RPi-Cam-Web-Interface

From eLinux.org

RPi Cam Web Interface is a web interface for the Raspberry Pi Camera module. It can be used for a wide variety of applications including surveillance, dvr recording and time lapse photography. It is highly configurable and can be extended with the use of macro scripts. It can be opened on any browser (smartphones included) and contains the following features:

- View, stop and restart a live-preview with low latency and high framerate. Full sensor area available.
- Control camera settings like brightness, contrast, ... live
- Record full-hd videos and save them on the sd-card packed into mp4 container while the live-preview continues
- Do timed or continuous video recording with splitting into fixed length segments
- Take single or multiple (timelapse) full-res pictures and save them on the sd-card (live-preview holds on for a short moment)
- Preview, download and delete the saved videos and pictures, zip-download for multiple files
- Trigger captures by motion detection using internal or external detection processes.
- Trigger captures by many scheduling-possibilities
- Circular buffer to capture actions leading up to motion detection
- Control Pan-Tilt or Pi-Light
- Shutdown/Reboot your Pi from the web interface
- Show annotations (eg timestamp) on live-preview and taken images/videos
- Supports selection from 2 cameras when used with a compute module

IMPORTANT NOTE: This is for the Raspberry Pi camera only. It does NOT support USB cameras.

It's been programmed by silvanmelchior as a client for RaspiMJPEG in 2013. Since then, thanks to the help of many other programmers, it has become the best interface to control the RPi Cam over your browser.

Remember, anyone can create an account on here and add to this wiki.

Contents

Installation Instructions

Basic Installation

Warning: The installer will replace various files, so backup all your data.

See also Addition section for tips on installing from scratch.

Step 1: Install Raspbian on your RPi
Step 2: Attach camera to RPi and enable camera support (http://www.raspberrypi.org/camera)
Step 3: Update your RPi with the following commands:

```
sudo apt-get update
sudo apt-get dist-upgrade
```

Occasionally if camera core software updates have been done then a sudo rpi-update may be used to benefit from these before they become available as standard.

Step 4:

For Jessie Lite run sudo apt-get install git

Clone the code from github and enable and run the install script with the following commands:

```
git clone https://github.com/silvanmelchior/RPi_Cam_Web_Interface.git
cd RPi_Cam_Web_Interface
./install.sh
```

Older versions needed the scripts to be made executable with chmod u+x *.sh If you get permission denied while trying to run the install scripts then try that step

5 separate scripts are provided to do separate installation and maintenance functions.

The scripts are

- install.sh main installation as used in step 4 above
- update.sh check for updates and then run main installation
- start.sh starts the software
- stop.sh stops the software
- remove.sh removes the software

- To run these scripts make sure you are in the RPi_Cam_Web_Interface folder then precede the script with a ./
- E.g. To update an existing installation ./update.sh
- E.g. To start the camera software ./start.sh
- E.g. To stop the camera software ./stop.sh

The main installation always does the same thing to simplify its logic. It gathers all user parameters first in one combined dialog and then always applies the parameters as it goes through the process. Autostart should be yes if you want this software to start automatically when raspberry boots up. jpglink should normally be no. Change it to yes if you have external software that needs direct access to the cam.jpg image file.

A phpversion parameter provides for a choice on which php version to use (5 or 7). php5 was used up until 2017-09-22. If there any difficulties using 7 or when upgrading older systems then try using 5.

A q (quiet) parameter may be used to skip this and give an automatic install based on config.txt All parameters are always in the config.txt file, a default version is created if one doesn't exist and is then changed just once after the initial user dialog. The installation always tries to upgrade the main software components and then functionally goes through the configuration steps for each area like apache, motion start up.

After the setup finishes it offers to start the camera system. It will also start on a reboot if autostart was configured.

Step 5: Use it

Open up any browser on any computer in your network and enter the url to access the camera web site. This will be http://ipAddress:port/subfolder. If the port had been left at default 80 during install then this may be left out. Similarly the subfolder (default html) can be left out if that was cleared during the install. So for a port 80, no subfolder install the url becomes http://ipAddress

IMPORTANT NOTE: If you need to change any files then you need to do these in the run time version (e.g. in the /var/www/html folder or /etc/raspimjpeg). Changing files in the RPi_Cam_Web_Interface folder has no immediate effect. These are just copies used during the install.

Web Server choice

The install offers a choice of apache/nginx/lighttpd for the web server to be used. It is best to choose one after the original rasbian install and stick to it as re-installing with a different choice is likely to lead to confusion with multiple servers fighting over port usage.

If you want to change to a different server then it is best to start from a fresh rasbian install.

Camera Choice

Remember, USB cameras are not supported!
However, there are a number of good alternatives out there that offer a wider field of view. For example:

Raspberry Pi Camera Board - Night Vision & Fisheye 160° Lens - https://www.modmypi.com/raspberry-pi/camera/camera-boards/raspberry-pi-camera-board-night-vision-and-fisheye-160°-lens-5mp/?limit=50
Raspberry Pi Camera Board - Fisheye 160° Lens - https://www.modmypi.com/raspberry-pi/camera/camera-boards/raspberry-pi-camera-board-fisheye-160°-lens-5mp/?limit=50
Raspberry Pi Camera Board - Fisheye 222° Lens - https://www.modmypi.com/raspberry-pi/camera/camera-boards/raspberry-pi-camera-board-fisheye-222°-lens-5mp/?limit=50
Raspberry Pi Camera Board - Night Vision "IR-CUT" - https://www.modmypi.com/raspberry-pi/camera/camera-boards/raspberry-pi-night-vision-camera-ir-cut/?limit=50

Startup Behaviour

To change the default startup-settings, edit the config-file /etc/raspimjpeg. If you want to disable autostart completely, navigate back to the directory, where you cloned the git-repo in Step 4 and run one of the following command: ./RPi_Cam_Web_Interface_Installer.sh autostart_no --> the interface doesn't start at startup, you need to run a command to use it (commands below) ./RPi_Cam_Web_Interface_Installer.sh autostart_yes --> the interface starts at startup and takes control over the camera (standard)

To temporarily start/stop or deinstall: Navigate back to the directory, where you downloaded the installer in Step 4. If you want to stop the interface temporarily, run "./RPi_Cam_Browser_Control_Installer.sh stop". To restart it, run "./RPi_Cam_Browser_Control_Installer.sh start". If you want to remove the interface completely, run "./RPi_Cam_Browser_Control_Installer.sh remove". Attention: It removes all files in /var/www.

!! it seems that we have to use "./**RPi_Cam_Web_Interface_Installer.sh**" for "stop" and "start" instead of "./**RPi_Cam_Browser_Control_Installer.sh** stop"

You may be have to do

```
sudo chmod 755 /etc/rc.local
and restart
```

Support

There is a very active (and long) thread at raspberrypi.org. We highly recommend you use the thread search facility to see if your question has been previously covered, but if not please feel free to post your questions there.
Forum: http://www.raspberrypi.org/forums/viewtopic.php?f=43&t=63276

Project itself: https://github.com/silvanmelchior/RPi_Cam_Web_Interface
RaspiMJPEG: https://github.com/roberttidey/userland/tree/master/host_applications/linux/apps/raspicam

Special thanks

To btidey for many, many new features
To jarrah31 for this wiki
To jonar for the github-repo
To James Cooke for the styling
To slabua for many fixes/improvements
To everybody else who helps on github, in the forum or here to develop the RPi Cam Web Interface

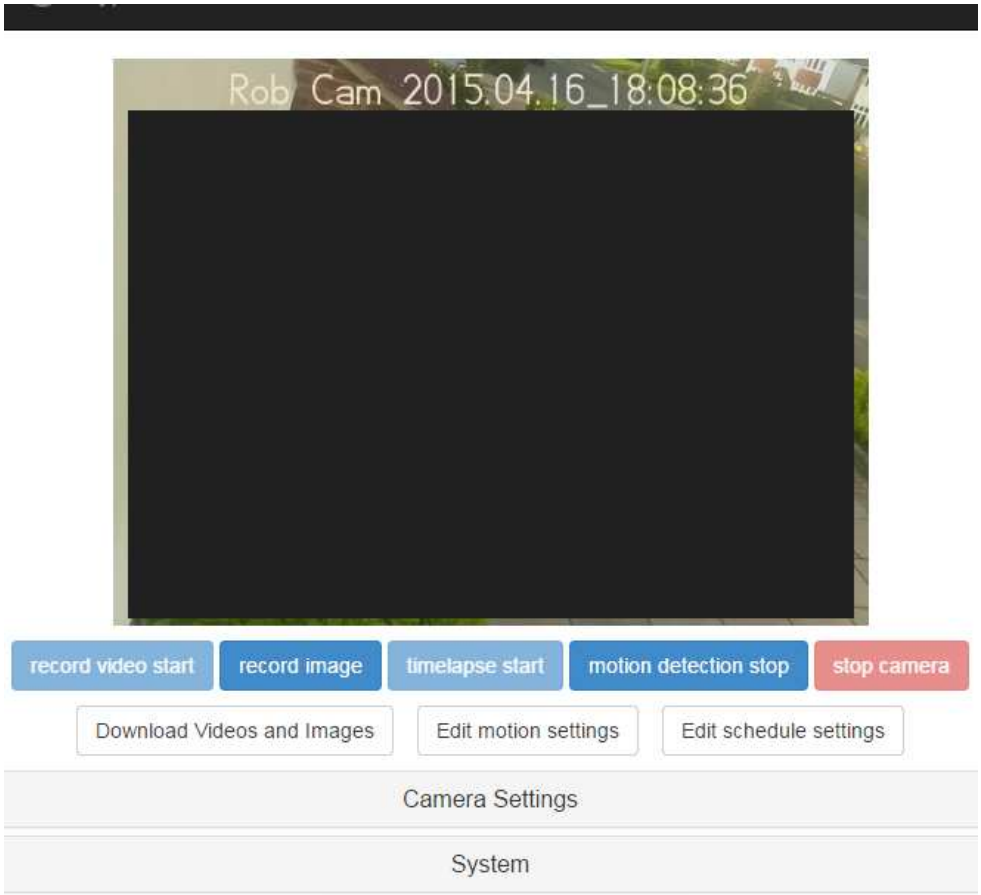
Ps: If you like Silvan Melchior's work and have some extra-money ;) :
Bitcoin: 398h4RVQhptrgN7eT9abAStCe1yu7zxBoV
PayPal: https://www.paypal.com/cgi-bin/webscr?cmd=_donations&business=HN25F6V378FB4&lc=CH&item_name=Private&currency_code=CHF&bn=PP%2dDonationsBF%3abtn_donateCC_LG%2egif%3aNonHosted

PPs: If you like Bob Tidey's (aka btidey from the forum) work and have some extra-money ;) :
PayPal: https://www.paypal.com/cgi-bin/webscr?cmd=_s-xclick&hosted_button_id=FWX7UL9NSNM6L

Usage

Basic usage

Navigating to the main web site brings up a medium resolution live view of the camera, access to a number of control buttons, and access to camera settings and system controls. On all screens the top navigation bar steps back one view.



The basic controls allow for capturing single images, videos, or time lapse sequences. It can also be put into a motion detection state where motion will trigger activity like a video recording.

The buttons under the main controls allow for Downloading and previewing any captures, plus access to motion detection and scheduling set up.

Below this are Camera settings and a system control bar.

