contained in them as obvious as possible. Earlier, we learned that reusable code blueprints that we can transform into objects are called classes. Classes are contained in these packages. We will see in our very first app how to easily import other people's packages along with specific classes from those packages for use in our projects. A class will almost always be contained in its own file, with the same name as the class, and have the .java file extension. In Java, we further break up our classes into sections that perform the different actions for our class. We call these sections methods. These are, most often, the methods of the class that we will use to access the functionality provided within all those millions of lines of code. We do not need to read the code. We just need to know which class does what we need, which package it is in, and which methods from within the class give us precisely the results we are after. The **next diagram** shows a representation of the Android API. We can think about the structure of the code that we will write in exactly the same way, although we will most likely have just one package per app. Of course, because of the object-oriented nature of Java, we will only be using selective parts from this API. Also note that each class has its own distinct data. Typically, if you want access to the data in a class, you need to have an object of that class.

5.7 Android UI Overview

5.7.1 Android App Structure and UI patterns.

Introduction

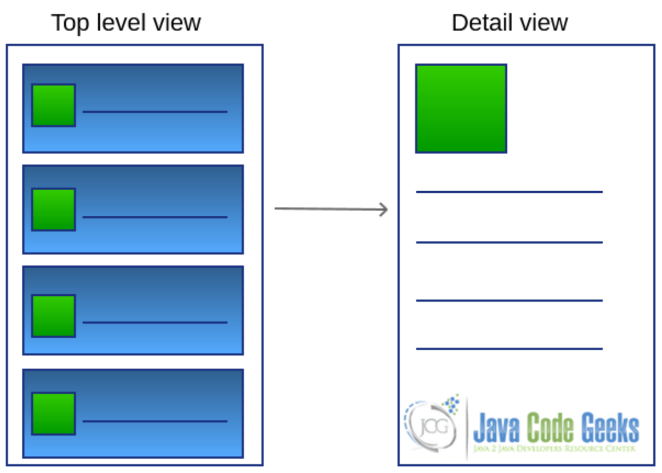
Android is a widely used OS made for smart phones and tablets. It is an open source project led by Google and it is released under Apache License. This permissive license helped this OS to be widely adopted and allows the manufacturers to freely modify and customize it. As matter of fact, despite Android being designed for smartphones and tablets, it is also used in TVs, cameras and so on. Moreover, Android has a very large community that extend its features and creates apps that cover almost all aspects. All android applications, called apps, are built on Android UI framework. App interface is the first thing a user sees and interacts with. From the user perspective, this framework keeps the overall experience consistent for every app installed in our smartphone or tablets. At the same time, from the developer perspective, this framework provides some basic blocks that can be used to build complex and consistent user interface (API).
Android UI interface is divided in three different areas:

• Home screen
• All apps
• Recent screen

The home screen is the "landing" area when we power our phone on. This interface is highly customizable and themed. Using widgets we can create and personalize our "home" screen. All apps is the interface where the app installed are displayed, while recent screens are the list of last used apps. Since its born, Android has changed a lot in terms of its features and its interfaces.

App Structure and UI patterns

Android apps are very different from each other because they try to address different user needs. There are simple apps with a very simple UI that has just only one view and there are other apps much more complex with a very structured navigation and multiple views.
In general, we can say an Android app is made by a top-level view and detail/level view.



5.7.1.1 Top level view

As said, the top level view is the "landing" area of our app, so we have to reserve to it a special attention, because this is the first thing a user sees of our app. There are some specific patterns that can be applied when designing this view depending on the type of information we want to show:

• Fixed tabs
• Spinner
• Navigation drawer

We have to choose one of them carefully depending on the nature of our app. We can use fixed tabs when we want to give to the user an overview of the different views present in our app, so that a user can switch easily between them to show different type of information. A typical example is a app that shows tech news: in this case we could use different tabs to group news like ('Android', iOS, 'Games' and so on). Spinner is used when we want to move directly to a specific view, this is the case of a calendar app when we can use spinner to go directly to a specific month. The navigation drawer is one of the newest patterns introduced by Google. This is a sliding menu, usually at the left side of the smartphone screen, that can be opened and closed by the user. This pattern can be used when we have a multiple top level view and we want to give to the user a fast access to one of them, or we want to give to the user the freedom to move to one low level view directly. This pattern replaces, somehow, an old pattern called dashboard widely used in the past. This pattern is simple a view where there are some big buttons/icons to access to specific views/app features.

5.7.1.2 Detail view

The detail view is a low level view where a user can interact with data directly. It is used to show data and edit them. In this kind of view the layout plays an important role to make data well organized and structured. At this level, we can implement an efficient navigation to improve usability of our app. In fact, we can use swipe view pattern so that user can move between different detail views. Depending on the type of component we use to show detail information to the user, we can implement some low level patterns that simplify the user interaction with our app.

5.7.1.3 Action Bar

The action bar is relatively new in Android and was introduced in Android 3.0 (API level 11). It is a well known pattern that plays an important role. An action bar is a piece of the screen, usually at the top, that is persistent across multiple views. It provides some key functions:

• App branding: icon area

• Title area
• Key action area
• Menu area

5.7.2 Android UI: Understanding Views

In this article, we want to explore more about Views and how we can use them to build great user interfaces. This article will cover concepts about Views and Adapters. Developing a user interface in Android can be quite simple since we can describe them using XML files and the system will do the heavy work converting those in real user interface components, placing them according to the screen size and resolution. The Android UI framework provides some basic UI components, that are called controls or widgets, helping developers while coding and building an app.

