USGS
science for a changing world

A Multipurpose Camera System for Monitoring Kīlauea Volcano, Hawai‘i

Chapter 2 of
Section A, Methods Used in Volcano Monitoring
Book 13, Volcano Monitoring

Techniques and Methods 13–A2

U.S. Department of the Interior
U.S. Geological Survey

COVER. Photograph of camera at Kīlauea Volcano, Hawai‘i, used to monitor ongoing summit lava lake activity at Halema‘uma‘u Crater (U.S. Geological Survey photograph by Matthew R. Patrick).

A Multipurpose Camera System for Monitoring Kīlauea Volcano, Hawai‘i

By Matthew R. Patrick, Tim Orr, Lopaka Lee, and Cyril Moniz

Chapter 2 of
Section A, Methods Used in Volcano Monitoring
Book 13, Volcano Monitoring

Techniques and Methods 13–A2

U.S. Department of the Interior
U.S. Geological Survey

U.S. Department of the Interior
SALLY JEWELL, Secretary

U.S. Geological Survey
Suzette M. Kimball, Acting Director

U.S. Geological Survey, Reston, Virginia: 2015

Suggested citation:
Patrick, M.R., Orr, T., Lee, L., and Moniz, C., 2015, A multipurpose camera system for monitoring Kīlauea Volcano, Hawai'i: U.S. Geological Survey Techniques and Methods, book 13, chap. A2, 25 p., http://dx.doi.org/10.3133/tm13A2.

ISSN 2328-7055 (online)

Contents

Figures

Tables

Supplementary Videos

[Available online only at http://pubs.usgs.gov/tm/13/a02/]

1. Halema‘uma‘u plume time-lapse. This video shows an image every 10 minutes, from February 3, 2014, at 0001 Hawai‘i Standard Time (HST) to February 9, 2014, at 2359 HST. The movie shows the commonly fluctuating wind directions typical of winter months, when the normally steady trade winds become unstable. The camera was positioned in the Hawaiian Volcano Observatory observation tower. In the lower right corner of the image is the public overlook at Jaggar Museum.

2. Halema‘uma‘u lava lake time-lapse. This video shows an image every minute, from February 14, 2014, at 1200 Hawai‘i Standard Time (HST) to February 15, 2014, at 1200 HST. The plot of RSAM (real-time seismic amplitude measurement), which can be taken as a proxy for the amplitude of seismic tremor, is shown below. Spikes in RSAM correspond with the appearance of additional spattering sources on the lake margin, whereas the sustained low level in RSAM after about 0800 on February 15 is an indicator of the absence of spattering at the lake and very quiet activity.

3. Halema‘uma‘u lava lake video clips. Four clips from February 2014 are shown, taken at the following times: (1) February 14, 1200 Hawai‘i Standard Time (HST); (2) February 14, 1800 HST; (3) February 15, 0000 HST; and (4) February 15, 0600 HST. Videos are shown at 3× speed.

A Multipurpose Camera System for Monitoring Kīlauea Volcano, Hawai‘i

By Matthew R. Patrick, Tim Orr, Lopaka Lee, and Cyril Moniz

Abstract

We describe a low-cost, compact multipurpose camera system designed for field deployment at active volcanoes that can be used either as a webcam (transmitting images back to an observatory in real-time) or as a time-lapse camera system (storing images onto the camera system for periodic retrieval during field visits). The system also has the capability to acquire high-definition video. The camera system uses a Raspberry Pi single-board computer and a 5-megapixel low-light (near-infrared sensitive) camera, as well as a small Global Positioning System (GPS) module to ensure accurate time-stamping of images. Custom Python scripts control the webcam and GPS unit and handle data management. The inexpensive nature of the system allows it to be installed at hazardous sites where it might be lost. Another major advantage of this camera system is that it provides accurate internal timing (independent of network connection) and, because a full Linux operating system and the Python programming language are available on the camera system itself, it has the versatility to be configured for the specific needs of the user. We describe example deployments of the camera at Kīlauea Volcano, Hawai‘i, to monitor ongoing summit lava lake activity.

Introduction

Remote field cameras are essential tools for monitoring volcanic activity, with the images providing an invaluable visual framework for interpreting the multitude of other data streams coming into a volcano observatory (Poland and others, 2008; Behncke and others, 2009). The U.S. Geological Survey (USGS) Hawaiian Volcano Observatory (HVO) has found automated camera systems to be particularly useful for monitoring highly dynamic activity at Kīlauea Volcano, Hawai‘i, over the past 6 years (Hoblitt and others, 2008; Orr and Hoblitt, 2008; Patrick and others, 2010a,b, 2011; Orr and Rea, 2012; Orr and others, 2013), and the history of remote field cameras on Kīlauea extends back several decades (Wolfe and others, 1988; Thornber and others, 1997). These camera systems have been both webcams (images are telemetered in real-time back to the observatory) and time-lapse systems (images are not telemetered but stored on the camera for periodic retrieval by field personnel). However, the specific needs for these camera systems vary widely based on the eruption style and other circumstances. Most affordable webcams available on the market, unfortunately, are difficult or impossible to customize to meet individual needs. For instance, we are not aware of any affordable webcam that time-stamps images based on Global Positioning System (GPS) input. Accurate timing of images is essential for comparison with other datasets (for example, geophysical data). To solve these limitations, we have assembled a simple camera system that uses a single-board computer (Raspberry Pi) running Linux and Python, enabling complete customization and tailoring to individual needs.

The Raspberry Pi (table 1) is a small, low-cost ($35) computer that runs the Linux operating system and has been used in many hobbyist projects, including those focusing on time-lapse photography. The system we describe here does not have any particularly novel improvements over previous projects shared on the Internet, as time-lapse and GPS time-syncing have already been done with the Raspberry Pi by many hobbyists (http://www.raspberrypi.org/tag/time-lapse/, accessed January 2015). We simply present a system that combines previous uses of the Raspberry Pi, coupled with our own customized scripts and modified enclosure, for effective use at active volcanoes. The camera may be used as a webcam (when combined with telemetry equipment) or as a time-lapse camera. The focus of this report is the camera system itself, and we do not go into detail on the power system or telemetry. Power and telemetry systems for volcano-monitoring cameras are described in Hoblitt and others (2008), Orr and Hoblitt (2008), Paskievitch and others (2010), and Patrick and others (2014). In this paper we describe two sample deployments on Kīlauea Volcano to monitor ongoing lava lake activity in Halema‘uma‘u Crater.

Equipment

The equipment needed for a compact multipurpose camera system for field deployment at active volcanoes based on a system used at Kīlauea Volcano, Hawai‘i, is described below.

Raspberry Pi Model B

The Raspberry Pi Model B is a small (fig. 1), low-cost ($35) single-board Linux computer that has a 700-megahertz (MHz) ARM processor with 512 megabyte (Mb) random access memory (RAM), two universal serial bus (USB) 2.0 ports, an Ethernet port, and a Secure Digital (SD) memory card slot for data storage and the operating system (http://www.raspberrypi.org; accessed April 2014).

Table 1. Equipment list and approximate costs for compact multipurpose camera system designed for field deployment at active volcanoes based on a system used at Kīlauea Volcano, Hawai‘i.

