HardwareX 8 (2020) e00160

Contents lists available at ScienceDirect

HardwareX

journal homepage: www.elsevier.com/locate/ohx

Hardware Article

Digital video recorder for Raspberry PI cameras with multi-camera synchronous acquisition

Ghadi Salem [a,*], Jonathan Krynitsky [a], Noah Cubert [a], Alex Pu [b], Simeon Anfinrud [a], Jonathan Pedersen [a], Joshua Lehman [a], Ajith Kanuri [a], Thomas Pohida [a]

[a] Signal Processing and Instrumentation Section, Office of Intramural Research, Center for Information Technology, National Institutes of Health, USA
[b] Division of Veterinary Services, Center for Biologics Evaluation and Research, U. S. Food and Drug Administration, USA

ARTICLE INFO

Article history:
Received 10 January 2020
Received in revised form 4 November 2020
Accepted 21 November 2020

Keywords:
Digital Video Recorder (DVR)
Raspberry PI camera DVR
Multi-camera synchronized acquisition

ABSTRACT

Video acquisition and analysis have become integral parts of scientific research. Two major components of a video acquisition system are the choice of camera and the acquisition software. A vast variety of cameras are available on the market. Turnkey multi-camera synchronous acquisition software, however, is not as widely available. For prototyping applications, the Raspberry Pi (RPi) has been widely utilized due to many factors, including cost. There are implementations for video acquisition and preview from a single RPi camera, including one implementation released by the RPi organization itself. However, there are no multi-camera acquisition solutions for the RPi. This paper presents an open-source digital video recorder (DVR) system for the popular RPi camera. The DVR is simple to setup and use for acquisition with a single camera or multiple cameras. In the case of multiple cameras, the acquisition is synchronized between cameras. The DVR comes with a graphical user interface (GUI) to allow previewing the camera streams, setting recording parameters, and associating "names" to cameras. The acquisition code as well as the DVR GUI are written in Python. The open-source software also includes a GUI for playback of recorded video. The versatility of the DVR is demonstrated with a life science research application involving high-throughput monitoring of fruit-flies.

Specifications table

Hardware name	*Raspberry Pi Multi-Camera Synchronous DVR*
Subject area	• Neuroscience • Biological Sciences (e.g. Microbiology and Biochemistry) • General
Hardware type	• Imaging tools
Open Source License	*BSD-2*
Cost of Hardware	*$2073.52 for 24 camera system*
Source File Repository	https://doi.org/10.17605/OSF.IO/RUD4A

* Corresponding author.
E-mail address: ghadi.salem@nih.gov (G. Salem).

https://doi.org/10.1016/j.ohx.2020.e00160
2468-0672/Published by Elsevier Ltd.

1. Hardware in context

Recent years have witnessed a dramatic increase in utilization of video technology. The increase is attributed to two major factors: the availability of low-cost cameras and the advances in automated video analysis methods. Aside from consumer applications, video systems have enabled novel data collection in various research disciplines [1]. In life sciences research, for example, video has been used to automatically profile research animals' behavior [2,3]. Some systems utilize several synchronous cameras for more comprehensive, detailed and accurate analysis [4,5].

Assembling a digital video acquisition system entails choosing the camera(s) and a recording software (i.e., digital video recorder). Camera choices are abundant. Many well-known companies (e.g., Allied Vision, Point Grey Research, Sony, Basler) offer a wide selection of digital cameras with different resolution, frame rate, bit depth among other specifications. One critical criterion is the camera cost. Low cost is one reason the RPi camera has been widely used, especially in prototyping applications [6–8]. Along with low-cost, other factors accounting for the popularity of the RPi in prototyping applications include: (1) RPi has a wide user-base for hobbyists and prototyping applications with excellent on-line support forums, (2) The base RPi board has compute power that can be harnessed for custom video analysis code, (3) The base board has general purpose I/O that are often needed to interface with external signals and systems, and (4) The remote-head camera board can be used with an array of ribbon cable lengths allowing for integration into various mechanical setups.

When deciding on a DVR, the choices are more limited, especially for multi-camera systems. Commercial digital video recording (DVR) systems [9,10] allow for various recording configurations and support many standard interface cameras (e.g., USB3 and GigE). For the RPi, there are many implementations for single camera acquisition [6–8,11,12], including a command line function for preview and acquisition released by RPi foundation itself. To our knowledge, however, there is not a multi-camera DVR for the RPi.

We present a light-weight, open-source, and versatile DVR for the RPi camera. The DVR software offers a platform for testing, developing, and running prototype video acquisition applications. The DVR supports single- and multi-camera acquisition for long durations (e.g., days). For multi-camera setups, frame acquisition (i.e., sampling) is synchronous across cameras. A Python-based graphical user interface (GUI) front-end offers many user-friendly features, including: (1) video preview, (2) adjustable recording parameters (e.g., start and end times of recording, clip length), (3) grouping and bundling of several cameras as one operational unit, (4) assignment of user-defined names to cameras, (5) definition of experiment parameters (e.g., *meta*-data regarding subjects recorded) for each camera or camera bundle, and (6) adjustable camera settings (e.g., recording resolution, frame-rate, gain-settings).

The DVR and associated GUI operate on a local network with a router that is not connected to external traffic. This method of operation lacks the convenience of remote network accessibility. However, it does ensure bandwidth for full frame rate acquisition from multiple cameras without disruption from external network traffic. Furthermore, it eliminates any remote security breach concerns for scientific studies.

One user-controlled parameter of the DVR is the time duration of video recording segments. Rather than recording a long-duration video to a single large file, the recording is split into smaller video files of a user-defined length (e.g., 2 min). One advantage of such a scheme is to reinforce synchronicity in multi-camera recording experiments. To enable seamless playback of a full experiment, a Python-based playback GUI was written to allow the user to select an entire experiment to view, rather than selecting individual files. The playback GUI gives the user typical playback functionality (e.g., reverse, forward, skip, etc) for the full recording length while relieving the user from manually navigating the video storage file structure. Additionally, the playback GUI allows the user to clip and save video segments of interest at prescribed start/end times (i.e., not aligned with the original video recording segments).

The DVR functionality is validated to demonstrate the system's full frame rate acquisition, synchronicity, and low latency for previewing. As an example use-case, a reference is made to a novel vision system for monitoring of Drosophila in high-throughput experiments [13]. The system employs the DVR to acquire video from 24 RPi cameras. The cameras were used to record 96 flies for a 24 h duration. This application is referenced as an example to demonstrate the scalability of the system and the adaptability of its components to new research venues that can be explored using the DVR.

