Time lapse Video

#### With Raspberry Pi and Windows Azure

# Objectives

* Produce time-lapse video, upload it onto cloud and store it as a movie file.

## Stretch Goals

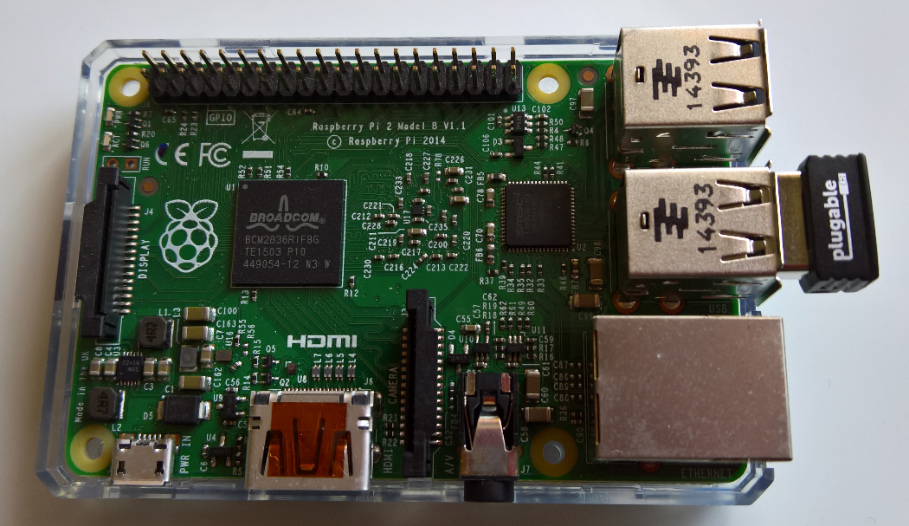
* Publish the movie using azure media services.
* send notifications of the new movie being available to the subscribers
* back-up the movie file after certain period.

# Ingredients

1. Raspberry Pi
2. Windows Azure Subscription
3. Raspberry Pi digital camera
4. Patience … lots off.

# Recipe

You’ll need a device which can capture the photo. For this project, I’ve chosen a Raspberry Pi (2) device with which a camera is attached. Few snaps of the device, camera module and connection are shown below.



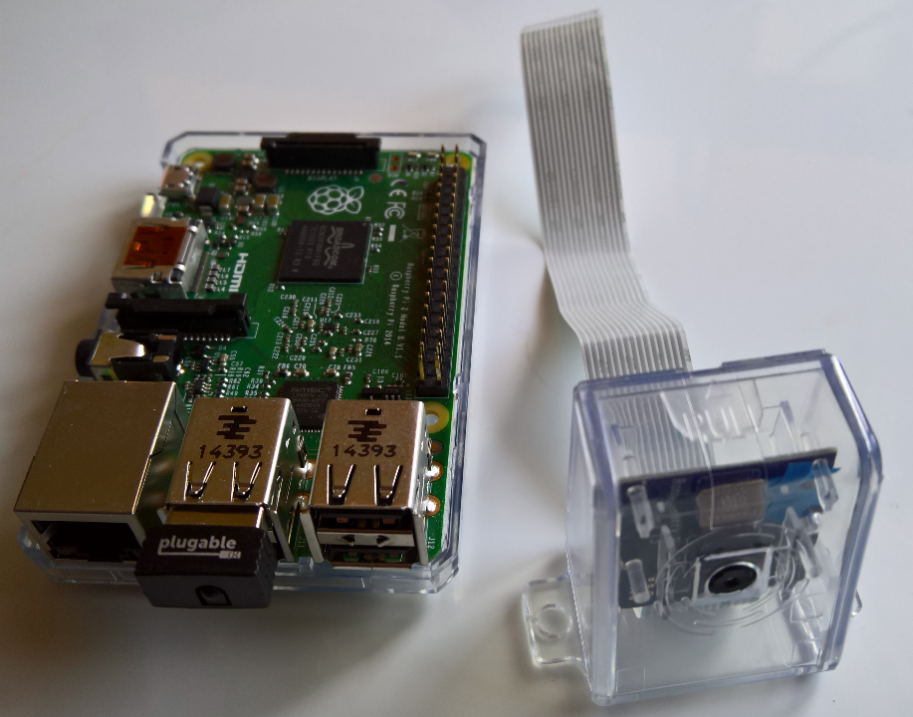




Figure 1: Raspberry Pi device and camera module

### Connecting camera module

Full documentation on how to connect the camera, configuration options to switch-on the camera operations plus trouble-shooting the same can be found [here](https://www.raspberrypi.org/documentation/usage/camera/raspicam/raspistill.md)

In order for the camera to take pictures, we’ll use the ‘raspistill’ application. This is a command line application used to capture still photographs. More technical information on it can be found [here](https://www.raspberrypi.org/documentation/usage/camera/raspicam/raspistill.md).