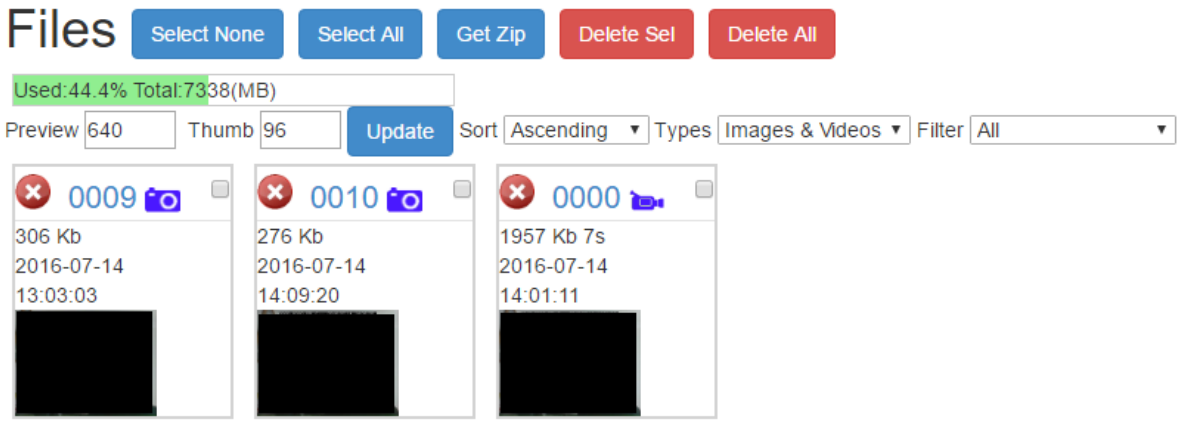
Clicking on the image will toggle between normal and full screen mode. BY default it starts up in normal mode but this can be altered by using the config variable fullscreen.

Clicking on Camera settings gives access to a wide range of camera set up controls from controlling annotation through to exposure set ups and image and video formats.

The system control bar allows for selection of streaming mode, a simple button on/off toggle, shutdown and reboot of the system, clearing settings and selection of custom styles. By default the system generates the live preview as a continuous set of captured images as this has maximum browser compatibility. MJPEG streaming may be selected but this may not work on all browsers. This is a per browser setting. The Simple button on/off controls whether a simple / full button appears next to the live preview. If enabled this allows the screen to be toggled between a simple preview only and the normal full display. Full will be also determined by any user level privileges set up.

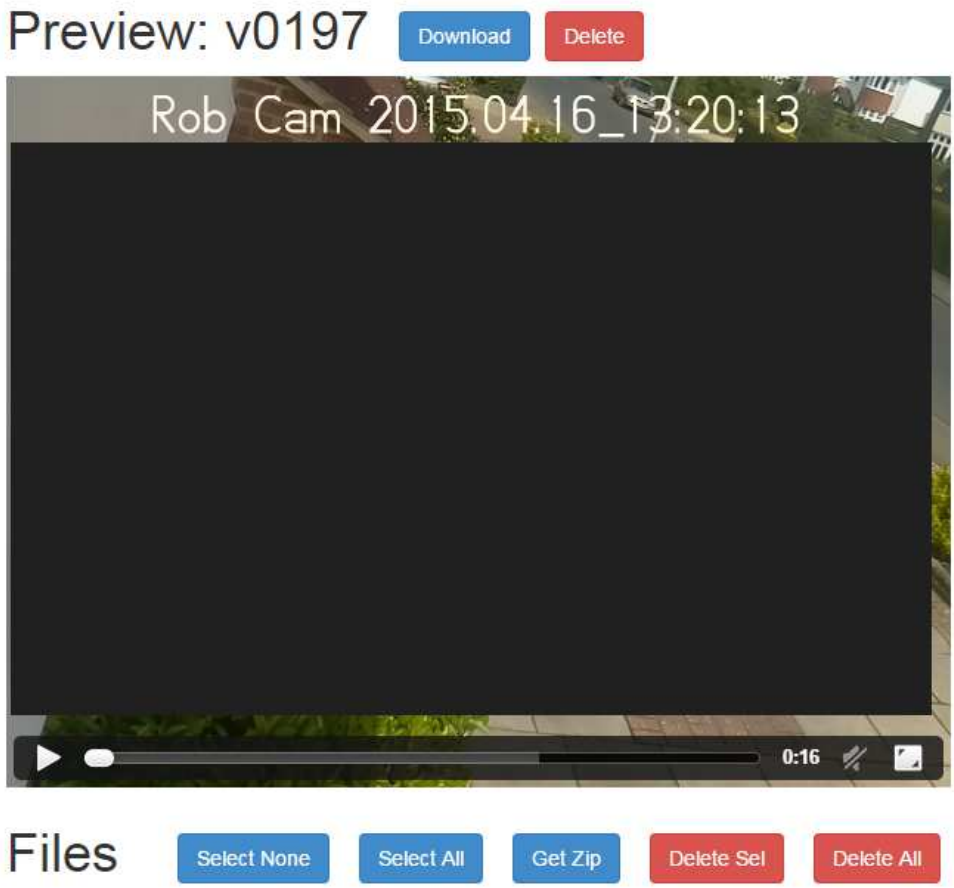
The custom style bar shows any extra styles that have been added. As installed there is the normal white default look, and a dark night mode.

Download Clicking the Download Videos and images button shows all previous captures as a series of thumbnails.



These may be selected individually or as a set and either deleted or zip downloaded.

Clicking on an individual thumbnail brings it up in a larger preview pane and videos may be played from here. Next and previous buttons step forwards or backwards through the files. The file number in each thumbnail is also a link which will open the file in a new tab in the browser.



The selected file may be deleted or downloaded as a single file. If it is a timelapse sequence it will still be zipped as it contains multiple files.

The size of thumbnails, size of main preview, image sort order, image type and a time filter may be updated and this will be remembered as a per browser setting. Sort, Type and Filter take effect when the setting is changed. Sizes need the update button to be pressed after new values are entered.

Scheduler

The same program file schedule.php is used both as the web facing settings interface and the run-time daemon which does the automation. All the settings used are contained within the schedule.json config file. **Important** The scheduler daemon will normally be stopped and started by regular update methods. If you copy a new scheduler.php in as a patch then the old scheduler will still be running and maybe out of sync. So Stop and Start the scheduler if you do this to make sure that the daemon is the new copy.

Save Settings | Backup | Restore | Show Log | Download Log | Clear Log | Stop

Parameter	Value	Parameter	Value
Fifo_In	/var/www/FIFO1	Fifo_Out	/var/www/FIFO
Cmd_Poll	0.03	Mode_Poll	10
Management_Interval	3600	Management_Command	
PurgeVideo_Hours	0	PurgeImage_Hours	0
PurgeLapse_Hours	0	GMTOffset	0
PurgeSpace_ModeEx	Select Mode Off	PurgeSpace_Level	10
DawnStart_Minutes	-180	DayStart_Minutes	0
DayEnd_Minutes	0	DuskEnd_Minutes	180
Latitude	52.00	Longtitude	0.00
Max_Capture	0	DayMode	Select Mode All Day
AutoCapture_Interval	0	AutoCamera_Interval	0

Time Offset: 0 | Sunrise: 05:43 | Sunset: 18:27 | Current: 09:40 | Period: AllDay

Period	Days Su-Sa	Motion Start	Motion Stop	Period Start
AllDay	✔✔✔✔✔✔✔			md 1;em night

Command reference

Different rows will be shown according to the day mode selected. For example if Fixed Times is selected then 12 time based rows appear.

Max_Capture	0	DayMode	Select Mode Fixed Times
AutoCapture_Interval	0	AutoCamera_Interval	0

Time Offset: 0 | Sunrise: 05:43 | Sunset: 18:27 | Current: 09:44 | Period: 1

Period	Days Su-Sa	Motion Start	Motion Stop	Period Start
09:00	✔✔✔✔✔✔✔			md 0;em night
10:00	✔✔✔✔✔✔✔			md 0
11:00	✔☐✔✔✔✔✔			md 0
12:00	✔✔✔✔✔✔✔			md 0
13:00	✔✔✔✔☐✔✔			md 0

The Scheduler will normally be started on boot up and can be left running all the time. A Stop button shows when the background scheduling program is running and turns into a start button if it is not. Normally it should just be left running. Note that if it is stopped then motion start stop triggers will not be actioned as they pass through the scheduler. If the software is updated without rebooting then stop and start scheduler to make sure the new version is running.

Save Settings saves the settings to schedule.json. Changes will be passed to the background program and take immediate effect. These settings may be backed up and restored.

- scheduleLog is the file where scheduling activity is recorded. The contents are displayed when the Show Log button is used and the log may be cleared from there.
- Fifo_In defines the named pipe that schedule monitors. It should be the same as where motion sends its commands. Fifo_Out is where scheduler sends its commands to raspimjpg Do not change these without good reason. Other configs are relying on these and would need to be changed as well.
- Cmd_Poll is how often the scheduler checks the Fifo_In for incoming commands. It should be kept quite short to avoid unnecessary delays.
- Mode_Poll is how often the scheduler checks for changes between the 4 main daily periods. Setting it to 10 means there might be a 10 second delay in determining when day starts for example.
- Max_Capture determines the maximum capture period. If motion sends a start command and doesn't send a stop command then the scheduler will automatically stop the capture after this interval. This can be used to make all recordings this length by configuring motion never to send stop commands. The scheduler will then always time out and stop the recording. A value of 0 means no timeout.

- Management_Interval causes Scheduler to run a task every Interval (seconds). This is an optional user command plus automated purging.
- Management_Command is a macro facility which will be run every Management_interval. macros are commands held in the macros folder in the web site typically shell scripts. This is a change from previous operation where any direct system command could be executed. The system installer must put any commands in the macros folder and give them execution rights. Take care when using this facility; test out any macros first.
- Purge settings. See below.
- Latitude, and Longtitude define where the camera is and allow the sunrise and sunset times to be calculated throughout the year.
- GMTOffset adjusts for sunrise / sunset calculation. Set in hours or TimeZone string
- DawnStart_Minutes through to DuskEnd_Minutes divide the day up into 4 periods based on sunrise and sunset. Dawn starts at sunrise + DawnStart_Minutes so would normally be negative to make Dawn start before sunrise. Similarly for the other 3 numbers. Night is from DuskEnd to DawnStart the next day.
- DayMode – This provides 3 different types of scheduling.
  - Sun based splits the day up into 4 periods (Night, Dawn,Day, Dusk) based on sunrise and sunset calculations and the offsets.
  - All Day uses just the Day settings for the whole 24 hour period.
  - Fixed Times splits the Day up into up to 6 periods based on fixed times. The commands in force are those for the time just less than current time. These do not have to be in order.
- AutoCapture_interval - If non zero sets the scheduler to issue capture starts at this interval. This is used in conjunction with Max_Capture to set up a repeating sequence of fixed length recordings of maxcapture duration at the autocapture interval. Note that Max_Capture should be set a little less than the AutoCapture_interval to make sure the video is stopped before the next is due to start. 1 second less should be sufficient.
- AutoCapture_interval - If non zero then this sets the system up to only run the camera system when there is an active browser client attached. The interval determines how long the camera stays on after the browser stops watching the stream. The camera will automatically start when the browser next views the stream. Only use in situations where you expect the browser to be used normally. Scheduling and motion detection will not work in this mode if no browser is attached.

The period table allows mode changes and start and end commands to be different in each of the daily periods. Times are to the nearest minute and there can be an additional delay of Mode_Poll seconds before a period change is detected. Each row in the table also has weekday checkboxes. Commands are only issued on days where they are checked. This may be used to skip or select particular commands on different days.

- Motion Start are used to start captures when a motion trigger start is received. If left blank (e.g. at night) then no capture happens when motion is detected.

- Motion Stop are used to end captures when a motion trigger stop.
- Period Start are used to send in commands at the start of each daily period. So, for example they may be used to control motion detection and change camera settings. Changing to night mode for example extends the usefulness of the camera in dusk and dawn periods. They may also be used to start and stop recordings at the beginning of each period.

Note the scheduler is calculating the day periods based on local time conditions. The raspberry pi should have the appropriate time zone set in raspi-config. This may be checked by issuing a date +%R command line which should show local time. It is also displayed on the schedule settings page.

Example: Enable motion detection all day
When scheduler starts up (or if the camera system is restarted) then it detects which period it is in and performs the commands configured in the period start for that period.

1. In All Day mode then there is only 1 period so any commands in period start are always sent. md 1 will therefore start motion detection
2. In Sun based or Fixed Times mode then the period start commands for the current period are sent. Say you are in Sun based and put md 1 in Day and md 0 in Dawn, Night, Dusk. After a reboot if the time is in a day period then motion will be started, else it will wait till next day period starts.

