**A report on Raspberry PI Volcano Monitoring Based NOIR Camera**

**Sudha Rani**

**Chaitra Bhaskarachary**

**Feven Legesse**

**Submitted in partial fulfillment of the requirement of the Geo-Informatics Lab Course.**

**May 24/2019**

**Acknowledgment**

We would like to thank our Instructor, Prof. Dr. Vikram Unnithan for his encouragement and guidance. We would also like to thank our Teaching assistant, Mr. Florian Neu and Mr. Dilip Hiremath for the valuable advice and support they gave us through all the processes of the project and the course and we also thank our classmates for their support when we had a problem.

**1.Introduction**

The Raspberry Pi Foundation provides high performance, low-cost Raspberry Pi computers to educate and solve real world problems. Raspberry Pi is one of the perfect device for aspiring computer scientists. The NoIR camera module is furnished to provide high definition quality images and records video in 1080 pixels. These camera systems have both webcam and time lapse systems and time-stamps images based on Global Positioning System(GPS) input.

The system we have used here has no improvement over the previous experiments done by many scientists on time lapse or GPS time syncing using Raspberry pi. This paper presents the ideas of previous projects along with our customised scripts and different place of experiment, for effective use at any outgassing labs , Volcanos or any Flume labs to check the quality of Flume.

Our main objective in this project is to analyze data based on the images obtained from the images that were captured by the NoIR camera of the Raspberry Pi and the information regarding the geographic location which is obtained through GPS puck connected to the Raspberry Pi. We have described the deployment of pi to monitor outgassing on Jacobs University Central Heating plant by running the customized python scripts. The scripts are described in the appendix section of the report.

**2.Data Resources and Analysis**

**2.1 Equipment’s used**

**2.1.1 Raspberry pi Accessories**

Raspberry pi includes a 8-gigabyte(GB) SD card , a 5-volt(V) micro USB to 12 -V direct current power adapter, Raspberry pi case and a USB to TTL cable.

**2.1.2 Power System**

The power system we used for our test deployment is 5-volt(V) portable Power bank.Based on this power draw power bank provides power for an entire day,which was sufficient for our test deployment.

**2.1.3 Noir Camera**

Raspberry pi Noir camera is designed specially for raspberry pi computer. This camera sensor is 5 megapixels and can take 1080p high definition video.Noir camera module has near infrared filter removed and is sensitive to both visible and near infrared light. It provides streamlined command line interface for acquiring still photographs and videos.



**Figure 1 : Raspberry Pi and Noir Camera**

**2.2 Equipment for initial setup**

The common equipments requires to setup raspberry pi and Gps unit are AC power adapter for micro USB power connector , ethernet cable ,USB keyboard ,USB mouse , monitor,DVI-to-HDMI adapter or HDMI cable and close care to check the compatibility of these equipments with raspberry pi.

**2.3 Image Acquisition and Management Scripts**

The scripts for all steps has been written in Python 2.7 which can be installed in Raspberry pi using the commands given below.

Installation of python 2.7 can be done using linux command:

**sudo apt-get install python**

**2.3.1 Script 1: Fast Time-Lapse Acquisition ( Appendix 2)**

We have used two different schemes of time-lapse: one fast and other slow.Fast time lapse take several pictures per minute and slow time lapse take an image per second.We have used this script to acquire images every 20 seconds. Fast time lapse script uses script 3 to acquire the incoming images.

**2.3.2 Script 2: Slow Time-Lapse Acquisition (Appendix 3)**

This scheme is meant to take single image every few minutes. This script acquires single image along with the image file time ,which is given by NTP(Network Time Protocol) using GPS unit or network connection. Longitude, Latitude and time syncing is stamped on the image acquired. The acquired image is then stored in form of JPEG file with file name as date and time when image was taken, the file name will be in the format of year-month-day-hour.

**2.3.3 Script 3: Archive Fast Time Lapse Images (Appendix 4)**

This script is schedules by cron to run every minute to acquire the images from script 1 along with timestamp.

**2.3.4 Script 4: Get GPS Position (Appendix 5)**

This script pulls first 30 NMEA sentences from GPS serial output, which determines the GPS lock, if so longitude and latitude is recorded ,which is passed to script 2 and script 3 to be directly stamped onto the images.

**2.3.5 Script 5 and 6 (Appendix 6)**

This script checks if NTP timeserver is setting system time correctly from GPS unit .

