The Hidden Earth Timetable Display – Part 2

The UK's annual caving conference – Hidden Earth – has experimented with a network of computers that provide timetable displays, as well as news and information. The Raspberry Pi module makes a very useful web client for such a system. In this article, ***David Gibson*** *describes how to get started with the Raspberry Pi. A future article will give actual code examples.*

The previous article in this series [Gibson, 2018] outlined the concept of a networked system of computer displays using the Raspberry Pi (RPi) computer module. Our aim is to run a web browser client on the RPi, in so-called 'kiosk' mode, and for it to automatically identify itself to a webserver. This article describes how to get started and covers some of the network issues involved.

These notes are *not* intended for beginners. It is assumed that the reader has some knowledge of webservers, HTML, HTTP and network issues, but that he may be a newcomer to Linux and RPi.

Setting-up the Raspberry Pi

Equipment List

The range of products and models can be a little confusing, and these instructions might not work with all devices (especially earlier models without a GUI). These notes are for a **Raspberry Pi 3, model B+**. So… you will need…

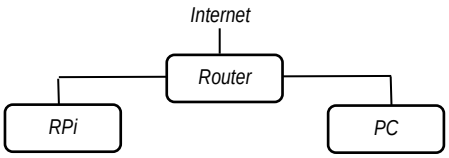
- A Raspberry Pi 3 model B+
- A housing to help prevent short-circuits, and to protect the board from damage
- An SD memory card with the RPi software
- A RPi power supply
- A monitor with an HDMI interface
- A cable to connect the monitor to the RPi
- A (preferably) wireless keyboard and mouse with (preferably) a combined USB dongle
- A router connected to the Internet, and (preferably) with a WLAN capability, to communicate with the RPi

Connecting and Powering Up

For this simple introduction and demonstration, a single RPi and a PC will be connected to the router – either wirelessly or by cable – see the diagram below.

First of all, find an enclosure for the RPi. Leaving it bare on the desk is asking for trouble. There are a large number of different styles of plastic case available.

As well as the RPi hardware, you need the software, which comes on a microSD card. I am using the NOOBS software from **uk.farnell.com**. NOOBS stands for 'new-out-of-the-box' and includes a number of configuration routines, in addition to the Raspian operating system. You can download NOOBS from the RPi website, but then you need to delve into the intricacies of programming an SD card in Windows (if you are using Windows) to run on a Linux system. It is easier just to buy an SD card pre-flashed with NOOBS.



- Plug the microSD card into the RPi
- Place the RPi into its housing

I suggest that, to avoid too many trailing leads, you use a wireless mouse and keyboard, and to save wasting USB slots you use a combined set, so that you only need one dongle for the two devices.

- Plug in the dongle(s) or cables for the mouse and keyboard.
- Connect the monitor to the RPi board and switch it on. If the monitor needs configuring to use its HDMI interface, do this.
- If you are using an Ethernet link to the router, connect the cable. A wireless connection is built into the RPi.
- Attach the RPi power supply and switch on

Because you are powering up for the first time, some configuration and installation will take place. Be prepared for a blank screen, and hundreds of rapidly-scrolling messages, followed by several long pauses. Eventually, the system should stabilise, showing a photograph (rather like Windows 10 produces) or the RPi desktop screen, with a row of icons along the top of the screen. To the horror of die-hard Linux users, the RPi has sensibly booted into a GUI !

Configuration

You now need to do some additional configuration. If you hover the mouse over the RPi icon, you should see a tool-tip message – *Click here to open Applications Menu*. Do that; select *Preferences* and then *Raspberry Pi Configuration*. There are, of course, many settings we can tweak, but the one we are looking for now is the *Localisation* pane. Again, there is plenty to tweak, but the setting we *need* to make is the *WiFi Country*[1].

Whilst you are making these settings, you might also want to set the time zone, which takes the form of *Country/City*.

Having set the WiFi country, the WLAN link to the router can be enabled, click the network icon and it will list the available routers. Select your router and, if it needs a password, type its 'pre-shared key'. This is often printed on the router.

Checking the Browser

Now that you have an Internet connection, open the RPi browser by clicking the web icon. You will notice that, once there is a connection, the date and time will be updated.