The manuscript is organized as follows. Section 2 describes the back-end components of the DVR software for both the host PC and RPi that handle video acquisition and network communication. Section 3 lists and describes the individual files that comprise the DVR software along with a graphical representation of the software components. Section 4 lists the materials needed to reproduce a 24-camera DVR. Detailed software installation instructions for the host PC as well as the RPi are given in Section 5. The GUI operating instructions are detailed in Section 6. Section 7 presents the validation work carried out to ensure that the DVR is acquiring at full frame rate and to quantify the latency.

2. Hardware description

An example hardware setup is shown in Fig. 1. The RPi cameras, which are shown to capture video of a time counter in the example, are connected to RPis. The RPis are connected to a router, which in turn is connected to the acquisition PC. The PC runs the DVR software. The DVR software is written in Python and integrates a network of RPi cameras. The DVR software is designed to be intuitive and open-source. Existing commercial DVRs have more refined interfaces with many convenience features. Commercial DVRs also support a wide range of cameras. These commercial DVRs, however, are proprietary and

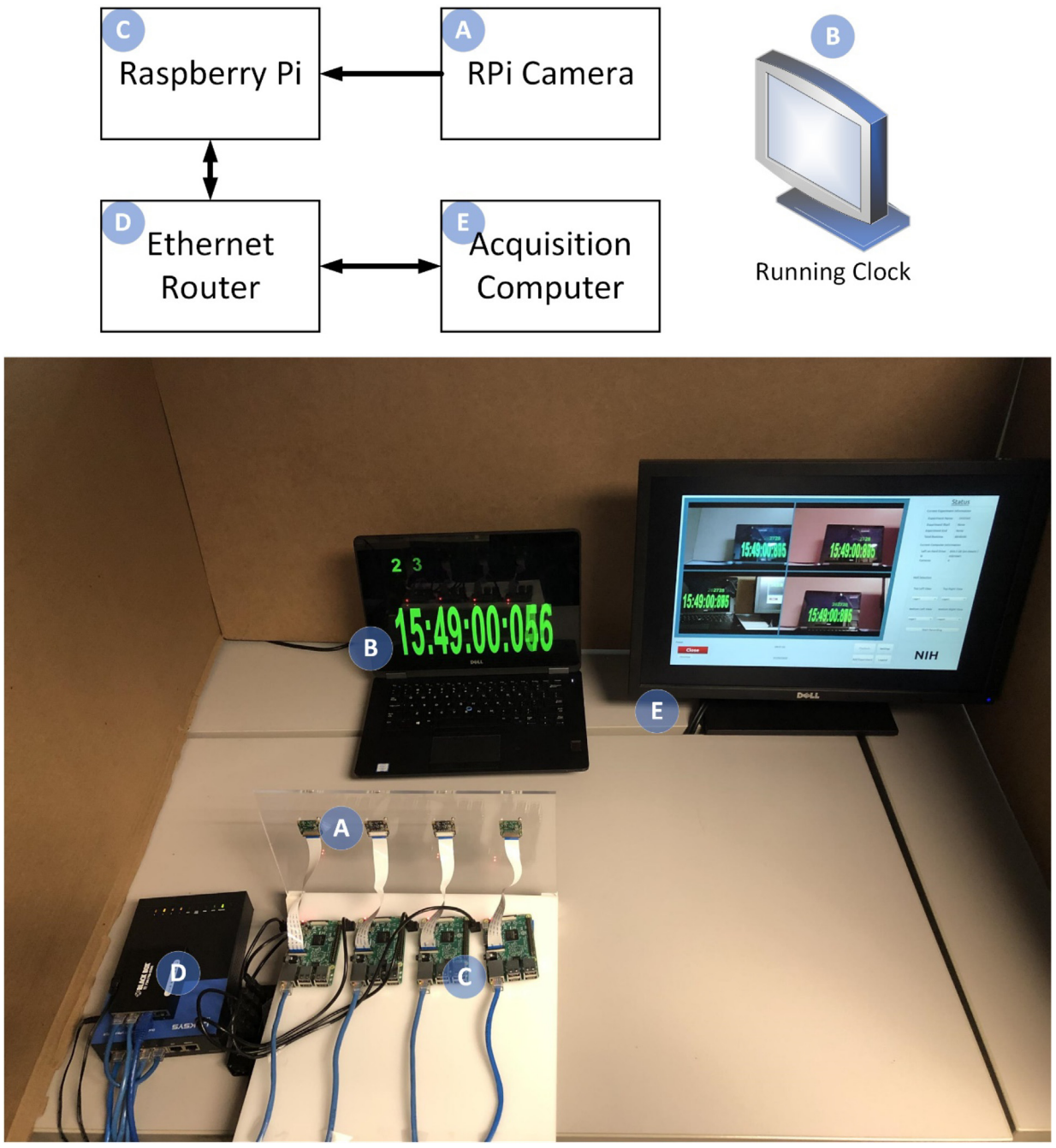

Fig. 1. Top: A diagram highlighting the components of the system setup used in testing. Bottom: An image of the actual experimental setup. Four RPi cameras (A) are pointed to a PC screen (B) displaying a running clock. The cameras are connected to the RPi boards (C), which in turn are connected to a router (D). The router is connected to an acquisition PC running the DVR software (E). The alphabetical letter designators in the top diagram refer to the same system components as outlined in the experimental setup image.

costly. The supported cameras also tend to be expensive. For example, many supported USB3 and GigE cameras exceed $350. For prototyping applications and scientific research experiments, the high cost of a feature-rich DVR (albeit with wide support for costly cameras) could be more limiting than a free, although basic, DVR with support for a single line of inexpensive, although capable, cameras. The DVR presented in this work will, as is, cover most use cases, especially in scientific research experiments. Additionally, users are free to customize or add functionality for more specialized applications because the DVR software is open-source.

The following sections describe an overview of the DVR software components. This overview, along with the detailed instructions in sections 5 and 6, suffice for a user to fully utilize the software. To aid users' customizing the software, the code has been thoroughly commented.

2.1. The open-source DVR software

- Offers a Python based platform for scientific experiments requiring short-term or long-term video acquisition.
- Enables single and multi-camera acquisition, with the capability of synchronized acquisition for multi-camera setups.
- Includes a GUI with many user-friendly features for setting up and launching recording experiments, modifying camera and recording parameters.
- Allows for easy customization through a fully Python-based implementation and open-source distribution.