Example: Send an email when motion is detected
If you want the email sent when a capture is done then use the job macro facilities. For example, a completed video capture will trigger end_box.sh if a file of this name is in the macros folder. The file should have execute permissions and preferably be owned by www-data. Job macros are passed the capture file name so there is no need for wild cards. This name is actually the capture file so if you want the thumbnail then the script needs to convert it. E.g.

```
#!/bin/bash
list=( $1*.th.jpg )
thumb="${list[-1]}"
```

$1 is the parameter passed to the script and $thumb will be the thumbnail name.

Purging

The scheduler has facilities for removing old files automatically. There are two mechanisms which are both checked and used at each Management_Interval. First the 3 PURGE_HOURS (one per file type) are checked and any file older than this is removed. So if PurgeVideo_Hours is set to 98 then any video captured more than a week ago will be removed. Setting the values to 0 disables this . This type of purging is useful when there is plenty of storage but you just want to remove old material. The second mechanism purges based on filing system space available or media space used. The detailed mechanism is selected with PurgeSpace_Mode which can be Off, Min Space % or Max Usage %, Min Space GB, Max Space GB. Min Space means that the available remaining space is checked and older files are removed until there is at least the level of free space set available on the filing system. Max Usage means that older files are removed so that the total space used by the

media is less than the level set. The level in either case may be set in GB or as a % of the total filing system size depending on which Mode was selected. Note that this is different to the previous scheme where % could be put in as a unit. % will now be ignored if entered in the level.

Logging A button Show Log shows scheduler and also raspimjpeg activity (if raspimjpeg log is set the same as scheduler). The log may be downloaded and cleared.

The scheduler has a facility for log file maintenance. This is controlled by log_size which may be set under the camera settings. If this is 0 then no logging is done and no maintenance is needed. Otherwise log_size sets the maximum number of lines in the log. This is checked every management interval and any excess old log lines are removed.

[] are scheduler events, {} are raspimjpeg events

Could I use the Scheduler to activate timelapse full-res picture capture on a sun-based schedule? The scheduler is fairly general purpose and not constrained to motion capture video.

Any sequence of the supported commands can be put into the motion start, motion stop, and period start fields. This can include changing camera settings, taking still images, videos, or controlling time lapse sequences.

Motion Start and Motion Stop commands get executed whenever a trigger (1=Start, 0=Stop) get entered into the scheduler FIFO1. Normally these would come from the motion detection logic but can come from anywhere (e.g. PIR motion detectors, door triggers etc).

The Period Start commands get executed whenever a period change is detected. In All Day mode there is only one set, but time based and Sun based allow different commands at different periods of the day. In a lot of cases these are used to determine what camera settings to use or whether to enable motion detection, but they can be used for anything. So for example, a time lapse sequence could be started at the beginning of a period and terminated at the next period start.

For additional flexibility one can use the Management command facility of the scheduler. Any commands here are periodically run at the Management interval. Typically one would use a sy command to execute a macro periodically that could do whatever logic was required and in turn issue further commands back to the command queue FIFO to initiate, image, video and time lapse operations.

Motion Detection

There are three motion detection modes available in RPi Web Cam Interface that can be selected within "Motion detect mode" on the "Camera Settings" section:

1. Internal - Uses a more efficient motion detection method
2. Monitor - Logs motion detection to a file rather that triggering a recording. Ideally suited to constant video recording along with the video split feature.
3. External - Uses the well known Linux Motion program.

Internal

v6 has a built in motion detection scheme. It is activated by selecting the motion detect mode under camera settings to be Internal. When this is done the original motion settings button disappears and a new Motion Settings accordion control appears on main page.

The settings here are likely to change as the detection algorithm is worked on. Currently there is a vector preview mode which allows the basic vector data to be seen in the live window. This does not work in all browsers and it is recommended to use mjpeg stream mode to minimise problems.

The current detection parameters are Noise Level, Threshold, Mask Image, Change frames to start and still frames to stop. The detection is working at full video frame rate (e.g. 25 fps) so one may want to use a fairly large still frame count to avoid early stop. These parameters are very likely to change in future versions.

The motionmask is like the external motion mask file mechanism. It is a grey scale pgm image file. It is used here as a binary mask. Changes in areas where the mask is non zero (not pure black black) are included. The mask file must be the same resolution as the motion vectors. This is essentially /1/16 of the video but rounded up by 1 in width ;121x68 for a 1920x1080 , 82x61 for a 1296x972, 49x36 for 768x576. It is best created by grabbing an example cam.jpg and then using an photo editor to first change it into a grey scale black white mask where white is the area of interest. Then resize it to the right dimensions and save as a pgm file.

You can see what is going on with internal motion detection by including %c and %f in the annotation string (the date/time info bar at the top of the image). %c will show the filtered changes occurring by frames. When this is above the 'threshold' value then the change frame counter increments (%f) and when this exceeds the start frame count then a capture is triggered. Lower the value of the threshold to increase the sensitivity to small changes.

There are some new pipe commands to control the internal motion detect. Those will get added to the documentation when the parameters finalises a bit more.

Alternate algorithm A second algorithm is available. To select this use Noise parameter values >=1000. The vector change data is first 2d filtered to remove isolated single change values within a frame. Next the accumulated changes in a frame are clipped so that a huge flash change in data does not contribute excessively to the detection. The clipping is based on the threshold value and the motion_clip factor determines the clip ceiling. So a motion_clip factor of 3 means that a single frame cannot contribute more than 3 times the threshold. Using higher values of clip factor allows individual frames to contribute more and could be useful if small threshold values are being used. The Noise parameter controls a frame based

moving average. Any value of this above 999 triggers the alternate algorithm. The averaging is done over 'Noise' - 999 frames so a value of 1009 smooths vector data over 10 frames. If this averaged value exceeds the threshold for 'start'successive frames then a trigger is generated. Similarly the average must be below the threshold for 'stop'successive frames to trigger a stop.

A really good detailed explanation was given by Robert here (copied below too):

https://www.raspberrypi.org/forums/viewtopic.php?p=1020905&sid=2d571cf720f615789b2815098bef3440#p1020905

I use the new > 1000 internal motion detection for all my needs.

The older internal method (<1000) counts vector changes in each frame which are above the 'noise setting' and if the total is above the threshold then it is a changed frame. If there is a sequence of changed frames more than the 'change frames to start' then a start trigger occurs. Reverse applies to generate a stop trigger.

The newer method (>1000) has quite a few optimisations. First it applies a simple 2d filter to changes so that only larger moving objects contribute to the detection. Second, the noise level now controls a moving average time filter to all the vector changes to smooth out the changes. Any frame which now has the smoothed vector change greater than the threshold now is a changed frame. The trigger logic on change frames is similar to the older method.

Starting from defaults (Second algorithm internal) decreasing the noise level (but still above 1000) lowers the amount of smoothing applied. This makes it more responsive to quick changes but can lead to unnecessary 'false' triggers. Similarly increasing this will apply more smoothing making it more necessary for a change to be sustained before a trigger occurs.

Increasing or decreasing the threshold changes the sensitivity to when a particular change level is regarded as a changed frame.

Change frames to start or stop then apply a secondary filter before triggers occur. Higher start numbers mean movement must be sustained longer before a start is generated. Higher stop numbers mean stillness must be sustained longer before a stop is generated. It is normally good to have a highish number for the stop as that means if there is a pause in movement and it restarts then the capture continues in one recording.

Use the %c and %f variables in the annotation string to see what is going on. %c is showing the filtered change level. When it exceeds the threshold then the frame counter %f ticks up until it exceeds the start frame trigger point. Similarly it ticks up for values below the threshold when trying to detect the end of motion.

Extra explanation regarding Noise: When noise level is 1000 or above it kicks in a different algorithm. Firstly a simple 2d filter is applied to the vector changes so that isolated single vector block changes are removed. The 'noise' level then controls a temporal moving filter average to the vector changes found. 999 is subtracted from the noise level and this gives the frame averaging factor.

Moving_Average_n+1 = Moving_Average_n * (Factor - 1) / Factor + Changes_n+1 / Factor

So if Noise = 1000 then Factor = 1 and effectively no averaging is done as the previous average is ignored. If Noise = 1009 then Factor = 10 and 90% comes from the current average and only 10% from new changes. High Noise factors then give 'smoother' changes and make it less susceptible to false triggering but also mean that changes need to be sustained for longer before a trigger occurs.

For the mask generation.

Grab a still image at the same aspect ratio of the video format you are using. E.g. if using 1296 x 972 video then grab an image at the same resolution (temporarily modify a still image resolution to match the video res to grab the image).

Download the jpg (using scp on Linux or WinSCP on Windows) and load it into an image editor (PaintShopPro, Irfanview, Gimp). Resize the image to be the same as the mask image (Log will show that). So for 1296x972 the mask is 82x61. Now use the image editor to paint the areas looking for change to be white and the areas to be ignored to be black. Convert the image to 8bit gray scale and save as a portable greymap file (*.pgm).

Copy this pgm file to the root folder of the webpage, eg. /var/www/html/ and then include the full path to the filename in the Mask Image field. e.g.

/var/www/html/mask-file.pgm

Monitor Mode

Normally motion detection is only active when motion detect is started up. A monitor mode can be selected which activates the base internal motion detection but does not generate triggers to the scheduler. When selected it will be active during normal recording or even when not recording providing video buffering is used.

The monitor mode will log stop and start events to the motionLog.txt file in the web folder (may be changed in raspimjpeg config file.
Motion detection will also trigger the macro motion_event.sh if set up (within /var/www/html/macros/)

A great use case for this mode is the ability to constantly record video all day, but still log motion detection. This is worth tying in with the Video Split setting to avoid having large video files. An example config is as follows:

```
$ cat /var/www/html/uconfig
annotation RPi Cam %Y.%M.%D_%h:%m:%s %c %f
anno_background 1
width 800
```

```
video_width 1296
video_height 972
video_buffer 2000
video_split 1800
motion_external 2
motion_threshold 60
motion_stopframes 1200
motion_clip 3
```

Setting the option to record video all day may trigger watchdog resets more often than normal (see the Watchdog Reset section for more details - http://elinux.org/RPi-Cam-Web-Interface#Watchdog_Reset). This effectively restarts the program, and at the default settings video won't resume recording.

To ensure video recording always resumes after either a reboot or reset, enter the text "ca 1" (minus speech marks) in the "Period Start" field within "Edit Schedule Settings".

External

The motion screen gives access to the motion config settings. Motion detection must be on for this to work.

Not all motion settings are relevant here. A filtered list is shown and the full list can be accessed if required. Settings can be saved backed up and restored. Saving does tell motion to start using the new settings.