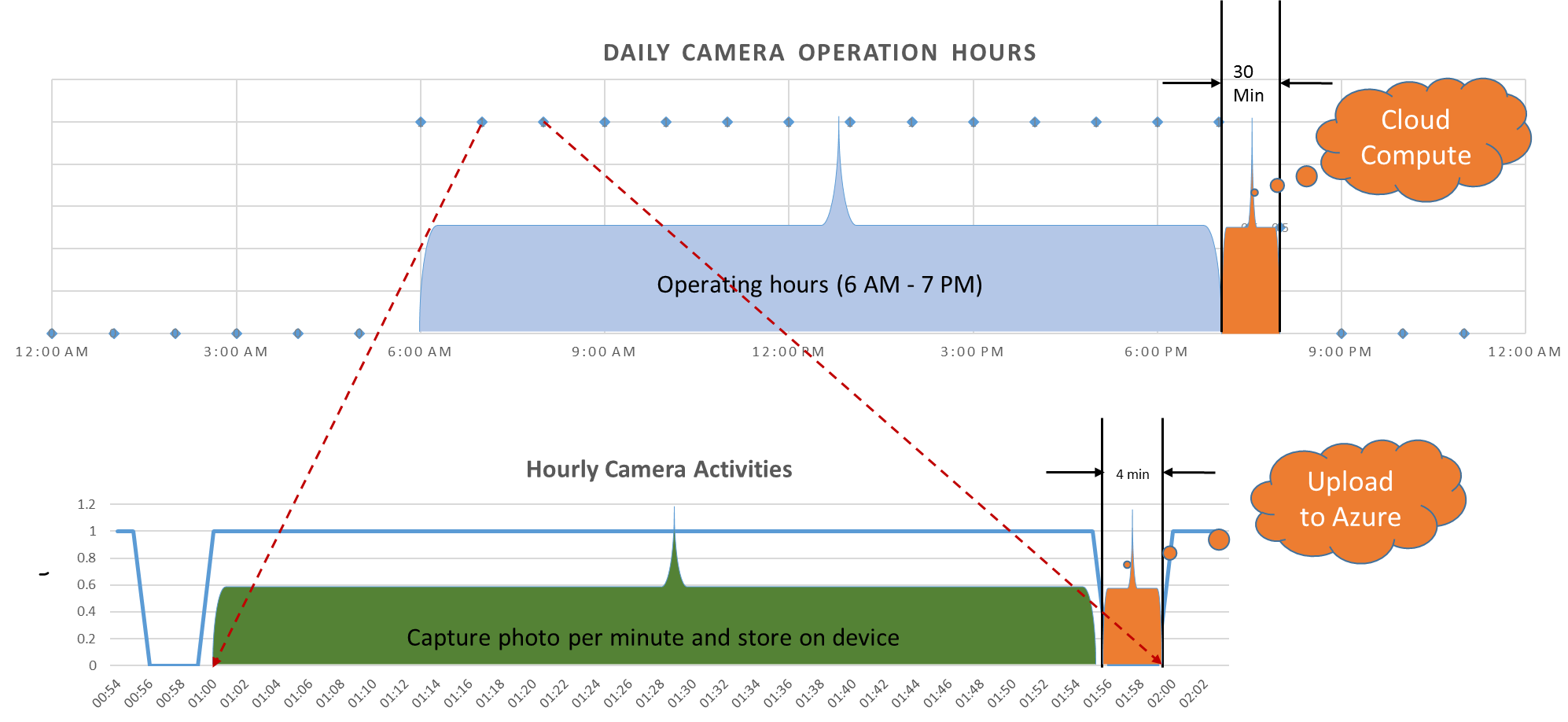
# Solution

## Plan

It’s important that we are aware of pay-as-you-go nature of the cloud economics and utilise cloud resources for the minimum time possible.

In order for us to achieve that, we would keep the device operational, taking pictures, only during the day-time, which in-turn means our Raspberry camera would be in operation between dawn and dusk. The device would take pictures through the Raspberry Pi camera module every minute, store it in a folder on the device. This will be done for the first 56 minutes of every hour. In the last four minutes of every hour, the captured photographs would be copied to Microsoft Azure storage.

