Setting Up the Raspberry Pi for Internet of Things

Introduction

This document provides installation and configuration instructions for six Internet of Things (IoT) projects demonstrating a variety of techniques for controlling devices over the Internet. Figure 1 shows a conceptual view of the relationship between an Internet client and an Internet accessable device connected to the Raspberry Pi. Each project's user interface design maintains a "thin client" philosophy so that no special software need be installed on the client machine accessing the Internet thing. The projects range from simple control of an LED to displaying recorded data from an altimeter sensor.

The instructions given in this document apply to most Linux distributions. The author has successfully run these projects on a Raspberry Pi 3 running the Raspberry Pi OS operating system. Network configuration and settings are not covered in this project hookup guide. It is assumed that the user has a means of connecting the Raspberry Pi to a local network and that the Internet is accessible from the local network. There are many possible ways of doing this. The networking aspect is beyond the scope of this document. Contact your local network administrator for instructions on how to connect your Raspberry Pi to your local network.

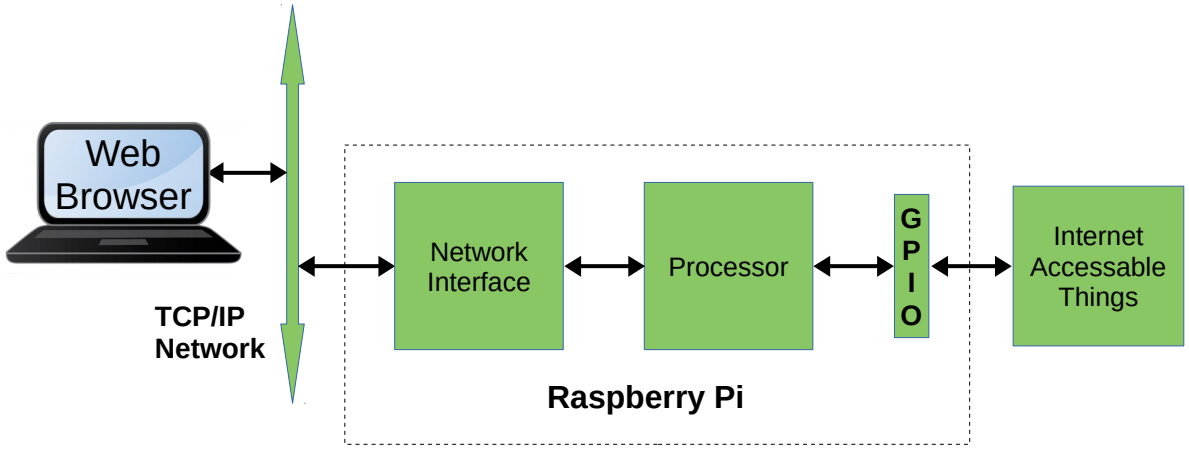

Figure 1. Overall conceptual view of Raspberry Pi internet accessible devices.

Raspberry Pi Internet of Things Overview

Hookup instructions and source code provided for the following six projects

Project	Description
LED	Turn on and off an LED
Push Button	Display on a web page the number of times a push button has been pushed
Servo	Control a servo by moving a slider on a web page
Fncgen	A simple, software defined function generator that demonstrates the uses of a digital to analog converter.

Oscilloscope	A software defined oscilloscope that demonstrates the use of an analog to digital converter.
Altimeter	An altimeter app that presents in charts and text format the data from an altitude sensor. Demonstrates the use of a round-robin database to store data and generate graphical charts.

Each project has its own folder in the Raspberry Pi Internet of Things (raspiot) repository. Each project folder has the following structure

- **docs** folder containing hookup instructions, schematics, and component data sheets
- **bin** folder containing executables that go in the *pi* user bin folder "/home/pi/bin"
- **html** folder containing html documents and php scripts that go in the *pi* user web document root folder "/home/pi/public_html".

Inside the document root "public_html" folder a sub-folder should be created for each project that will be installed. For example, if you choose to install the LED project, you should create a folder "/home/pi/public_html/led" to contain all the files and sub-folders that are in the repository LED html folder. The files in each project bin folder always go into the common "/home/pi/bin" folder.