5.7.2.1 Views

The View class is the basic class that all the components extend. A View draws something on a piece of screen and it is responsible to handle events while user interacts with it. Even the generic View Group class extends View. A View Group is a special View that holds other views and places these views following some rules. We will see that Android provides some specialized views that helps us to handle text, images, user inputs, buttons and so on. All these views have some key aspects in common:
• All views have a set of properties: These properties affect the way the view is rendered. There is a set of properties common to all views, while there are some other properties depending on the type of view.
• Focus: The system manages the focus on each view and depending on the user input, we can modify and force the focus on a specific view.
• Listeners: All views have listeners which are used to handle events when the user interacts with the view. We can register our app to listen to specific events that occur on a view.
• Visibility: We can control if a view is visible or not and we can change the view visibility at runtime too. A view property is a key value pair that we can use to customize the view behavior. The property values can be:
• a number
• a string
• a reference to a value written somewhere else
The first two options are trivial, the third is the most interesting one because we can create a file (always in XML) that holds a list of values and we can reference it in our property value. This is the best approach to follow especially when we use themes and styles or we want to support multi-language apps. One of the most important property is view id: this is a unique identifier for the view and we can look up a specific view using this id. This is a “static” property meaning we cannot change it once we have defined it. When we want to set a view property, we have two ways to do it:
• using XML while we define our view
• programmatically

5.7.2.2 Image View component

This component is used to show an image on the screen. The image we want to show can be placed inside our apk or we can load it remotely. In the first case, our image has to be placed under the res/drawable directory. Base on what we discussed in our previous chapter, we already know we have to keep in mind that our app can run on multipe devices with different screen density. To better support different screen densities, we can create different images with different resolutions. The res/drawable directory is the default directory where the system looks if it cannot find an image with the right density for the screen. Generally speaking, we have to create at least four different image with different resolutions, in fact we can notice in our IDE that there are at least five directories:

• drawable-ldpi (not supported any more)
• drawable-mdpi (medium dpi)
• drawable-hdpi (high dpi)
• drawable-xhdpi (extra-high dpi)
• drawable-xxhdpi (x-extra-high dpi)
• drawable

An important thing to remember is the following: the images must have the same name. Once we have it, we can use the Image View in this way:

```
<ImageView
android:id="@+id/img"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/Refrigertoreye" />
```

If the image has to be loaded remotely, there are some considerations to keep in mind while doing it. First, you should consider that loading an image from a remote server is a time consuming operation and you should do it in a separate thread so that you can avoid ANR (Application Not Responding) problem. If you want to keep things simple you could use an AsyncTask or you can use the Volley lib. The second thing to remember is that you cannot set the image in an XML file, but you can define only the ImageView without referencing an image. In this case, you can set the image using:

```
img.setImageBitmap(your_bitmap)
```

where your_bitmap is the image you have loaded remotely.

5.7.2.3 Input controls

Input controls are components that the user can interact with. Using a component like that, you can, for example, give to the user the chance to insert some values. The Android UI framework provides a wide range of input controls: text field, input field, toggle buttons, radio buttons, buttons, checkbox, pickers and so on. Each of them has a specialized class and we can build complex interfaces using these components. There is a list of common components, below, with the class that handles it:

Component	Description	Class
Button	This one of the most used components. It can be pressed by a user and when pressed we can launch an action.	Button
TextFields	This is an editable text and we give to the user the chance to insert some data. Edit Text is the classic input field while AutoCompleteTextView is a component we can use when we have a pre-defined list of result and we want the system to complete some words inserted by user	EditText AutoCompleteTextView
CheckBox	This is an on/off component. We can use it when we want user to select one or more choices.	CheckBox

Radio buttons	Very similar to the checkbox except for the fact that the user can select only one item.	RadioGroup RadioButton
Toggle button	On/off component with a state indicator.	ToggleButton
Pickers	Component that helps the user to select one value using up/down buttons or gesture. Usually we use DatePicker and TimePicker	DatePicker TimePicker

5.7.3 Layout overview

we can implement when we create UIs so that we can guarantee consistency to our app interface. In this article we want to explore how we can organize such views and how these views can be placed on the screen. In other words, we will analyze in detail layout managers, or simply layouts. When we create our app interface, we use some special view that acts as container. These special views control how other views are placed on the smartphone/tablet screen. Android provides a collection of Layout Managers and each of them implements a different strategy to hold, manage and place its children. From the API point of view, all the Layout managers derive from the ViewGroup class. There are some layouts that place their children horizontally or vertically, and others that implement a different strategy. We will analyze them in details later in this article. In Android, layouts can be nested so we can use different layouts for different areas of our interface. However, please ve aware that it is not advisable to create too complex layouts, because this can affect the overall app performance. We can declare a layout in two ways:

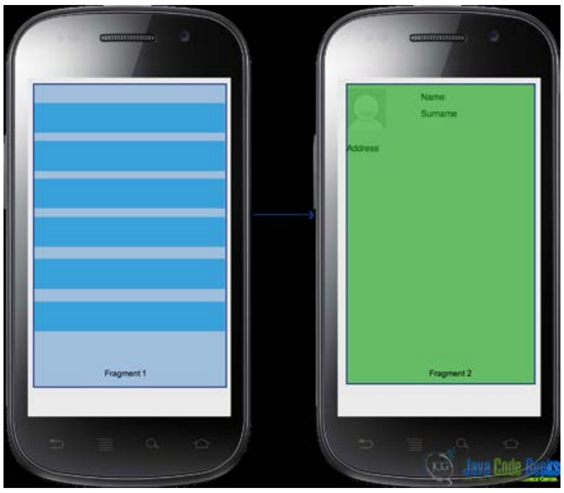
• Using XML: In this case, using an XML file we “describe” how our user interface looks like. We define the elements (views and sub layouts) that appear in the user interface. At the same time, we define their attributes, as we saw in the previous.