At the end of day, we would start our compute instance under Azure. The job for this compute instance is to concatenate all the photographs and convert the day’s capture into a movie file. Pictorially, whole process can be depicted as following.



ASSUMPTION: OS installed on the Raspberry Pi device is the **Raspbian** Linux. The user logged on to the device is **‘pi’** and on the device, her home folder location is ‘**/home/pi’**

On the Raspberry Pi device, we created two folders (under the home folder of user ‘**pi’**).

1. **/home/pi/pictures**
2. **/home/pi/UploadPhoto**

Script for capturing a photo is contained inside **‘camera.sh’** file which is copied under the **~/pictures** folder on the Raspberry pi device and can be found under the ‘**Scripts’** folder inside the **UploadPhoto** project. Script is reproduced below. It uses raspistill application to capture the picture. The app stores the output to a file, the file name of which is derived from the date-minute stamp of the device clock.

#!/bin/bash

DATE=$(date +"%Y-%m-%d\_%H%M")

raspistill -o /home/pi/pictures/$DATE.jpg

Under the ‘**/home/pi/UploadPhoto**’ folder, app responsible for uploading all the photographs to the cloud – **UploadPhotoApp.exe and dependent assemblies** are copied. This application is written in .NET. In order to run this application on the Raspberry Pi device, we need to install the ‘mono’ project on the device. For information on how to install C# compatible run-time libraries for Raspberry Pi device, please see [this](http://www.raspberry-sharp.org/) web site.

NOTE: You may also need to install additional certificates on the device

The code of ‘**UploadPhotoApp.exe’** app is copied under the Github repository folder ‘**UploadPhoto’**. In order to compile and build the solution, load the **‘UploadPhoto.sln’** inside Visual Studio 13.

Following table describes the tasks that need to be undertaken to build, install and make the whole operation possible s well as including information on the various scripts and program files that facilitate the operations.

## Build, Installation and Operating Tasks

1. Create two file storage accounts under the Azure subscription. One account should be called ‘**Share**’ and other ‘**backup**’. The script for creating both the accounts can be found under the folder ‘**UploadPhoto\UploadPhoto\Scripts\**’ of the GitHub repository and is called ‘**CreateAzureShare.ps1**’. Update the scripts with values relevant to your Azure subscription and execute the script couple of time. First time to create the file share named ‘**Share**’ and second time to create the file share called ‘**backup**’.
   1. ‘**Share**’ cloud file storage is used to upload all the photographs from the device per hour. The movie file produced as a result of concatenation of all the photographs by the compute instance is also copied here.
   2. ‘**backup**’ cloud file storage is used for back-up purposes. Once the cloud compute instance has processed all the pictures and created the movie file, all the photographs are moved into this folder for the archiving purpose

The whole data movement operation can be pictorially represented as following.