2.2. Software description

The Python-based video acquisition software operates on a master/slave network structure. Each RPi camera is connected to a RPi base board which runs what hereinafter is referred to as the client software. The DVR software runs on a Host PC and is connected to all the RPi base boards over a wired local network. The DVR software has the role of master and actively controls the slave RPis with commands to initiate recording and streaming, change camera properties, and change video format. When prompted, each RPi transmits video over the local network to the Host PC where it is saved to an internal or external hard drive.

2.3. Master/Slave network structure

The DVR software, as the master, is the driving component in acquiring video from each slave RPi module. The RPi base boards act as slaves and control their attached camera based on commands sent by the DVR software. Once the RPis receive power and complete an initial boot sequence, they await a UDP packet on port 8890 with the command to initiate a connection with the DVR software. The DVR software will continually broadcast UDP packets with the Host PC IP address and TCP interaction port number until connections are established. Once a connection packet is received, the RPis initiate a connection with the Host PC and wait for commands. The commands include start recording, split recording, stop recording, start previewing, stop previewing, and reboot. When given a command that involves streaming, such as recording or previewing, the RPis initiate a new connection with the Host PC to stream the data. The connection parameters are provided by the DVR software within the start recording command message.

2.4. User interface

The user interface, shown in Section 6, was developed with PyQt5 and has been tested on Windows 7, Windows 10, and Ubuntu PCs. The user interface consists of video preview window, computer status indicators, and control buttons. The video preview window is split into four separate sub-windows, allowing the user to preview up to 4 video streams at once. To the top right of the preview window is experiment information such as experiment name, start and end time, and duration. Below that window is a section displaying computer information such as Host PC IP address, free memory space, and number of connected cameras. Finally, below the preview window are buttons that the user can use to initiate experiments, alter camera settings, and assign names to cameras.

3. Design files

Design Files Summary

Design file name	File type	Open source license	Location of the file
launcher.py	Python File	BSD-2	OSF Repository
runacquistion.bat	Bat File	BSD-2	OSF Repository
settings.json	Json File	BSD-2	OSF Repository
utils.py	Python File	BSD-2	OSF Repository
init.py	Python File	BSD-2	OSF Repository
main.py	Python File	BSD-2	OSF Repository
protocol.py	Python File	BSD-2	OSF Repository
screen.py	Python File	BSD-2	OSF Repository
bundle.py	Python File	BSD-2	OSF Repository
cam_set.py	Python File	BSD-2	OSF Repository

a (*continued*)

Design file name	File type	Open source license	Location of the file
exp_updater.py	Python File	BSD-2	OSF Repository
gplayer.py	Python File	BSD-2	OSF Repository
updater.py	Python File	BSD-2	OSF Repository
acquisition_window.py	Python File	BSD-2	OSF Repository
add_experiment.py	Python File	BSD-2	OSF Repository
bundle_gui.py	Python File	BSD-2	OSF Repository
cam_set_gui.py	Python File	BSD-2	OSF Repository
camera_labler.py	Python File	BSD-2	OSF Repository
cropping_gui.py	Python File	BSD-2	OSF Repository
legend_gui.py	Python File	BSD-2	OSF Repository
server.py	Python File	BSD-2	OSF Repository
server_protocols.py	Python File	BSD-2	OSF Repository
client.py	Python File	BSD-2	OSF Repository

Launcher.py: This file provides the logic for the front-end user facing code. This file exists to link the frontend GUI code to the backend DVR code.

runacquistion.bat: The purpose of this file is to provide a simple means of running the software by double clicking this file instead of launching from the command line.

settings.json: Contains camera settings such as FPS and zoom that can be altered by the user in the GUI.

utils.py: A collection of miscellaneous functions used by the software. These functions are isolated in this separate file to enhance code readability.

init.py: Initializes the program and checks if the program is being run on the correct platform.

main.py: Utility file to read command line arguments.

protocol.py: Defines communication messages between the DVR software and the Raspberry Pi cameras. Messages are sent through network sockets and can pass several types of data, including integers, double floating-point numbers, and strings.

screen.py: This module contains the user interface for the optional Raspberry Pi screen.

bundle.py: Contains the definition of a GUI and object that helps organize client cameras into groups.

cam_set.py: Defines functions that enable the user to modify the DVR software settings.

exp_updater.py: Defines functions, data structure, a GUI that enable the user to define and store experiments.

gplayer.py: Initializes an object that generates streaming preview camera stream on a GUI.

updater.py: This module updates all the information of the GUI such as date/time, cameras, IP addresses, and remaining free space on the computer.

acquisition_window.py: Contains the GUI elements and layout for the acquisition screen.

add_experiment.py: Contains the GUI elements and layout for the add experiment screen.

bundle_gui.py: Contains the GUI elements and layout for the bundler window.

cam_set_gui.py: Contains the GUI elements and layout for the camera settings window.

camera_labler.py: Contains the GUI elements and layout for the camera labeler window.

cropping_gui.py: Module defining the GUI window for setting the cropping area for each camera.

legend_gui.py: Contains the GUI definition and layout for the legend.

server.py: Contains DVR software function definitions. This file can be run manually as a standalone application through the command line or automatically through the Launcher.

client.py: Contains Raspberry Pi software function definitions. This file is set up to automatically run when the Raspberry Pi boots. It handles video acquisition from the Raspberry Pi camera and transmission of video to the DVR PC.

server_protocols.py: Defines the protocol functions for the DVR software. It defines MessageHandler functions (for incoming messages) and MessageBuilder functions (for outgoing messages). Each message ID is associated with a function that responds to or constructs messages, respectively.

In addition to the higher level software operation description given in Section 2, and the documentation comments within each python code file, Fig. 2 offers a graphical model of the software. The code files in the diagram are grouped by function. The diagram shows the hardware component on which each file resides. The interdependencies between the files are also shown.