Hardware Hookup

Each Raspberry project has its own hookup guide in its documents folder. Refer to the project hookup guide for instructions on how to hookup the hardware components and install the software. Each project involves connecting devices to the Raspberry Pi GPIO. Figure 2 shows the location of the GPIO header on the Raspberry Pi printed circuit board and provides explanations of the GPIO pin outs. The hardware is best hooked up using standard bread-boarding components along with a Raspberry Pi GPIO breakout board. Figure 3 shows a photo of all six projects hooked up using only one breadboard.

Parts List

Project	Parts	Supplier / Part Number
LED	• LED (150-200mcd) • 220 Ohm ¼ Watt Resistor	Adafruit, Digikey, Sparkfun
Push Button	• Normally Open Single Pole Single Throw Push Button Switch (breadboard mountable) • 10 Kiloohm ¼ Watt Resistor	Adafruit, Digikey, Sparkfun # COM-09190
Servo	• SpringRC SM-S2309S Micro Servo (or equivalent) • 1 Kiloohm ¼ Watt Resistor	Adafruit, Digikey, Sparkfun
Fncgen	• MCP4725 Digital to Analog Converter	Sparkfun # BOB-12918
Oscilloscope	• ADS1115 Analog to Digital Converter	Adafruit # 1085
Altimeter	• MPL3115A2 Altimeter	Sparkfun # SEN-11084

Required for all projects	• Raspberry Pi 3 (or higher) • GPIO Breakout Connector • Solderless Breadboard • Jumper Wires	• Adafruit # 3055, Sparkfun # DEV-14643 • Adafruit # 2028, Sparkfun # BOB-13717 • Adafruit # 239, Sparkfun # PRT-12615 • Adafruit # 3314, Sparkfun # PRT-00124

Supplier Websites

Supplier	URL	Parts
Adafruit	https://www.adafruit.com	LED, Push Button, Resistors, ADS1115, Pi 3, Breakout Connector, Solderless Breadboard, Jumper Wires
Digikey	https://wwww.digikey.com/	LED, Push Button, Resistors, Pi 3, Breakout Connector, Solderless Breadboard, Jumper Wires
Sparkfun	https://sparkfun.com	LED, Push Button, Resistors, MCP4725, MPL3115, Pi 3, Breakout Connector, Solderless Breadboard, Jumper Wires