**2.3.6 Script 7 (Appendix 7)**

This script creates video clip using default function raspivid .During the recording of video it temporarily suspends the cron scheduler only till duration of video clip so that overlapping requests are not made to the camera module. The script renames the video file with start date and time and moves it to the date-based folder structure which is used by time lapse images. Final output of this script is the metadata file which contains the gps coordinates and time synchronization status of video.

**3.Deployments**

The camera was first used in Clamv lab at jacobs University, Bremen to test the settings and working of camera for few weeks capturing images every once in a minute and 30 sec video. Gps was also able to maintain the lock and provide accurate time stamps . Images were captured from Jacobs University Central Heating plant, Bremen.

****

**Figure 2: Jacobs University Central Heating plant, Bremen**

After the setting up of our instruments, we deployed the system in Jacobs University Bremen to ensure the camera, GPS puck are working properly. The image above was captured on 12 April evening, the time and the GPS location is stamped on the image using our python script.

****

**Figure 3: Jacobs University Central Heating System**

This image was captured near the Jacobs University Central Heating System. The idea behind this is to capture the outgassing from the central heating system imitating the volcano monitoring system.

****

**Figure 4: Jacobs University Central Heating System**

****

**Figure 5: Jacobs University Central Heating System**

This image was captured near the Jacobs University Central Heating system keeping the camera close to the flumes coming out of the nitrogen cylinder proving the capability of taking pictures at odd temperatures.

****

**Figure 7: Heincke (Smoke coming out of ship funnel)**

The system was deployed at Heincke on 30-04-2019. The image above was taken to test the smoke coming out of ship funnel.

****

**Figure 8: At Heincke (Smoke coming out of ship funnel) at night**

The system was set-on overnight to capture the images in the low light environment.

**4.Conclusion**

As mentioned in the scripts above, we used different time-lapse: fast and slow. Time-lapse is where a camera takes a flow of images of a subject with a gap of time among each image. In order to acquire several images per minute, we use fast time-lapse script and for every 10 seconds it acquires an image. Whereas, the slow time-lapse script captures image every minute with latitude, longitude, and time-syncing status.

Raspberry Pi NOIR Camera which is small size around 25mm x 24mm x 9mm and over 3g weight is a custom designed as a supplement or additional for Raspberry pi. Basically, it doesn’t have ‘IR cut filter’ installed. We can attach the camera to Raspberry Pi through the two small sockets on the board upper surface. It comes without infrared filter and basically it is not for regular daylight photography. It uses CSI interface which is designed for interfacings to camera. CSI bus is capable of extremely high data rate and it exclusively carries pixel data. The camera supports 1080p30, 720p60 and 640X480p60/90 video and 2592 X 1944-pixel static images. Pi NOIR Camera is used for Black and white IR photos and low-light photography with IR lighting.

Using GPS receiver, we could get both positioning and high accuracy time capabilities of the location.

**5. Appendix A**

**5.1. System Setup**

1. Connect Raspberry Pi to monitor, mouse, keyboard, power supply and to Jacobs network using Wifi. Put in the SD card containing the linux operating system. Boot sequence shows up on the monitor and perform the following steps :

* Choose “Enable camera” to allow the operating system to work with Raspberry Pi camera.
* Choose “Internationalization Options” and change timezone to Europe/Berlin.
* Reboot the system

2. Login with user : pi and password : raspberry

3. Update Linux. In terminal type :

sudo apt-get update

sudo apt-get upgrade

sudo reboot

4. Install Python modules

sudo apt-get install python-scipy python-matplotlib

5. Install gpsd service

sudo apt-get install gpsd

6. Install ImageMagick

sudo apt-get install imagemagick

**5.2. Webserver Setup**

1. Install Apache webserver, to access the camera images data on Raspberry Pi using a web browser making the camera system behave like webcam.

sudo apt-get install apache2

sudo service apache2 restart

2. Change Apache log directory and edit the configuration file:

sudo nano /etc/apache2/envvars

change the line :

export APACHE\_LOG\_DIR=/var/log/apache2$SUFFIX

change this line to :

export APACHE\_LOG\_DIR=/var/log

**5.3. Set up Camera**

1. Shut down the computer

sudo shutdown now

1. Plug the camera ribbon into the Camera Serial Interface(CSI) connector on the Raspberry Pi computer.
2. Restart the computer and log back in.
3. Create a folder for both the USB flash drive mounting and also the camera images.