You should find that the supplied browser is Chromium – much better than the earlier offering, Epiphany. Should you need to update the software and/or install Chromium, I will explain how to do that later[2]. Although, upgrading the software is probably one of the first things to do.

The Command-Line Interface

Although there is plenty to do without needing the command-line interface[3] (CLI), you will need to learn where it is to carry out any specialised tasks. Just click the Terminal icon and a terminal window will open.

Accessing the CLI Remotely

For some of my tasks, I prefer to talk to the RPi over the network, rather than via its keyboard, because using two keyboards and mouses at the same time can get con-

1 This is necessary on the Pi 3 Model B+ so that the 5G networking can choose the correct frequency bands. In the UK you need to set it to GB.
2 See a future article.
3 It's not that I am against CLIs – there is a time and place for them. It's just that a GUI is arguably easier for many operations.

fusing. To access the RPi over the local network, the stages involved are

- On the RPi, enable its SSH interface
- *Possibly* establish its IP address
- Run an SSH terminal app. on the PC

To enable remote connection to the ℜπ, go to the Applications Menu and select *Preferences / Raspberry Pi Configuration / Interfaces*. Enable SSH (which stands for secure shell and is a terminal application for use over a network link).

If you do not already have it, you will need to install an SSH terminal app on the PC. A commonly used package is puTTY from **putty.org**. Assuming you are running puTTY, it will ask you for *Hostname or IP address* and it will suggest you connect on port 22, which is the standard SSH port and is where the RPi is listening. Normally, we would connect to an Internet address and would know the domain name. For this local connection, we need either the IP address on the local network or – more conveniently – a local domain name.

If this is the only RPi that is connected, we can take a reasonable guess that its hostname is **raspberrypi** and so the router should recognise **raspberrypi.home** or **raspberrypi.local** as local destinations. We can also take a shortcut by specifying the default login username as part of the URL, viz.: **pi@raspberrypi.home**.

As SSH is a 'secure' connection, it will prompt for information about security keys – for which you need to reply with the common-sense answers. A terminal window will then open and will ask for a password. The RPi factory setting is **raspberry**. The RPi command prompt will then be displayed, looking something like …

```
pi@raspberrypi:~ $
```

thus demonstrating that the remote login to the CLI was successful. Of course, this is only the CLI, not a friendly GUI. In fact, it *is* possible to access the GUI remotely as well, but it requires further software to be installed and – in my experience – it is not easy to get it to work satisfactorily.

File Transfer with FTP

As well as CLI operations, it is useful to be able to transfer files to/from the RPi without going to the lengths of burning a USB memory stick. We therefore need an FTP (file transfer protocol) application. Some browsers have FTP capability, but this is limited. We really need a 'proper' package like, say, winSCP from **winscp.net**, and we connect using secure FTP (SFTP) on port 22. Using FTP remotely is similar to using the RPi's GUI and clicking on the File Manager icon. A secure connection is not essential on the local network, but you might as well get into the habit as there is a trend for servers to disallow plain FTP.

Hostname and IP Address

Asking the Raspberry Pi

If the **.local** and **.home** domains do not work, you will need to establish for certain the hostname or IP address of the RPi. Open a terminal window on the RPi and, at the command prompt, enter the command **hostname –I**. Representing the dialogue with **$** for the command prompt, like this…

```
$ hostname -I
192.168.1.74
```

…I am showing that the RPi responds by printing its IP address[4]. For hostname …

```
$ hostname
raspberrypi
```

As with most commands, terse help is available with the --help option…

```
$ hostname --help
```

Asking the Router

It is easier to ask the RPi; but the router knows everything about the network – possibly more than the PC and the RPi – and it is useful to learn how to find your way around in the router. To access the router's control panel, you need to know its domain name (e.g. **BTHome Hub.home**, for my router, supplied by the telecoms provider BT), or its IP address. A 'generic' router is likely to have the IP address **192.168.1.1** or perhaps it will be on port 8080 as **192.168.1.1:8080**, but you cannot guarantee either of those possibilities (my router is at **192.168.1.254**). As with all things, Google (probably) knows.