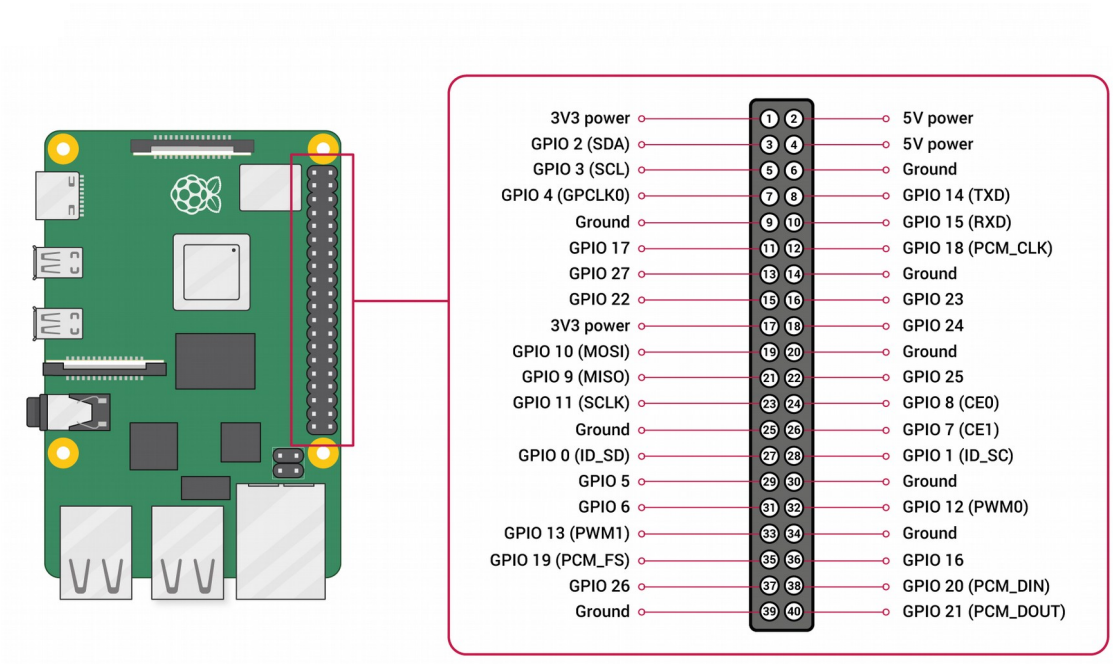

Figure 2. Diagram showing GPIO pin outs and location of GPIO header on Raspberry Pi PCB.

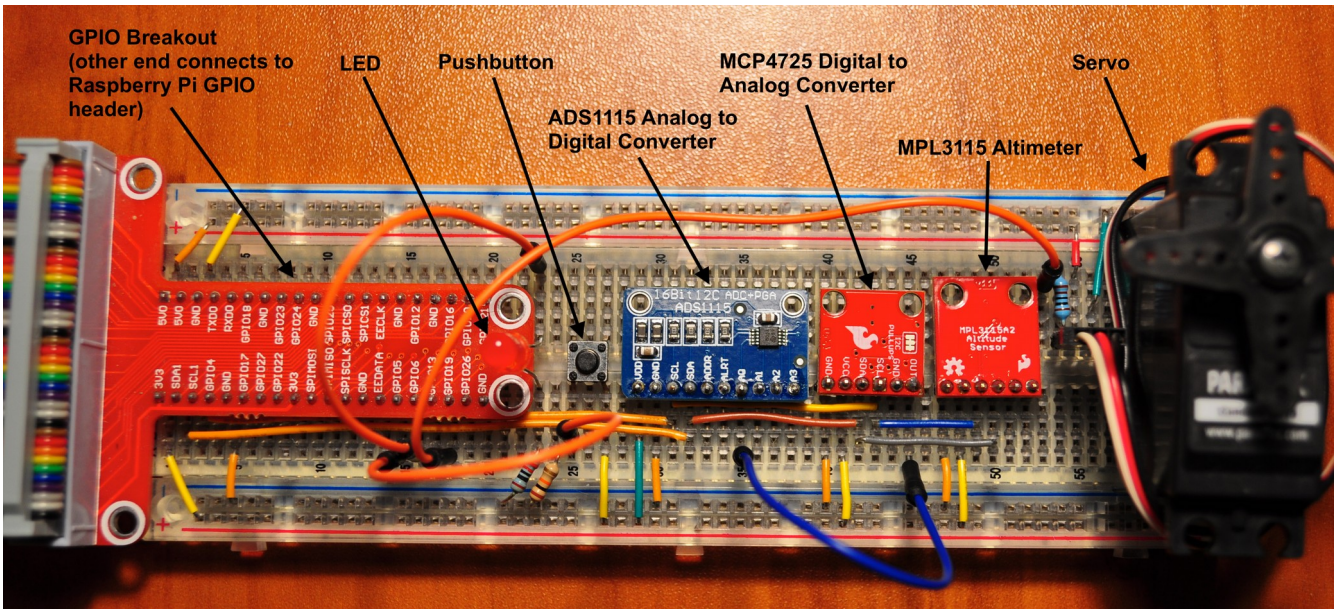

Figure 3. Photo of all six projects mounted on a single breadboard.

Server Software

Installing the Server Software

Dependencies:

1. The Raspberry Pi IoT apps should be installed on a Linux host which meets the following requirements:
    - A recent Raspberry Pi Linux distribution such as Debian, or Raspberry Pi OS. (The author has successfully developed and installed the software on a Raspberry Pi running the Raspberry Pi OS.)
    - Apache 2 should be installed and configured to allow serving HTML documents from the user's public_html folder.
    - PHP 7 should be installed and configured to allow PHP scripts to run from the user's public_html folder.
    - The Python I2C smbus library should be installed.
    - Python 3.0 usually comes pre-installed in virtually all Linux distributions. Type "python3" at a command line prompt to verify Python 3 has been installed.
2. See the Appendix for detailed notes on how to set up a Raspberry Pi server, and how to install and configure the components in note 1 above.