• At runtime: In this case, we code our layout instantiating our ViewGroup and attaching Views to it. We can manipulate their properties programmatically setting their attributes. We can use both approaches in our app. We could, for example, use XML to create the user interface and assign to its Views some properties. At run time, we can find (or lookup) this Views and ViewGroup (layout) and change their properties programmatically. We could for example, have a View with red background and at runtime we change it to green color. Android is very powerful and flexible from this point of view. Using XML, we somehow decouple the presentation from the code that handles its behavior. In XML, the UI description is external to the source code, so theoretically we could change the presentation, meaning just the XML file, without touching our source code. This is the case when, for example, we want to adapt our user interface for multiple screen dimensions. In this case, we define different layouts having the same name but in different directories, and the system chooses the one that best matches the screen dimensions. This is the standard approach taken by Android in order to handle multiple screen sizes. Moreover, we will see later that we can use another technique based on Fragments. If we use XML, it is possible to use draw the layout and debug it easily. From the API point of View, each ViewGroup defines a nested class called LayoutParameter that holds some parameters that define size and position for each views belonging to the ViewGroup. All ViewGroup have in common two parameters called width and height (or layout_width and layout_height) that every View must define. These two parameters represent the width and the height of the View. We can specify a numeric value or more often we can use two constants:

• wrap_content, meaning that the dimension of the view will depend on the actual content
• fill_parent (or match_parent), meaning that the view has to become as big as its parent holding it A view in Android is a rectangle, and the view location is expressed as a pair of coordinates left and top. These two values determine the View position inside its ViewGroup. Another important View property inside a Layout is padding, expressed with four values (let, top, right, bottom). Using padding we can move the content of the View. Android provides several standard layout managers:
• Linear Layout
• Table Layout
• Relative Layout
• Frame Layout
• Grid Layout

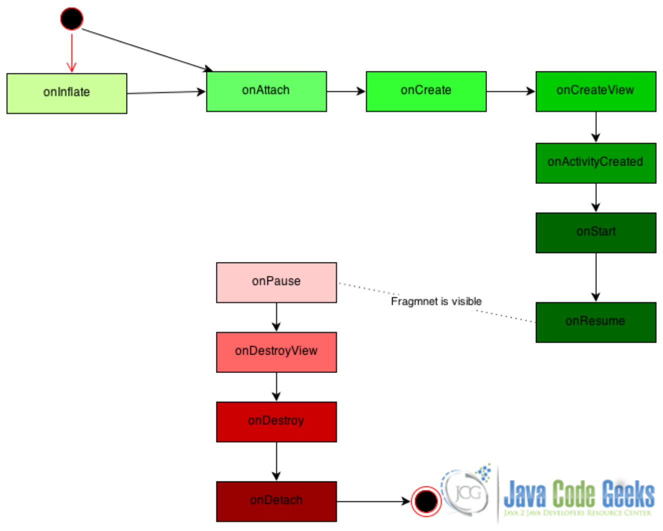
5.7.4 Fragments

By now, we have seen how we can build UIs using layouts and components. In this situation, we have an Activity with its layout and components. We know how we can adapt our layout for multiple screens but this technique sometimes is not enough, especially when we want to support tablets and smartphones. We have talked about this already, and we know that the smartphone screen size is very different from the screen of a tablet. Moreover, it is necessary to make the UI more dynamic and flexible in a large screen tablets. For these reasons, Android 3.0 (API level 11) has introduced the fragments concept. What is a fragment? A fragment is a piece of user interface in an Activity. We can combine multiple fragments to create complex UIs. Each fragment has its own lifecycle and we can manage the lifecycle of each fragment independently. A fragment exists just inside an Activity that acts as its container. When we add a fragment inside a layout, it lives inside the ViewGroup that represents the layout. A fragment is a very powerful component that helps developers to create dynamic UI and support, at the same time, multiple screen size. A fragment can be seen as a reusable piece of code with its own interface. A classic example is an app that has a list of contacts where, when a user clicks on a contact, the app shows the contact's details. In a smartphone we can move from one activity to another showing a different layout, list and details However, in a tablet this behavior would result in a poorly appealing app that does not use all the screen size available. In the tablet, we would like to have a list and the details at the same time on the screen.



5.7.4.1 *Fragment lifecycle*

Now that we know when we should use fragments, we need to know how they work before using them. A fragment has its own lifecycle as an Activity has, but it is more complex in comparison to the activity lifecycle. Moreover, a fragment lives only inside an Activity that acts as its container. Below, the fragment lifecycle is shown:



As we can see, this lifecycle has more states in comparison with the activity lifecycle. Moving from the top to the bottom, we have:

• onInflate: This method is called only if we define fragment directly in our layout using the tag. In this method we can save some configuration parameter and some attributes define in the XML layout file.