4. Bill of materials

Bill of Materials

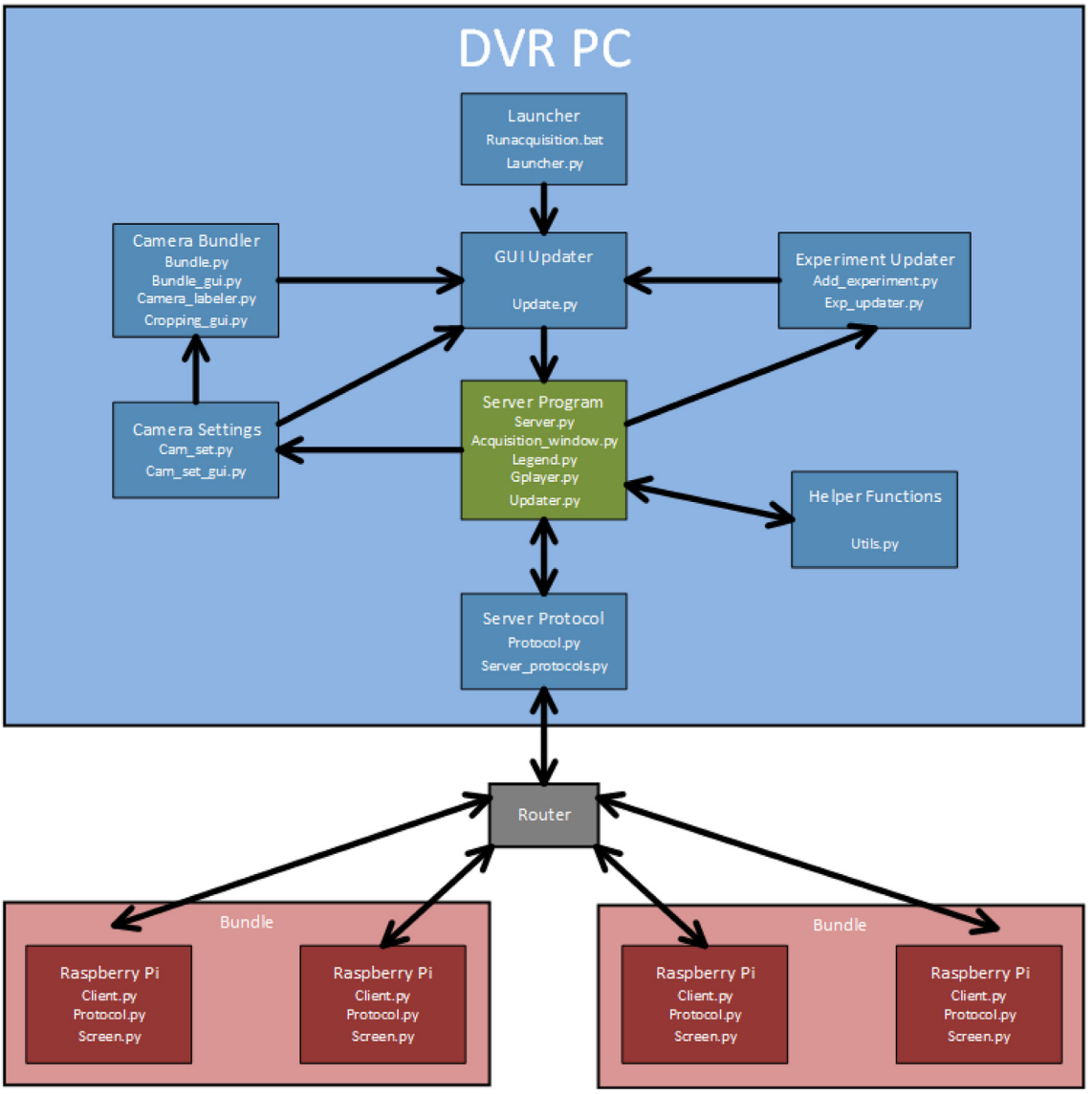

Fig. 2. The DVR software model showing the grouping of code files in terms of higher level function. The diagram shows the code-file to hardware device pairing. In this example diagram, the box labelled DVR PC refers to the computer to which the video is streamed and stored. The example assumes four RPis each two are bundled as one unit (e.g., two cameras per cage). The diagram further shows the interactions between the different code-files and software functions.

Designator	Component	Count	Cost per unit – USD	Total cost – USD	Source of materials
R1	Raspberry Pi 3 Model B	24	37.55	901.20	https://www.newark.com/raspberry-pi/raspberrypi3-modb-1gb/sbc-raspberry-pi-3-mod-b-1gb-ram/dp/77Y6520?ost = 77Y6520&ddkey = https%3Aen-US%2FElement14_US%2Fsearch
C1	Raspberry Pi Camera Board, Version 2	24	23.75	570	https://www.newark.com/raspberry-pi/rpi-8mp-camera-board/camera-board-8-mp-raspberry-pi/dp/77Y6521?ost = 77Y6521&ddkey = https%3Aen-US%2FElement14_US%2Fsearch

a (*continued*)

Designator	Component	Count	Cost per unit – USD	Total cost – USD	Source of materials
E1	Gigabit Ethernet Router	1	199.99	199.99	https://www.cdwg.com/product/Linksys-LRT224-router-desktop/3192333?
E2	Gigabit Ethernet 5-Port switch	1	76.18	76.18	https://www.newark.com/netgear/gs105na/gs105-5port-copper-gigabit/dp/15P9155?ost = GS105NA&ddkey = https%3Aen-US%2FElement14_US%2Fsearch
E3	Gigabit Ethernet 24-port switch	1	79.99	79.99	https://www.amazon.com/TRENDnet-24-Port-GREENnet-Switching-TEG-S24D/dp/B013OP3I4M
E4	Cat 6 Ethernet cables	26	5.01	130.26	https://www.newark.com/amphenol-cables-on-demand/mp-64rj45unnb-006/patch-cable-rj45-plug-6-blue/dp/77Y9670?st = ethernet
P1	6-Port USB Wall Charger	4	28.99	115.96	https://www.anker.com/products/variant/powerport-6/A2123123

The above list reflects the materials needed to build our example use case system (Section 7) which utilizes 24 cameras. Note that items R1 and C1 match the required camera count. Item E4 is also equal to the camera count plus, in the case of a 24 camera system, two additional cables to connect the router to the host PC and connect the switches to the router. Other items, such as E1, are system wide requirements and are therefore needed regardless of camera count. The rest of the items also scale based on camera count, but not in a one-to-one correspondence. The cost for a lower camera count would factor in these dependencies. As an extreme case, a one-camera system would require count 1 of R1, C1, E1, and P1 as well as count 2 of E4 (one from camera to router, and one from router to PC). Hence the cost of a one camera system would be $300.30.