Type the domain name or IP address into the web browser. Some navigation (possibly visiting 'advanced settings' if you have a dumbed-down router like mine) will then lead you to a page that lists, for each connected device, its network name, IP address, and MAC address.

Network Issues

Before completing our goal of running a web client on the RPi there are some network issues that we need to consider.

Finding the MAC Address

I explained, above, how we can find the IP address of the RPi. This is useful if we want to *initiate* a network connection remotely (e.g. for SSH, FTP or if the RPi were running a webserver – which I'll get to in a future article). But we want the RPi's browser to identify itself without our needing to program the RPi individually, and the IP address is less useful for this because it can change. Ideally, we need to ascertain the MAC address and arrange to send it, automatically, to the server; e.g. as **hidden-earth.local/?mac=B8-27-EB-36-40-08**.

The MAC address is stored in a text file on the RPi. The Linux command CAT[5] will print the file; so we can inspect it as …

```
$ cat /sys/class/net/eth0/address
b8:27:eb:63:15:5d
$ cat /sys/class/net/wlan0/address
b8:27:eb:36:40:08
```

Note that there are two MAC addresses – one for the Ethernet interface and one for the wireless LAN. If you happen to be using a RPi with additional interfaces, then you will have more MAC addresses to look for. You can get a complete picture from

```
$ ifconfig
```

You might think that this is starting to get complicated – how can we tell which interface the RPi is actually using? In fact, we do not need to know. We can use *any* of its MAC addresses to identify any particular RPi. The webserver does not know that the MAC address we are sending at the HTTP application level is not the actual MAC address of the network adapter. In fact, if the webserver is not on the local network it has no way of knowing the MAC address at all – which is why we are going to such lengths to obtain and send it. Also; if it turns out to be convenient, we only need to use the last three bytes of the address because the first three identify the product and manufacturer and so are likely to be the same for all the RPi boxes.

Talking to a Webserver

A webserver on the Internet is identified by its URL but it is likely we will be operating on the *local* network, so what hostname should we use? The router can tell us the PC's name, but the PC knows too, of course. At a Windows prompt …

```
$ ipconfig /all
```

will produce a list of information. Look for the hostname listed under *Windows IP Configuration*. Alternatively, look for the value assigned to USERDOMAIN in…

```
$ set user
```

4 Be sure to type a capital I, not a lower-case i, which would give you the loopback IP address, which is of no use whatsoever.

5 If you are new to Linux, CAT is similar to TYPE and PRINT on other systems

We want to have a sensible hostname, like **hidden-earth** rather than **HP-LAPTOP-A1B23D4**. There are several ways to change a PC's hostname. In Windows 10, navigate to **start/settings/advanced/about /rename this PC**, or ask Google.

My PC is (re-)named **red-laptop** so to access it from the RPi I simply use the URL **http://red-laptop.local**. Note that the http prefix is needed in Chromium's address bar because, without it, it seems not to recognise the top-level domain (TLD) **.local** and will open a search engine instead.

I have a webserver running on my PC (I use the XAMPP package) so accessing it from the RPi via **red-laptop.local** is just like asking for the **localhost** page on my PC.

Home and Local Addresses

The top-level domain **.local** is *usually* interpreted by the router as referring to the local network. (Note that this is *not* the same as using the domain name **localhost**, which is a local loop-back *within* the computer). Another similar domain is **.home** but this has no official status as a local domain. The two are not synonymous and it is possible that you will need to try both. There is no standard for this, and I have not got to the bottom of how and why my router seems to treat the two TLDs differently. *Sometimes* it seems that it is using one on the Ethernet and one on the WLAN. I will return to this at a later date.

Setting Up the Web Client

So, eventually, we have reached the point where we can write a few lines of code to run in a start-up file on the RPi.

To recap, the aim is to run multiple RPi boxes on the network, each acting as a web client, identifying itself to a web-server and receiving individually-tailored webpages to display. Periodically, the web client must request an update because (to keep the system simple) the server cannot push data to the RPis without being asked.