• onAttach: This method is called as soon as the fragment is “attached” to the “father” activity and we can use this method to store the reference about the activity.

• onCreate: It is one of the most important steps, our fragment is in the creation process. This method can be used to start some thread to retrieve data information, maybe from a remote server.

• onCreateView: It is the method called when the fragment has to create its view hierarchy. During this method we will inflate our layout inside the fragment. During this phase we cannot be sure that our activity is still created so we cannot count on it for some operation.

• OnActivityCreated: In this method, we are notified when the “father” activity is created and ready for use. From now on, our activity is active and created and we can use it when we need.

• onStart: Here we do the common things as in the activity onStart, during this phase our fragment is visible but it isn’t still interacting with the user.

• onResume: This method is called when the fragment is ready to interact with user. At the end of this phase our fragment is up and running! Then, it is possible that the activity might be paused and so the activity’s onPause is called. Well, in this case the onPause fragment method is called too. After that, it is possible that the OS decides to destroy our fragment view and so the onDestroyView is called. After that, if the system decides to dismiss our fragment, it calls the onDestroy method. Here we should release all the active connections because our fragment is close to shutting down. Even during the destroy phase, it is still attached to the father activity. The last step is to detach the fragment from the activity and it happens when the on Detach is called.

5.7.5 Drawable

In Android, a Drawable is a graphical object that can be shown on the screen. From API point of view all the Drawable objects derive from Drawable class. They have an important role in Android programming and we can use XML to create them. They differ from standard widgets because they are not interactive, meaning that they do not react to user touch. Images, colors, shapes, objects that change their aspect according to their state, object that can be animated are all drawable objects. In Android under res directory, there is a sub-dir reserved for Drawable, it is called res/drawable Under the drawable dir we can add binary files like images or XML files. As we saw in the previous chapters, we can create several directories according to the screen density we want to support. These directories have a name like drawable-<>. This is very useful when we use images; in this case, we have to create several image versions: for example, we can create an image for the high dpi screen or another one for medium dpi screen. Once we have our file under drawable directory, we can reference it, in our class, using R.drawable.file_name. While it is very easy add a binary file to one of these directory, it is a matter of copy and paste, if we want to use a XML file we have to create it.

There are several types of drawable:

• Bitmap
• Nine-patch.
• Layer list
• State list
• Level list
• Transition drawable
• Inset drawable
• Clip drawable
• Scale drawable
• Shape drawable

An interesting aspect is that we can create such elements using XML or directly from code. There is a correspondence between the elements shown above and the API class. We can add the Drawable suffix and we create the corresponding class name: for example if the corresponding class of Bitmap drawable is BitmapDrawable and so on.You can have a look here if you want to have more information. We will not cover all these objects in this article but only the most popular.we covered some important UI aspects, concepts and best practices we should follow when developing an Android app. In this last article we saw how to apply all this knowledge to build a real app using all the topics covered in the previous chapters. Of course, this simple app can be improved and you can have it as an exercise to try to add new functionalities. For example, we did not cover how to modify or delete the items in the list. In this case we could listen when a user clicks on an item and you could show a menu with several options, or we could use the action bar changing it according to the user actions. For example, we could create a multiple selection list view and when user clicks on “trash” icon in the action bar we remove all the items selected. There are different aspects we can improve in this app and it will be a good exercise if you want to go deeper in Android development aspects.

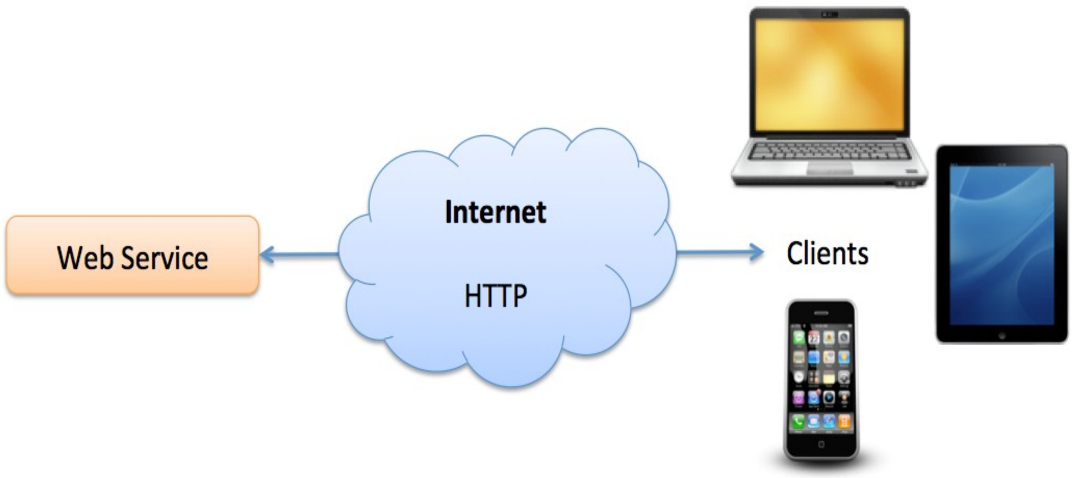
5.8 Connection between backend and frontend