5. Build instructions

5.1. Software

Setting Up the Raspberry Pi Operating System

1) Go to (https://www.raspberrypi.org/downloads/) to download and install the Raspberry Pi Imager for Ubuntu or Windows depending on what operating system you are using.
2) Open the Raspberry Pi Imager program and connect a microSD card to your computer.
3) Select the correct Raspbian OS (the default/recommended version) and microSD card from the respective menus.
4) Click the "Write" button to write the Raspbian OS image to the microSD card.
5) After the install is finished, remove the microSD card from the PC and insert it into a RPi.

Setting up Software on Raspberry Pi's

1) Connect the RPi to a keyboard, mouse, HDMI compatible screen, and USB power source.
2) When the RPi first boots up, select the preferred time locale and keyboard layout and click "Next".
3) If desired, set up a password and/or unique username. Click "Next" when done.
4) Connect the RPi to a WIFI network and click "Next" when done.
5) Update the built-in software as necessary and click "Next" when done.
6) Reboot the RPi after any updates before continuing.
7) Open the Raspberry Pi Configuration window by clicking on raspberry symbol on the top left followed by "Preferences" and "Raspberry Pi Configuration".
8) In the Raspberry Pi Configuration window, navigate to "Interfaces" and enable both "Camera" and "SSH".
9) Reboot the RPi.
10) Open a terminal (CTRL + Alt + T) and type the following commands to install the software dependencies:
11) sudo apt-get install -f gstreamer1.0
12) sudo apt-get install -f gstreamer1.0-tools
13) sudo apt-get install python3-pyqt5
14) sudo apt install python3.4 (**Whenever you are prompted with the "Do you wish to continue [Y/N]" just type in "Y" and hit "Enter." This prompt will reoccur frequently throughout the setup process.**)
15) Disable the WIFI by clicking on the WIFI symbol in the top right corner of the screen followed by "Turn off WIFI".
16) Disable Bluetooth by clicking the Bluetooth symbol in the top right corner of the screen followed by "Turn off Bluetooth".

17) Enter the following command in a terminal window:
18) crontab – e (**Type in 1 if prompted with some text and hit "Enter."**)
19) Scroll to the bottom of the file (using the arrow keys only) and type the following all on one line: @reboot sleep 10; export XAUTHORITY=/home/pi/.Xauthority; export DISPLAY=:0.0; python3.4 /home/pi/scripts/ (**To exit, hit CTRL + X then "Y" then "Enter". This will be used frequently when exiting a document utilizing the nano text editor.**)
20) Return to the terminal and enter the following command:
21) sudo nano /etc/lightdm/lightdm.conf
22) Scroll using the arrows until you find the text "[Seat:*]" in the document that pops up.
23) Underneath this text type:
xserver-command = X – s 0 dpms (**To exit, hit CTRL + X then "Y" then "Enter".**)
24) Return to the terminal and enter the following command to open the config document:
25) sudo nano /boot/config.txt
26) Scroll to the line with the text "#hdmi_force_hotplug = 1" and delete the "#" symbol. This step is only recommended if multiple types of displays (old and new) will be used with this specific RPi. However, SKIP this step if only one display type will be used to interface with the RPi. Also SKIP this step if SSH is used to control the RPi from a PC.
27) In the same config text document, scroll to the bottom and type in the following two lines to disable the Bluetooth and WIFI modules: (**SKIP this step if WIFI or Bluetooth will still be needed**)
dtoverlay = pi3-disable-bt
dtoverlay = pi3-disable-wifi
28) Save and exit the config document. (**This will cause the RPi to not be connected to a WIFI or Bluetooth source again without going back into this config.txt document and deleting/commenting out these two lines.**)
29) Right click on the desktop menu bar and click on "Panel Settings". Click "Advanced" and check "Minimize panel when not in use".
30) Use one of the below two methods to transfer the VideoAPA software to the RPi:
31) Flash Drive Method:
32) Insert a FAT32 formatted flash drive into a PC with the DVR Software.
33) Navigate to the VideoAPA folder in the "Place in RPi" directory and copy the file called "scripts" to the flash drive.
34) Remove the flash drive from the PC and connect it to the RPi.
35) A window should pop up indicating the drive was connected, hit "OK".
36) Open a terminal and type the command: cp -r /media/pi/<FLASH DRIVE NAME>/scripts /home/pi/
37) Remove the flash drive from the RPi.
38) File Transfer Method: Add the provided "scripts" folder to the "/home/pi" directory using a file transfer client such as WinSCP or FileZilla using an SFTP connection. More information on this can be found at: https://www.raspberrypi.org/documentation/remote-access/ssh/sftp.md
39) The RPi is now set up. Make additional copies of the RPi image for use with other RPis using one of the below methods:
40) RPi Method:
41) Connect a blank microSD card to the already set up RPi using a USB microSD card reader.
42) Open the main RPi menu (raspberry symbol on top left), navigate to "Accessories", and open "SD Card Copier".
43) Select the already imaged card as the "Copy From Device", select the blank microSD as the "Copy To Device", and hit "Start".
44) Wait for the operation to complete, remove the new microSD card, and test it on another RPi.
45) Win32 Method for Windows Users:
46) Turn off the RPi and remove the microSD.
47) Download and install Win32: https://sourceforge.net/projects/win32diskimager/
48) Follow the instructions on the following page to copy RPi images: https://raspi.tv/2012/how-to-make-a-raspberry-pi-disk-image-to-sd-card-with-win32diskimager

Host PC for Windows:
1) Install Python 3.4.4 found on the python website: https://www.python.org/downloads/
(**Python 3.4 is used due to its compatibility with PyGObject on Windows**)
2) Install Gstreamer along with PyGObject on Windows using the following link: https://sourceforge.net/projects/pygobjectwin32/files/?source = navbar. (**When prompted, add PyGObject to Python 3.4.4 path**)
3) Install PyQt5 from:
https://sourceforge.net/projects/pyqt/files/PyQt5/PyQt-5.4.1/
4) Navigate to the following link, scroll down to "Do you want an older version", and click the link for 2010 products:
https://visualstudio.microsoft.com/vs/older-downloads/
5) Download and install Visual Studio 10.0 Express from the list of Microsoft 2010 products (registration required). *(**Visual Studio 10.0 isn't necessary to run the software however some of the packages that are installed (specifically OpenCV-python) require Microsoft Visual C++ build tools. Version 10.0 is required for Python 3.4. Visual Studio 14.0 is recommended for Python 3.6 **)*
6) Type "cmd" into the search bar next to the Windows icon on your PC and open the Command Prompt.