[Excludes costs for tripod, power system, and telemetry. SD, Secure Digital; USB, universal serial bus; TTL, transistor–transistor logic; DC, direct current]

Item	Manufacturer/supplier	Cost in U.S. dollars
Raspberry Pi Model B	Raspberry Pi	35
Raspberry Pi NoIR camera module	Raspberry Pi	30
Ultimate GPS module version 3	Adafruit	40
SD memory card (8 gigabyte)	SanDisk	8
DC 12 volt to USB 5-volt power adapter	Fulree, Drok	12
USB to TTL serial cable	Adafruit	10
Raspberry Pi case	SB components	9
Raspberry Pi camera mount	Pimori	10
Pelican case 1120	Pelican	25
Low-profile USB flash drive	Various	15
Total		194

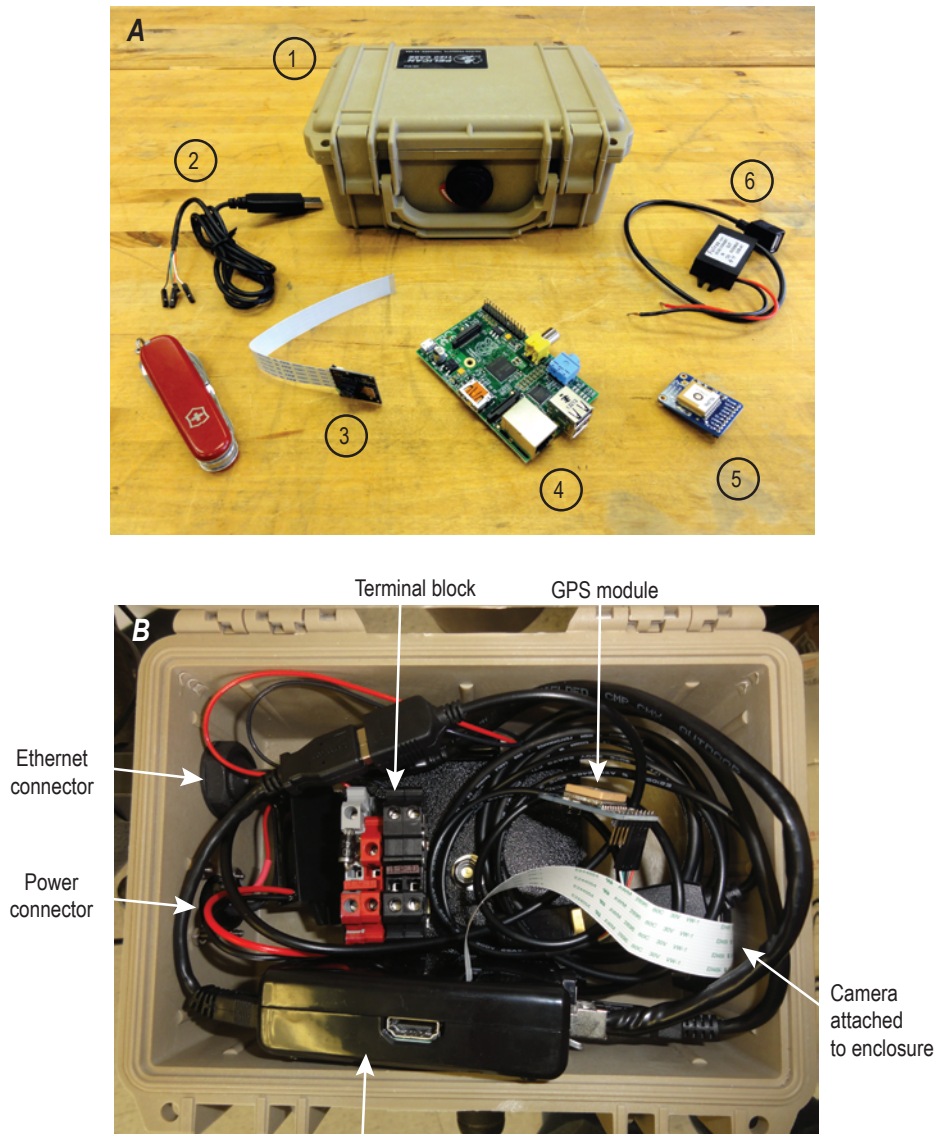

Figure 1. Photographs of the compact multipurpose camera system used at Kīlauea Volcano, Hawai‘i. *A*, Main camera components—the camera system is enclosed in a waterproof Pelican case (1) and includes the Raspberry Pi computer (4), the Raspberry Pi NoIR camera module (3), and the Adafruit Ultimate GPS module (5), which is connected by the USB to TTL (universal serial bus to transistor–transistor logic) cable (2). The power connection in the enclosure consists of the 12-volt direct current (DC) to 5-volt USB converter (6) and DIN-rail terminal block (not shown). A few minor cables and other components are not shown. The low-profile USB flash drive is not shown. *B*, Camera system within its enclosure (tan Pelican case). Downward view of the components connected within the Pelican case. The camera module (partially obscured) is attached to the right face of the case. Power and Ethernet are connected at the left face. (U.S. Geological Survey photographs by Matthew R. Patrick.)

Raspberry Pi NoIR Camera Module

The Raspberry Pi NoIR camera module is designed specifically for the Raspberry Pi computer and is very inexpensive ($30). The camera sensor is 5 megapixels (2,592×1,944 pixels) and can do 1080p (1,080 horizontal lines of vertical resolution) high-definition video at 30 frames per second. It is a fixed-focus camera with a horizontal field of view of about 55°. The default software, available from RaspberryPi.org, provides a streamlined command-line interface for acquiring still photographs and video that can be easily added to scripts.

This NoIR camera module is "low-light" in the sense that it has the near-infrared filter removed and is thus sensitive to both visible and near-infrared light. HVO has had good success with such low-light cameras for two reasons. First, they tend to be very sensitive to the near-infrared radiation emitted by active lava and thus can be used to detect active lava far better than visible-wavelength cameras. Second, we have found that such low-light cameras tend to provide better views through thick volcanic fume than visible-wavelength cameras, presumably due to the longer wavelengths (Patrick and others, 2012; Orr and others, 2013). This ability to "see" through fume provides a great benefit when monitoring the active lava lake at Halemaʻumaʻu Crater. The lake is contained within the "Overlook crater," a nested crater within Halemaʻumaʻu Crater, and is often obscured by thick fume to the naked eye.

Ultimate GPS Module Version 3

The Adafruit Ultimate GPS module is a small (fig. 1) low-cost ($40) GPS unit that is easy to interface with and commonly used for Raspberry Pi projects. Although the unit can provide PPS (pulse per second) timing accuracy (which can be used to get microsecond timing accuracy), we use only the National Marine Electronics Association (NMEA) timing from the serial feed, which is only accurate to within about a second. The NMEA protocol was simpler to work with and the timing is adequate for our camera system. This unit is attached to the USB port on the Raspberry Pi using the USB to TTL (transistor–transistor logic) cable. It includes a slot for a small battery that allows the module to retain a relatively accurate time even when the power to the Raspberry Pi is cut off, and makes the Ultimate GPS module act as a real-time clock (RTC). The battery also allows faster GPS locks.

Raspberry Pi Accessories

Accessory Raspberry Pi equipment includes an 8-gigabyte (Gb) SD card, a 5-volt (V) micro-USB to 12-V direct-current (DC) power adapter, a Raspberry Pi case, a 16-Gb USB flash drive and a USB to TTL cable. The USB drive needs to be a low-profile form factor in order to fit into the enclosure.

Enclosure: Pelican Case Model 1120 and Accessories

A small enclosure is needed to protect the camera system from rain and volcanic gas. We choose Pelican cases because they are water tight and plastic and therefore do not corrode like metal enclosures. These types of enclosures have a good track record in thick volcanic fume on Kīlauea (Harris and others, 2005; Orr and Hoblitt, 2008; Patrick and others, 2014).

The Pelican case enclosure was modified in several ways. First, we cut a small hole in the front of the box and covered the hole with a piece of plexiglass to provide a window for the webcam. Two size 2-56 screws passed through the front of the box to attach the plexiglass to the outside of the case, as well as fasten the Raspberry Pi camera to the inside of the case (using the Raspberry Pimoroni camera mount to aid in a tight fit). At the opposite end of the box we cut two holes and installed connectors for the power and Ethernet cables. We attached the bottom of the Pelican case to a ¼-inch camera screw that had a ¼-inch female attachment at the bottom to fasten to a standard camera tripod. All holes cut into the Pelican case were well sealed with silicone sealant to ensure waterproofness (following Harris and others, 2005; Hoblitt and others, 2008).

The power cable for the Raspberry Pi computer (which also powers the camera and GPS module) is a 5-V micro-USB connector, and we plugged in a micro-USB to normal USB cord, which is then plugged into a 5-V USB to 12-V DC converter. The positive and negative 12-V wires from this converter are then plugged into a small DIN-rail terminal block (fig. 1) inside the enclosure. The switch on the terminal block was removed and a diode, used to avoid short-circuiting, was soldered in its place. A 3-ampere (A) fuse was also attached to the terminal block.

Tripod

To save costs, we used an old Tiltall tripod (fig. 2) from our inventory. New Tiltall tripods cost about $120. Large rocks are piled around the legs of the tripod to ensure stability in high winds. We emphasize that stability of the tripod and camera is vital to construct steady long-term time-lapse sequences.

