Raspberry Pi Setup

A Raspberry Pi 2 Model B along with the official Raspberry Pi Camera Module and Raspberry Pi 2 Model B Case were purchased to serve as the basis of the portable portion of the project that would reside at the site that required technical support. With the Camera Module being more than capable of capturing high resolution and high frame rate video, the option of powering the Raspberry Pi 2 Model B via mini-USB, the low power requirements of the Raspberry Pi as shown by the article referenced above in the literature search, as well as the relatively cheap cost, it seemed the perfect device to build the system upon, even if only for a proof-of-concept if not for a fully realised solution. Along with this device, an EasyAcc Ultra 16000mAh power pack was procured to serve as a portable power source, a Microsoft Lifechat LX-3000 was borrowed from WVN to provide audio support for the voice chat portion of the project, a Dynamode Nano USB WL-700N-RXS Wifi Adapter was bought to allow the Raspberry Pi to connect to Wireless Networks, and a Kingston 16GB Micro SD Card was purchased to act as the Hard Drive of the Raspberry Pi.

The first step to set up the Raspberry Pi was to select and install an Operating System (OS). An OS called Raspbian (based on Debian Wheezy) was chosen due to its status as the officially supported OS for Raspberry Pi, and it’s widely adopted use by Raspberry Pi users. Once the image file for this OS was downloaded, it was written to the SD Card using a tool called Win32DiskImager. Once written, the SD Card simply needed to be inserted into the Raspberry Pi and it could be powered and used.

While a Raspberry Pi is simple to set up to operate via SSH, first-time set up of the Raspberry Pi requires a screen and keyboard to provide the interface to configure SSH, wireless settings, and a handful of other settings. A small portable screen and keyboard were provided by the developer for the duration of the development process to serve this purpose, though these accessories are not required for use in the finished product.

Once powered, the Raspiconfig utility was loaded and settings to enable the Camera Module, to enable SSH, and to enable WiFi were set using the utility. Once these settings were enabled, the Raspberry Pi was rebooted and connected to a Router (again supplied by the developer) via Ethernet cable to allow for access via SSH. Once powered up again, an attempt was made to configure WiFi settings, which is performed by editing the WPA\_Supplicant.conf file located in /etc/WPA\_Supplicant. The following snippet was appended to the file and saved:-

network={  
ssid="NETWORKNAME"  
psk="PASSWORD"  
}

Adding this snippet should have allowed the Raspberry Pi to connect to the router wirelessly, but this didn’t work. After a brief investigation it was found that the Nano USB WiFi Adapter was faulty, and would quickly overheat and malfunction. After a cursory search, it was found this was a common problem with this adapter. To remedy this, an Edimax EW-7811UN Wireless Nano USB Adapter was purchased as a replacement, chosen largely due to its purported reliability and widely adopted use in other Raspberry Pi projects.

Once the new WiFi adapter was installed, the Raspberry Pi quickly connected to the router, and SSH was possible over the Wireless Local Area Network (WLAN) provided by the router. SSH control was established using Putty, a common tool used for this purpose. A similarly common tool called WinSCP was used to transfer files to and from the Raspberry Pi.

The first part of the device portion of this project tackled was providing a live video feed over the internet. It was decided due to time constraints that it would be far better to find and use existing software to provide this functionality rather than to design and build something that likely already existed. After reading various documents (blogs, articles, and forum posts) regarding other Raspberry Pi projects used for IP Camera, WiFi Camera, and Security Camera purposes (purposes similar to those specified by this project), it was found that the most commonly used package was Motion. Motion provided a basic video feed accessible over the internet, and had a large variety of configuring options, including password protection, motion detection, among other features.

Installing packages to the Raspberry Pi is usually as simple as giving a single command t the Raspberry Pi, in this case apt-get install motion. Upon retrieving this instruction, the APT (Advanced Packaging Tool) provided by Raspbian would lookup the package in its stored Debian Sources file found in /etc/apt/sources.list (which is updated using apt-get update), and then calculate all dependencies of the package that weren’t already installed, and download and install the package and all its dependencies. However, this required an internet connection to work, and it proved impossible to connect the Raspberry Pi to the internet provided by the university, Eduroam. Unfortunately, the developer lived in university accommodation and had no other access to the internet. Instead, an attempt was made to install Motion and its dependencies manually by finding them in the Raspbian Repository Archive, writing them to the SD card, and de-packaging them on the Raspberry Pi. This did not work however, and progress was halted until internet access could be established. A full reinstall of Raspbian was then performed to remove the mismatched software and dependencies and to reset any configuration options that were changed in efforts to connect the Raspberry Pi to Eduroam.

Internet access was eventually achieved by registering the Raspberry Pi to another internet service provided by the university, Swis-lite, that is usually reserved for game consoles. Once connected, running the command apt-get install motion correctly installed the program to the Raspberry Pi. It was at this time that it was discovered that since the program was designed to be used as Security Camera software, it deliberately delivered only low frame-rate video that was remotely accessible, and came with options to begin recording locally when it detected motion. Tests showed that this frame rate was unacceptably low (2-3 FPS) and could not provide higher FPS due to the limitations of the software.

A deeper search of software available for the Raspberry Pi revealed a package called RPi Cam Web Interface. Though it utilised Motion to perform some of its functionality, it was designed not as a Security Camera-like application, but was designed to provide a full HD high frame rate camera feed, as well as a comprehensive interface that allowed the user to change the desired frame rate, capture resolution, brightness and contrast settings, and a host of other configuration options, as well as access to some system tools, including buttons to shut down or reboot the Raspberry Pi (this interface would serve well to allow limited remote operation of the Raspberry Pi without full SSH connection). All of this functionally was provided as a web page viewable in any browser and accessible by the IP address of the Raspberry Pi through port 80 (meaning no port forwarding was required). This software seemed like it would well satisfy the video capabilities of the project.

As it was not an archived Raspbian package, installation of RPi Cam Web Interface was not as simple as that of Motion, and required the source code to be cloned from GitHub and installed using the following set of commands:-

git clone https://github.com/silvanmelchior/RPi\_Cam\_Web\_Interface.git

cd RPi\_Cam\_Web\_Interface

chmod u+x RPi\_Cam\_Web\_Interface\_Installer.sh

./RPi\_Cam\_Web\_Interface\_Installer.sh install

At this point an installer menu provided access to installing the program itself as well as a pre-configured Apache client to provide the web capabilities.

From this point on, RPi Cam Web Interface required no further configuration. It provided a sufficiently high quality video stream that was expandable to see more detail, as well as all the interaction utilities it advertised in advance.

Next, a lightweight application that was developed alongside the Camera Registrar server was installed to the Raspberry Pi. This program would access a file called details.conf and register itself to the server with the location specified on the first line of details.conf, and receive and store a unique ID. Once registered, the application then pings the Registrar every second to inform it that the Raspberry Pi is online and to allowed the Registrar to register the IP address by which the device could be accessed.

TODO

Voice chat functionality

Red = references required

Blue = add to abbreviations section