Software Inventory

The following software items need to be installed in the documents root folder, e.g., "/home/pi/public_html".

HTML folder:

- index.html
- apps.html
- favicon.ico
- static/bg_apps.jpg

bin folder:

- startup.sh

Getting the Raspberry Pi IoT Software

From the github repository:

1. On a computer connected to the Internet download the zip file from

   https://github.com/fractalxaos/raspiot/archive/master.zip

2. From the downloaded file **raspiot-master.zip**, extract the **raspiot-master** folder to the desktop.

3. From the **raspiot-master** folder, copy files to the Raspberry Pi as directed in the following section and it the instructions for each individual project.

Installing on a Raspberry Pi

Figure 1 shows a block diagram of a Raspberry Pi connected to the local area network. The Pi can be either connected to the network via built-in WiFi adapter, or by Ethernet cable. Note that the following installation procedure assumes that the document root for the IoT HTML documents will be the user's public_html folder. Typically the full path name to this folder will be something like **/home/{user}/public_html**, where **{user}** is the name of the user account hosting the IoT project apps. The following steps will assume the apps are running under the **pi** user account. The **pi** account is the default account when Raspberry Pi OS is first installed on a Raspberry Pi.

1. Follow the instructions in the appendix to install and configure web services on your Raspberry Pi.

2. If necessary, turn on the I2C bus by running

   sudo raspi-config

   Select Interface Options. Select I2C and set to ON.

3. Determine if the python I2C smbus library has been installed. Start the python interpreter by running

   python3

   At the python prompt enter

   >>>**import smbus**

   If you get an error, then the smbus library has not been installed. Type **Ctrl-D** to exit the python interpreter.

4. If necessary, install the I2C smbus python library by running

   sudo apt-get update

sudo apt-get install python-smbus

5. If it doesn't already exist, use **mkdir** to create a folder **public_html** in the **pi** user's home folder. The working folder should look like **/home/pi/public_html**.

6. Move all the contents of the repository **raspiot/html** folder to the **public_html** folder created in step 5. This includes the **static** subfolder and its contents.

7. If it does not already exist, use **mkdir** to create a folder named **bin** in the **pi** user's home folder. For example, the full path name should look like **/home/pi/bin**.

8. Move all the contents of the **raspiot/bin** subfolder in the **raspiot** folder to the **bin** folder created in step 7.

9. Follow the individual hookup instructions for each project you choose to install.

This completes installation of the server software.

References and Resources

The following resources describe how to configure a Raspberry Pi to be a server

- Using an SSH key pair:
  https://help.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

- Reducing wear on the SD card:
  https://www.zdnet.com/article/raspberry-pi-extending-the-life-of-the-sd-card/

- Installing a web server:
  https://www.raspberrypi.com/documentation/computers/remote-access.html#setting-up-an-apache-web-server

- Linux system administration and devops
  https://www.guru99.com/unix-linux-tutorial.html
  http://devopsbootcamp.osuosl.org/
  http://lug.oregonstate.edu

The following tutorials are useful for more in depth understanding of the server software:

- Javascript http://www.w3schools.com/js/default.asp
- PHP http://www.w3schools.com/php/default.asp
- HTML http://www.w3schools.com/html/default.asp
- Rrdtool http://oss.oetiker.ch/rrdtool/
- Python http://greenteapress.com/thinkpython/thinkpython.html

Appendix

The following steps describe how to build a web server on a Raspberry Pi. The steps below configure the Raspberry Pi to serve web pages from the user's document root folder. When user html folders are enabled, Apache looks for the files in the user's **public_html** folder. For example, a browser request **http://raspi.local/~pi/myDocument.html** would cause the host **raspi.local** to look for **myDocument.html** in the folder **/home/pi/public_html**. With the setup in the steps below, documents can also be served virtually, as if from the host's **/var/www** folder. For example, **http://raspi.local/myDocument.html** would also cause Apache to look for **myDocument.html** in the folder **/home/pi/public_html**.

1. If upgrading from a previous Raspberry Pi OS version, first backup the user home folder by running

   sudo tar -zcpvf /home/pi_bak.tar.gz /home/pi

   Copy the **pi_bak.zip** file to a thumb drive or some other external storage media.