Power System

The power system we used for our test deployments consisted of four 10-A-hour Lithium polymer batteries (fig. 2). We used an Extech 380942 clamp meter to monitor the power consumption of the system and measured an average draw of roughly 250 milliamps (at 12 V). Based on this average power draw (3 Watts, W) the battery pack would provide about a week of power, which was sufficient for our test deployments. For long-term deployments, we typically use a solar-powered system consisting of a combination of 12-V lead/acid batteries and 80-W solar panels (Hoblitt and others, 2008; Orr and Hoblitt, 2008; Patrick and others, 2014).

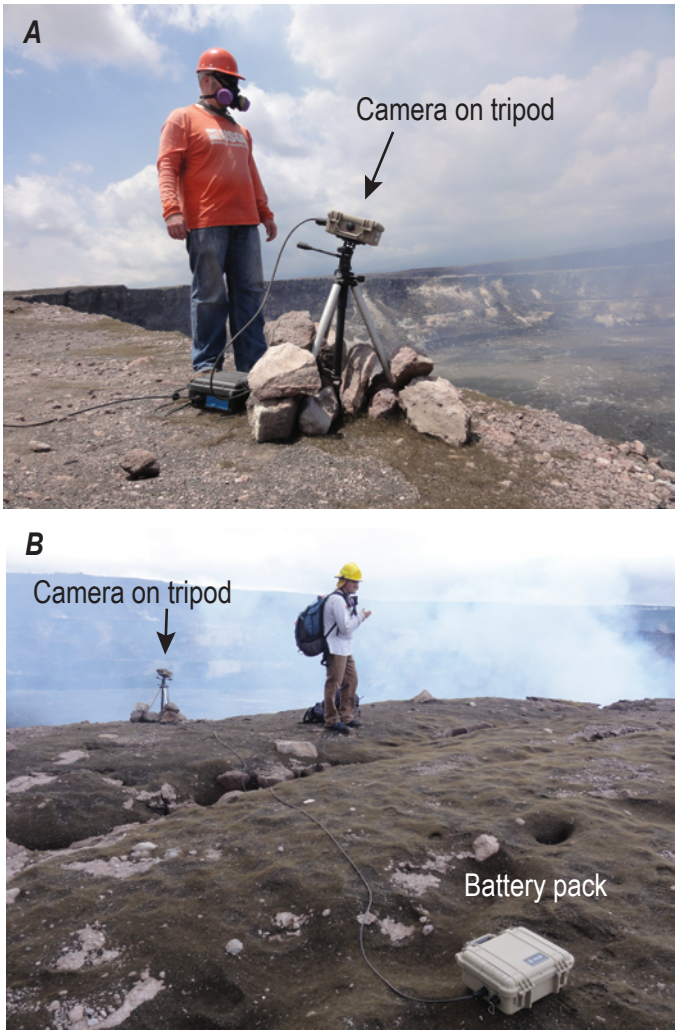

Figure 2. Photographs of the compact multipurpose camera system in the field at Kīlauea Volcano, Hawai‘i, at the edge of Halema‘uma‘u Crater. *A*, The camera is within the tan Pelican case, atop a standard photography tripod, and pointed towards the active lava lake (out of view at the right). Note that the tripod legs are secured with rock piles to ensure stability of the camera. *B*, The battery pack (40 ampere-hour lithium polymer batteries) is shown for the temporary deployment and can supply about a week of power. (U.S. Geological Survey photographs by Matthew R. Patrick.)

Telemetry

The focus of this paper is the camera system itself so we do not discuss in detail the telemetry system for the camera, should it be used as a webcam. Nevertheless, to transmit webcam images back to the observatory we normally use one of two methods. The first is a WiFi (local-area wireless) radio system, such as the one described by Hoblitt and others (2008). The second is a cellular modem that can transmit images when the camera is within cellular phone reception. We have used a Sierra Wireless AirLink Raven XE cellular modem with a high-gain antenna, which cost about $600 to purchase. Data access cost about $40 a month through Verizon Wireless. Choice of telemetry system can greatly increase power consumption.

Equipment for Initial Setup

Some equipment is needed to set up the Raspberry Pi and GPS unit and configure them for field operation. Most of this equipment is freely available in a typical office environment, and so we do not include it in the cost of the system. This equipment includes an AC (alternating current) power adapter for the micro-USB power connector, Ethernet cable, powered USB hub, USB keyboard, USB mouse, monitor, DVI-to-HDMI (Digital Visual Interface to High Definition Vector Imaging) adapter (assuming monitor has DVI input) or HDMI cable (if monitor has HDMI input). Care must be taken to ensure that this equipment is compatible with the Raspberry Pi.

Image Acquisition and Management Scripts

All scripts are written in Python 2.7, which comes preinstalled on the Raspbian Debian Linux Wheezy distribution.

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

We use two different time-lapse acquisition schemes—one fast and the other slow. The fast time-lapse script (script 1) is meant to acquire several images per minute. We have used this script to acquire an image every 10 seconds (s) with good success. Note that this script requires a few seconds to archive each image, so we do not recommend intervals less than 5 s—for shorter intervals, periodic video acquisition can be scheduled. The fast time-lapse setup requires an associated script (script 3) to run alongside it and archive the incoming images.

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

The slow time-lapse scheme (script 2) is meant to acquire an image every few minutes, with the fastest rate limited by the "cron" scheduler at one image per minute. This script is run by the "crontab" at an interval chosen by the user; in other words, the crontab interval determines the imaging frequency. The script acquires a single image from the camera (using function raspistill), gets the image file time from the computer, which is itself set by Network Time Protocol (NTP) using either the GPS unit or network connection, and then imprints this time on the image. The latitude and longitude, and time-syncing status are also stamped on the image. The script then exports this image to a high-resolution JPEG file, with the file name based on the image date and time. Once an image is acquired, it is moved to a folder within a date-time directory structure with the following hierarchy: year-month-day-hour. The script archives the images onto the USB flash drive, which can then be easily swapped in the field without powering down the camera. The SD card also allows image storage, but to get these files the SD card must be removed in the field (abruptly shutting down the system). We found that

these shutdowns occasionally corrupted the SD card, sometimes losing data. Therefore, writing data to a removable USB flash drive is preferred.

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

Script 3 is scheduled to run every minute by cron in order to timestamp and archive the incoming images from script 1 (fast time-lapse). The script also stamps the geographic coordinates. Image archiving is done on the USB flash drive.

Script 4: Get GPS Position (Appendix 5)

Script 4 is run by scripts 2 and 3 immediately before time-stamping the image. The script pulls 30 NMEA sentences from the GPS serial output and finds a NMEA GPRMC (recommended minimum specific GPS/transit data) sentence. From this sentence it determines if a GPS lock is active, and if so, it records the latitude and longitude. The lock status, and latitude and longitude are saved as variables that are passed to scripts 2 and 3, and then stamped directly on the image.

Scripts 5 and 6: Ensure NTP Server Working Correctly (Appendix 6)

Scripts 5 and 6 ensure that the NTP timeserver is setting the system time correctly from the GPS unit.

Script 7: Acquire Video (Appendix 7)

Script 7 simply runs the default Raspberry Pi video function (raspivid) with a single input for video clip duration. The script provides some additional functionality. First, it temporarily suspends the cron scheduler for the duration of the video clip so that overlapping requests (from video and time-lapse image acquisitions) are not made to the camera module; simultaneous requests can make the camera freeze and require reboot. Second, the script renames the video file based on the start date and time and moves it into the date-based folder structure used by the time-lapse images. Finally, the script creates a metadata file for the video that contains the GPS coordinates and time-synchronization status.

Script 8: Ensure Flash Drive Does Not Fill Up (Appendix 8)

Script 8 simply checks the available storage space on the flash drive each day. If the free space falls below a specified threshold (for example, 500 Mb), the script deletes the oldest day's worth of data and repeats this until the free space gets above the threshold. In this manner, the system maintains only the most recent data on the flash drive.