sudo mkdir /media/usb

sudo mkdir/media/usb/webcam

sudo chown pi:pi /media/usb

**5.3. Set up GPS unit and time-syncing**

1. Shutdown the computer

sudo shutdown now

2. Connect the GPS puck to Raspberry Pi

3. Confirm the GPS is working by running sudo cat /dev/ttyUSB0 command.

4. Modify NTP timeserver to read GPS data. The NTP configuration file is modified by module called gpsd.

5. Test NTP to get the GPS time.

**5.4. Setting up image capturing and scheduling scripts**

Crontab is used to run the python scripts on schedule specified . (crontab -e)

**Script 1 :** Fast time-lapse (several images per minute) - \*/10 \* \* \* \* /usr/bin/python2.7 fasttimelapse.py Every 10 mins

**Script 2 :** Slow time-lapse - \*/2 \* \* \* \* /usr/bin/python2.7 slowtimelapse.py - One image is captured every 2 minutes

**Script 3 :** Archive fast-time lapse images - \*/1 \* \* \* \* /usr/bin/python2.7 putfiles.py - Every minute

**Script 5 :** Force NTP time sync - 1 \*/2 \* \* \* /usr/bin/python2.7 ntpset.py - Every couple hours

**Script 7 :** Run video script - \*/10 \* \* \* \* /usr/bin/python2.7 getvideocmd.py -i 30 - Every 10 mins and video acquisition for 30 seconds

**Scripts 8 :** Free disk space check - 0 0 \* \* \* /usr/bin/python2.7 freespace.py - Every day

**Appendix B ( Scripts)**

1. **Fast Time-Lapse script. (fasttimelapse.py)**

|  |
| --- |
| import os #Acquire image and name file based on date-time os.chdir('/media/usb/webcam/') os.system('kill $(pgrep raspistill)') os.system('raspistill -o timelapse1%04d.jpg -q 90 -w 1024 -h 768 580000 -tl 10000 -vf -hf -ex auto -awb auto -n');  #acquire image path='/media/usb/webcam/timelapse\*jpg' |

**2.** **Slow Time-Lapse script (slowtimelapse.py)**

|  |
| --- |
| import os from pytz import timezone import time as ti import os.path import datetime import subprocess import shlex from gpspuller3 import gpspull import pytz import shutil import glob   tic=ti.time() ss=100000000 #Acquire image and name file based on date-time os.chdir('/media/usb/webcam') #acquire image. Image size is 1024 x 768. Camera waits 3 sec before capture to settle exposure. os.system('raspistill -o webcam2.jpg -q 90 -w 1024 -h 768 -t 3000 -vf -hf -n') path='/media/usb/webcam/webcam2.jpg' #stamp on gps fix [gpsfix,lat,lon,gpstime]=gpspull() print ("pull done") # debug to check that gpspull is not haenging #stamp on network connection (for time sync info) command\_line="ping -c 1 www.google.com" args=shlex.split(command\_line) try:  subprocess.check\_call(args,stdout=subprocess.PIPE,stderr=subprocess.PIPE)  s="Internet time-sync: yes" except subprocess.CalledProcessError:  s="Internet time-sync: no" for fname in glob.glob(path): print (fname) #debug name of file  t3=os.path.getmtime(fname) #get system time of filename  tt=ti.localtime(t3)  t4=datetime.datetime.fromtimestamp(t3)  s1=fname #stamp on gps coordinates if lat=='nan':  latstring='Lat: nan'  lonstring='Lon: nan'  print ("no gps fix") #debug no gpsfix else:  latstring='Lat: '+str(lat)  lonstring='Lon: '+str(lon)  print ("gps fix:", lat, lon) #debug no gpsfix #make big string at bottom t4s=str(t4) tz=ti.strftime('%Z',ti.gmtime()) bigstring=t4s+' '+tz+' | '+s+' | '+gpsfix+' | '+latstring+' | '+lonstring #save image with date filename t5=ti.strftime('%Y%m%d%H%M%S',tt) imagename=t5+'.jpg' d0='/media/usb/webcam/'+imagename cmdstring="/usr/bin/convert "+s1+" -pointsize 17 -fill white -annotate +20+760 '"+bigstring+"' "+d0 os.system(cmdstring) #Archive current image in date-time folder structure year=ti.strftime('%Y',tt) month=ti.strftime('%m',tt) day=ti.strftime('%d',tt) hour=ti.strftime('%H',tt) t=pytz.timezone('UTC') t4aware=t.localize(t4) if not gpstime=='nan':  difft=gpstime-t4aware  ss=difft.total\_seconds() if ss<120 and gpsfix=='GPS time-sync: yes' and not gpstime=='nan':  d1='/media/usb/webcam/'+year+'/'+month+'/'+day+'/'+hour+'/'+imagename Else:  d1='/media/usb/webcam/unsuretimestamp/'+year+'/'+month+'/'+day+'/'+hour+'/'+imagename  d2=os.path.dirname(d1) #if file path does not exist, make it if not os.path.exists(d2):  os.makedirs(d2)  shutil.copyfile(d0,'/var/www/image.jpg') #copy image to webserver directory #print d0 #print d1 shutil.move(d0,d1) #move file to new date folder #write metadata text file for webserver f=open('metadata.txt','w') f.write(t5+'\n') f.close() shutil.move('metadata.txt','/var/www/metadata.txt') toc=ti.time() print (toc-tic) |