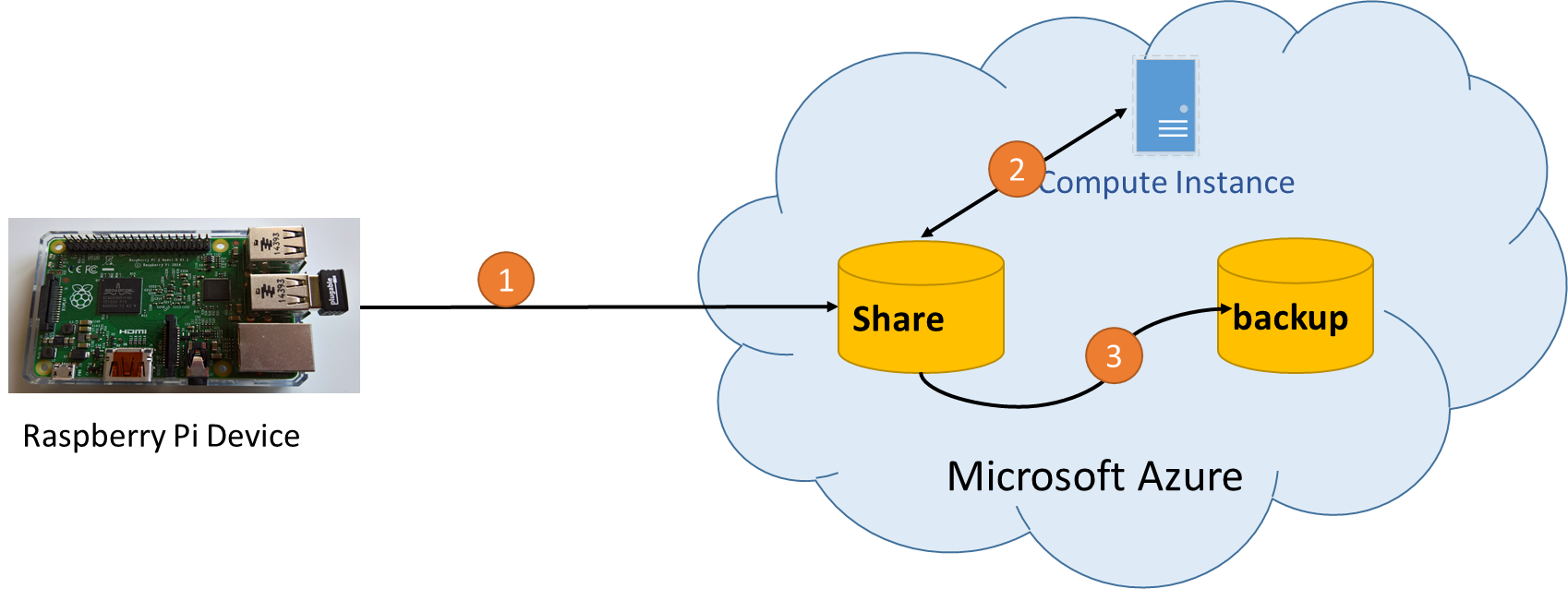


Figure 1: Data Movement

1. **UploadloadPhotoApp.exe** copies all the photographs from the device to the Azure file share – ‘**Share**’.
2. Linux virtual instance processes all the photographs at the end of day, and creates the movie file – avi format – and stores it in the same file share
3. Same virtual instance copies all the processed photographs to the backup file share.
4. Compile, build and install the **UploadPhotoApp.exe** and dependent assemblies
5. Download the code from the **Github** repository and open the solution file ‘**UploadPhoto.sln**’ inside Visual Studio 13.
   1. Update the ‘UploadPhotoApp.exe.config’ file. Modify the following line with the Azure storage that you created in the previous instance.
      1. <add key="AccountKey" value="[insert your key here]" />
6. Compile a ‘**Release**’ build. After successful compilation, you are ready to copy the binaries to the Raspberry Pi device.
7. To install the .NET application and its dependencies, use the ‘**Photo.cmd**’ batch file. This file can be found under the ‘**UploadPhoto\Scripts**’ folder.
   1. The batch file depends on ‘**Putty(putty)**’ and ‘**Putty secure copy (pscp)**’ application to copy the binary files to the device. The **putty** program and its dependent programs can be easily downloaded from the web and installed.
   2. Open ‘**Photo.cmd**’ file in an editor (Notepad?) and edit the following settings:
      1. set puttydir= [specify the folder where putty and pscp programs are copied]
      2. set prjdir= [Complete path name of Visual Studio project directory, for e.g. C:\UploadPhoto\UploadPhoto\]
      3. set rpi\_ip= [The ip address of Raspberry Pi device. For e,g, 192.168.0.8]
      4. set rpi\_usr= [Name of the authenticated user on the Raspberry Pi device. By default, its ‘**pi**’]
      5. set rpi\_pw=[insert password for the device user]
   3. Execute the **‘Photo.cmd’** batch program from the command line. It should copy the .NET binaries to the device’s ‘**/home/pi/UploadPhoto’** folder.
      1. Batch program not only copies the UploadPhotoApp binaries, it also marks the application as executable under the device system.
      2. It copies ‘**Upload.sh**’ shell program and it contains the instructions on how to invoke the .NET application using ‘**Mono**’ framework. This program is copied to **/home/pi/UploadPhoto** folder.
8. Now we are ready to configure the device to capture photographs and upload the same to Azure storage. In order for us to achieve the [**Plan**](#_Plan) described earlier, we will update the **crontab** on the Raspberry Pi device.
   1. Open the **crontab** schedule on the Raspberry pi device by executing following command
      1. >> sudo crontab -e
   2. Insert following lines at the end of the scheduler file.
      1. This following content are available inside the file ‘linuxmachine - crontab’, under the folder ‘Scripts’.
      2. # add following lines at the end of crontab file
      3. # capture the pictures every minute
      4. 0-56 0-10 \* \* \* /home/pi/pictures/camera.sh 2>&1
      5. # upload the captured images 57th minute of every hour - Total 57 pictures should be uploaded every hour
      6. 57 \* \* \* \* /home/pi/UploadPhoto/upload.sh 2>&1
      7. # Example of how to capture picture every five minutes
      8. #0,5,10,15,20,25,30,35,40,45,50,55 \* \* \* \* /home/pi/pictures/camera.sh 2>&1
   3. Save the file (CTRL+O) and exit (CTRL+X).
   4. The system should indicate that new scheduler has been created
   5. Verify that the new scheduler is capturing the photos every minute by keeping a watch on the folder **/home/pi/pictures.** 
      1. **Watch /home/pi/picture &**
   6. Verify that the photos are getting uploaded to your Azure share every hour. You can use the **Microsoft Azure Resource Explorer** tool to do this
9. Now that we’ve successfully uploaded the photographs to Azure, as next step we need to convert these photographs into a movie file. We will utilise ‘**mencoder**’ application to do this.
   1. Create an instance of a Linux machine inside Azure. For this sample, we suggest creating an instance of **Ubuntu 14.0**
   2. Start the instance, logon to instance and create/copy a script file that will map the Azure file shares and invoke the ‘**mencoder**’ application to create the movie file. The script file to be copied can be found under ‘**Scripts**’ folder and is named ‘**\UploadPhoto\Scripts\createavi.sh**’
      1. Copy the script file to the home folder of the default user.
      2. Update the script files with the storage keys of the ‘**Share**’ and ‘**backup**’ Azure storage.
   3. Mark the script file as executable by changing its mode
      1. chmod + createavi.sh
   4. Edit the /etc/rc.local file and insert the following line at the end of the file. This will ensure that the photo processing happens once the machine boots up.
      1. /home/linuxuser/createavi.sh &
      2. An example of what the **/etc/rc.local** file should look like is provided in the file ‘**machinestartup.sh**’ under the folder ‘**\Scripts**’
   5. Restart the instance
   6. Verify that the .avi file is created from all the existing photographs in the share and that all the \*.jpg’s are moved to the ‘backup’ folder. Use the **Microsoft Azure Resource Explorer** tool to do this.
   7. If you’ve come this far, give yourself a pat on the back and make sure that you provide feedback on the blog.
10. Next steps (TBC)
    1. Automate start and shutdown of the Linux instance using Azure workflows and Runbooks
11. Next steps (TBC)
    1. Automate the publication of movie file via Azure media service