Show All | Save Settings | Backup | Restore

framerate	2
minimum_frame_time	0
netcam_url	http://localhost:6655/cam_pic.php
netcam_userpass	password
switchfilter	off
threshold	1500
threshold_tune	off
noise_level	80
noise_tune	on
despeckle	EedDl
area_detect	(null)
mask_file	/home/pi/RPi_Cam_Web_Interface/motionmask.pgm
smart_mask_speed	0
lightswitch	0
minimum_motion_frames	1
gap	3
on_event_start	echo -n '1' >/var/www/FIFO1
on_event_end	echo -n '0' >/var/www/FIFO1
on_motion_detected	(null)
on_area_detected	(null)

Option	Range/Values Default	Description
framerate	Values: 2 - 100 Default: 100 (no limit)	Maximum number of frames to be captured from the camera per second.
minimum_frame_time	Values: 0 - 2147483647 Default: 0	Minimum time in seconds between the capturing picture frames from the camera. Default: 0 = disabled - the capture rate is given by the camera framerate.
netcam_url	Values: Max 4095 characters Default: Not defined	Specify an url to a downloadable jpeg file or raw mjpeg stream to use as input device. Such as an AXIS 2100 network camera.
netcam_userpass	Values: Max 4095 characters Default: Not defined	For network cameras protected by username and password, use this option for HTTP 1.1 Basic authentication. The string is specified as username:password. Do not specify this option for no authentication.
switchfilter	Values: on, off Default: off	Turns the switch filter on or off. The filter can distinguish between most switching noise and real motion. With this you can even set roundrobin_skip to 1 without generating much false detection.
threshold	Values: 1 - 2147483647 Default: 1500	Threshold for declaring motion. The threshold is the number of changed pixels counted after noise filtering, masking, despeckle, and labelling.
threshold_tune	Values: on, off Default: off	Activates the automatic tuning of threshold level. (It's broken)
noise_level	Values: 1 - 255 Default: 32	The noise level is used as a threshold for distinguishing between noise and motion.
noise_tune	Values: on, off Default: on	Activates the automatic tuning of noise level.
despeckle	Values: EedDl Default: Not defined	Despeckle motion image using combinations of (E/e)rode or (D/d)ilate. And ending with optional (l)abeling.
area_detect	Values: 1 - 999999999 Default: Not defined	Detect motion center in predefined areas. A script (on_area_detected) is started immediately when motion center is detected in one of the given areas, but only once during an event even if there is motion in a different configured area.
mask_file	Values: Max 4095 characters Default: Not defined	PGM file to use as a sensitivity mask. This picture MUST have the same width and height as the frames being captured and be in binary format.
smart_mask_speed	Values: 0 - 10 Default: 0 (disabled)	Slugginess of the smart mask. Default is 0 = DISABLED. 1 is slow, 10 is fast.
lightswitch	Values: 0 - 100 Default: 0 (disabled)	Ignore sudden massive light intensity changes given as a percentage of the picture area that changed intensity.
minimum_motion_frames	Values: 1 - 1000s Default: 1	Picture frames must contain motion at least the specified number of frames in a row before they are detected as true motion. At the default of 1, all motion is detected. Valid range is 1 to thousands, but it is recommended to keep it within 1-5.
gap	Values: 0 - 2147483647 Default: 60	Gap is the seconds of no motion detection that triggers the end of an event. An event is defined as a series of motion images taken within a short timeframe.
on_event_start	Values: Max 4095 characters Default: Not defined	Command to be executed when an event starts. An event starts at first motion detected after a period of no motion defined by gap. You can use ConversionSpecifiers and spaces as part of the command.
on_event_end	Values: Max 4095 characters Default: Not defined	Command to be executed when an event ends after a period of no motion. The period of no motion is defined by option gap. You can use Conversion Specifiers and spaces as part of the command.
on_motion_detected	Values: Max 4095 characters Default: Not defined	Command to be executed when a motion frame is detected. You can use Conversion Specifiers and spaces as part of the command.
on_area_detected	Values: Max 4095 characters Default: Not defined	Command to be executed when motion in a predefined area is detected. Check option area_detect.

Configuration

Configuration Scheme

A number of configuration files control how the overall system operates. The web pages give browser access to some of these.

raspimjpeg file in /etc is read whenever the raspimjpeg process starts up including if it is stopped and started from the browser. It contains basic

paths and information to allow raspimjpeg to do its job plus the camera settings that will be used by default.

uconfig file in the /var/www folder is used to hold any changes from the defaults in raspimjpeg applied from the web browser. To start with it doesn't exist but new values will be added if changes are made from the browser. Changes apply directly because the cmd_pipe process has fed those into the raspimjpeg process, but they are also written by raspimjpeg to the uconfig file. When raspimjpeg starts it reads the factory defaults file first and then overwrites any settings that are in the uconfig file. You can also clear the uconfig file from the web browser to effectively return to factory settings. When this is done then the previous settings are moved to uconfig.bak. The file does not hold real-time commands like capture start/end commands or motion enable.

The main index page also reads raspimjpeg config (via a link) and the uconfig file so that it can show the current settings. When a setting is changed here this triggers a cmd_pipe command into raspimjpeg. That updates uconfig and the web page reloads the config files to show the change.

motion.conf in the /etc/motion folder is read by motion to determine its operating characteristics. As motion is being used here in a failry simple mode of motion detection then many of its parameters are irrelevant. The primary ones of interest are those setting the motion detection characteristics like mask files, thresholds, noise levels. motion provides a web api to view and edit these settings and this is used by the motion.php page to show and allow altering the settings.

schedule.json in the /var/www folder is used by the scheduling process to determine the characteristics of the automation. It is read and edited using the browser based schedule.php. It is also read by the same file running in command line mode as a background daemon. If changes are made then the scheduler daemon must be started and stopped via theweb page to allow it to see the new settings.

uPresets.html is an optional file in the /var/www folder which if present controls the video and image format presets present on the main page. If not present then internal values are used. The file is just a list of html option fields. An example uPresetsV2.html is included which can be used as a starting point for creating a custom list.

Naming Scheme

raspimjpeg uses a name scheme for captures and annotation that are controlled by the settings in the config files. These have a number of %X substitution parameters which will be filled in as follows. X must be one of the the following characters. They can be put in any order and repeated if required for a maximum total of 16 substitutions. Any other characters are passed through, but do not use any 'illegal' filename characters or the subdir_char (@). Extra path separators may be put in below the media folder level.

%variable	Substitution
%%	Single %
%Y	4 digit year
%y	2 digit year
%M	2 digit month
%D	2 digit day
%h	2 digit hour
%m	2 digit minute
%s	2 digit second
%u	3 digit millisecond
%k	2 digit video frame counter
%v	4 digit video index
%i	4 digit image index Also used for time lapse batch
%t	4 digit lapse index Starts at 1 in each batch
%a	user annotation Include the content from user_annotate file (/dev/shm/mjpeg/user_annotate.txt)

So a config image_path of /var/www/media/im_%i_%Y%M%D_%h%m%s.jpg

will give a name like im_0005_20150309_093057.jpg

Lapse Index starts at 1 for a particular Time Lapse set and increments.

Thumbnails are named with the base capture name appended with ' . [vit]IndexNumber.th.jpg' where [vit] is a single character for video, image and lapse recordings. This allows cross referencing back to the real capture files independent of the actual base name.

The %i IndexNumber is shared between thumbnails and images. Vidoes have their own IndexNumber %v. raspimjpeg when starting scans the highest number in each category and uses these as the base. A pipe command (sc) can also be used to trigger a scan and set the indexes.

By default the numbers for %i,%v and %t start as 4 digits long but will increase if necessary. The default may be changed by editing the count_format config variable in raspimjpeg. Changing to %05d will give 5 digits by default.

Image Quality and Compression

There are 2 settings to control quality and compression. quality (1-100) controls the quality of the live image preview, and image_quality (1-100) controls quality of still image captures. The number does not match normal jpeg Q factor as the Raspberry camera compression software Q factor is quite non linear. The default 10 is approximately equivalent to 75 in normal JPEG usage and gives a decent trade-off between quality and file size. 7 is about Q 50, and 20 is about Q 95.

```
Raspberry compression discussion (https://www.raspberrypi.org/forums/viewtopic.php?f=43&t=73174&p=527300#p527300)
```

Frequently Asked Questions

How do I change the path for the video images and pictures?

There are separate path settings for video, images and time lapse in the /etc/raspimjpeg config file.

These allow changing the naming scheme of the files and separating out into sub-folders and even including date parameters for these sub-folders.

If you change the 'root' from the media folder in the web install then you also have to change that setting in the web side config.php.

Instead use standard linux link/bind (fstab) facilities to 'relocate these to onto other locations like USB sticks, disk drives and network storage locations. When you do that bear in mind the linked location must have read/write permissions for user www-data

I recommend using doing it that way and leaving the logical location root location as media under the web install as this avoids having to change the web side config and also means that if the software is upgraded then nothing else needs to change each time.

Is there any way to have the "UPTIME" displayed on the Web Cam Interface?

You can use the user annotation variable %a to put any other info in the annotation. The info is picked up from /dev/shm/mjpeg/user_annotate.txt each time the annotation is generated. So use a CRON job or something similar to update this file with a string representing UPTIME.

have you considered adding some additions to this program to monitor the temperature?

Adding a temperature sensor (e.g. DS18B20) should be reasonably straightforward. There are lots of articles on how to connect and read theses on the Pi. It is more a question of how to display it. One way that could be used even with the current software would be to use a program to regularly read the temperature and write it to /dev/shm/mjpeg/user_annotate.txt. The %a annotation variable can then be used to incorporate this in the video annotation. See wiki for the annotation variables.

How to I enable motion edition to start automatically after the Pi powers up?

Motion Detection can be controlled at start up in 2 basic ways.

1) You can edit the /etc/raspimjpeg main config file and change motion_detection from false to true
2) You can configure the scheduler so that it has a period start command of md 1. The Period Start command will be executed when the raspberry starts up.

Method 2 is better in my view as it allows control if you have set up the Scheduler to have different periods. E.g. in Sun based mode you could choose to have motion detection only active during the Day (md 1 in Day Period Start and md 0 in other Period Starts). When the Raspberry boots then it will turn on motion detection if it starts in a Day period otherwise it would wait for the next Day period to start. If you don't want to split the Day up then you use All Day mode and put a md 1 in its one Period Start entry so the Raspberry always goes into motion detection no matter when it boots up.

How do I record video all day?

Use the Video Split setting within Camera Settings to save multiple files of a set length to prevent files becoming too big.
e.g. a value of 1800 will split the video into 30 min segments.

An unfortunate side effect of recording all day is the increased likelihood of the camera stream freezing and triggering a watchdog reset. This problem is outside of Raspimjpeg's control but isn't as bad as it sounds. A reset usually occurs if no updates have occurred on cam.jpg for 3 seconds (customisable option), and video recording can be automatically re-enabled by making a change within the scheduler (see below).

To make this work effectively, a few options need to be set within "Edit schedule settings":
PurgeSpace_ModeEx - Min Space %
PurgeSpace_Level - 30 (or lower if you have a larger SD card)

Period Start - ca 1 (this starts video recording whenever the Pi or the software restarts

If a watchdog reset occurs, it will probably leave behind an unconverted .h264 file in /var/www/html/media/. You need to be on v6.3.8 or higher and enable execution rights on the startstop.sh macro as shown below:

```
sudo chmod +x /var/www/html/macros/startstop.sh
```

The default script will do the job ok, but if you would like more features such as the ability to only have one script running at once or be able to handle errors, replace the file contents with the following:

```
#!/bin/bash
#Check if script already running
mypidfile=/var/www/html/macros/startstop.sh.pid

NOW=`date +"-%Y/%m/%d %H:%M:%S-"`
if [ -f $mypidfile ]; then
        echo "${NOW} Script already running..." >> /var/www/html/scheduleLog.txt
        exit
fi
#Remove PID file when exiting
trap "rm -f -- '$mypidfile'" EXIT

echo $$ > "$mypidfile"

#Do conversion
if [ "$1" == "start" ]; then
  cd $(dirname $(readlink -f $0))
  cd ../media
  shopt -s nullglob
  for f in *.h264
    do
      f1=${f%.*}
        NOW=`date +"-%Y/%m/%d %H:%M:%S-"`
        echo "${NOW} Converting $f" >> /var/www/html/scheduleLog.txt
        #set -e;MP4Box -fps 25 -add $f $f1 > /dev/null 2>&1;rm $f;
        if MP4Box -fps 25 -add $f $f1; then
                NOW=`date +"-%Y/%m/%d %H:%M:%S-"`
                echo "${NOW} Conversion complete, removing $f" >> /var/www/html/scheduleLog.txt
                rm $f
        else
                NOW=`date +"-%Y/%m/%d %H:%M:%S-"`
                echo "${NOW} Error with $f" >> /var/www/html/scheduleLog.txt
        fi
   done
fi
```

CIFS / Samba / Windows Share

To mount a Windows share on boot, put the following line in /etc/fstab.

```
//<IP of Server>/<share name>/<optional subfolder> /var/www/html/media cifs defaults,rw,credentials=/home/pi/.smbcredentials,uid=www-data,gid=www-d
```

e.g.

```
//192.168.0.2/raspishare/cam1 /var/www/html/media cifs defaults,rw,credentials=/home/pi/.smbcredentials,uid=www-data,gid=www-data,iocharset=utf8 0 0
```

Create a .smbcredentials file in /home/pi/

```
nano ~/.smbcredentials
```

Enter these lines, substituting msusername and mspassword with your Windows credentials.

```
username=msusername
password=mspassword
```

Change the permissions of that file

```
chmod 600 ~/.smbcredentials
```

Mount the share

```
sudo mount -av
```

It is highly recommended to set a local boxing folder so that conversion from .h264 to .mp4 occurs locally.