7) Type the following commands into the Command Prompt window to set up software dependencies:
8) easy_install pip==8.0.1
9) pip install numpy==1.15.0
10) pip install opencv-python==3.3.0.9
11) pip install setuptools==18.2
12) pip install requests==2.23.0
13) pip install pytz==2020.1
14) pip install yaml==5.3.1
15) pip install urllib3==1.25.9
16) "C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\vcvarsall.bat" && pip install crypto==1.4.1
17) pip install paramiko==1.18.0
18) pip install matplotlib==2.0.2
19) pip install typing
20) If vcvarsall.bat can't be found automatically, search for it in File Explorer where Microsoft Visual Studio 10.0 was downloaded.
21) Reopen the "Command Prompt" and type in:
cd C:\Users\[YOURUSERNAME]\Downloads\[DVR SOFTWARE CONTAINING FOLDER] \VideoAPA_For_Host_PC\acquisition > runacquisition.bat
22) Type into the command prompt:
ipconfig
23) If the return of the 'ipconfig' command contains the text 'vEthernet' (Default Switch) go to 'Control Panel -> Programs -> Programs and Features -> Turn Windows features on or off' and check off the box 'Hyper-V' thereby disabling the vEthernet (Default Switch) and restart your computer. The vEthernet (Default Switch) is a default ethernet connection that interferes with the connection between your host PC and the RPI's

Host for PC (Ubuntu 18.04)

(**Note whenever the terminal displays a prompt with: [sudo] password for < YOUR USERNAME>:
type in your password. The password will not show up on the terminal as it is typed, however the terminal is registering it. Press enter when done.**)
(** Also, during these instructions you may get prompts such as Do you want to continue? [Y/n]
Please type "y" and hit enter **)

1) Download specified RPi DVR software.zip file from the repository.
2) Copy the repository to your Documents folder and extract all files.
3) Open a terminal (CTRL + Alt + T).
4) Type the following commands into terminal:
   a. sudo apt-get install python3-pyqt5
   b. pip3 install numpy==1.15.0
   c. pip3 install opencv-python==3.3.0.9
   d. pip3 install paramiko==1.18.0
   e. pip3 install matplotlib==2.0.2
   f. pip3 install typing
   g. pip3 install pygo
   h. pip3 install pygi
   i. sudo apt-get install git
   j. sudo apt-get install emacs
   k. cd
5) Carefully install python3.4 by typing the following commands into the terminal (See full instructions here https://www.tutorialspoint.com/how-to-install-python-3–4-4-on-ubuntu):
   a. sudo apt-get install build-essential checkinstall
   b. sudo apt-get install libgdbm-dev tk-dev libncursesw5-dev libssl-dev libsqlite3-dev libreadline-gplv2-dev libc6-dev libbz2-dev
   c. cd /usr/src
   d. sudo wget https://www.python.org/ftp/python/3.4.4/Python-3.4.4.tgz
   e. sudo tar xzf Python-3.4.4.tgz
   f. cd Python-3.4.4
   g. sudo ./configure
   h. sudo make altinstall
6) Then type in the following and install gstreamer1.0:
   a. sudo apt-get install -f gstreamer1.0
   b. sudo apt-get install -f gstreamer1.0-tools

(**The above may not always work so you may have to install manually as seen in the following command. We recommend doing both, just in case.**)

c. sudo apt-get install libgstreamer1.0–0 gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-bad gstreamer1.0-plugins-ugly gstreamer1.0-libav gstreamer1.0-doc gstreamer1.0-tools gstreamer1.0-x gstreamer1.0-alsa gstreamer1.0-gl gstreamer1.0-gtk3 gstreamer1.0-qt5 gstreamer1.0-pulseaudio

7) Install the MP4Box application and dependencies that are required following directions found on https://gpac.wp.imt.fr/tag/mp4box/. *(***Update as of 7/17/2020 MP4Box doesn't work as expected with our software on Linux due to recent update by gpac. Please download an older version****)*

8) Open a terminal and type the following commands:

a. git clone https://github.com/gpac/gpac.git
b. cd gpac
c. ./configure --static-mp4box --use-zlib = no
d. make -j4
e. sudo make install
f. cd

9) To check that MP4Box was installed correctly, type "which MP4Box" into the terminal and it should echo with "/usr/local/bin/MP4Box".

10) Type the following commands into the terminal:

a. cd Downloads/<INSERT the containing directory for VideoAPA_For_Host_PC>/VideoAPA_For_Host_PC/acquisition/
b. chmod + x linux_vr_run.sh
c. ./linux_vr_run.sh

11) The program should now run. There should be 4 black screens and a window open. Now close that window. Whenever you want to run the program again, go to the same location and type "./linux_vr_run.sh" again. *(**Whenever you close out of the application you cannot immediately open the application again without seeing an error. This is normal as you must allow time for the RPis to reboot and reconnect to your PC. You should wait about 30–40 s before each close and open session. **)*

12) If Ubuntu was just installed, you may get an error in server.py get_ip_address. This is because the ethernet identifier may have been changed during installation of Ubuntu. To double check the configuration, type the following commands in the terminal:

a. sudo apt-get install net-tools
b. ifconfig -a

13) If nothing called eth0 appears, complete the following instructions. Otherwise, you are finished setting up the Host PC!

14) Type the following command into the terminal:

a. sudo nano /etc/default/grub

15) Change the line that reads "GRUB_CMDLINE_LINUX=" to:

GRUB_CMDLINE_LINUX="net.ifnames = 0 biosdevname = 0″

16) Return to the terminal and type the following command

a. sudo update-grub

17) Reboot the computer

5.2. Hardware

1. Turn off the WIFI on the host PC and disconnect it from any wired networks.
2. Connect each RPi camera (C1) to a Raspberry Pi (R1).
3. Connect the Host PC and each RPi to any of the numbered ports (i.e. not the WAN or LINK ports) on the Gigabit Router (E1) with a Category 6 cable (E4). If the router does not have enough ports, connect an Ethernet Switch (E2 and/or E3) to a numbered port on the router and use its extra ports for the remaining connections.
4. When the host PC connects to the router for the first time, follow the Windows prompts to set up the network as a home network. This can also be done through "Network and Sharing Center". Also go to "Windows Firewall", select "Allow a program or feature through Windows Firewall", and ensure Python is allowed through every type of network.
5. Power each RPi with the Micro USB power port (P1). We suggest using USB charging hubs with at least 2 Amps of current capacity for RPi 2 or 3′s and 3 Amps of current for RPi 4′s. Underpowering the RPis can lead to issues powering cameras.