5.8.1 Web Services Overview

- A Web Service is
- A Web API

- A Standard defined by W3C
- Cross platform and Platform independent Communication
- Distributed Application Development

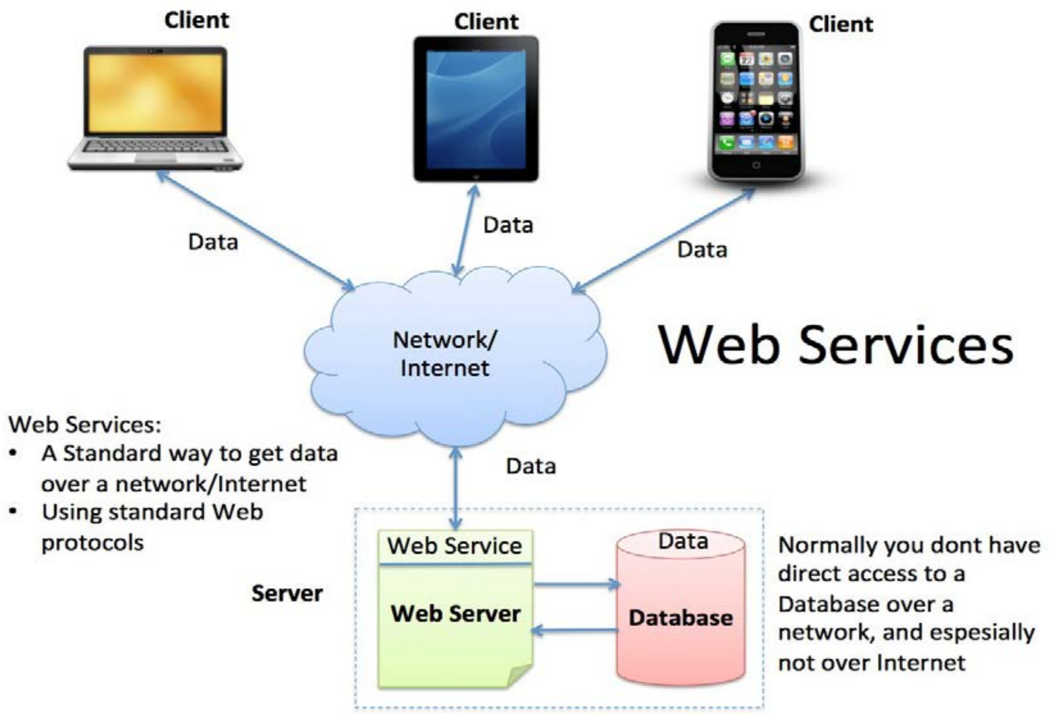
Web Services can be implemented and used in most Programming Languages (C#/ASP.NET,
PHP, LabVIEW, Objective---C, Java, etc.) Web Services uses standard Web technology (Web protocols) such as HTTP, REST, SOAP,XML, WSDL, JSON, etc.



Web Services technology used in Web Services:

- HTTP --- Hypertext Transfer Protocol
- XML – Extensible Markup Language
- WSDL --- Web Services Description Language
- SOAP --- Simple Object Access Protocol
- REST --- Representational State Transfer

A Web Service is typically deployed on a web server, similiar as ordinary web pages.


Client
Client
Client
Data
Data
Data
Network/
Internet
Web Services
Web Services:
• A Standard way to get data over a network/Internet
• Using standard Web protocols
Data
Server
Web Service
Web Server
Data
Database
Normally you dont have direct access to a Database over a network, and espesially not over Internet

5.8.2 API Introduction

The Discovery Service provides the developer with details of the home operator that serves the end user. Once you obtain the details for a specific user you can perform the Mobile Connect API calls to authorise that user. API allows you to interact with our system programmatically from your own application. Using the API you interact with Resources such as:Products , Subscriptions and Customers.The API attempts to conform to the RESTful design principles. You interact with the resources exposed via the API by accessing resource collection and element URIs using the HTTP verbs (GET, POST, PUT, and DELETE). Chargify accepts and returns both JSON and XML data via the API.You'll likely need access to a web developer or programmer (if you're not one) to get the most use out of the API.

5.8.2.1 API Authentication

The API Authentication is implemented as HTTP Basic Authentication over TLS (HTTPS). Your API login credentials are not the same as the credentials you use to log in to the web interface. You must obtain your API credentials. Separately You will use HTTP Basic Authentication to verify your identity via the API. All requests must come overTLS/HTTPS, and be to the subdomain of the Site you wish to access.

An example using authentication via curl:

```
https://acme.chargify.com/customers.json
```

5.8.2.2 API Resources

The fundamental concept in any RESTful API is the resource. A resource is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it. It is similar to an object instance in an object-oriented programming language, with the important difference that only a few standard methods are defined for the resource (corresponding to the standard HTTP GET, POST, PUT and DELETE methods), while an object instance typically has many methods.Resources can be grouped into collections. Each collection is homogeneous so that it contains only one type of resource, and unordered. Resources can also exist outside any collection. In this case, we refer to these resources as singleton resources. Collections are themselves resources as well.Collections can exist globally, at the top level of an API, but can also be contained inside a single resource. In the latter case, we refer to these collections as sub-collections. Sub-collections are usually used to express some kind of "contained in" relationship. We go into more detail on this in Relationships. We call information that describes available resources types, their behavior, and their relationships the resource model of an API. The resource model can be viewed as the RESTful mapping of the application data model.in the table below we used these methods to call data from server.