/var/www/html/h264 - local boxing folder
/var/www/html/media - remote share where mp4 files are copied to after being converted.

To do this, first of all create the folder:

```
sudo mkdir /var/www/html/h264
sudo chown www-data:www-data /var/www/html/h264
```

Edit the config file

```
nano /etc/raspimjpeg
```

Find the boxing_path line and add the relevant path

```
macros_path /var/www/html/macros
user_annotate /dev/shm/mjpeg/user_annotate.txt
boxing_path /var/www/html/h264
subdir_char @
count_format %04d
```

Save the file and restart the raspimjpeg service or reboot the Pi

Example Configurations

To help new people to the software, please post your example /var/www/html/uconfig examples below. Note that this file is normally used to persistently store custom values set from the web interface, but you can manually paste your settings into this file as long as you restart the server (stop/start from within ~/RPi_Cam_Web_Interface/).

This sets the maximum camera view, a larger preview width, splits video every 30 mins when constantly recording video, uses Monitor motion mode.

```
annotation RPi Cam %Y.%M.%D_%h:%m:%s %c %f
anno_background 1
width 800
video_width 1296
video_height 972
video_split 1800
motion_external 2
motion_threshold 60
motion_image /var/www/html/cam-mask3.pgm
motion_stopframes 1200
motion_clip 3
```

Additions & Tricks

This section will be updated to contain links to posts with useful information and answers to popular questions. Anyone can add to this, so please create an eLinux account and help contribute!

Pre-Event Motion Capture Buffer

There is a video_buffer setting in /etc/raspimjpeg (default 0) and it can also be set from the camera settings (buffer). The value is a time duration estimate in milliseconds of the circular buffer.

If the value is 0 then operation is as it were originally; a video capture starts capturing the h264 camera stream to a file. With the various latencies using motion detection then the trigger and the start of a capture can be a second or so after motion really started, so the initial action is not recorded.

If the value is set to say 3000 (3 seconds) then raspimjpeg captures video data all the time into a RAM memory circular buffer sized to contain nominally 3 seconds, but in practice normally more as static video compresses well. When a video capture then starts, most[*] of the contents of this buffer is then prepended into the video file which is now capturing data. This means that file recording actually contains video from before the trigger. You control the amount before by adjusting the buffer.

The thumbnail image is still taken from the trigger moment so it helps show the cause. You will also notice that the thumbnail has a time stamp later than the start of the video. The difference is effectively the size of the buffer.

[*]Due to h264 interframe compression (eg. next frame may depend on previous frame), a video file must start at a "stand-alone" I-frame, which may occur only once every 60 video frames. Buffer sizes below 3000 are not recommended because an I-frame may not be found in the buffer if it is too small.

Pan-Tilt or Pi-Light

Information about Pi-Pan and Pi-Light: http://www.mindsensors.com/12-rpi

Needed Hardware for this addition:

- Pi-Pan (MindSensors)
- Pi-Light (MindSensors)
- Pi-Case to mount Pi-Pan recommended (e.g. MindSensors)

In case you want an alternative minimal hardware (cheaper) option go to the end of this section.

With these code-changes and additions, it's possible to control Pi-Pan and Pi-Light from the RPi Cam Web Interface. Just follow these steps:

- Assembly and install Pi-Pan: http://www.mindsensors.com/content/27-assembly-instructions-for-pi-pan-kit
- Check if pipan and pilight work with the scripts as described on the website
- Add the file "pipan_pipe.py" to the pipan-files with the following content:
- Edit the line with /var/www/FIFO_pipan if using a subfolder. E.g. to /var/www/html/FIFO_pipan

```
#!/usr/bin/env python

import time
import os, sys
import pipan
import pilight

p_servo = pipan.PiPan()
p_led = pilight.PILIGHT()

while True:
  pipein = open("/var/www/FIFO_pipan", 'r')
  line = pipein.readline()
  line_array = line.split(' ')
  if line_array[0] == "servo":
    p_servo.do_pan(int(line_array[1]))
    p_servo.do_tilt(int(line_array[2]))
  elif line_array[0] == "led":
    p_led.createPiLight(int(line_array[1]),int(line_array[2]),int(line_array[3]))
  pipein.close()
```

- Navigate to "/var/www" (or to /var/www/subfolder) and add a named pipe with the following commands:

```
sudo mknod FIFO_pipan p
sudo chmod 666 FIFO_pipan
```

- Edit "/etc/rc.local": add the following line above the exit-command (change the path to the directory where you extracted the pipan-files):

```
python /home/pi/pi-pan/pipan_pipe.py &
```

- Go to /var/www and rename the file "pipan_off" to "pipan_on" and "pilight_off" to "pilight_on"

- Reenable camera-support in raspi-config

That's it. After rebooting your Pi, you should be able to control Pi-Pan with the new Buttons "Up", "Down", "Left" and "Right" or on the keyboard with "W", "S", "A" and "D". The Pi-Light can be controled in the settings-table or on the keyboard with "F". If you have a touch-device (Android or iOS), you can pan/tilt by dragging the preview-image around. To change the minimum/maximum pan/tilt angles, edit the settings in /var/www /pipan.php.

Pi-Pan uses servoblaster to generate the signal for pwm. However, it might be that servoblaster will interfere with 3.5mm jack audio output. If so, edit /etc/init.d/servoblaster.sh and add the option --pcm to the servod-command on line 17.

If you get the error "IOError: [Errno 2] No such file or directory", run raspi-config and disable device tree in advanced options and then reboot.

- If you extract the pi-pan files in a different directory, the camera will not reset to neutral position on startup. To fix, verify the file "servoblaster.sh" in the pi-pan and the etc/init.d folders correctly references the pi-pan directory:

```
do_start () {
        /usr/local/sbin/servod --idle-timeout=2000 --p1pins=16,18 >/dev/null
    chmod a+rw /dev/i2c* > /dev/null 2>&1
        sleep 2
        python ~pi/pi-pan/neutral_servo.py 2>&1 &
        #VERIFY PATH ABOVE IS CORRECT
}
```

Minimal hardware (cheaper) option;

Servoblaster

There is also the possibility to just use ServoBlaster and an own servo-construction to move the camera; more information here: https://github.com /skalad/RPi_Cam_Web_Interface_ServoBlaster_pan_tilt. (But please note that this is now outdated. Use the functionality within this git as described below.)

Equivalent functionality can be turned on by renaming the file in the web folder servo_off to servo_on. Make sure that pipan_on is not also present as that takes precedence. Rename it to pipan_off if used before.

The co-ordinates and some set up information is preserved in the servo_on file in json format. This may be edited to change the behaviour. In particular the buttons can be mapped onto different servo directions, and the min, max and step rates can be changed for each axis. This file may be edited with a text editor. Only change the values, not the key names. The code uses the default servoblaster pins. You can set them using "sudo ./servod" from the user folder inside the ServoBlaster directory.

Remote access to website with User/Pass and changing port

1) Set up security on your camera by installing or re-installing with username:password and optionally choose a different port for access other than the default 80. Changing the port is not necessary but can be helpful in avoiding conflicts with other web uses and provides a bit of extra security against web scanners. It also allows multiple cameras by using different ports on each one.

2) Test security locally e.g. http://cameraip:port and make sure login is OK.

3) Login to your router and

a) Find its WAN IP address. This will be needed to get access from internet.

b) Set up port forward so that the port selected is redirected to the cameraip

c) The way to do this is obviously dependent on the actual router model. Check out http://www.howtogeek.com/66214/how-to-forward-ports-on-your-router/

4) To use a regular name from the WAN rather than the WAN IP of your router then set up a dynamic DNS account at a service like http://freedns.afraid.org

Central Motion Detection from Multiple Cams using iSpy Connect

http://www.raspberrypi.org/forums/viewtopic.php?p=518500#p518500

Now that an MJPEG stream has been implemented, you can configure iSpy with the following:

MJPEG URL tab within Video Source:

```
http://<IP>/cam_pic_new.php?
```

View video stream on an iDevice / Smartphone

Credit goes to Oke for the original post. A few more tweaks added for iPhone app. http://www.raspberrypi.org/phpBB3/viewtopic.php?p=507756#p507756

(note - cam.jpg created during the installation now)

- Download "IP Cam View Pro" by NibblesnBits (I used IP Cam View Lite on the iPhone - can upgrade to Pro)

https://itunes.apple.com/us/app/ip-camera-viewer/id402656416?mt=8&ls=1

- Select the menu icon
- Press Manage Cameras
- Select Add Camera then scroll down to find Raspberry, and then select RPi Cam Web Interface
- Give your cam a name
- Enter your URL/IP, e.g. 192.168.0.1
- Enter user/pass if configured
- Ch.# is the folder name, such as html
- Press Test
- If it works, press Save

Simple live-preview only page

A file min.php is included in the install which gives a basic live preview only. To use just browse to http://yourcamerip/min.php

The minimal code to embed the preview:

php:

```
<!DOCTYPE html>
<html>
  <head>
    <title>RPi Cam Preview</title>
    <script src="script_min.js"></script>
  </head>
  <body onload="setTimeout('init();', 100);">
    <center>
```

```
      <div><img id="mjpeg_dest" /></div>
    </center>
  </body>
</html>
```

js:

```
var mjpeg_img;

function reload_img () {
  mjpeg_img.src = "cam_pic.php?time=" + new Date().getTime();
}
function error_img () {
  setTimeout("mjpeg_img.src = 'cam_pic.php?time=' + new Date().getTime();", 100);
}
function init() {
  mjpeg_img = document.getElementById("mjpeg_dest");
  mjpeg_img.onload = reload_img;
  mjpeg_img.onerror = error_img;
  reload_img();
}
```

The size can be changed either as parameter in /etc/raspimjpeg or with CSS. To add further features (change settings, record images/videos), study the existing homepage.

Adjusting Live Preview bandwidth usage

The live video on the main screen is produced by a mjpeg feed either as a true stream or as succession of jpegs. Either way this can consume quite a bit of bandwidth. This section describes how this may be adjusted.

There are a number of factors affecting the live previews; width, divider, quality. These can all be controlled from the web interface. In addition you can choose between default streaming where it is fetching a sequence of jpegs or a true mjpeg stream. This doesn't fundamentally change the bandwidth but the mjpeg stream has less to and fro with the web server so may be smoother.

Width controls the size of the mjpeg images generated and will have a significant impact on bandwidth. It also controls the size on the display. By default it is 512px. If you can live with say 384 then that should halves the bandwidth.

Quality is the jpeg compression factor (0-100). Lower numbers give better compression at the expense of quality. It is set to 25 by default which gives pretty good compression and not too much degradation. You can try lowering it to say 15 which will significantly lower sizes further but you will start to see some degradation. Lowe than this it will start to get bad.

Divider is the rate at which new data is fetched. It is the video fps (default 25) divided by 'Divider'. Nominally that would mean it is trying to update at 25fps but in practice other delays lower this a bit. Increasing the divider lowers the fetch frame rate so 3 would give a nominal rate of 8fps. This will also have a dramatic effect on bandwidth.