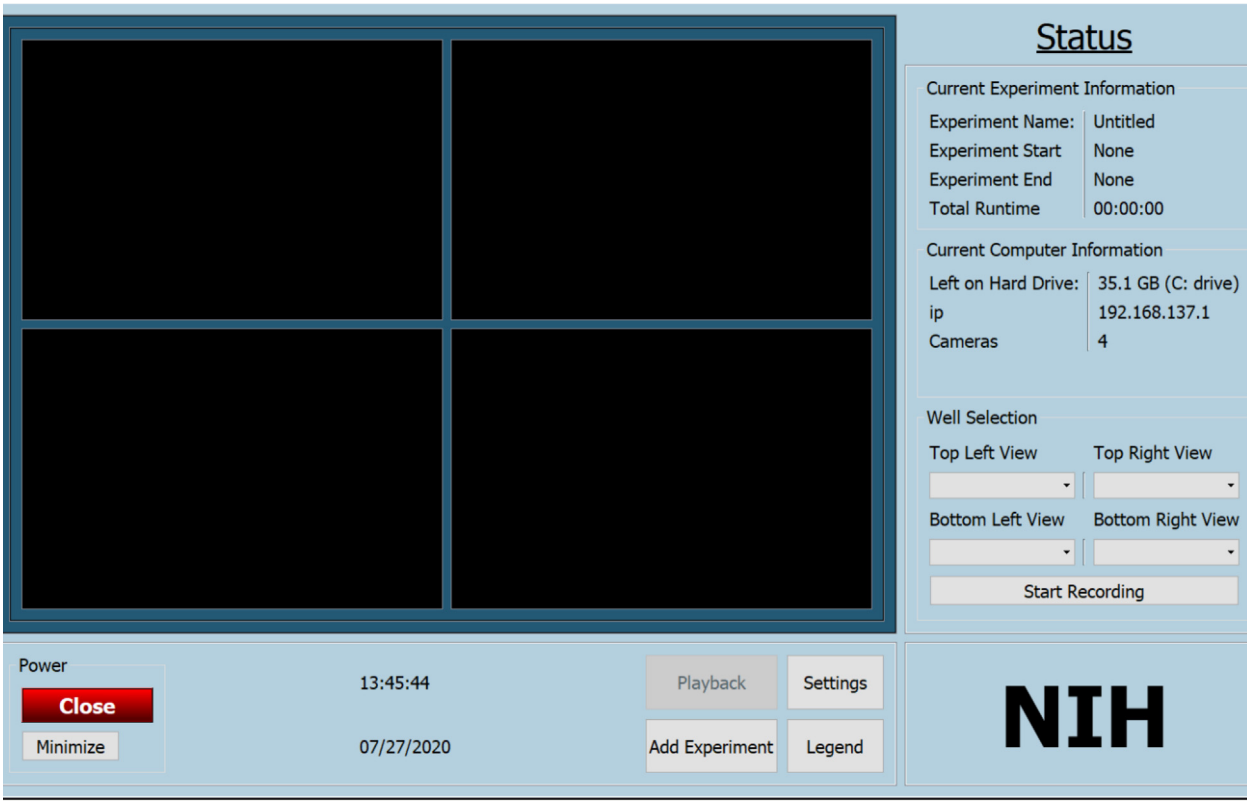

Fig. 3. View of GUI at launch.

6. Operation instructions

6.1. Navigating the software

1) Run the runacquisition batch file. The following window (see Fig. 3) will appear:

6.2. Changing settings

1) From the main menu, click on Settings. The following window (see Fig. 4) will appear.
2) On this settings menu, the user can change settings such as frames per second (fps), ISO, resolution, and the orientation of the video. The user is also able to set the length for each video segment, as well as the save path for storing the settings.

6.3. Bundling cameras

1) From the settings menu, click "Bundle Settings." The following window (see Fig. 5) will appear:
2) In the dropdown menu in the middle of the screen, select "Add new cage." A window will allow the user to define a name for the cage.
3) In the same dropdown menu, select the newly defined cage.
4) Select the desired camera ID on the left side of the window. A preview (see Fig. 6) will appear.
5) To assign the selected camera to the selected cage, click "Set Camera as Main," "Set Camera as Front," or "Set Camera as Rear." This will allow you to assign which view you can see.
6) Repeat steps 3–5 to add each desired cage/camera.

6.4. Unbundling cameras from cages

1) From the main menu, click the "Settings" button (See Fig. 3, Settings is in the bottom right corner).
2) On the settings window, click "Bundle Settings" (See Fig. 4).

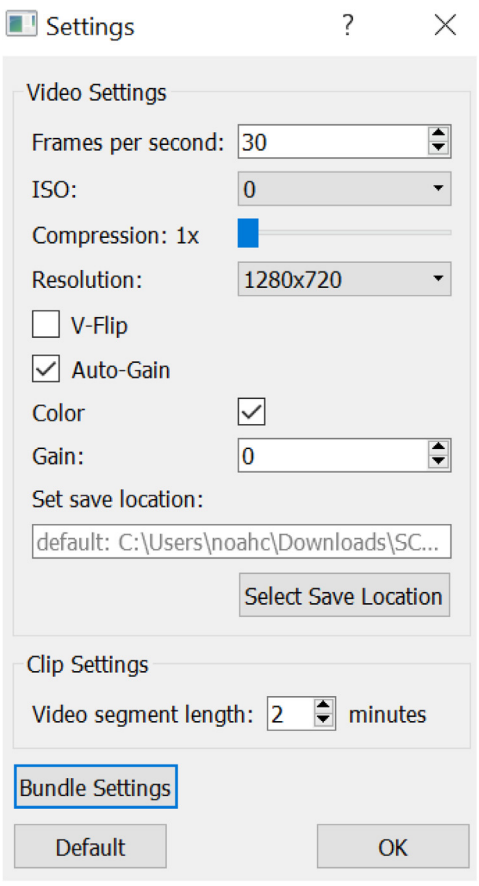

Fig. 4. Settings menu GUI.