Resource/URI	GET	POST	PUT	DELETE

Subscriptions /subscriptions	List subscriptions	Create new subscription	–	–
Subscription /subscriptions/<x>	Read subscription	–	Update subscription	Cancel subscription
Subscription Charges /subscriptions/<x>/charges	–	Create one-time charge	–	–
Product Family Components /product_families/<x>/components	List components available on a given Product Family	–	–	–

5.8.2.3 Application Data

Resources have data associated with them. The richness of data that can be associated with a resource is part of the resource model for an API.We define the data that can be associated with a resource in terms of the JSON data model, using the following mapping rules:

- Resources are modeled as a JSON object. The type of the resource is stored under the special key:value pair "_type".
- Data associated with a resource is modeled as key:value pairs on the JSON object. To prevent naming conflicts with internal key:value pairs, keys must not start with "_".
- The values of key:value pairs use any of the native JSON data types of string, number, boolean, null, or arrays thereof. Values can also be objects, in which case they are modeling nested resources.
- Collections are modeled as an array of objects.

We will also refer to key:value pairs as attributes of the JSON object, and we will be sloppy and use that same term for data items associated with resources, too. This use of attributes is not to be confused with XML attributes.

5.8.3 Representation

We have defined resources, and defined the data associated with them in terms of the JSON data model. However, these resources are still abstract entities. Before they can be communicated to a client over an HTTP connection, they need to be serialized to a textual representation. This representation can then be included as an entity in an HTTP message body.The following representations are common for resources. The table also lists the appropriate content-type to use:

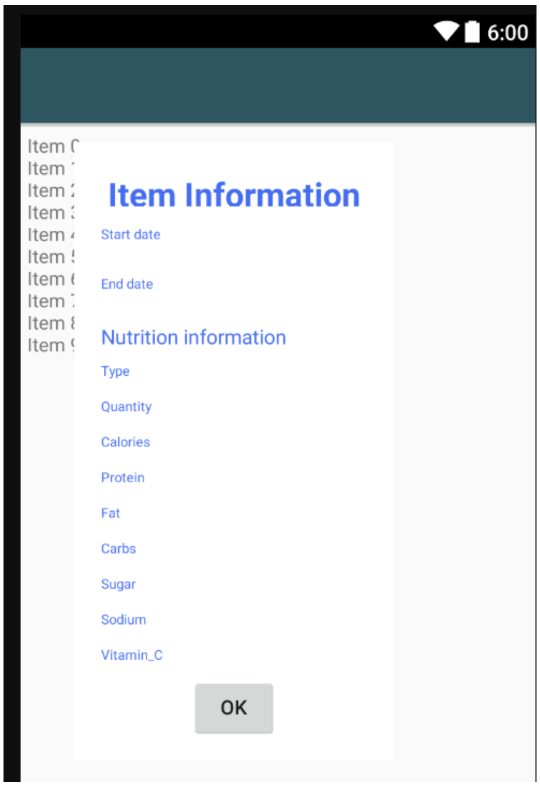
Type	**Content-Type**
JSON	application/x-resource+json application/x-collection+json
YAML	application/x-resource+yaml application/x-collection+yaml
XML	application/x-resource+xml application/x-collection+xml
HTML	text/html

Note: all these content types use the "x-" experimental prefix that is allowed by RFC2046.

5.9 Conclusion

we covered some important UI aspects, concepts and best practices we should follow when developing an Android app. In this last article we saw how to apply all this knowledge to build a real app using all the topics covered in the previous headings. Of course, this simple app can be improved and you can have it as an exercise to try to add new functionalities.For example, we did not cover how to modify or delete the items in the list. In this case we could listen when a user clicks on anitem and you could show a menu with several options, or we could use the action bar changing it according to the user actions.For example we could create a multiple selection list view and when user clicks on "trash" icon in the action bar we remove all the items selected.There are different aspects we can improve in this app and it will be a good exercise if you want to go deeper in Android development aspects.

Xml design :



The code of xml explain part of above screen shot and all will be the same.

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Start date"
    android:textColor="#456df1"
    android:padding="5dp"
    android:textSize="10dp" />

<TextView
    android:id="@+id/tv_start_date"
    android:layout_width="match_parent"
    android:layout_height="15dp"
    android:padding="5dp" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="End date"
    android:textColor="#456df1"
    android:padding="5dp"
    android:textSize="10dp" />

<TextView
    android:id="@+id/tv_end_date"
    android:layout_width="match_parent"
    android:layout_height="15dp"
    android:padding="5dp" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Nutrition information "
    android:textColor="#456df1"
    android:padding="5dp"
    android:textSize="15dp" />
```

Java code in design
It explains how to program buttons and and textview.

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_food_manager);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        recyclerView = (RecyclerView) findViewById(R.id.food_manager_List);
        layout = (ScrollView) findViewById(R.id.details_layout);
        ok = (Button) findViewById(R.id.ok);

        start = (TextView) findViewById(R.id.tv_start_date);
        end = (TextView) findViewById(R.id.tv_end_date);
        type = (TextView) findViewById(R.id.tv_type);
        quantity = (TextView) findViewById(R.id.tv_quantity);
        calories = (TextView) findViewById(R.id.tv_calories);
        protein = (TextView) findViewById(R.id.tv_protein);
        fat = (TextView) findViewById(R.id.tv_fat);
        carbs = (TextView) findViewById(R.id.tv_carbs);
        suger = (TextView) findViewById(R.id.tv_sugar);
        sodium = (TextView) findViewById(R.id.tv_sodium);
        vitamin = (TextView) findViewById(R.id.tv_vitamin);
```

