Raspberry Pi Security Camera System

Sean Rider

**Overview:**

**Project:** The idea behind this project was to create a system of wireless security cameras that would back up to a master computer via WiFi to prevent loss of footage in the event of camera failure. This project will be double sided; programming will be necessary for both the camera software itself as well as for the master computer to receive and handle the incoming footage and in the future control the cameras remotely with greater ease of access than SSH through the terminal.

**Importance:** This project is designed to provide a cheap alternative to pricey home security systems and to demonstrate my knowledge of networking concepts.

**Learned:** This project taught me a lot about the Raspberry Pi hardware and Linux command line environment. It also taught me about things like automation using Chron jobs. It was also good practice for my python writing skills.

**Introduction:**

This project was designed to provide a cheap and secure alternative to home security camera systems. The main purpose of the project was to create portable, wireless security cameras using Raspberry Pi 3s connected to a battery and WiFi network to record footage and back it up to a master computer located elsewhere. This is so that in the event of camera error, failure, or damage, the footage which reveals what happened to the camera will be accessible. Below is an outline of how the security system should operate.

* Camera is powered on and connected to the internet. Recording begins.
* One hour of footage is recorded.
* Footage is backed up via WiFi. Second recording begins.
* Footage recording and backup loop continues.
* Footage is deleted from the camera’s storage as soon as it has been backed up.
* Footage is deleted from Master storage after a certain amount of time unless it is saved.

**Sections:**

This project consisted of two main sections: python scripts running in a Linux environment and batch scripts running on Windows.

**Python:** Object oriented programing language which can be compiled and run by the Linux command line. Easy to pick up syntax and good readability make this programing language easy to pick up. The scripts in this project were used to control the functioning of the camera hardware. They set the interval of recording, initialize the camera and start and stop the recording. They also are used to create file names based on the date of filming for easy lookup later on.

**Batch Scripts:** Text files with commands that can be run by the Windows command line at time of execution. Set using Cron jobs to run automatically to back up the footage files using WinSCP’s built in command line tools, and to delete files which are of a certain age to prevent filling up the storage device.

**Networking:**

The networking component of this system required several things. The interaction between the Lightweight Raspbian distro that the PiCameras are running on and a Windows 10 environment for the Master PC (hopefully can be expanded to IOS and Linux once desktop app development is completed). In both components security was important, because a security camera system is not something that you would want to get hacked easily. The windows section of the security aspect was already covered as my home PC has been set up with anti-virus, firewall, and VPN since it was set up. Securing the Pi’s took a bit of research but was also very simple. The Linux distro is set up with a default username and password which can be easily googled. Therefore the first step is changing the password to something secure. Next the firewall needed to be enabled. I did this using a software call Uncomplicated Firewall which basically acts as a front end to editing the IPtables stored on the OS. Iptables are what are used to manipulate netfilter, which is the built in firewall on Raspbian (by default this firewall is set to allow all incoming connections). From Ubuntu’s website:

Getting started with ufw is easy. For example, to enable firewall, allow ssh access, enable logging, and check the status of the firewall, perform:

