# Field camera technical paper

By - Larry Bonnette

**Scope:**

For most of our members the flying field is a good distance away from home. Therefore, it would be nice to have a way for our members to see if there is anyone at the field before making that long drive.

The purpose of the field camera is to provide members with the ability to see if there are people at the field.

**Access:**

On the Tri-countybarnstormer.com web site in the "Additional Features" drop down menu you will find the "Field Camera" selection.

This selection will take you to the Field Camera "near live" page.

The page displays the latest photo of the interior of the pavilion facing south.

The camera is mounted on the impound area. The photo is updated every 5 minutes from the hours of 7:00 AM until 8:00 PM, and the page itself auto refreshes every 1 minute.

**Technical:**

The camera is capable of providing live streaming video at a resolution of 720 or 1080.

The camera can provide snapshots at maximum interval of once every 7 seconds.

The snapshot quality can be varied to adjust the size of the jpg file.

The snapshot is set to medium quality (to save data usage).

Internet access is through an At&t "Unite" cell phone hotspot.

The cost of a 2GB data service is $25 per month.

Because part of the goal is to keep the data costs low, we opted to provide only snapshots instead of streaming video to our web site.

The max interval of snapshots coming from the camera is 7 seconds. Because sending a photo to the web site every seven seconds would eat up too much data, we had to find a way to buffer the photos and display only the newest photo every 5 minutes.

Why every 5 minutes?
We are limited to 2GB of data transfer to our web site due to cost limitations.
Each photo consumes 53K of data passing through the hotspot heading to our web site. One photo every 5 minutes will consume 636K of data per hour. 636 K of hotspot data times 13 hours a day is 8.2 MB a day. We consume 8.2 MB of hotspot data per day, and that times 31 days is 256 MB (under our 2GB limit).

Keep in mind that the 53K number is an average. Some data is larger (up to 1 MB) some is smaller.

So, how do we limit the data going to the web site to one photo every 5 minutes?
We added a Raspberry Pi.

A Raspberry Pi is a small computer (3.3" x 2.2") that runs Linux. Using this computer, we can allow the camera to connect to it via FTP and send snapshots at 7 second intervals to the Pi.

We can then program the Pi to wait 5 minutes and then, find the newest photo and send it to the web site.

The Pi uses FTP to connect to the Tri-countybarnstormer.com web site. The camera uses FTP to connect to the Pi. Using the FTP link, the camera transfers one photo every 7 seconds to the Pi. The Pi waits 5 minutes and then sorts through all of the photos sent in the last 5 minutes and finds the newest one. The Pi uses FTP to transfer that photo to our web site. When you call up the camera page on your browser you see the newest photo that was sent.

The camera page has a "refresh" set on it to force your browser to refresh every 60 seconds. This is so you can stay on the page and get the photo updates as they come in from the Raspberry Pi.

At the end of the day the camera has dumped 6,685 photos onto the Pi consuming approximately 349 MB of space. We can't let this go on or the 16 GB disk drive on the Pi will fill up. So, we created a program to remove all of the files at the end of each day. We limit the hours of operation to daylight hours so we don't have to pay for transmitting photos at night.

It may be valuable to store more than 1 days' worth of snapshots on the Pi. So, to try and preserve 1 weeks' worth of snapshots, at midnight each day. We copy all of the snapshots to separate directories in the Photo account on the pi. These directories are "Arch1 thru Arch7". At the end of the week we overwrite the oldest data.

Camera data operations are accomplished on the Raspberry Pi using Perl scripts.

These Perl scripts are launched at certain intervals using "cron" on the Pi.

The software utility "cron" is a time-based job scheduler in UNIX computer operating systems. Unix system managers who set up and maintain software environments use "cron" to schedule jobs (commands or shell scripts) to run periodically at fixed times, dates, or intervals.

"cron" is driven by a crontab (cron table) file, a configuration file that specifies shell commands to run periodically on a given schedule.

There are 3 Perl scripts that are used to manage the camera data going to the web site.

movepic.pl – is a Perl script that sorts all of the photos currently stored (by the camera) on the Pi. From this sorted list the script finds the newest photo and selects the photo that was saved just before the newest. The reason it chooses the photo just before the newest is because if the newest is chosen it could possibly be caught in the middle of a transfer from the camera and would therefore be corrupt. The script then uses FTP to copy this selected photo from the Pi to the tri-county web site. This script is executed using "cron" every 5 minutes.

cleanup.pl – is a Perl script that saves all current photos into an archive directory. There are 7 archive directories (arch1-7) and they are located in the "photo" account on the Pi. Each night at midnight this script copies all of the photos from the current day's photo directory into an archive directory. The script then deletes the current day's photos in order to make room for the next day's photos. At the end of one week there are 7 archive directories' that contain one week's worth of photos. The oldest archive directory is deleted. This script is executed using "cron" every night at midnight.

rstrt.pl – is a Perl script that restarts the Raspberry Pi. After running the Pi during the testing period, I found that the Pi becomes unstable after about 16 hours of operation. To get around this I use the restart Perl script to reboot the Pi twice a day. Once at 1:00 PM and once at 1:00 AM. This script is executed using "cron" twice a day (once at 1AM and once at 1PM).

**Costs:**

25-foot extension cord ……. $14.97
Power strip …… $3.97
Raspberry Pi …… $35.99
Pi Power supply …… $9.99
16 GB Micro SD card …….. $5.36
Hot Spot ……. $69.99
Camera ……. $119.99
USB Cable …….. $2.63

Hot Spot service (consumed during testing) 4 months at $25/mo. = $100

Printed Pi case = $0
Printed Hot Spot case = $0
Printed camera cable cover = $0

Total = $262.89 (not including hot spot service for testing). Development time 4 months.

Issues corrected:
Photos scheduled to display at 7:00 AM were not displayed until 7:05. This occurred because the camera was not sending out photos until 7:00. When the script ran at 7 there were no photos to choose from because they were all deleted at midnight. To fix this I set the camera to start sending photos at 6:55 AM. This provides approximately 40 photos to sort through at 7:00 AM.

During an evening of temperatures around 25 degrees. The Hot spot stopped working. I visited the field and warmed the hotspot up a few degrees and it started to work OK. I think that if I place a small 40 to 60-watt incandescent lamp near the hotspot it should keep it warm enough to continue to work when the weather gets cold. Update: During a second week of cold weather hovering around 20 degrees. I added a small lamp with a 20-watt bulb. The hotspot remained up and running throughout the cold period.

Operation of the weather station had been steady for around 2 weeks. I noticed that if we encountered cloudy days of more than 3 a week the battery would die. I replaced the 2200 mah battery with a 6600 mah battery and that seemed to fix the problem.

The weather proof box is suffering from condensation (inside). I may have to drill a vent hole in the box.