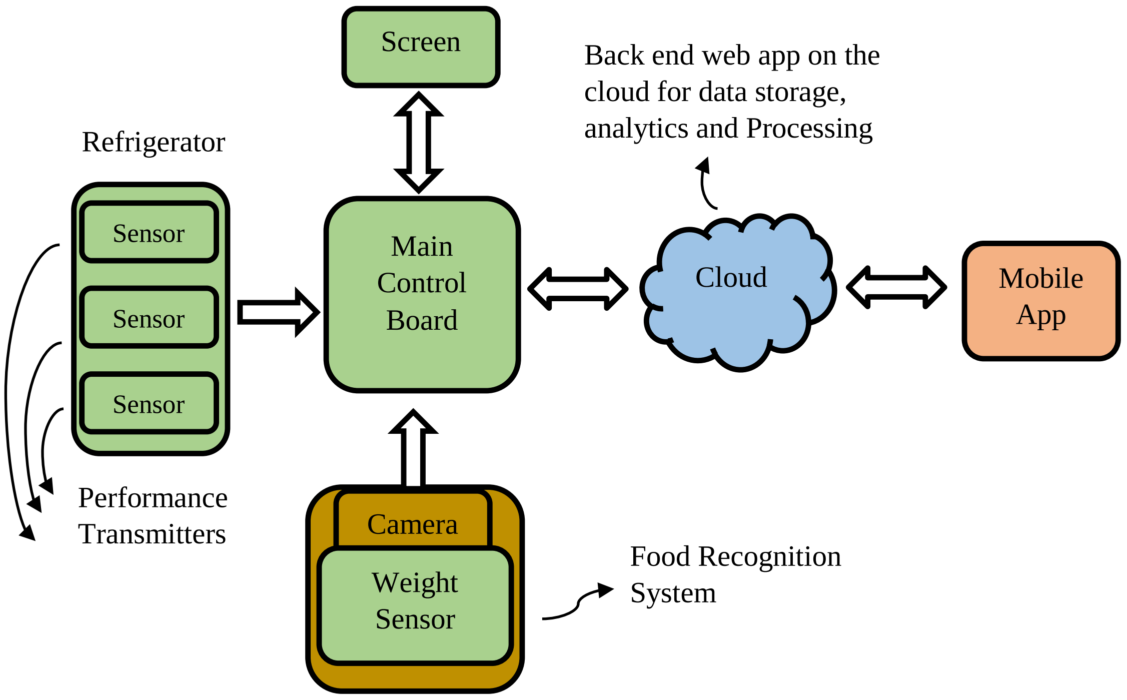
Finally: In the end this is just a simple example in the part of the project I have, but as I mentioned earlier is a mobile application for what has been described in all chapter parts this simple example is only applicable to certain parts of the chapter.