Move saved images and videos to another folder

Detailed instructions on how to map a network NFS share to /var/www/media - http://www.raspberrypi.org/forums/viewtopic.php?p=531344#p531344

When to do the move - http://www.raspberrypi.org/forum/viewtopic.php?p=515967#p515967

Mounting a share - http://www.raspberrypi.org/phpBB3/viewtopic.php?p=513781#p513781

Mounting a Windows Share - http://www.stuffaboutcode.com/2012/05/raspberry-pi-connect-nas-windows-share.html

If you've mounted a network share to something other than /var/www/media, such as /mnt/myshare, you can bind the two together using this command:

```
sudo mount --bind /mnt/myshare /var/www/media
```

Overlay/Composite two images together

One way to add an image over the top of another is with ImageMagick. I've constructed a crude example below, and it works! But it bogs down the refresh rate, drives the load level to 1.5 and is barely acceptable. I'd like to toggle it on/off with a new button, and improve it's performance, so please edit this article better.

```
<?php
/// derived from http://www.php.net/manual/en/imagick.compositeimage.php
/// requires 'sudo apt-get install php5-imagick' on RPi
/// backup /var/www/cam_pic.php, then replace it with this
/// goal is to put backup lines on reverse truck/trailer camera(s)

    $tlines = new Imagick('/var/www/lines/TrailerLines0deg.png');
    /// TrailerLines0deg.png is a image of the guidelines with a transparent background
    /// TODO: use steering wheel to select correct line graphic from library
```

```
    $camimg = new Imagick('/dev/shm/mjpeg/cam.jpg');
    $camimg->compositeImage($tlines, Imagick::COMPOSITE_DEFAULT, 70, 20);
    header("Content-Type: image/png");
    echo $camimg;
?>
```

h264 to mp4

To convert a h264 video file to mp4, the interface uses MP4Box. If you set MP4Box Off in /etc/raspimjpeg, perhaps so that the interface is more responsive to frequent start stop recording requests, you can later use MP4Box to convert any saved h264 videos file to MP4. The bash command is

```
$ MP4Box -add source.h264 destination.mp4
```

The internal boxing command is now configurable by changing the MP4Box_cmd setting in /etc/raspimjpeg. The default value is

```
(set -e;MP4Box -fps %i -add %s %s > /dev/null 2>&1;rm "%s";) &
```

If replacing this then note the following. A set -e is used to cause termination of the command sequence if a cmd fails. This means that here if the MP4Box fails then the rm comd to remove the source is not executed. Four % parameters in the string are replaced before execution with fps, source filename (h264), dest filename (mp4), and source filename again. It is important that any replacement command line specifies all 4 parameters in that order. The line finishes with an & that allows this command to run in the background.

Add extra users to security

The install.sh sets up the web security to allow one user to login with a password. If you want to add more users with different passwords then use the following command.

```
sudo htpasswd /usr/local/.htpasswd newusername
```

Installation tips

If installing from scratch for semi-dedicated camera usage then these are extra steps I take to minimise install time and free extra space.

Step 1: Write latest full raspbian jessie image using a tool like win32diskimager

Step 2: Write a ssh file to boot folder and optionally write a configured wpa-supplicant.conf file to /boot to allow headless operation and wifi access if used.

Step 3: Place card in Pi and start up and login into Pi using ssh client (e.g. Putty)

Step 4: Change passwd

Step 5: Do not run raspi-config yet

Step 6: Remove extra material from full raspbian install and update

```
sudo apt-get remove --purge wolfram-engine scratch minecraft-pi sonic-pi dillo gpicview oracle-java8-jdk openjdk-7-jre oracle-java7-jdk openjdk-8-j
sudo apt-get clean
sudo apt-get autoremove
sudo apt-get update
sudo apt-get dist-upgrade
```

Step 7: Now run sudo raspi-config. Expand file system, Change Time Zone, Enable camera

Step 8: Now proceed with normal installation instructions from Step 4.

Construction of a Pi Zero Security camera

There is an instructable on how to put together a Pi Zero Security camera in a dummy camera housing.

http://www.instructables.com/id/Zero-Security-Camera/

Multiview: Simple multi-camera display

The install set has a simple multi-camera display. There is no functionality other than live preview of mutliple cameras in a simple web page.

To set up you need a multiview.json file in the web folder. A prototype multiview.jsonD is provided. First copy this to multiview.json. You can do the edits in a text editor as it is very simple. This contains two arrays. hosts is just a set of urls for the camera hosts. Note that the host url required is the same as used to access the normal camera but must also have the trailing '/' character. E.g. http://192.168.0.100/html/ Space is allowed for up

to 4 hosts but more could be added. If ports are used then these should be included. delays is a set of microsecond counts which controls the refresh rate of the displays; 40000 corresponds to 25 frames per second. Use larger values to slow down refresh rate.

The multiview is accessed by ip/multiview.html

If less than 4 cameras are in use then just comment out the extra cameras in multiview.html e.g. <----->

If more cameras are wanted then you will need to cut and paste extra elements into the multiview.html and add corresponding entries in multiview.json

User Access levels: Simple user access control of web display functions

If using login methods to restrict access to the web display then you may optionally give different users different functionality on the web interface. There are 3 levels of functionality defined USERLEVEL_MIN 0 gives just a basic preview screen with no user functionality USERLEVEL_MEDIUM 3 gives control over video recording, image capture and time lapse together with access to the download page. Settings cannot be changed. USERLEVEL_MAX 6 gives full control with access to all settings

Note that this is not designed to be secure; it just removes direct web access to facilities. A knowledgable user could still access settings even with lower User level by constructing web queries.

To set up use the normal login configuration and then add further users to htaccess.

Now set up a text file in the web install folder called userLevel; an example userLevelExample is included. Each line in this file is username:Level where Level is 0,3 or 6 If login is used and this file does not exist then all users get full rights. If the file exists but a username doesn't exist then they get the minimum level. This can be used with a single username to give access to all settings and then all other unknown users only get the minimum preview display.

User buttons

The main page has an option to add user buttons to the interface which can trigger activity by using macros.

This is controlled by a userbuttons file in the main web install folder (e.g. /var/www/html). If the file does not exist then the page displays without any extra buttons. If the file does exist then it can contain the definitions for up to 6 user buttons,

The file is just a text file and contains definition lines which are just

```
buttonName,macroname.sh
```

When a button is pressed then the corresponding macro is executed.

An example file is included called userbuttonsD. If this is renamed to userbuttons then two extra buttons will appear.

Tutorials

How to Make a £59 Fully Featured Raspberry Pi Home Security Camera – Newbie Guide

https://quavoce.wordpress.com/2017/10/25/how-to-make-a-59-fully-featured-raspberry-pi-home-security-camera-newbie-guide/

Turn on and off motion detection using IFTTT Do Buttons

https://quavoce.wordpress.com/2017/01/17/rpi-cam-web-interface-turn-on-and-off-motion-detection-using-ifttt-do-buttons/

Full Reverse Proxy Instructions with Dynamic IP and HTTPS Encryption

https://quavoce.wordpress.com/2017/06/04/full-reverse-proxy-instructions-with-dynamic-ip-and-https-encryption/

Telegram Alerts with Motion Detection sending pictures to your phone with RPi Web Cam Interface

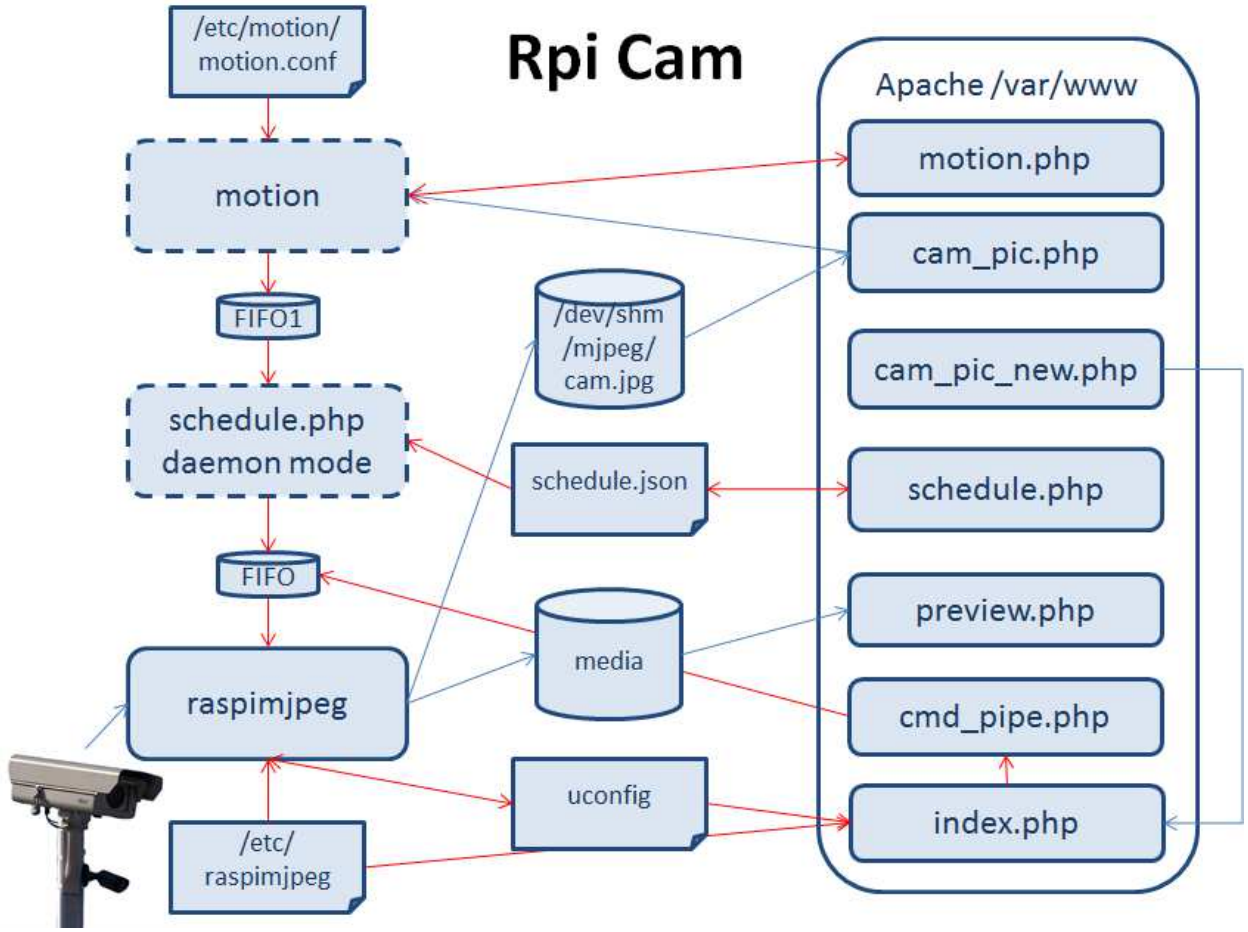
https://quavoce.wordpress.com/2017/09/29/telegram-send-messages-photos-animated-gifs-from-your-raspberry-pi-and-rpi-web-cam/

Access Raspberry Pi Filesystem from Mac OSX using Netatalk AFP

https://quavoce.wordpress.com/2017/06/13/access-raspberry-pi-filesystem-from-mac-osx-using-netatalk-afp/

Architecture

The overall functionality is quite complex but centres around the raspimjpeg process which accesses the camera data. The following diagram shows the major components. Blue lines indicate data flows. Red lines indicate control.



Data Flow

In normal monitoring mode raspimjpeg makes a connection to the camera (MMAL) and generates a continuous stream of preview jpeg captures in the /dev/shm/mjpeg directory all with the same name cam.jpg. This folder is RAM memory based so does not strain the SD card memory.

These captures may be accessed via URLs on the Apache Web server. cam_pic.php process returns the latest one and cam_pic_new.php merges them into a mjpeg stream. When a browser has logged into the web server then the main page (index.php) will use cam_new_pic.php to give a moving representation of the camera output. If motion detection is active then the motion process also accesses these images via cam_pic.php in order to analyse differences between successive frames and detect motion.