Acquisition Schemes

Time-Lapse System Setup

Following the setup detailed in appendix 1, the camera system will be configured to act as a time-lapse camera system, so no additional work is needed for the camera to operate in this fashion. Periodically, personnel will need to visit the camera and move the camera images off of the flash drive or simply swap flash drives. Although the images would not be available for real-time operational monitoring, they could be helpful for research or monitoring long-term processes that pose no immediate hazard.

Webcam System Setup

In the webcam mode, the camera images are acquired in the same manner as the time-lapse setup but are transmitted to an observatory in near-real-time (within minutes), which is essential for real-time operational monitoring. The only change to the camera system is that it must be connected to a network, which can be done in one of two ways. First, the Raspberry Pi might be connected, using an Ethernet cable or WiFi adaptor, to an existing network that transmits data, either directly or by the Internet, to an observatory (Hoblitt and others, 2008). Second, the Raspberry Pi might be plugged in to a cellular or satellite modem to transmit the images through the Internet to an observatory. For the past several years we have used the Sierra Wireless Airlink Raven XE cellular modem, which connects to the Internet using the Verizon cellular network, with other camera systems. We tested this Raspberry Pi camera system (with the Apache webserver installed as described below) with this cellular modem successfully.

At the observatory, a script must be run to periodically (for example, every few minutes) connect to the remote Raspberry Pi computer in the field and pull the most recently acquired image. The time-lapse acquisition script that we use on the Raspberry Pi (script 2 or 3) saves the current image to a constant file name (overwriting the file with the current image) in the webserver directory, along with a metadata file with image time. The webserver on the Raspberry Pi allows the current image and metadata file to be accessed using a Web browser or to be automatically downloaded using the "wget" command.

Video Acquisition Setup

For most long-term continuous monitoring of volcanic activity, low image-acquisition rates (for example, an image every 1–10 minutes) are adequate and, in fact, preferred to higher frame rates to keep storage space manageable. In special circumstances, however, high frame-rate acquisition may be desired for very dynamic processes, such as lava fountaining or lava lake draining. The Raspberry Pi camera module uses a video program called "raspivid" that makes video acquisition very straightforward. In script 7 (appendix 7) we have included this program along with time-stamping, and this script can be scheduled using a cron job or called on demand.

Deployments

The camera system was deployed in several locations to monitor the ongoing eruption at the summit of Kīlauea Volcano on the Island of Hawai‘i. The summit eruption began in March 2008 (Wilson and others, 2008) and continues at the time of writing (early 2015) (Patrick and others, 2013, 2014). For the past several years, the activity has consisted of a lava lake in Halema‘uma‘u Crater, which is in the southwestern part of Kīlauea's summit caldera. The lake itself is deep within a new crater (informally called the "Overlook crater," due to its position immediately below the site of the Halema‘uma‘u visitor overlook). The lake undergoes frequent changes in lava level and emits a continuous plume of volcanic gas. This plume normally drifts southwest with the trade winds, and reacts in the atmosphere to form volcanic fog, or "vog." Vog is a respiratory irritant that affects the health of residents downwind and negatively impacts the local agricultural industry (Patrick and others, 2013).

Monitoring the Outgassing Plume From the HVO Tower

The camera was first deployed and operated within the HVO observation tower to test some of the configuration settings and ensure the camera system was working correctly. The system ran for more than a week (February 3–9, 2014) without problems, acquiring an image every minute (script 2) and a 30-s video sequence every 6 hours (script 7). The GPS unit was able to maintain a lock through the windows of the tower and provide accurate time stamps.

The camera captured the variations in the outgassing plume (fig. 3, supplementary video 1), which fluctuated in direction with the prevailing winds. Normally, the plume is carried to the southwest with the trade winds (Elias and Sutton, 2012), but trade winds are unstable in the winter months. The direction the plume is carried determines which parts of the island experience the unpleasant effects of vog. During the week of observations, the wind direction and plume shifted many times.

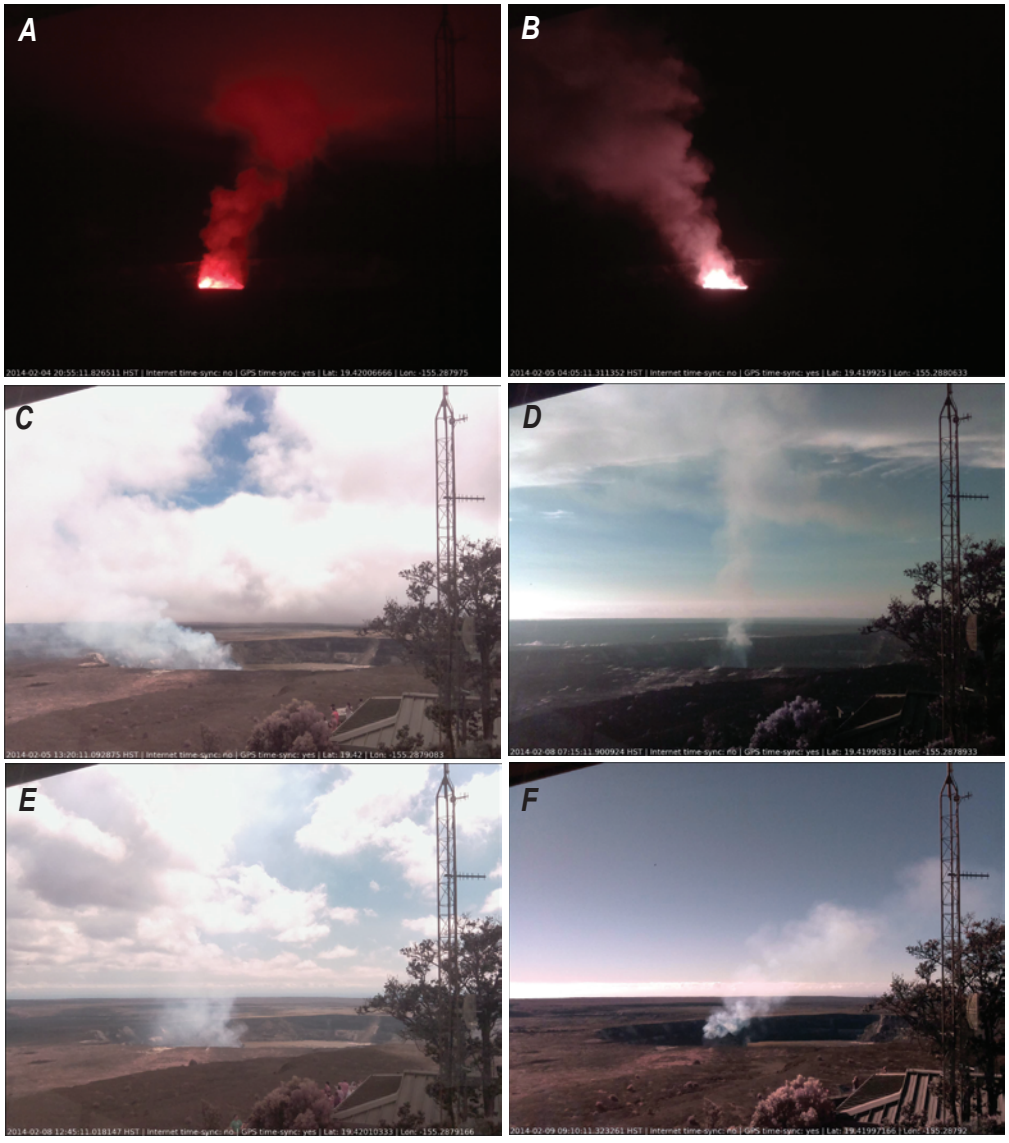

Figure 3. Photographs of the summit eruption plume at Kīlauea Volcano, Hawai‘i, taken by the compact multipurpose camera system in early February 2014. The photographs show the changing plume behavior due to fluctuations in wind direction and wind speed. The time and position imprints are at the bottom of each image in white text but are too small to read in this figure. *A*, February 4, 2055 Hawai‘i Standard Time (HST); *B*, February 5, 0405 HST; *C*, February 5, 1320 HST; *D*, February 8, 0715 HST; E, February 8, 1245 HST; and F, February 9, 0910 HST. (U.S. Geological Survey photographs.)