Chapter 6
Test and Results

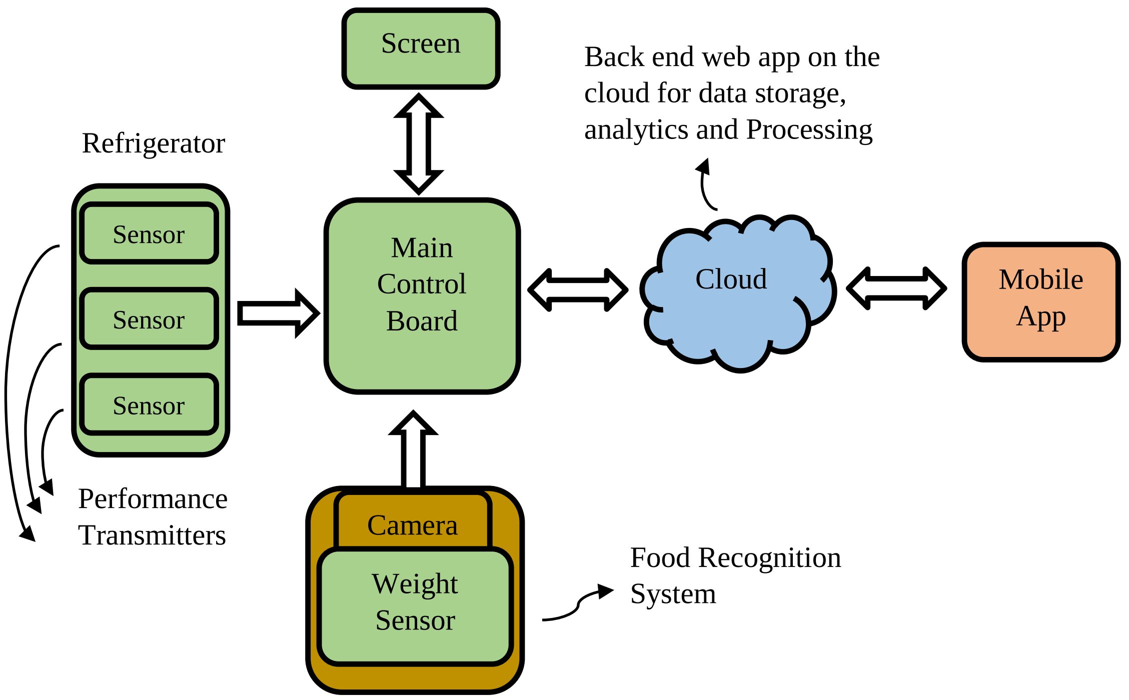
6 Overview

This chapter explains how the system works, indicating the resulted output of each stage of the system and how this output is connected to the next stage. The chapter also shows the tools used in Microsoft Azure cloud, what data is stored and how it is analyzed. The features provided by both the mobile application and the screen application are also discussed and tested.

The chapter also includes available future work in the control system, screen application, recognition system, cloud system and the mobile application.



6.1 Implementation



6.1.1 Control System

6.1.2 Fruits and Vegetables Recognition

Once the image is captured and stored in a specific path the code runs and produces a list of 42 features describing the fruit or vegetable type.

The code sent the features to the web service established on Microsoft Azure machine learning studio to recognize the type of fruit or vegetable. According to the type, nutrition information and expiration date are generated. Type, nutrition information and expiration date are then displayed on the system screen.

6.1.3 Cloud System

6.1.4 Mobile Application

6.2 Future Work

This section presents probable reforms and additional features that can de add to the system to enhance its performance and improve its functionality.

6.2.1 Future Work Concerning Control System

The whole system will be integrated in a single design to facilitate system movement. The camera will be triggered by the weight sensor. Barcode will be added to the system to recognize canned food and detect its expiration date efficiently. Different sensors will be added to the system to describe the refrigerator state efficiently such as pressure sensor and rotten food sensors.

The touch screen will enable the user to access the type, quantity and expiration date easily to change any error or incorrect information. This requires a new sophisticated screen application GUI.

New features will be added to the screen application for the user luxury such as mobile screen mirroring, music player and children monitoring screen.

6.2.2 Future Work Concerning Food recognition

The food recognition system will be enhanced to include canned food using barcode. Larger number of fruits and vegetables features will be added to the training data set so that the system recognizes more types.

The recognition technique can be modified so that users can train the system to recognize new types, which will increase the recognition efficiency in a short time. This technique of users training needs advanced authentication technique to ensure that the training is done by an actual user not by a robot.

6.2.3 Future Work concerning Cloud and Data Analytics

Data analytics will be involved with a much higher degree, as it will be used to suggest recipes to users according to the food stored in the refrigerator with valid expiring date. Data analytics will also be used to predict serious errors in the refrigerator by analyzing the different sensors readings.

6.2.4 Future work concerning the Mobile Application

Diet monitoring feature will be added to the mobile application so that each user monitors the calories in his food on his profile, store his current weight, his targeted weight, the time of losing weight and his activity type. The application will calculate the average allowed calories per day and subtract the calories that the user acquires when eating.

The user will submit his location to the mobile application. The mobile application will determine the closest supermarkets to the user and will order the approved shopping list from the nearest supermarket having all the food required in the shopping list.

6.3 Conclusion

It is obvious now that color and texture information represented as statistical features of fruits and vegetables can be used as features to recognize fruits and vegetables types. Multiclass neural networks based classifier can efficiently be trained to recognize fruits and vegetables.

Data analytics can be used to increase the life time of electrical appliances by analyzing its performance and comparing it with an ideal appliance.

IoT, mobile applications and embedded systems will entirely change the shape of the future.

References

[1] Raspberry Pi Cookbook Software and Hardware Problems and Solutions - 2nd Edition - Simon Monk – 2016

[2] Sams Teach Yourself: Big Data Analyticswith Microsoft HDInsight® in 24 Hours - Arshad Ali & Manpreet Singh – 2016

[3] Microservices, IoT, and Azure: Leveraging DevOps and Microservice Architecture to Deliver SaaS Solutions - Bob Familiar – 2015

[4] Azure Machine Learning: Microsoft Azure Essentials - Jeff Barnes – 2015

[5] Microsoft® Azure™ SQL Database Step by Step - Leonard G. Lobel & Eric D. Boyd – 2014

[6] Data Science in the Cloud with Microsoft Azure Machine Learning and R - Stephen F. Elston – 2015

[7] Predictive Analytics with Microsoft Azure Machine Learning - 2nd Edition - Roger Barga &Valentine Fontama & Wee Hyong Tok – 2015

[8] Pro SQL Database for Windows Azure - 2nd Edition - Scott Klein & Herve Roggero – 2012

[9] Windows Azure Mobile Services - Bruce Johnson – 2013

[10] Digital Image Processing - 3rd Edition - Rafael C. Gonzalez & Richard E. Woods – 1977

[11] Programming Computer Vision with Python Tools and algorithms for analyzing images - Jan Erik Solem – 2012

[12] Essentials of cloud computing - K. Chandrasekaran – 2015

[13] Android Studio Development Essentials – 6th Edition - Neil Smyth – 2015

[14] Android Programming for Beginners - John Horton – 2015

[15] Android™ Application Development All-in-One for Dummies® - 2nd Edition - Barry Burd – 2015