If raspimjpeg is put into a capture mode (described below) then the flow of preview images is maintained but an extra recording is made of either a single image, a time lapse sequence of single images, or a full video recording which can be at any format the camera can support including HD normal frame rates. This data is stored in the media folder. For the images and tile lapse images these are stored directly in jpg format. The video is initially stored as a raw h264 stream from the camera but can be automatically formatted into mp4 when the recording ends. The boxing mode (MP4Box) controls this. Three options are provided; false leaves the files in raw h264, true converts them inline but no further recording is possible till this completes, background spawns a separate process to do the boxing operation. Normally the raw h264 captures and box operations take place within the same folder. This can cause quite a lot of network traffic and potential problems if the media folder has been remotely mounted. A config option (boxing_path) may be defined as a separate local folder. If that is used then the capture is to the boxing_path folder and the boxing operation is from the boxing_path to the final destination. This limits the network traffic to one pass and also takes it out of the real time capture operation.

When raspimjpeg initiates any of these sequences it also grabs the current cam.jpg from the preview stream and stores it in the media folder with a thumbnail name tied to the captured data. These thumbnails are used by the preview_php process to give a representation of each capture when the download button is pressed.

Control Flow

Raspimjpeg can be controlled by sending in commands into a named FIFO pipe in the /var/www folder.

The commands give access to most of the camera settings plus stopping and starting the capture processes. Commands can come from effectively 3

sources but one is directed through a secondary process for scheduling purposes.

- Commands can come from the web browser via the cmd_pipe.php web page. This is used to allow changing the camera settings and manually starting and stopping captures.
- Commands can come from the scheduling process (scheduler.php – daemon) which can be used to change various modes, camera settings at different times based on sunrise and sunset.
- Commands can also come from motion and these are used to start and stop captures based on motion detection. In the original scheme these commands went straight to raspimjpeg. In the modified scheme they are sent through the scheduling process daemon so that it may change the nature of stop and start based on the daily periods. To achieve this motion now sends its commands to a secondary named FIFO pipe (FIFO1). The scheduler is monitoring this for motion start / stops and then sends in translated commands to the main raspimjpeg process.

RaspiMJPEG

Config file

This is held in /etc/raspimjpeg and consists of keyword value lines. It is read at start up and when the samera is restarted. There is a uconfig file held in the web folder which allows user overrides to many of these values.

Keyword	Default Value	Description
annotation	RPi Cam %Y.%M.%D_%h:%m:%s	See an command
anno_background	false	See ab command
anno3_custom_background_colour	0	See ac command
anno3_custom_background_Y	0	See ac command
anno3_custom_background_U	128	See ac command
anno3_custom_background_V	128	See ac command
anno3_custom_text_colour	0	See at command
anno3_custom_text_Y	255	See at command
anno3_custom_text_U	128	See at command
anno3_custom_text_V	128	See at command
anno_text_size	50	See as command
sharpness	0	See sh command
contrast	0	See co command
brightness	50	See br command
saturation	0	See sa command
iso	0	See is command
metering_mode	average	See mm command
video_stabilisation	false	See vs command
exposure_compensation	0	See ec command
exposure_mode	auto	See em command
white_balance	auto	See wb command
autowbgain_r	150	See ag command
autowbgain_b	150	See ag command
image_effect	none	See ie command
colour_effect_en	false	See ce command
colour_effect_u	128	See ce command
colour_effect_v	128	See ce command
rotation	0	See ro command
hflip	false	Sets horizontal flip on /off
vflip	false	Sets vertical flip on /off
sensor_region_x	0	See ri command
sensor_region_y	0	See ri command
sensor_region_w	65536	See ri command
sensor_region_h	65536	See ri command
shutter_speed	0	See ss command
raw_layer	false	See rl command
camera_num	0	See cn command
minimise_frag	0	controls MMAL MINIMISE_FRAGMENTATION 0/1 on/off
mmal_logfile		Set to filepath (//dev/shm/mjpeg/mmallogfile) to enable callback logging. Debug use only
width	512	Sets preview width
quality	10	Sets preview jpeg compression
divider	1	Sets preview fps relative to video fps
video_width	1920	See px command
video_height	1080	See px command
video_fps	25	See px command
video_bitrate	17000000	See bi command
video_buffer	0	See bu command

h264_buffer_size	131072	Sets h264 buffer size (0 default 65536) Higher values helps callback smoothness
h264_buffers	0	Sets number of buffers used by capture system
video_split	0	See vi command
MP4Box	background	See bo command
MP4Box_fps	25	See px command
MP4Box_cmd	dflt	Sets cmd used during boxing operations. See Additions and Tricks section
image_width	2592	See px command
image_height	1944	See px command
image_quality	10	See qu command
tl_interval	30	See tv command
motion_external	true	Select external motion detetc
vector_preview	false	Show vector display rather than image
motion_noise	20	Internal motion detect parameter
motion_threshold	100	Internal motion detect parameter
motion_image		Internal motion detect image mask
motion_startframes	5	Internal motion detect parameter
motion_stopframes	50	Internal motion detect parameter
motion_pipe	/var/www/FIFO1	Where to send internal detect triggers
motion_file	0	Turn on vector capture to file
base_path	/var/www	Base path of web install
preview_path	/dev/shm/mjpeg/cam.jpg	RAM folder for preview jpgs
image_path	/var/www/media/im_%i_%Y%M%D_%h%m%s.jpg	Template for image capture names
lapse_path	/var/www/media/tl_%i_%t_%Y%M%D_%h%m%s.jpg	Template for time lapse capture names
video_path	/var/www/media/vi_%v_%Y%M%D_%h%m%s.mp4	Template for video capture names
status_file	/var/www/status_mjpeg.txt	raspimjeg status file
control_file	/var/www/FIFO	Listening pipe for commands
media_path	/var/www/media	Base storage for media files
macros_path	/var/www/macros	Base storage for macros
boxing_path		Temp folder for capture and boxing operations
subdir_char	@	Character used to flatten paths in thumbnail names
count_format	%04d	format string for capture numbering. e.g. %05d for 5 digits
start_img	start_img.sh	Image capture start macro name
end_img	&end_img.sh	Image capture complete macro name
start_vid	start_vid.sh	Video capture start macro name
end_vid	end_vid.sh	Video capture complete macro name
end_box	&end_box.sh	Boxing complete macro name
thumb_gen	vit	Thumbnail generation enables
autostart	standard	Whether to start camera at start up
motion_detection	false	Whether to enable motion detect at start up
watchdog_interval	30	Check interval to see if preview still working
watchdog_errors	3	Trigger point for preview failure
user_config	/var/www/uconfig	User config file path
log_file	/var/www/scheduleLog.txt	Logging path
fullscreen	false	If true start up display fullscreen
log_size	5000	Maximum number of lines in log. 0 means no logging

Pipe

The following commands are supported by raspimjpeg when received from the pipe. Many are equivalents of values in the config file. Others are associated with several parameters. They are sent in as a serial stream as a 2 character command, space, and space separated parameters.

The original scheme did not use LF terminators. LF terminators are now supported and recommended to improve reliability in command recognition. The old scheme is still supported. To send from a command line use something like

```
echo 'im' >/var/www/FIFO
```

(where this may need to be adjusted according to the install folder.

The main Pipe is FIFO which is used by the web interface and scheduler. Commands may also be sent to extra Pipes named FIFO11 up to FIFO19. All commands are treated the same no matter which pipe is used. This can be useful to make sure that commands from other sources don't get mixed up. The installer makes FIFO11 as an input; if others are required then they should be made manually and will then be recognised automatically next time the camera system is started.

Cmd	Parameters	Description
an	Text	set annotation
ab	0/1	annotation background off/on
px	AAAA BBBB CC DD EEEE FFFF	set video+img res video = AxB px, C fps, box with D fps, image = ExF px)
bu	Number	set pre-trigger video buffer in mSec (approx)
as	Number	Set text size 0-99
at	E YYY UUU VVV	Set Text colour E (0/1 enable) Colour as Y:U:V
ac	E YYY UUU VVV	Set background colour E (0/1 enable) Colour as Y:U:V
sh	Number	set sharpness (range: [-100;100]; default: 0)
co	Number	set contrast (range: [-100;100]; default: 0)
br	Number	set brightness (range: [0;100]; default: 50)
sa	Number	set saturation (range: [-100;100]; default: 0)
is	Number	set ISO (range: [100;800]; default: 0=auto)
vs	Number	0/1 turn off/on video stabilisation
ec	Number	set exposure compensation (range: [-10;10]; default: 0)
em	Keyword	set exposure mode (range: [off/auto/night/nightpreview/backlight/spotlight/sports/snow/beach /verylong/fixedfps/antishake/fireworks]; default: auto)
wb	Keyword	set white balance (range: [off/auto/sun/cloudy/shade/tungsten/fluorescent/incandescent /flash/horizon]; default: auto)
ag	RRR BBB	set white balance off red_gain blue gain (100 = 1.0; default: 150)
mm	Keyword	set metering mode (range: [average/spot/backlit/matrix]; default: average)
ie	Keyword	set image effect (range: [none/negative/solarise/posterize/whiteboard/blackboard/sketch/denoise/emboss /oilpaint /hatch/gpen/pastel/watercolour/film/blur/saturation/colourswap/washedout/posterise /colourpoint /colourbalance/cartoon]; default: none)
ce	A BB CC	set colour effect (A=enable/disable, effect = B:C)
ro	Number	set rotation (range: [0/90/180/270]; default: 0)
fl	Number	Set horisontal flip(hflip) and vertical flip(vflip). 0={hflip=0,vflip=0}, 1={hflip=1,vflip=0}, 2={hflip=0,vflip=1}, 3={hflip=1,vflip=1}, default: 0
ri	AAAAA BBBBB CCCCC DDDDD	set sensor region (AAAAA BBBBB CCCCC DDDDD, x=A, y=B, w=C, h=D)
ss	Number	Set shutter speed
qu	Number	set output image quality (range: [0;100]; default: 85)
pv	QQ WWW DD	set preview quality (0-100) default 25 Width (128-1024) default 512, Divider (1-16) default 1

Job Macros

Raspimjpeg supports calling macros for every capture job, some system events, and error handling. These are shell scripts held in the macros folder. They must be created and stored there and given suitable ownership and permissions. Typically they should be owned by www-data and have execution privileges by that user (e.g. 764)

start_img.sh if present is called just before starting to capture a still images. It is passed a parameter containing the image file name. By default it is configured as a synchronous macro which means it will complete before the image is captured.

end_img.sh if present is called at the completion of capturing still images. It is passed a parameter containing the image file name.

start_vid.sh if present is called at the start of raw video capture. It is passed a parameter containing the mp4 video file name.

end_vid.sh if present is called at the end of raw video capture. It is passed a parameter containing the h264 video file name. Note that if boxing is being done then this file will disappear after boxing is complete.

end_box.sh if present is called at the end of any boxing command. It is passed a parameter containing the mp4 file prodcued by boxing.

error_hard.sh if present is called if any fatal error occurs. It is passed a parameter containing the error string.

error_soft.sh if present is called if any non-fatal error occurs. It is passed a parameter containing the error string.

do_cmd.sh if present is called if some selected pipe commands are processed. It is passed a parameter containing the command string. Currently only called by motion detect (md) commands and ru commands.

motion_event.sh if present is called when internal motion detection finds a start or stop condition. It is passed a parameter 0 for stop and 1 for start. This works even when using motion monitoring mode.

startstop.sh if present is called when camera software starts or stops. It is passed a parameter 'stop' for stop and 'start' for start. AN example is included which tries to convert any orphan .h264 files on start up. This would need to be renamed and given execute permissions to work.

If you want the thumbnail in the script then convert it from supplied filename. E.g.

```
#!/bin/bash
list=( $1*.th.jpg )
thumb="${list[-1]}"
```

$1 is the parameter passed to the script and $thumb will be the thumbnail name.

Note this will not work directly with end_vid.sh as the capture at that point is a .h264 file. Use the end_box.sh where the conversion to .mp4 has taken place.