**3.** **Archive Fast Time-Lapse Images ( putfiles.py)**

|  |
| --- |
| # Script 3 takes incoming images and 1) puts a timestamp on them and puts files in a date-based folder structure.Call this script in a crontab once every minute. This #function should run alongside fasttimelapse.py. This script requires that ImageMagick be installed.  import os import time as ti import os.path import datetime import subprocess import shlex from gpspuller3 import gpspull import pytz import shutil import glob tic=ti.time() path='/media/usb/webcam/timelapse\*jpg' #stamp on gps fix [gpsfix,lat,lon,gpstime]=gpspull() #stamp on network connection (for time sync info) command\_line="ping -c 1 www.google.com" #test ping address args=shlex.split(command\_line) try:  subprocess.check\_call(args,stdout=subprocess.PIPE,stderr=subprocess.PIPE)  s="Internet time-sync: yes" except subprocess.CalledProcessError:  s="Internet time-sync: no" #go through all the images that are timelapse\*jpg, stamp on text and put in date folders for fname in glob.glob(path):  t3=os.path.getmtime(fname) #get system time of filename  tt=ti.localtime(t3)  t4=datetime.datetime.fromtimestamp(t3)  s1=fname #stamp on gps coordinates if lat=='nan':  latstring='Lat: nan'  lonstring='Lon: nan' else:  latstring='Lat: '+str(lat)  lonstring='Lon: '+str(lon) #make big string at bottom t4s=str(t4) tz=ti.strftime('%Z',ti.gmtime()) bigstring=t4s+' '+tz+' | '+s+' | '+gpsfix+' | '+latstring+' | '+lonstring #save image with date filename t5=ti.strftime('%Y%m%d%H%M%S',tt) imagename=t5+'.jpg' d0='/media/usb/webcam/'+imagename cmdstring="/usr/bin/convert "+s1+" -pointsize 17 -fill white -annotate +20+760 '"+bigstring+"' "+d0 os.system(cmdstring) os.remove(fname) #Archive current image in date-time folder structure year=ti.strftime('%Y',tt) month=ti.strftime('%m',tt) day=ti.strftime('%d',tt) hour=ti.strftime('%H',tt) t=pytz.timezone('UTC') t4aware=t.localize(t4) if not gpstime=='nan':  difft=gpstime-t4aware  ss = difft.total\_seconds()  if ss<180 and gpsfix=='GPS time-sync: yes' and not gpstime=='nan':  d1='/media/usb/webcam/'+year+'/'+month+'/'+day+'/'+hour+'/'+imagename  else:  d1='/media/usb/webcam/unsuretimestamp/'+year+'/'+month+'/'+day+'/'+hour+'/'+imagename  print d1  d2=os.path.dirname(d1) #if file path does not exist, make it  if not os.path.exists(d2):  os.makedirs(d2)  shutil.copyfile(d0,'/var/www/image.jpg') #copy image to webserver directory #os.rename(d0,d1) #move file to new date folder  shutil.move(d0,d1) #write metadata text file for webserver f=open('metadata.txt','w') f.write(t5+'\n') f.close() shutil.move('metadata.txt','/var/www/metadata.txt') toc=ti.time() print (toc-tic) |