2. If you are building the web server on an existing SD card with Raspberry Pi OS already installed, skip to step 11.

3. Copy the new Raspberry Pi OS disk image to the SD card. Follow Raspberry Pi instructions for copying the disk image to the SD card.

4. After copying the disk image to the SD card, mount the SD card **boot** partition on the computer you used to copy the disk image. Navigate to the root of the **boot** partition and create an empty file named **ssh** by running

   touch ssh

   This will enable setting up the Raspberry Pi without needing to attach a keyboard, mouse, and monitor.

5. If you are connecting to the Pi via WiFi, in the root of the **boot** partition use a text editor to create a file named **wpa_supplicant.conf** and add the following lines. Otherwise, if you are connecting via wired Ethernet, skip this step.

   ```
   ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
   update_config=1
   country=US
   network={
     ssid="your_wifi_network_name"
     psk="your_wifi_network_password"
     key_mgmt=WPA-PSK
   }
   ```

   Be sure to substitute your WiFi network name (SSID) and WiFi password.

6. Unmount and remove the SD card.

7. Insert the SD card into the Pi and connect the Pi to its power source. The Pi will boot up.

8. Once the Pi has booted, on your computer open a terminal window and run the command

   ssh pi@raspberrypi.local

The default password is **raspberry.**

9. Run the command

   sudo raspi-config

   In System Options change

   Hostname to your host name
   Password to your password

   In Localisation Options change

   Locale to en_US.UTF-8 UTF-8
   Timezone to your timezone
   Keyboard US

10. Reboot the Pi by running

    sudo reboot

11. Use ssh to login as user pi with the password set in step 9

    ssh pi@{your host name}.local

12. [Optional] Run all software updates

    sudo apt-get update
    sudo apt-get upgrade
    sudo reboot

13. Update the package database and install vim

    sudo apt-get update
    sudo apt-get install vim

    vim is an editor that makes it easy to make changes in configuration files. Alternatively you can use **nano** or some other terminal session based editor of your choice.

14. [Optional] On the client computer, create an ssh key pair. In the Raspberry Pi user **pi** home folder, create a folder named **.ssh**. In the **.ssh** folder create a file named **authorized_keys**, and copy the public key to it. For more information on how to create and use key pairs see https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent .

15. Note that all procedures below that require editing a system file have to be done acting as superuser. For example, to edit the file in step 16 run the command

    sudo vi /boot/config.txt

    Before editing a system file, be sure to make a backup copy. For example, before running the above command make a backup copy by running

sudo cp /boot/config.txt /boot/config_bak.txt

16. [Optional] If they will not be used, disable WiFi and Bluetooth. After creating a backup copy, modify **/boot/config.txt** by adding the following two lines to the end of the file:

```
dtoverlay=pi3-disable-wifi
dtoverlay=pi3-disable-bt
```

    and run the once off command:

    sudo systemctl disable hciuart

17. [Optional] For added security turn off password authentication. You must have set up ssh keys as discribed in step 14. *Do not do this step unless you are familiar with ssh keys and how they work.* You will not be able to login via ssh to the Pi with a password. The above enhances security when using secure shell (ssh) to access the Raspberry Pi. Backup and then modify **/etc/ssh/sshd_config** as follows

```
#PasswordAuthentication yes
PasswordAuthentication no
```

18. System log files get get overwritten frequently; similarly output data and chart files get overwritten frequently. On SD card systems, such as a Raspberry Pi, it is inadvisable to do frequent writes to any file system mounted on the SD card. To use the RAM based temporary file system (tmpfs) to store these files, continue with step 19. Otherwise, to write dynamic content to the disk drive, continue with step 20.