Macros can be updated and enabled / disabled from the web interface. This uses the um (update macro) pipe command. This takes 2 variables. The first is a number (starting from 0) indexing the macros ('error_soft','error_hard','start_img','end_img','start_vid','end_vid','end_box','do_cmd','motion_event','startstop') and the second is the filename of the macro in the macros folder. Disabling of macros is done by prepending a '-' character to the name. This causes the macro not to be found and therefore not executed. So, for example, um 6 -end_box.sh would disable the end_box script.

Trouble Shooting

This section contains trouble shooting tips.

General principle is that it is always easier to troubleshoot starting from an operational system and finding what breaks it rather than starting from a broken system and finding what makes it work. So install basic system leaving all config as default as this is most likely to work. Then make any changes you want one at a time.

Loading

Web interface starts up but just shows 'Loading'. There are a number of causes for this but most are caused by the core raspimjpeg process not running or not running correctly. First, Have you enabled autostart during installation and rebooted? If not then either run the install again to enable it or run the install script with the keyword start on the end.

```
./RPi_Cam_Web_Interface.sh start
```

Check that raspimjpeg process is running (ps -A) should show 2 instances of the process. Check that a thumbnail is being produced at the location /dev/shm/mjpeg There should be a file called cam.jpg which is being regularly updated. Check also the scheduler log available from the scheduler log pagae to see if any errors are being logged.

Motion Detection

Motion detection relies on a chain of events so trouble-shooting is deciding where the chain has broken down.

If using the external detection the motion process must be running, second it must get a continuous feed of updated cam.jpg images to analyse, third it must detect motion based on its settings, fourth it must issue start and stop commands, fifth the commands must get picked up by the scheduler and translated to raspimjpeg capture commands, and sixth raspimjpeg must action those commands. Working on these in turn and using a terminal session (e.g. ssh)

- When motion detection is started then a ps –A cmd should show a motion process in the list. Similarly when motion detection is stopped then there should be no motion process in the list.
- Motion picks up the images via the netcam_url setting. This defaults to http://localhost/cam_pic.php which should work with default set up. If the web site port has changed that it needs the port added and if a password has been added then the netcam_userpass setting must be changed. Test by browsing to http://cameraIPaddress/cam_pic.php . You shoud see the latest preview image. Change something in from to of the camera and refresh to check t is getting fresh images.
- For the motion detection and event triggers the best way to debug is to temporarily change the on_event_start from echo -n '1' > /var/www /FIFO1 to echo 'start' >> ~/motion.txt and on_event_end echo -n '0' > /var/www/FIFO1 to echo 'start' >> ~/motion.txt This will then write start and stop into that text file if motion triggers. Change threshold and noise settings in motion set up, and check for triggers occurring.

If using internal motion detection then no extra processes are used.

For both types the detection produces a 1 (motion start detected) and a 0 (motion stop detected which are sent into the FIFO1 queue for the scheduler to decide what to do. This can be simple video start / stop or more complex command sequences. THe action can also depend on the day and time depending on how scheduler is set up.

- Check that scheduler is running (Schedule screen should show a stop button, if it says start then start it). For test purposes set Scheduler to All Day mode and make sure ca 1 is in Motion Start and ca 0 is in Motion stop. Now manually send a trigger by issuing a echo -n '1' > /var/www/FIFO1 command. Scheduler log should show start trigger received and should send on the command to raspimjpeg which should start the recording.
- You can also manually check raspimjpeg itself by echo -n 'ca 1' > /var/www/FIFO which should start a video recording and echo -n 'ca 0' > /var/www/FIFO which should stop it.

When using internal detection it can be useful to view the detection values by using two special annotation characters. Add a %f %c to the fronto f the annotation string under camera settings. The annotation will now show the running changed frame count and the change value within the current frame. This can aslo be useful when tuning detection values for the best results.

Viewing and Saving Motion Values

Configuration values for the external motion program are maintained via a http interface supported by motion. This means that motion must be running for this to work. Normally this should just work and you should see current values, be able to edit them and save the settings back. If this is not working then try the following steps.

- Check that motion is running as the correct user by issuing a ps -Af command. motion should show up as running under user www-data
- Check that the motion.conf program has the correct ownership and permissions. A sudo ls -l /etc/motion/motion.conf should show the file belonging to www-data and with rw user permissions.

You can try manually fixing any incorrect permissions as follows

```
sudo chown www-data:www-data /etc/motion/motion.conf
sudo chmod 664 /etc/motion/motion.conf
```

- Check that the motion.conf program has the correct web api set up. If an older version of motion has been used before then it is possible that its motion.conf file is confusing the install. In particular check and correct the web api parameters. They should be like

```
webcontrol_port 6642
webcontrol_localhost on
webcontrol_html_output off
```

- You can also check the api by turning motion detection off, changing webcontrol_localhost to off so it can be accessed from other host, and then restarting motion detection. A web browse to http://cameraipaddress:6642/0/config/list will then show the motion parameters.

Network speed / choppy video

This section contains hints on optimising the set up for lower network usage of the video viewing primarily of the live preview. Network bottlenecks particularly on wifi links can cause choppy video.

- Default or MJPEG streaming. The default streaming is to fetch a series of jpegs from the server to give the equivalent of a live video feed. Each fetch is produced by a request from the client at a particular frame rate. An alternative method called MJPEG streaming may be selected under the system settings. Here a single request is made and the server responds with an MJPEG stream of data at the particular frame rate. Although the volume of data being sent from the server is approximately the same in both cases. The MJPEG stream avoids the multiple requests and can help smooth the flow. MJPEG stream should be used whenever possible but can cause problems with some browsers.

- Preview Size. Controlled by a camera setting. It determiens the width in pixels of the jpeg images prodcued for the live video feed. The height is scaled from this to give the right aspect ratio. Decreasing the width will make the jpeg data smaller and lower the volume which needs to be sent to the browser. Halving the width from its default 512 will cut the volume of data almost by a factor of 4.
- Quality. Controlled by a camera setting this determines the jpeg compression factor used in producing the live preview images. If th e quality is lowered then the data size of the jpegs is reduced. The default is 25 to give high quality images. It can be reduced to say 15 before significant degradation in quality takes place. This can halve the data size
- Divider. Controlled by a camera setting this determines the frame update rate of the live preview relative to the frame rate of the video capture. So a divider of 1 means the preview attempts to be the same as video frame rate (e.g. 25 fps). A divider of 5 lowers the preview frame rate to 5 fps. A low frame rate will obviously tend to a more jerky view but lowering it a bit can help if the network is congested.

Taken together these 3 factors can be used to lower the network bandwidth requirement by an order of magnitude.

Choppiness when viewing video captures cannot be directly optimised as these are high quality videos with their own bandwidth requirements. One can slight lower the data rate by redusing the Video bit rate down from its maximum default to say 2Mbits but this is not a direct reduction as the video encoder is already doing a good compression job. One can change the capture resolution set up or the frame rate to lower the speed requirement or download the material for local playback on the client machine.

Buttons don't work

Web interface is showing video OK but the action buttons like record video don't work or are disabled.

The most common cause of this is a problem getting the status of the raspimjpeg process which then controls the state of the buttons.

The status of raspimjpeg is maintained in /run/shm/mjpeg/status_mjpeg.txt which in turn is linked to by a status_mjpeg.txt in the web install folder. The web interface retrieves the status via the link. The link should have been set up correctly during the installation process providing the install.sh method was used but earlier versions sometimes didn't set the link correctly.

First check the link file exists in the web install folder e.g. ls -l /var/www/html There should be an entry status_mjpeg.txt shown as a link to the real file.

If there is not then either re-run the install.sh making sure it is the most recent version or create the link manually by

```
sudo ln -sf /run/shm/mjpeg/status_mjpeg.txt /var/www/subfolder/status_mjpeg.txt
```

replace the subfolder or leave it out according to where the install is.

You can also check the contents of the file by cat /var/www/subfolder/status_mjpeg.txt It should show 1 word typically somethink like ready for a running but idle system.

If the status file looks OK and is linked properly then the next check is to activate developer tools in the browser e.g. Chrome. You should then see this status being retrieved about every 5 seconds under the network tab.

Time Lapse conversion issues

The time lapse converter now uses gstreamer to do the work which utilises the GPU and makes it very fast.

All the necessary components are included in a full Jessie image but not in Jessie Lite. The camera install adds on the extra gstreamer-tools package but not the base gstreamer s/w which is quite a lot of stuff.

There are a few options here.

a) Install gstreamer onto Jessie Lite. (sudo apt-get install gstreamer1.0) I have done this and it then works, but I now normally use method c)

b) Use full Jessie

c) Use full Jessie and remove the major unnecessary components as described in Installation tips section.

In addition because the conversion uses the GPU the image format needs to be compatible with the GPU conversion processing. Full static image resolution conversion will not work but resolutions up to 1920 x 1080 will. If you wish to use higher resolution formats then I suggest downloading the image set for external conversion.

Watchdog Reset

The video feed from the camera can sometimes fail due to a number of reasons, be it insufficient power or camera interface errors that occur outside of raspimjpeg's control. Before the reset feature was implemented, a camera error would just cause the feed to freeze and would stay like that until a manual restart was performed.

When raspimjpeg starts it creates a child process of itself then just sits in the background doing nothing waiting for its child to terminate. If that happens then it starts another copy. The current child is what is doing the work.

The current child is also monitoring the production of cam.jpg images if it is not in an idle state. It does this by comparing the file time stamp at regular intervals. Providing the timestamp is greater than the previous then all is regarded as OK. If they are the same then this is regarded as a potential error. If there are a sequence of such errors then the child regards something as broken and exits. At that point the parent monitoring process kicks in and starts another copy.

Another explanation is as follows:
The watchdog is looking for the preview stream jpg file to be regularly updated. If that stops for more than the watchdog interval then that triggers the watchdog process. The intention here is to use the preview stream to check that the camera interfacing is functioning normally. Stopping of the stream can happen outside of raspimjpeg control with no accompanying errors. Those are normally to do with powering or cabling. Other stops (more common) will log camera interface errors in scheduler log just prior to the watchdog kicking in.

Archived or Old Material

Original Installation method

This installer method is left in for compatibility. Please use the install.sh method as described above for normal usage.

The original installation used just 1 combined script. It was set up originally for wheezy and has had some changes to make it Jessie compatible but these are not complete.

Follow Steps 1 to 3 of the install method above then

```
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
```

The script by default will install in the normal web root (/var/www). It supports installing in a subfolder of /var/www. When run it will ask a question as to what folder to use. Either hit enter to use the default or enter just the subfolder name to be used (e.g. rcam will install in /var/www /rcam). You can also edit the script and put the subfolder name in directly.

After the setup finishes, you should be able to run the ./start.sh script. If there any problems then reboot the Pi. Now just open up any browser on any computer in your network and enter the IP of the RPi as URL.

When you want to update an existing install of the software to a later version first run the script with 'update' instead of 'install'. Then if differences are detected then you should run the script a second time using 'install'

```
./RPi_Cam_Web_Interface_Installer.sh update
# if needed
./RPi_Cam_Web_Interface_Installer.sh install
```

Jessie: With the introduction of the Jessie version of the Raspberry OS the original installer was no longer working by default. The install.sh method works fine with Jessie. This is primarily due to changes on the Apache side and also to the www-data user set up. Apache now has a default web file location of /var/www/html and the www-data has nologin set up. Also the default Apache site-locations file has changed.

To make it work I set the subdirectory in the install to html to make it compatible with the Apache default. I then edited the /etc/passwd file. Some of these changes have subsequently been added to the script but the preferred method is to use install.sh as the installation method now.

www-data:x:33:33:www-data::/usr/sbin/nologin changed to www-data:x:33:33:www-data:/var/www:/bin/bash

I then separately installed libav-tools

sudo apt-get install libav-tools

With these changes it was working OK for me.

If you always get blank page in your browser, you can edit /etc/nginx/sites-enabled/rpicam, add

fastcgi_param SCRIPT_FILENAME $document_root/$fastcgi_script_name;

below "include fastcgi_params;", then restart nginx.

The installer is being worked on to try to make it work OK for both Wheezy and Jessie and to remove the need for these adjustments.