Assignment Number - 11

Title: Create a small dashboard application to be deployed on cloud.

Problem Definition: Create a small dashboard application to be deployed on cloud. Different publisher devices can publish their information and interested applications can subscribe.

Objectives:

➔ To develop a comprehensive approach towards building small low cost embedded IoT systems.
➔ To understand different sensory inputs.

Outcomes:

Students will be able to:

➔ Perform the connectivity with Raspberry-Pi, BeagleBoard, Arduino and other microcontrollers.
➔ Implement cloud applications with the help of client server programming by using Python.

Software and Hardware Requirements:

Raspberry-Pi, Cloud (ThingSpeak), client server model, controller / processor, Python, PC with the configuration as Latest Version of 64 bit Operating Systems, Open Source Fedora-GHz, 8 GB RAM, 500 GB HDD, 15'' Color Monitor, Keyboard, Mouse.

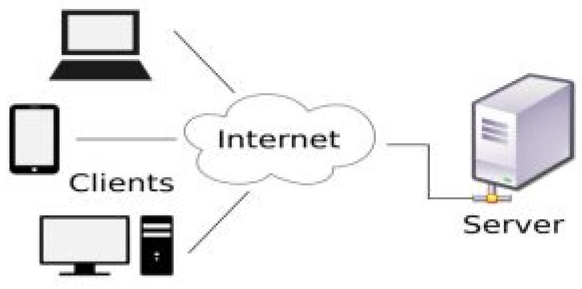
Theory:

ThingSpeak

ThingSpeak is an open source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP protocol over the Internet or via a Local Area Network. ThingSpeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates. ThingSpeak was originally launched by ioBridge in 2010 as a service in support of IoT applications. ThingSpeak has integrated support from the numerical computing software MATLAB from MathWorks, allowing ThingSpeak users to analyze and visualize uploaded data using MATLAB without requiring the purchase of a MATLAB license from MathWorks.

Client - Server Model

The client - server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Examples of computer applications that use the client - server model are Email, network printing, and the World Wide Web.



Setting Up an APACHE Web Server on a Raspberry-Pi:

Apache is a popular web server application you can install on the Raspberry Pi to allow it to serve web pages. On its own, Apache can serve HTML files over HTTP, and with additional modules can serve dynamic web pages using scripting languages such as PHP.

Install APACHE:

First install the apache2 package by typing the following command in to the Terminal:

```
sudo apt-get install apache2 -y
```

Test the Web Server:

By default, Apache puts a test HTML file in the web folder. This default web page is served when you browse to http://localhost/ on the Pi itself, or http://192.168.1.10 (whatever the Pi's IP address is) from another computer on the network. To find the Pi's IP address, type hostname -I at the command line (or read more about finding your IP address).

Changing the default Web Page:

This default web page is just a HTML file on the filesystem. It is located at /var/www/html/index.html.
Note: The directory was /var/www in Raspbian Wheezy but is now /var/www/html in Raspbian Jessie
Navigate to this directory in the Terminal and have a look at what's inside:

```
cd /var/www/html
ls -al
```

This will show you:

```
total 12
drwxr-xr-x 2 root root 4096 Jan 8 01:29 .
drwxr-xr-x 12 root root 4096 Jan 8 01:28 ..
-rw-r--r-- 1 root root 177 Jan 8 01:29 index.html
```

This shows that there is one file in /var/www/html/ called index.html. The . refers to the directory itself /var/www/html and the .. refers to the parent directory /www/.

What the Columns mean:

1. The permissions of the file or directory
2. The number of files in the directory (or 1 if it's a file).
3. The user which owns the file or directory
4. The group which owns the file or directory

5. The file size
6. The last modification date & time

As you can see, by default the html directory and index.html file are both owned by the root user. In order to edit the file, you must gain root permissions. Change the owner to your own user with sudo chown pi: index.html before editing. Try editing this file and refreshing the browser to see the web page change.

Your Own Website:

If you know HTML you can put your own HTML files and other assets in this directory and serve them as a website on your local network.

Source Code:

PHP Code:

```
<html>
<head>
        <meta name="viewport" content="width=device-width" />
        <title>LED Control</title>
</head>
<body>
        <form method="get" action="gpio.php">
                <input type="submit" value="ON" name="on">
                <input type="submit" value="OFF" name="off">
        </form>
        <?php
                $setmode17 = shell_exec("/usr/local/bin/gpio -g mode 17 out");
                if(isset($_GET['on'])){
                        $gpio_on = shell_exec("/usr/local/bin/gpio -g write 17 1");
                        echo "LED is on";
                }
                else if(isset($_GET['off'])){
                        $gpio_off = shell_exec("/usr/local/bin/gpio -g write 17 0");
                        echo "LED is off";
                }
        ?>
</body>
</html>
```

Mail Code:

```
import RPi.GPIO as GPIO
from subprocess import call
import time
import os
import glob
import smtplib
import base64
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
import subprocess
gmail_user = "checking999mail@gmail.com"
gmail_pwd = "mail999checking"
FROM = 'checking999mail@gmail.com'
TO = ['hyd.embedded@pantechmail.com'] #must be a list
i=1
while (i):
        i=i-1
        subprocess.Popen( "fswebcam -r 1280x720 /home/pi/Downloads/pan.jpg", shell=True )
        time.sleep(1)
        msg = MIMEMultipart()
        time.sleep(1)
        msg['Subject'] ="testing msg send from python"
        time.sleep(1)
        fp = open("/home/pi/Downloads/pan.jpg", 'rb')
        time.sleep(1)
        img = MIMEImage(fp.read())
        time.sleep(1)
        fp.close()
        time.sleep(1)
        msg.attach(img)
        time.sleep(1)
        try:
            server = smtplib.SMTP("smtp.gmail.com", 587) #or port 465 doesn't seem to work!
            print "smtp.gmail"
            server.ehlo()
            print "ehlo"
            server.starttls()
            print "starttls"
            server.login(gmail_user, gmail_pwd)
            print "reading mail & password"
            server.sendmail(FROM, TO, msg.as_string())
            print "from"
```

```
                server.close()
                print 'successfully sent the mail'
            except:
                print "failed to send mail"
    sudo apt-get install apache2
    sudo apt-get install php5 libapache2-mod-php5
    sudo apt-get install git-core
    git clone git://git.drogon.net/wiringPi
    cd wiringPi
    ./build
```

Conclusion:

We have successfully created a small IoT application which was deployed on cloud.