$ sudo ufw allow ssh/tcp

$ sudo ufw logging on

$ sudo ufw enable

$ sudo ufw status

Firewall loaded

To Action From

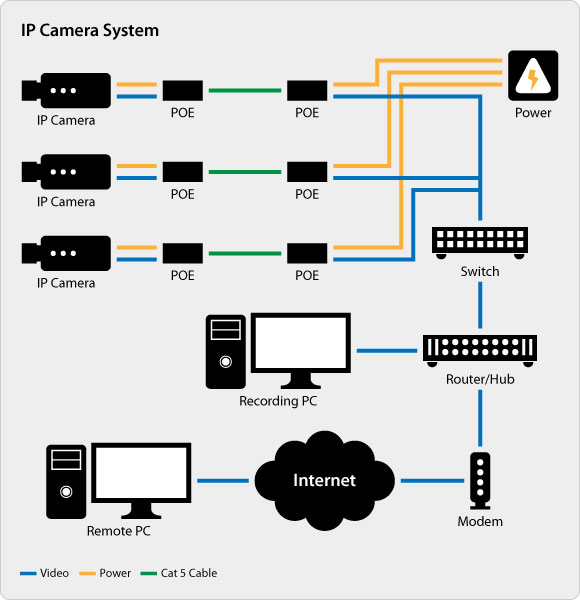
-- ------ ----

22:tcp ALLOW Anywhere

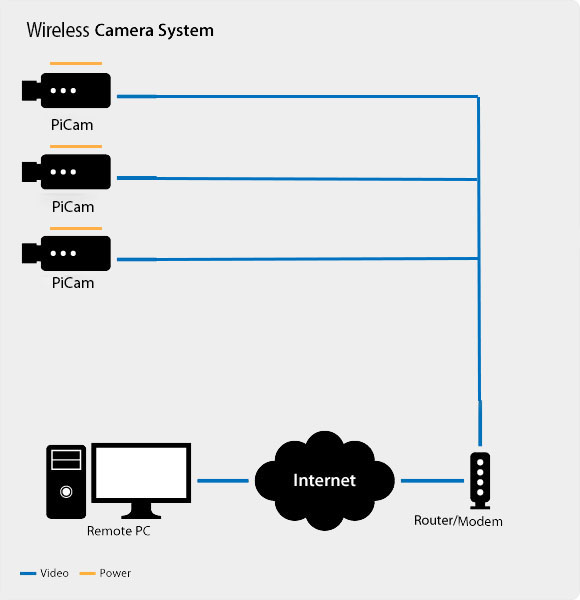
This shows just how simple setting up the firewall on a Pi can be. Using these tools I configured mine to allow incoming SSH connections only (will have to allow incoming requests through another port when setting up live-view through a browser eventually).

Now that both the Pis and the Master PC are relatively secure, the next step was getting them to work together to allow seamless and endless recording and backing up of footage possible using just a WiFi network. This was made incredibly easy via WinSCP’s built in command line library which makes automation of SSH connections a breeze. Using the username, password, and IPaddress of the PiCamera, as well as an SSH hostkey and the file location of the footage on the camera, WinSCP can be used via scripting from the command line to open an SSH connection with the Pi and copy files to the MasterPC. From there it is short step to automate the running of that script using a Cron job to backup the files at an interval which seems appropriate. I chose every 1 hour just because that is how long each recording is and I would rather the bulk of the files get backed up promptly, so that the small SD card holding the Pi OS is not in danger of becoming full. I want to improve upon this process by setting the backup to happen instantly when a recording has finished or been interrupted.

To illustrate how the Raspberry Pis are simplifying the security camera solution let us first look at a diagram of a traditional wired camera system:



As you can see there is much extra hardware and wiring necessary when the cameras are not built as standalone units. Giving the cameras the processing and storage power of the RPi gives the system much more flexibility when it comes to organization. Power lines and video lines must be implemented in location to connect the system. In the diagram below you can see how simplified the new system is. No Cat 5 cable is necessary as long as the WiFi network can handle the data transfer of video files. The Master PC handles switching and camera control, and the cameras themselves handle their own recording and power needs. Not shown in the diagram are firewalls on both the Master PC as well as each camera. This new model creates a portability that is limited only by access to Wireless internet.



**Conclusion and Future Plans:**

This project was basically for laying the bare bones of the security camera system. Much more is planned for its future. Eventually I want to implement an installable program file which will be installed by the user and will automatically handle the operation of the cameras as well as file management etc. Live views of each camera will be implemented so users can view their feed live without interrupting recording. This will most likely be done through the app as well as through web browser so cameras can be checked even when they are not home. The Raspbian distro and settings as well as the python script installation should also be more fully automated. After User Interaction is taken care of a handful of security measures will be implemented. Among them are motion detection, camera failsafe (when one camera fails the others backup instantly and the user is sent an alert).