**4**. **Get GPS Position (gpspuller.py)**

|  |
| --- |
| #Script 4 reads in NMEA sentences from a serial GPS device plugged into the Raspberry #Pi. Looks at 30 sentences and takes out GPRMC line, and then reads lat and lon and time.  import os import time import re import datetime as dt import pytz def gpspull():  print ('pulling GPS time...')  a=os.listdir('/sys/bus/usb-serial/devices')  devicesetup='/dev/'+a[0]  #ttyline= 'stty -F ' + devicesetup + ' 4800'  #print (ttyline) #debug ttyline  #os.system(ttyline)  #Acquire image and name file based on date-time  os.chdir('/media/usb/webcam')  #s='head --lines=30 /dev/'+a[0]+' > gpsinfo4.txt'  s='gpspipe -r -n 30 > gpsinfo4.txt'    os.system(s)  isactive='nan'  lat='nan'  lon='nan'  gpstime='nan'  fh=open('gpsinfo4.txt')  for line in fh.readlines():  #print line[1:6]  if line[1:6]=='GPRMC':  #print 'yes'  s=re.split(',',line)  isactive=s[2]  if isactive=='A':  slat=s[3]  slat1=int(float(slat)/100)  slat2=float(slat)-(slat1\*100)  slat2=slat2/60  lat=slat1+slat2  slon=s[5]  slon1=int(float(slon)/100)  slon2=float(slon)-(slon1\*100)  slon2=slon2/60  lon=slon1+slon2  lat=str(lat)  lat=float(lat[0:11])  lon=str(lon)  lon=float(lon[0:11])  if s[4]=='S':  lat=lat\*-1  if s[6]=='W':  lon=lon\*-1  t1=s[1]  thour=int(t1[0:2])  tmin=int(t1[2:4])  tsec=int(t1[4:6])  t2=s[9]  tday=int(t2[0:2])  tmonth=int(t2[2:4])  tyear=int(t2[4:6])+2000  utc=pytz.UTC  gpstime=dt.datetime(tyear,tmonth,tday,thour,tmin,tsec,0,utc)  if isactive=='nan':  gpslock='GPS time-sync: no'  elif isactive=='V':  gpslock='GPS time-sync: no'  elif isactive=='A':  gpslock='GPS time-sync: yes'  fh.close()  return (gpslock, lat, lon, gpstime) |

**5.** **NTP Time Set (ntpset.py)**

|  |
| --- |
| #setapproxtime #Script5: ntpset.py #this function tries to ensure the NTP server has the correct time  import os from setapproximate2 import setapproxtime setapproxtime() os.system('sudo service ntp stop') os.system('sudo ntpd -q') os.system('sudo service ntp start') |

**6. Approximate Time Set (setapproximate2.py)**

|  |
| --- |
| #Script6: setapproximate2.py #this function uses the RTC time on the Ultimate GPS module (gps lock or not) #to roughly set NTP, to ensure it is close enough that NTP can do self correcti$    import os import time import datetime as dt from gpspuller3 import gpspull from pytz import timezone def setapproxtime():  [gpsfix,lat,lon,gpstime]=gpspull()  print(gpstime)  t=gpstime.astimezone(timezone('UTC'))  gpstime=t  year=str(gpstime.year)  month=str(gpstime.month)  day=str(gpstime.day)  hour=str(gpstime.hour)  mint=str(gpstime.minute)  sec=str(gpstime.second)  s1=year+'-'+month+'-'+day  s1b='sudo date --set="'+s1  s2=hour+':'+mint+':'+sec  s3=s1b+' '+s2+'"'  os.system(s3) |