The salient point is that the RPi boxes should boot directly into a web browser (so-called kiosk mode) and should not need individually programming with an ID. The code we need to run is …

1) On power-up of the Raspberry Pi…

- Disable the energy-saving feature on the monitor; and its screen saver
- Establish a MAC address of the RPi
- Open a web browser in its full-screen mode
- Bypass the browser's initialisation screen
- Construct a URL and query string like **http://red-laptop.local/?mac=36-40-08**
- Ask the browser to fetch that address

Subsequent operations are managed by sending pages (and AJAX data) to the browser.

2) Periodically, via scheduled tasks…

- Find the browser's process id, kill the job and restart the browser
- Less frequently, reboot the RPi

A periodic reboot of the browser helps to avoid the problem that it is *pulling* pages from the server, rather than the server *pushing* them; which means that if the browser crashes, the server loses contact. Rebooting is useful during development but, in 'production' it causes a problem due to the browser not closing down cleanly. I will discuss this in *Part 3*.

Concluding Remarks

I have given an introduction to the Raspberry Pi and explained how to use SSH and FTP to access it remotely. I have outlined the code that we need to run. In *Part 3*, I will give some *actual* code examples. Further topics to discuss include…

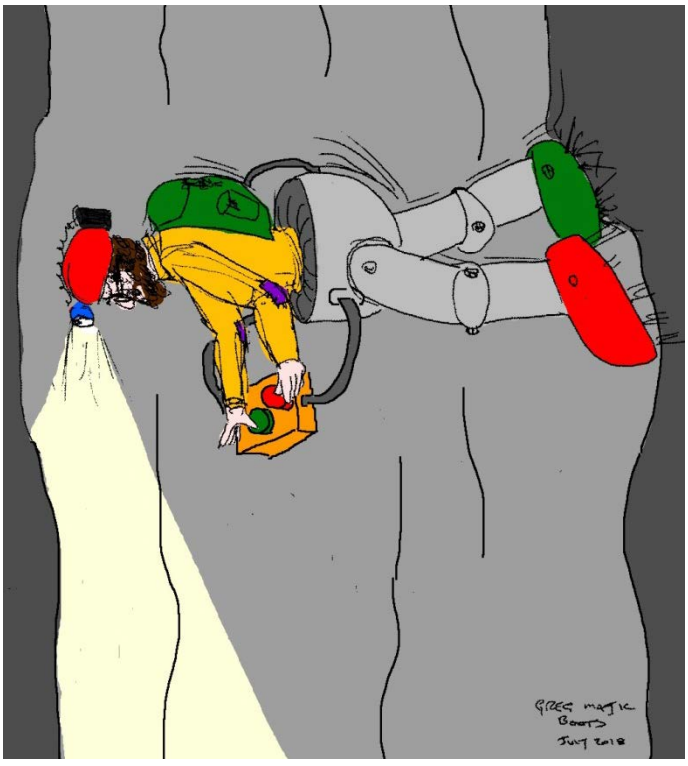
- How to upgrade the Raspberry Pi firmware
- An issue with Chromium – what happens if it doesn't shut down cleanly?
- How to test without a local webserver (using my demo pages on the Internet)
- How to run a webserver on the RPi
- A further outline of the 'Information Screens' system, explaining the client / server dialogue
- How to develop the 'Information Screens' system without using a router – using a simpler network switch instead
- More on the **.local** and **.home** domains

Reference

Gibson, David (2018). *The Hidden Earth Timetable Display*, CREGJ **102**. pp5-7 (June 2018)

THE ADVENTURES OF GREG

Illustration: Adrian Higgins
Words: Mike Bedford



1. Suction was surely the way to go up and down pitches and Greg was going to prove it. No more ropes, Petzl Stops or cow's tails; with a pump on his back, and a pair of suction boots, he'd be able to walk on vertical surfaces with ease. Greg knew his limitations and designing an automated control system was just asking for trouble but with just two switches all that hassle would be eliminated. What could possibly go wrong?

2. Greg soon got into the rhythm. Switch left boot off, move left foot, switch left boot on, switch right boot off, move right foot, switch right boot on, switch left boot off…

It really couldn't be simpler; in fact, Greg soon found that he barely needed to think about it. Move left foot, switch left boot on, switch right boot off, move right foot, switch left boot off.

… ooops.