Monitoring the Lava Lake From the Rim of Halemaʻumaʻu Crater

The second test deployment was at the rim of Halemaʻumaʻu Crater, about 200 meters (m) west of the closed visitor overlook, to image the Overlook crater and directly monitor the lava lake (figs. 2 and 4). The camera operated for 1 day (February 14, 2014, 1200 Hawaiʻi Standard Time (HST) to February 15, 2014, 1200 HST), using the lithium polymer battery pack for power. Still images were acquired every minute (script 2) and 30-s video clips were acquired every 6 hours (script 7). Image acquisition ran correctly, as did the GPS time synchronization.

The images depicted the lava lake well (fig. 5, Supplementary video 2). The lake surface normally consists of large crustal plates (black to the naked eye) separated by thin incandescent cracks (red to the naked eye) (fig. 5). These incandescent cracks can have surface temperatures of more than 500 degrees Celsius (°C) based on analysis with a thermal camera (Patrick and others, 2014). The cracks show up as white in the Raspberry Pi NoIR camera images, presumably due to the large amount of near-infrared radiation these incandescent cracks are emitting. A persistent spattering source is normally present in the southeast part of the crater at the lake margin (right side in images, fig. 5*A*, *B*).

The image sequence captures several interesting changes in the lava lake that correlate clearly with fluctuations in seismic tremor—in figure 5 shown as RSAM (real-time seismic amplitude measurement) (Endo and Murray, 1991; see also supplementary videos 2 and 3). On several occasions a transient spatter source appears on the east margin of the lake (far side of lake in images), and this spatter source migrates towards the southeast, presumably carried by the prevailing current in the lake (similar to the "traveling fountains" mentioned by Perret (1913) in the early Halemaʻumaʻu lava lake, as well as the migrating fountains observed by Patrick and others, 2011). The RSAM spikes when these brief spatter sources appear, suggesting that spattering and the associated shallow outgassing are closely related to the seismic tremor.

Figure 4. Photograph taken February 14, 2014, of camera monitoring the lava lake at the summit of Kīlauea Volcano, Hawaiʻi. The camera is positioned on the rim of Halemaʻumaʻu Crater, 200 meters (m) west of the "Overlook crater" (center of image), which contains the active lava lake. For scale, the Overlook crater is 160 × 210 m in size, and the lava lake on this date was about 50 m below the rim of the Overlook crater. The Overlook crater is contained within Halemaʻumaʻu Crater (about 1 kilometer diameter), whose far walls are marked by white alteration near the top of the photograph. (U.S. Geological Survey photograph by Matthew R. Patrick.)

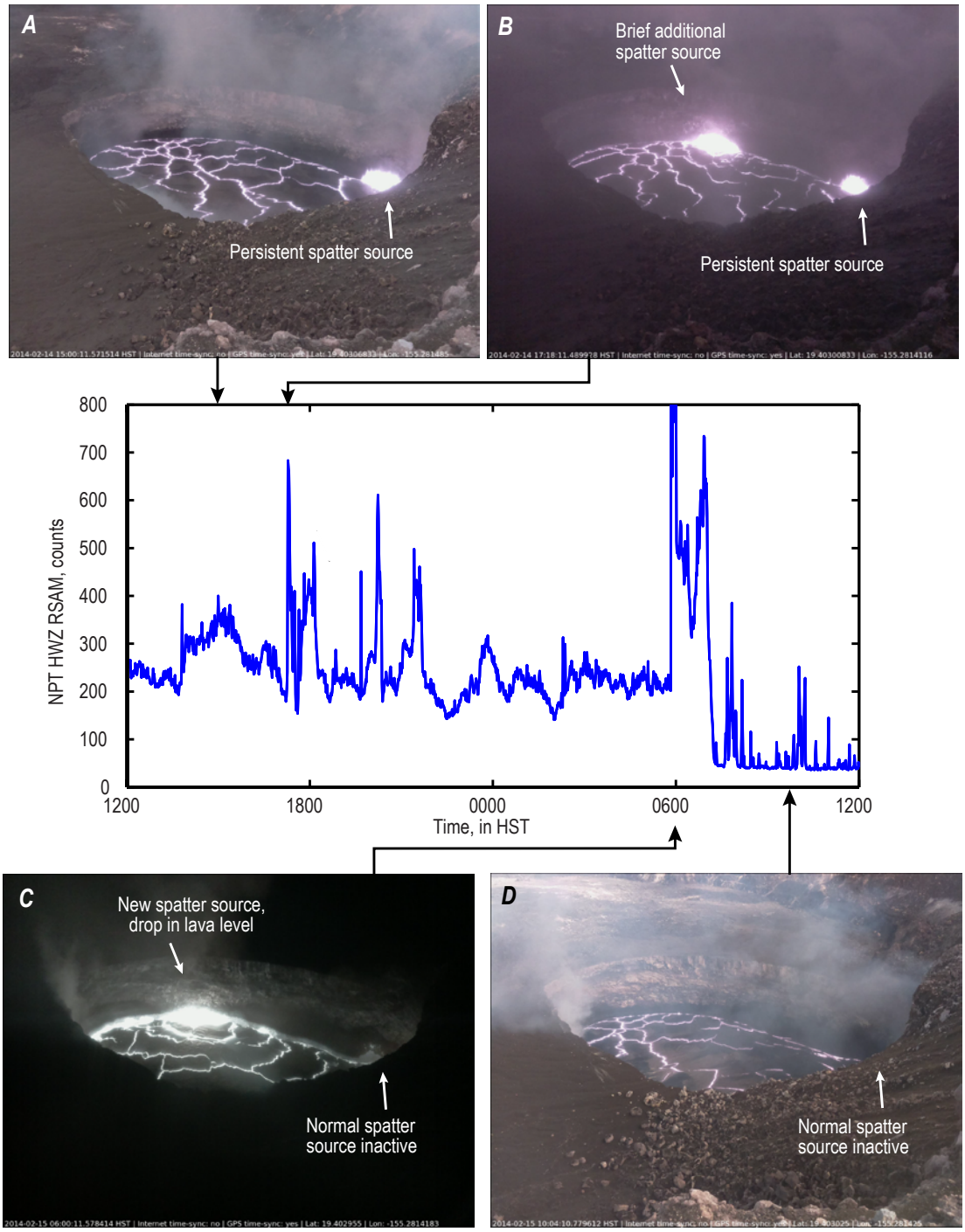

Figure 5. Photographs of the lava lake in Halema‘uma‘u Crater at the summit of Kīlauea Volcano, Hawai‘i, taken by the compact multipurpose camera system on February 14–15, 2014. The photographs are compared with a graph showing real-time seismic amplitude measurements (RSAM) recorded at station NPT (HWZ indicates the frequency band and directional component of the seismometer). Time on the graph is shown in Hawai‘i Standard Time (HST). *A*, Photograph taken during normal behavior and normal seismic tremor (moderate RSAM values); there is a single, persistent spattering source in the southeast part of the lake. *B*, Photograph taken during a rapid increase in seismic tremor, shown by a spike in RSAM, that occurred with the brief appearance of a second spattering source that was carried east in the prevailing current. *C*, Photograph taken during an apparent collapse on the northwest margin of the lava lake (left side in this view) that triggered increased spattering and a drop in lava level, presumably as gas was released from the shallow parts of the lake. The increased outgassing and spattering were associated with elevated RSAM values due to high seismic tremor. *D*, photograph showing that this change in the lake was associated with the persistent spatter source shutting down. The absence of spattering in the lake was associated with a reduction in seismic tremor, shown by the “flatline” in RSAM. (U.S. Geological Survey photographs.)

At 0551 HST in figure 5, a large spatter source appears as the RSAM spikes, and the lava level briefly drops. This is probably related to "gas piston" activity, a common process in the lava lake, which has been interpreted as the accumulation and release of gas beneath the lava lake surface (for example, Swanson and others, 1979; Patrick and others, 2010). This phase of increased activity ends with the persistent spattering source dying out, at which time the RSAM drops to unusually low levels. The correspondence of diminishing spattering with plummeting seismic tremor reinforces the observation that surface outgassing is closely related to seismic tremor. Overall, the images from these deployments demonstrate the value that continuous detailed surface observations have in interpreting geophysical data.