19. Setup the temporary file system (tmpfs) by backing up and then modifying **/etc/fstab**. Add the following lines to the bottom of the **fstab** file

```
# Added 2022-01-01 by admin to store logs and temporary files
# in ram to reduce wear on the SD card due to frequent writes.
tmpfs /tmp tmpfs nodev,nosuid,size=10m 0 0
tmpfs /var/tmp tmpfs defaults,noatime,nosuid,size=10m 0 0
tmpfs /var/log tmpfs defaults,noatime,nosuid,mode=0755,size=10m 0 0
tmpfs /var/spool/mqueue tmpfs defaults,noatime,
    nosuid,mode=0700,gid=12,size=10m 0 0
```

    Highly recommended, this step configures the Raspberry Pi to store all log files and temporary files in RAM. Unless configured to use an external hard drive, the Raspberry Pi mounts the root file system on the SD card. Log files and temporary files are frequently written to the file system, resulting in wear on the SD card. Storing these files in RAM saves the SD card from such wear. Note that all log files and temporary files are lost upon reboot or power down.

20. Reboot the Raspberry Pi.

21. Use ssh to login as user pi with the password set in step 9. If you have set up ssh keys per step 14, you will not need a password.

    ssh pi@{your host name}.local

22. Backup **/etc/rc.local** and then add the user start up script. For example, add the following line at the end of the file **/etc/rc.local**

    (su - pi -c "bin/startup.sh")&

23. Install LAMP. This is the standard Linux web server “stack”. For more information about installing LAMP see https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md . Run the following commands

    Apache
    ======
    sudo apt-get install apache2 -y
    sudo a2enmod rewrite
    sudo systemctl restart apache2

    PHP
    ===
    sudo apt-get install php libapache2-mod-php y
    sudo systemctl restart apache2

    [Optional] SQL
    ==========
    sudo apt install mariadb-server mariadb-client php-mysql -y

24. Backup and then modify **/etc/apache2/mods-available/userdir.conf** to allow user **.htaccess** files. For example,

    ```
    # Changed 2022-01-01 by admin to allow user .htaccess files.
    #AllowOverride FileInfo AuthConfig Limit Indexes
    AllowOverride All
    ```

    This enables Apache to use the .htaccess file in the user’s document root folder.

25. Enable user directories in Apache by running

    a2enmod userdir

26. Backup and then modify **/etc/apache2/mods-available/php7.3.conf** to allow PHP scripts to run in user directories by commenting the lines at bottom of file. For example,

    ```
    # Changed 2022-01-01 by admin to enable user php scripts.
    #<IfModule mod_userdir.c>
    #   <Directory /home/*/public_html>
    #       php_admin_flag engine Off
    #   </Directory>
    #</IfModule>
    ```

27. Enable PHP in Apache by running

    a2enmod php7.3

28. Backup and then modify **/etc/apache2/sites-available/000-default.conf** to make the pi user **public_html** folder the web server document root. For example,

    ```
    # Changed 2022-01-01 by admin to make user pi the html document root.
    #DocumentRoot /var/www/html
    DocumentRoot /home/pi/public_html
    ```

    This makes the pi user public_html folder the document root for the Raspberry Pi. For example a client request **http://raspberrypi.local/myDocument.html** would cause Apache to look for the document in

the folder **/home/pi/public_html**.

29. Backup and then modify **/etc/apache2/envvars** to create Apache logs in tmpfs. Add the following lines at the top of the file

```
# Added 2022-01-01 by admin to allow apache to use tmpfs.
if [ ! -d /var/log/apache2 ]; then
    mkdir /var/log/apache2
fi
if [ ! -d /var/log/mysql ]; then
    mkdir /var/log/mysql
fi
```

    Without these lines Apache will fail to start when the system reboots. When the system reboots the folders **/var/log/apache2** and **/var/log/mysql** will not exist if logs are using the temporary file system in RAM. Apache will fail to start if these folders are not present. Adding the above lines causes Apache to create these folders when the system boots up.

30. Enable Apache to access files in the the temporary file system **/tmp** folder by backing up and then modifying **/lib/systemd/system/apache2.service** as follows:

```
# Changed 2022-01-01 by admin to enable apache to follow
# symlinks to the /tmp folder in tmpfs.
#PrivateTmp=true
PrivateTmp=false
```

    This change enables Apache to follow symbolic links in the user document root (public_html) folder to folders and files in the temporary file system **/tmp** folder.

31. Reload system deamons by running

    sudo systemctl daemon-reload

    and restart Apache service

    sudo systemctl restart apache2

32. Reboot the Raspberry Pi.

33. Test all the above modifications.