**7.** **Acquire Video (getvideocmd)**

|  |
| --- |
| #Script 7 acquires h264 format video for a set duration, which is set in the command line #Script uses the Raspivi function, currently set at frame rate of 10 fps. Script then #writes a metadata file with time and position and archives video and metadata file in #date based folder structure. Script can be scheduled in cron to run at intervals.   import os import time as ti from gpspuller3 import gpspull import datetime import subprocess import shlex import argparse import shutil #manages input from command line, where you input duration of video parser=argparse.ArgumentParser(description='this takes video') parser.add\_argument('-i','--input',help='Input time in sec',required=True) arg=parser.parse\_args() print arg d=arg.input print d print type(d) d=int(d) x=d\*1000 os.chdir('/media/usb/webcam/') #use raspivid to acquire video at 10 fps try:  os.system('sudo /etc/init.d/cron stop')  ds=str(x)  tt0=ti.localtime(ti.time())  t50=ti.strftime('%Y%m%d%H%M%S',tt0)  print(t50)  a='raspivid -o video.h264 -hf -vf -b 120000000 -fps 10 -t '+ds  os.system(a) except:  print('oops') os.system('sudo /etc/init.d/cron start')   t3=os.path.getmtime('video.h264') tt=ti.localtime(t3) t5=ti.strftime('%Y%m%d%H%M%S',tt) imagename=t5+'.h264' d0='/media/usb/webcam/'+imagename os.rename('video.h264',imagename) #Archive current image in date-time folder structure year=ti.strftime('%Y',tt) month=ti.strftime('%m',tt) day=ti.strftime('%d',tt) hour=ti.strftime('%H',tt) d1='/media/usb/webcam/'+year+'/'+month+'/'+day+'/'+hour+'/'+imagename d2=os.path.dirname(d1) if not os.path.exists(d2):  os.makedirs(d2) shutil.move(d0,d1) #write metadata text file f=open('metadata.txt','w') [gpsfix,lat,lon,gpstime]=gpspull() latstring='Lat: '+str(lat) lonstring='Lon: '+str(lon) #stamp on network connection (for time sync info) command\_line="ping -c 1 www.google.com" args=shlex.split(command\_line) try:  subprocess.check\_call(args,stdout=subprocess.  PIPE,stderr=subprocess.PIPE)  s="Internet time-sync: yes" except subprocess.CalledProcessError:  s="Internet time-sync: no" currenttime='Start time of acquisition (HST): '+t50 xx='Raspberry Pi camera module' f.write(xx+'\n') f.write(currenttime+'\n') f.write(latstring+'\n') f.write(lonstring+'\n') f.write(s+'\n') f.write(gpsfix) f.close() os.chdir('/media/usb/webcam/') metaname=t5+'.txt' dx='/media/usb/webcam/'+metaname os.rename('metadata.txt',metaname) #Archive metadata in date-time folder structure year=ti.strftime('%Y',tt) month=ti.strftime('%m',tt) day=ti.strftime('%d',tt) hour=ti.strftime('%H',tt) d1='/media/usb/webcam/'+year+'/'+month+'/'+day+'/'+hour+'/'+metaname d2=os.path.dirname(d1) if not os.path.exists(d2):  os.makedirs(d2) shutil.move(dx,d1) |

**8. Maintain Free Space on Flash Drive (freespace.py)**

|  |
| --- |
| #!/usr/bin/python  # function checks space on flash drive and deletes directories as necessary  #to maintain sufficient free space to keep acquiring images import os import shutil path='/media/usb/webcam' st=os.statvfs(path) free=(st.f\_bavail\*st.f\_frsize) free=free/(1e9) thresh=6 #Gb to be left on disk  #keep deleting date folders until sufficient space cleared while free<thresh:  os.chdir(path) #remove a folder to make room  s=os.listdir(path)  year=min(s)  if os.listdir(year)==[]: #delete directory  shutil.rmtree(year)  else:  os.chdir(year)  s=os.listdir('./')  month=min(s)  if os.listdir(month)==[]: #delete directory  shutil.rmtree(month)  else:  os.chdir(month)  s=os.listdir('./')  day=min(s)  shutil.rmtree(day) #recalculate space  st=os.statvfs(path)  free=(st.f\_bavail\*st.f\_frsize)  free=free/(1e9) |

**References**

[1] MR Patrick, TR Orr, L Lee, CJ Moniz - 2015, “A Multipurpose Camera System for Monitoring Kilauea Volcano, Hawaii, retrieved from <https://pubs.usgs.gov/tm/13/a2/tm13-A2.pdf>

[2] Grey Loyse (2017), “Raspberry pi Documentation Release 0.0”

[3]“How to use your Raspberry Pi as a wireless access point”, retrieved from <https://thepi.io/how-to-use-your-raspberry-pi-as-a-wireless-access-point/>

[4]“How to install /use the Raspberry Pi Camera”, retrieved from <https://thepihut.com/blogs/raspberry-pi-tutorials/16021420-how-to-install-use-the-raspberry-pi-camera>

[5] Kevin Townsend, “Introduction”, retrieved from <https://learn.adafruit.com/adafruit-ultimate-gps-on-the-raspberry-pi?view=all>

[6] Andrew Back, “Add GPS Time and Location to a Raspberry Pi Project”, retrieved from <https://www.rs-online.com/designspark/add-gps-time-and-location-to-a-raspberry-pi-project>

[7] Raspberry Pi Camera Products, retrieved from <https://www.adafruit.com/product/1567>

[8] “How to do time lapses with the Raspberry Pi”, retrieved from <https://www.raspberrypi.org/forums/viewtopic.php?f=43&t=67109&sid=1101873fbb0ad8c01312fa5c06c93d24>

[9] S Saha, A Singh, P Bera, MN Kamal, S Dutta, U Goria - 2017 8th IEEE Annual Information technology, 2017 retrieved from <https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8103827>