Discussion

A major advantage of this camera system is that it is very versatile, and can be customized to the specific needs of the user, in terms of both hardware and software. For hardware, either a standard or low-light camera module can be used, and other peripherals could be attached to the Raspberry Pi computer. For software, the Python scripts, and their scheduling with the crontab, can be modified to customize time-lapse intervals, frequency of video acquisition, and image resolution. The system allows much more innovative customization than standard camera systems normally allow. Onboard image processing could be done with Python, and internal triggering could be used. For instance, if infrequent time-lapse images begin to show some quality that might indicate increased activity at the volcano, such as elevated brightness during nighttime hours, then the image interval could automatically be changed or video sequences could be started. Internal triggering could be accomplished with motion detection as well, which would be straightforward with this setup (for example, using the Linux software package "Motion" for tracking scene changes). External triggering is another innovative option. The Raspberry Pi could be connected to other data streams (such as a seismometer) to allow external signals to trigger image or video acquisition.

The camera system could be improved in several ways. First, a compass module could be attached to allow the camera's viewing azimuth to be stamped on the image. By combining the camera viewing geometry (that is, camera location, viewing angle, and camera field of view) with a high-resolution digital elevation model (DEM), quantitative approaches could be used to locate and measure features observed in the images (for example, James and others, 2010). Photogrammetric approaches could be applied using several cameras positioned around a target. Compass modules, such as the Parallax 3-Axis HMC5883L Compass Module, have already been used with the Raspberry Pi by others (http://www.raspberrypi.org/forums/viewtopic.php?f=44&t=17107, accessed January 2015). Second, a major weakness of the Raspberry Pi for long-term field deployments is its continuous power consumption. In Hawai'i, with abundant sunlight, solar-powered setups such as those in Orr and Hoblitt (2008) and Patrick and others (2014) can easily provide adequate power. In more challenging locations, such as in monitoring the remote, active volcanoes of Alaska (Paskievitch and others, 2010; Sentmen and others, 2010), sunlight is minimal in winter months and would require more efficient power management. The Sleepy Pi (http://spellfoundry.com/products/sleepy-pi/, accessed April 2014) is a power management unit developed for the Raspberry Pi that can switch the unit on and off at intervals. In Alaska, for example, the camera system described here could be turned on for several minutes each hour to acquire an image or short video sequence, then put to sleep. This would result in a substantial reduction in power consumption and minimize the number of batteries and solar panels required. Finally, the Raspberry Pi might be used to interface with many other types of instruments (for example, gas sensors, temperature sensors, accelerometers), not just cameras. Combining the computer and instruments with a cellular modem makes a very powerful real-time field-sensor package.

Acknowledgments

We thank Loren Antolik (HVO) for advice regarding the webserver. We also thank Richard LaHusen (Cascades Volcano Observatory) and John Paskievitch (Alaska Volcano Observatory) for helpful reviews. Internet forums (many can be found through http://www.raspberrypi.org/) and blogs detailing similar applications with the Raspberry Pi were essential to the success of this project.

References

Behncke, B., Falsaperla, S., and Pecora, E., 2009, Complex magma dynamics at Mount Etna revealed by seismic, thermal, and volcanological data: Journal of Geophysical Research, v. 114, B03211, doi:10.1029/2008JB005882.

Elias, T., and Sutton, A.J., 2012, Sulfur dioxide emission rates from Kilauea Volcano, Hawai'i, 2007–2010: U.S. Geological Survey Open-File Report 2012–1107, 25 p., http://pubs.usgs.gov/of/2012/1107/.

Endo, E.T., and Murray, T., 1991, Real-time seismic amplitude measurement (RSAM)—A volcano monitoring and prediction tool: Bulletin of Volcanology, v. 53, p. 533–545.

Harris, A., Pirie, D., Horton, K., Garbeil, H., Pilger, E., Ramm, H., Hoblitt, R., Thornber, C., Ripepe, M., Marchetti, E., and Poggi, P., 2005, DUCKS—Low cost thermal monitoring units for near-vent deployment: Journal of Volcanology and Geothermal Research, v. 143, p. 335–360.

Hoblitt, R.P., Orr, T.R., Castella, F., and Cervelli, P.F., 2008, Remote-controlled pan, tilt, zoom cameras at Kīlauea and Mauna Loa volcanoes, Hawai'i: U.S. Geological Survey Scientific Investigations Report 2008–5129, 22 p., http://pubs.usgs.gov/sir/2008/5129/.

James, M.R., Pinkerton, H., and Ripepe, M., 2010, Imaging short period variations in lava flux: Bulletin of Volcanology, v. 72, p. 671–676.

Orr, T.R., and Hobblit, R.P., 2008, A versatile time-lapse camera system developed by the Hawaiian Volcano Observatory for use at Kīlauea Volcano, Hawai'i: U.S. Geological Survey Scientific Investigations Report 2008–5117, 16 p., http://pubs.usgs.gov/sir/2008/5117/.

Orr, T.R., and Rea, J.C., 2012, Time-lapse camera observations of gas piston activity at Pu`u `O`o, Kīlauea volcano, Hawai`i: Bulletin of Volcanology v. 74, p. 2353–2362, doi:10.1007/s00445-0120-0667-0.

Orr, T.R., Thelen, W.A., Patrick, M.R., Swanson, D.A, and Wilson, D.C., 2013, Explosive eruptions triggered by rockfalls at Kīlauea volcano, Hawai`i: Geology, v. 41, p. 207–210.

Paskievitch, J., Read, C., and Parker, T., 2010, Remote telemetered and time-lapse cameras at Augustine Volcano, chapter 12 *of* Power, J.A., Coombs, M.L., and Freymueller, J.T., eds., The 2006 eruption of Augustine Volcano, Alaska: U.S. Geological Survey Professional Paper 1769, p. 285–293, http://pubs.usgs.gov/pp/1769/.

Patrick, M.R., Kauahikaua, J.P., and Antolik, L., 2010a, MATLAB tools for improved characterization and quantification of volcanic incandescence in webcam imagery; applications at Kīlauea Volcano, Hawai'i: U.S. Geological Survey Techniques and Methods 13–A1, 16 p., http://pubs.usgs.gov/tm/tm13a1/.

Patrick, M.R., Orr, T.R., Wilson, D., Sutton, A.J., Elias, T., Fee, D., and Nadeau, P.A., 2010b, Evidence for gas accumulation beneath the surface crust driving cyclic rise and fall of the lava surface at Halema'uma'u, Kīlauea Volcano: American Geophysical Union, Fall Meeting 2010, San Francisco, Calif., 13–17 December, abstract V21C-2339,

Patrick, M.R., Orr, T., Wilson, D., Dow, D., and Freeman, R., 2011, Cycles of spattering, seismic tremor and surface fluctuation within a perched lava channel, Kīlauea Volcano: Bulletin of Volcanology, v. 73, p. 639–653, doi:10.1007/s00445-010-0431-2.

Patrick, M.R., Orr, T.R., Antolik, L., Lee, R., and Kamibayashi, K., 2012, Recent improvements in monitoring Hawaiian volcanoes with webcams and thermal cameras: American Geophysical Union, Fall Meeting 2010, San Francisco, Calif., 13–17 December, abstract V33E-01.

Patrick, Matthew, Orr, T.A. Sutton, Jeff, Elias, Tamar, and Swanson, D., 2013, The first five years of Kīlauea's summit eruption in Halema'uma'u Crater, 2008–2013: U.S. Geological Survey Fact Sheet 2013–3116, 4 p., http://dx.doi.org/10.3133/fs20133116.

Patrick, M.R., Orr, T.R., Antolik, L., Lee, L., and Kamibayashi, K., 2014, Continuous monitoring of Hawaiian volcanoes with thermal cameras: Journal of Applied Volcanology, v. 3, no. 1, 19 p.

Perret, F.A., 1913, The lava fountains of Kilauea: American Journal of Science, ser. 4, v. 35, art. 206, p. 139–148, doi:10.2475/ajs.s4-35.206.139.

Poland, M.P., Dzurisin, D., LaHusen, R.G., Major, J.J., Lapcewich, D., Endo, E.T., Gooding, D.J., Schilling, S.P., and Janda, C.G., 2008, Remote camera observations of lava dome growth at Mount St. Helens, Washington, October 2004 to February 2006, chapter 11 *of* Sherrod, D.R., Scott, W.E., and Stauffer, P.H., eds., A volcano rekindled—The renewed eruption of Mount St. Helens, 2004–2006: U.S. Geological Survey Professional Paper 1750, p. 225–236, http://pubs.usgs.gov/pp/1750/.