# Appendix A

## Project files and their description

|  |  |
| --- | --- |
| File location | Description |
| UploadPhoto\UploadPhotoApp.exe.config | Configuration files for the UploadPhotoApp.exe application. Modify this file to specify the storage key and storage name assigned to |
| UploadPhoto\ bin\Release\UploadPhotoApp.exe and other dependent assemblies | Executable which copies all the files from ~/pictures folder to the Azure file share ‘Share’. This executable and other dependent files are copied to ~/UploadPhoto folder on the Raspberry pi device |
| UploadPhoto\UploadPhoto\Scripts\camera.sh | The script file which contains the command to take a photograph. Internally, the scripts use **raspistill** application to capture a snap. This scripts file is copied to the **~/pictures** folder on the device and is invoked by the **crontab** schedular. |
| UploadPhoto\UploadPhoto\Scripts\createavi.sh | Script file to consolidate all the time-lapse photographs and convert them into a movie file. Internally, the script first maps all the Azure file shares as cifs shares. Next it invokes **memcoder** application to create a movie file from the time-lapse snapshots. And finally it moves all the photographs from the ‘**Share**’ file share to ‘**backup**’ file share. This script is called when the Azure Linux instance is started as its referenced in **/etc/rc.local** file |
| UploadPhoto\UploadPhoto\Scripts\CreateAzureShare.ps1 | Azure script files to create the two file instances that are needed by the program. Use Azure command shell to invoke this application. You’ll need to provide the credentials to logon to Azure using the ‘**publishsettings**’ file. Where this file is used is described [here](#_Solution) |
| UploadPhoto\UploadPhoto\Scripts\linuxmachine - crontab.txt | Contains sample of how the schedular on Raspberry Pi can be scheduled to capture the snapshots every minute and upload them to the cloud. Invoke command ‘sudo crontab -e’ and add the relevant schedule at the end of the file. Make sure that you update and save the file such that system can run the latest schedule asap |
| UploadPhoto\UploadPhoto\Scripts\Photo.cmd | Script to install the UploadPhotoApp.exe and its dependencies onto the Raspberry Pi device. It also installs the script to execute **UploadPhoto** application. |
| UploadPhoto\UploadPhoto\Scripts\upload.sh | Script which specifies how the .NET application – **UploadPhotoApp.exe** – can be executed upon the framework – **Mono**. |