Sentman, D.D., McNutt, S.R., Stenbaek-Nielson, H.C., Tytgat, G., and DeRoin, N., 2010, Imaging observations of thermal emissions from Augustine Volcano using a small astronomical camera, chapter 24 *of* Power, J.A., Coombs, M.L., and Freymueller, J.T., eds., The 2006 eruption of Augustine Volcano, Alaska: U.S. Geological Survey Professional Paper 1769, p. 569–578, http://pubs.usgs.gov/pp/1769/.

Swanson, D.A., Duffield, W.A., Jackson, D.B., and Peterson, D.W., 1979, Chronological narrative of the 1969–71 Mauna Ulu eruption of Kilauea volcano, Hawaii: U.S. Geological Survey Professional Paper, 1056, 55 p., http://pubs.er.usgs.gov/publication/pp1056/.

Thornber, C.R., 1997, HVO/RVTS-1—A prototype remote video telemetry system for monitoring the Kilauea east rift zone eruption: U.S. Geological Survey Open-File Report 97–0537, 18 p., http://pubs.er.usgs.gov/publication/ofr97537/.

Wilson, D., Elias, T., Orr, T., Patrick, M., Sutton, A.J., and Swanson, D., 2008, Small explosion from new vent at Kīlauea's summit: Eos (Transactions of the American Geophysical Union), v. 89, p. 203.

Wolfe, E.W., Neal, C.A., Banks, N.G., and Duggan, T.J., 1988, Geologic observations and chronology of eruptive events, *in* Wolfe, E.W., ed., The Puu Oo eruption of Kilauea Volcano, Hawaii—Episodes 1 through 20, January 3, 1983, through June 8, 1984: U.S. Geological Survey Professional Paper 1463, http://pubs.er.usgs.gov/publication/pp1463/.

Appendix 1. System Setup

Set up Raspberry Pi computer

1. On a Windows machine, download and install the application Win2Disk (freeware). Also download the image file (.img) for the Raspbian wheezy Debian Linux distribution (free of charge). Write the .img file to an empty SD card (>4 Gb capacity) using Win2Disk. Detailed instructions on this step are available on the Raspberry Pi Web site (http://www.raspberrypi.org) and other sites.
2. Connect Raspberry Pi to peripherals (monitor, USB hub with keyboard and mouse plugged in) and then to power. Connect the Raspberry Pi to the network with the Ethernet cable. Put in the SD card containing the operating system. A boot sequence should show up on the monitor.
3. A setup window should appear.

   A. Choose "Expand filesystem" to make the computer recognize the full size of the SD card.
   B. Choose "Enable camera" to let the operating system work with the Raspberry Pi camera.
   C. Choose "Internationalization options" and "Change timezone" to change the timezone to your location.
   D. Reboot the computer.

4. Login with user:pi and password:raspberry
5. To enter a windows-style environment, type "startx" at command prompt.
6. Update Linux. In a terminal window, type:

```
sudo apt-get update
sudo apt-get upgrade
sudo reboot
```

7. Install Python modules.

```
sudo apt-get install python-scipy python-matplotlib
```

   Check IDLE path browser to make sure matplotlib is installed.

8. Login again, type "startx" and open a terminal window. Install gpsd service, as it appears to improve interfacing with the Ultimate GPS module.

```
sudo apt-get install gpsd
```

9. Install ImageMagick

```
sudo apt-get install imagemagick
```

10. The operating system assumes a United Kingdom keyboard layout. Change to USA keyboard layout, if necessary.

```
setxkbmap us
```

Set up webserver

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

```
sudo apt-get install apache2
sudo service apache2 restart
```

2. Change Apache log directory. Edit the configuration file:

```
sudo nano /etc/apache2/envvars
```

About halfway down this file there is a line that says this:

```
export APACHE_LOG_DIR=/var/log/apache2$SUFFIX
```

Change this line to:

```
export APACHE_LOG_DIR=/var/log
```

Set up camera

1. Shutdown the computer.

```
sudo shutdown now
```

2. Plug the camera ribbon into the CSI (Camera Serial Interface) connector on the Raspberry Pi computer. Demonstrations of this are shown on the Raspberry Pi Web site (http://www.raspberrypi.org/camera; last accessed April 2014).
3. Restart the computer and log back in.
4. 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
```

Set up GPS unit and time-syncing

1. Shutdown the computer.

```
sudo shutdown now
```

2. Solder the pins to the Ultimate GPS module.
3. Insert a CR1220 coin battery in the GPS module.
4. Connect the Ultimate GPS module to Raspberry Pi using the USB to TTL cable.
5. Restart the computer and log back in.
6. Confirm the Ultimate GPS is working. Put the GPS unit in a window with a clear view of the sky. While the unit is searching for satellites it will blink a red LED at 1 Hz. Once a lock is established, the unit will blink once every 15 seconds. To confirm that GPS data are coming in type the following:

```
sudo cat /dev/ttyUSB0
```

A stream of data should appear on the screen; hit control-c to stop the stream. Some of the sentences should begin with"$GPRMC", which has the following format (http://aprs.gids.nl/nmea/, accessed January 2015):

```
$GPRMC,182423.000,A,1936.8246,N,15512.7806,W,173.8,231.8,021013,004.2,W*70
```

Where:

```
1   182423.000 Time Stamp
2   A          validity - A-ok, V-invalid
3   1936.8246  current Latitude (19° 36.8246')
4   N          North/South
5   15512.7806 current Longitude (155° 12.7806')
6   W          East/West
7   173.8      Speed in knots
8   231.8      True course
9   021013     Date Stamp
10  004.2      Variation
11  W          East/West
12  *70        checksum
```

The second variable shows whether the GPS has a lock (A) or not (V).

7. Modify NTP timeserver to read GPS data. The NTP configuration file (ntp.conf, in the folder /etc) controls where the NTP service looks for an accurate time server, which in our case will either be a) an internet time-server when the camera is connected to the network or b) the Ultimate GPS module when the camera is not connected to the outside world. Note that other hobbyists using a Raspberry Pi computer with the Ultimate GPS receiver have installed a module called gpsd, and modified the NTP configuration file to interface with gpsd. However, we found that gpsd was unreliable, and so we put in the address of the GPS unit directly. To edit the file, type:

```
sudo nano /etc/ntp.conf
```

To the end of the NTP configuration file we added these lines:

```
server 127.127.20.0 mode 17 minpoll 3 iburst true prefer
fudge 127.127.20.0 flag 1 time2 0.496
```

After this change is made, restart the NTP service:

```
sudo service ntp restart
```

8. Test NTP to make sure it is getting GPS time. Disconnect the Ethernet cable so that the Raspberry Pi is no longer connected to the network. Unplug the power and wait an hour or so. Reconnect the power so that the camera boots up and go to the X windows environment. The time will likely be off by an hour or more. Watch the Ultimate GPS module and identify when it gets a GPS lock (LED going from 1 Hz to 1/15 Hz). This can also be checked by looking at the serial output as described above. Once the GPS has a lock, check the time on the computer screen to confirm that it has been corrected.

Set up image acquisition scripts and scheduling

1. Copy over Python scripts for image acquisition and management. Scripts are provided in the other appendices. Copy these scripts to the /home/pi directory.

2. Set crontab (type "crontab –e") to run the selected Python scripts on a schedule.

Script 1: Fast time-lapse (several images per minute)	Every 10 minutes
Script 2: Slow time-lapse (image every few minutes)	Every few minutes
Script 3: Archive fast time-lapse images	Every minute
Script 5: Force NTP time sync	Every couple hours
Script 7: Run video script	Every couple minutes, hours
Script 8: Free disk space check	Every day

We also add a cronjob in which the Raspberry Pi computer reboots once a day, a few minutes after midnight. For the fast time-lapse script the complete crontab then looks like this, with the script acquiring several images per minute,

continuously for 10 minutes, and then restarting:

```
*/10 * * * * /usr/bin/python2.7 fasttimelapse.py
*/1 * * * * /usr/bin/python2.7 putfiles.py
0 0 * * * /usr/bin/python2.7 freespace.py
11 0 * * * sudo shutdown -rF now
1 */2 * * * /usr/bin/python2.7 ntpset.py
5 */2 * * * sudo service ntp start
@reboot sleep 400 && /usr/bin/python2.7 ntpset.py
@reboot sleep 300 && sudo mount /dev/sda1 -o remount,rw
@reboot sudo mount /dev/sda1 -o remount,rw
*/2 * * * * sudo mount /dev/sda1 -o remount,rw
*/2 * * * * sudo chown -R pi /var/www
```

For the slow time-lapse script, the crontab looks like this, in which the crontab frequency determines the frequency of image acquisition (one image every 2 minutes in this example):

```
*/2 * * * * /usr/bin/python2.7 slowtimelapse.py
0 0 * * * /usr/bin/python2.7 freespace.py
11 0 * * * sudo shutdown -rF now
1 */2 * * * /usr/bin/python2.7 ntpset.py
5 */2 * * * sudo service ntp start
@reboot sleep 400 && /usr/bin/python2.7 ntpset.py
@reboot sleep 300 && sudo mount /dev/sda1 -o remount,rw
@reboot sudo mount /dev/sda1 -o remount,rw
*/2 * * * * sudo mount /dev/sda1 -o remount,rw
*/2 * * * * sudo chown -R pi /var/www
```

For periodic video acquisition, the crontab would look like this, in which the video acquisition lasts 30 seconds and is run every 10 minutes:

```
*/10 * * * * /usr/bin/python2.7 getvideocmd.py -i 30
0 0 * * * /usr/bin/python2.7 freespace.py
11 0 * * * sudo shutdown -rF now
1 */2 * * * /usr/bin/python2.7 ntpset.py
5 */2 * * * sudo service ntp start
@reboot sleep 400 && /usr/bin/python2.7 ntpset.py
@reboot sleep 300 && sudo mount /dev/sda1 -o remount,rw
@reboot sudo mount /dev/sda1 -o remount,rw
*/2 * * * * sudo mount /dev/sda1 -o remount,rw
*/2 * * * * sudo chown -R pi /var/www
```

Setup SD card as read-only to avoid SD card corruption

1. Change the fstab file to set the root directory as read-only.

   ```
   sudo nano /etc/fstab
   ```

   Edit the file so that it looks like this:

```
proc            /proc           proc    defaults          0       0
/dev/mmcblk0p1  /boot           vfat    ro                0       2
/dev/mmcblk0p2  /               ext4    ro                0       1
/dev/sda1       /media/usb      vfat    rw,uid=1000,gid=1000       0       2
tmpfs           /tmp            tmpfs   defaults,noatime,nosuid,size=100m    0   0
tmpfs           /var/lib/lightdm/       tmpfs             defaults,noatime,nosuid,mode=0755,size=30m   0    0
tmpfs           /var/log/       tmpfs   defaults,noatime,nosuid,mode=0755,size=100m      0     0
tmpfs           /var/run/       tmpfs   defaults,noatime,nosuid,mode=0755,size=2m   0   0
tmpfs           /var/spool/mqueue     tmpfs   defaults,noatime,nosuid,mode=0700,gid=12,size=30m   0  0
tmpfs           /var/www/       tmpfs   rw,noatime,user,nosuid,mode=0755,size=100m   0   0
# a swapfile is not a swap partition, so no using swapon|off from here on, use  dphys-swapfile swap[on|off]  for that
```

2. If further changes to the scripts or operating system are necessary, write permission must be reset:

   ```
   sudo mount / -o remount,rw
   ```

   The fstab will then reset the root directory to read-only upon the next reboot.

Appendix 2. Fast Time-Lapse (Script 1: fasttimelapse.py)

In the Python code below, green text is a comment, bold blue text is control flow, magenta text is a string:

```
#Script 1.This acquires a high rate timelapse sequence (currently 10 images per minute). The acquisition
#runs for 10 minutes, and should thus be scheduled by cron to restart every 10 minutes.  The script
#"putfiles.py" should also be scheduled by cron, preferably once per minute, to move the incoming image
#files to their date directories.

# Matt Patrick
# US Geological Survey - Hawaiian Volcano Observatory
# Mar 20, 2014

import os

#Acquire image and name file based on date-time
os.chdir('/media/usb/webcam/')

os.system('kill $(pgrep raspistill)')

#Raspistill in this setting acquires a timelapse burst of images (ie several images per minute) -
# currently set at 10 images per minute
#image size = 1024 x 768 (about 400 kb)
#runs for 570 seconds (a little under 10 minutes, because internal timer appears to run 4.5% longer
#than stated
#acquires an image every 10 seconds
os.system('raspistill -o timelapse%04d.jpg -q 90 -w 1024 -h 768 -t 580000 -tl 10000 -vf -hf -ex auto -awb auto -n');
#acquire image
path='/media/usb/webcam/timelapse*jpg'
```

Appendix 3. Slow Time-Lapse (Script 2: slowtimelapse.py)

In the Python code below, green text is a comment, bold blue text is control flow, magenta text is a string:

```
#Script 2. Call this script in a crontab to acquire an image every few minutes (cron can do no faster
#than one image per minute in this fashion.  To acquire several images per minute, use
#fasttimelapse.py as well as putfiles.py together in the crontab.

# Matt Patrick
# US Geological Survey - Hawaiian Volcano Observatory
# Mar 20, 2014

import os
#import pylab as plt
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()


#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):

   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)

    #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(ti.tzname[1])
    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
```

Appendix 4. Archive Fast Time-Lapse Images (Script 3: putfiles.py)

In the Python code below, green text is a comment, bold blue text is control flow, magenta text is a string:

```
#Script 3 takes incoming images and 1) puts a timestamp on them and 2) 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.

# Matt Patrick
# US Geological Survey - Hawaiian Volcano Observatory
# Mar 20, 2014


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(ti.tzname[1])
    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
```

Appendix 5. Get GPS Position (Script 4: gpspuller3.py)

In the Python code below, green text is a comment, bold blue text is control flow, magenta text is a string:

```
#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.

# Matt Patrick
# US Geological Survey - Hawaiian Volcano Observatory
# Mar 20, 2014


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')

    #Acquire image and name file based on date-time
    os.chdir('/media/usb/webcam')
    s='head --lines=30 /dev/'+a[0]+' > 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)
```

Appendix 6. NTP Time Set (Script 5: ntpset.py) and Approximate Time Set (Script 6: setapproximate2.py)

In the Python code below, green text is a comment, bold blue text is control flow, magenta text is a string:

```
#Script5: ntpset.py
#this function tries to ensure the NTP server has the correct time

#Matt Patrick
#US Geological Survey-Hawaiian Volcano Observatory
#Mar 20, 2014

import os
from setapproxtime2 import setapproxtime

setapproxtime()
os.system('sudo service ntp stop')
os.system('sudo ntpd -q')
os.system('sudo service ntp start')
```

```
#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 correction

# Matt Patrick
# US Geological Survey - Hawaiian Volcano Observatory
# Mar 20, 2014


import os
import time
import datetime as dt
from gpspuller3 import gpspull
from pytz import timezone

def setapproxtime():
    [gpsfix,lat,lon,gpstime]=gpspull()

    t=gpstime.astimezone(timezone('HST'))

    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)
```

Appendix 7. Acquire Video (Script 7: getvideocmd.py)

In the Python code below, green text is a comment, bold blue text is control flow, magenta text is a string:

```
#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.

# Matt Patrick
# US Geological Survey - Hawaiian Volcano Observatory
# Mar 20, 2014

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)
```

Appendix 8. Maintain Free Space on Flash Drive (Script 8: freespace.py)

In the Python code below, green text is a comment, bold blue text is control flow, magenta text is a string:

```
#Script 8: freespace.py
#function checks space on flash drive and deletes directories as necessary
#to maintain sufficient free space to keep acquiring images

# Matt Patrick
# US Geological Survey- Hawaiian Volcano Observatory
# Mar 20, 2014

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)
```

Page intentionally left blank.

Menlo Park Publishing Service Center, California
Manuscript approved for publication February 18, 2015
Edited by James W. Hendley II
Layout and design by Cory Hurd

ISSN 2328-7055 (online)
http://dx.doi.org/10.3133/tm13A2