3) In the dropdown menu in the middle of the page, select the cage name you want to unbundle from the camera (See Fig. 5).
4) Select the camera you would like to remove from the bundle from the list under "Cameras in Selected Cage".
5) Press "Remove Camera" to remove the camera from the bundle.
6) To remove a cage from the list, use the dropdown menu in the middle of the Bundler window to select "Delete cage" and select the cage to delete from the list that appears.
7) Repeat steps 3–6 for each desired camera/cage pair.

6.5. Previewing active cameras

Once cameras are bundled as cages, the user will be able to preview up to four video streams in the main GUI window (Fig. 7). Each quadrant of the main GUI's live preview display is controlled by each of the 4 corresponding dropdown menus in the lower right side of the GUI window (See Fig. 8 for an enlarged image of the camera selection section in the main GUI). The user can set each preview quadrant to any of the active cages to preview the stream from the cage's camera. Once a quadrant is set to preview a cage, it can be changed to another cage at any time by the user.

6.6. Starting an experiment

1) From the main menu, click on Add Experiment. The following window as show in Fig. 9 will open.
2) The left panel allows the user to set the experiment name, start/finish times, and save location. The right panel allows the user to select which cameras to record video from.
3) If the experiment is set incorrectly (e.g. the start time is before the end time but after the current time, etc.), the user will be prompted to fix the experiment parameters.
4) The software will initiate recording automatically at the start time. Likewise, the software will automatically terminate recording at the end time. If the user would like the start or finish early, they can manually use the "Start/Stop Recording" button in the lower right-hand corner of the main GUI.

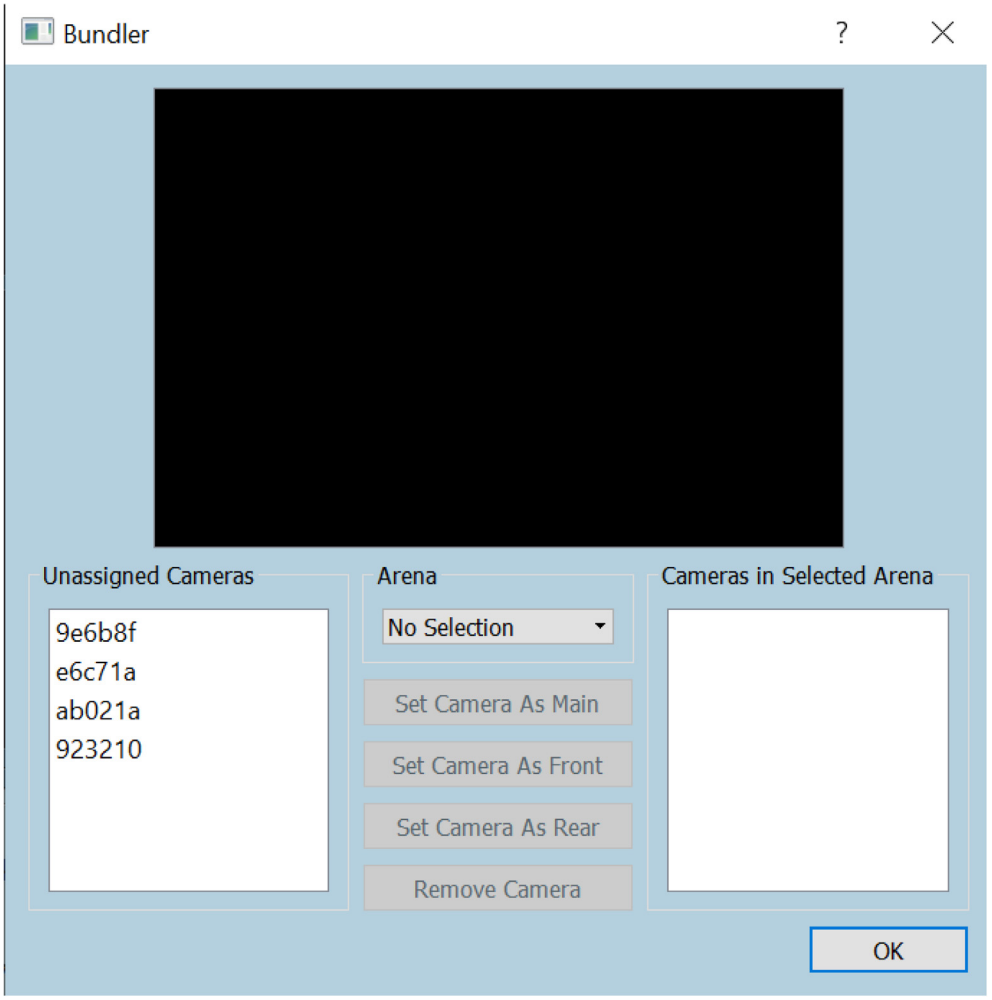

Fig. 5. Sample view of bundler GUI for grouping cameras.

7. Validation and characterization

Two versions of the RPi camera were used for validation and testing of both DVR hardware and software; Version 1 (OV5647) and Version 2 (IMX219). Both cameras can acquire video using a range of settings which are detailed in Tables 1 and 2.

The software works with Version 1 camera. Thorough testing was done with Version 2 camera. More specifically, tests included mode 1 (See Table 2) at 30 fps, mode 6 both at 30 fps and 60 fps, and mode 7 at 30, 60, and 90 fps. In order to assess the accuracy of the DVR software in terms of frame synchronicity and frame acquisition, a simple clock program was written to display the current time with millisecond resolution. The program also displays a counter value from 1 to 30 incremented each 33 ms. As shown in Fig. 1, four cameras were positioned in view of a screen displaying this program output and video was recorded using the DVR software simultaneously from all four cameras. The videos from each camera were visually compared to assess frame offsets and timing differences based on the observed video from the millisecond clock program. Video was found to stream reliably for all for cameras for at least a 2 min period before any frame drift occurred. Also, based on this test, the timing between all 4 cameras was accurate within a 33 ms period as seen in Fig. 10. In all the tests, the DVR had no dropped frames.

The image shown in Fig. 10 was used to assess preview latency. In the image, it is seen that the video preview in the DVR software lags behind the counter by approximately 170 ms.

The scalability and adaptability of this system is demonstrated in an example use case in fruit fly monitoring. Over the past decade, there has been a growing interest in the development of monitoring devices and automated systems for detecting behavioral changes in Drosophila melanogaster in response to different stimuli. These automated systems aim to reduce the cost and duration of standard assays. The measures derived from automated monitoring systems aid in phenotyping

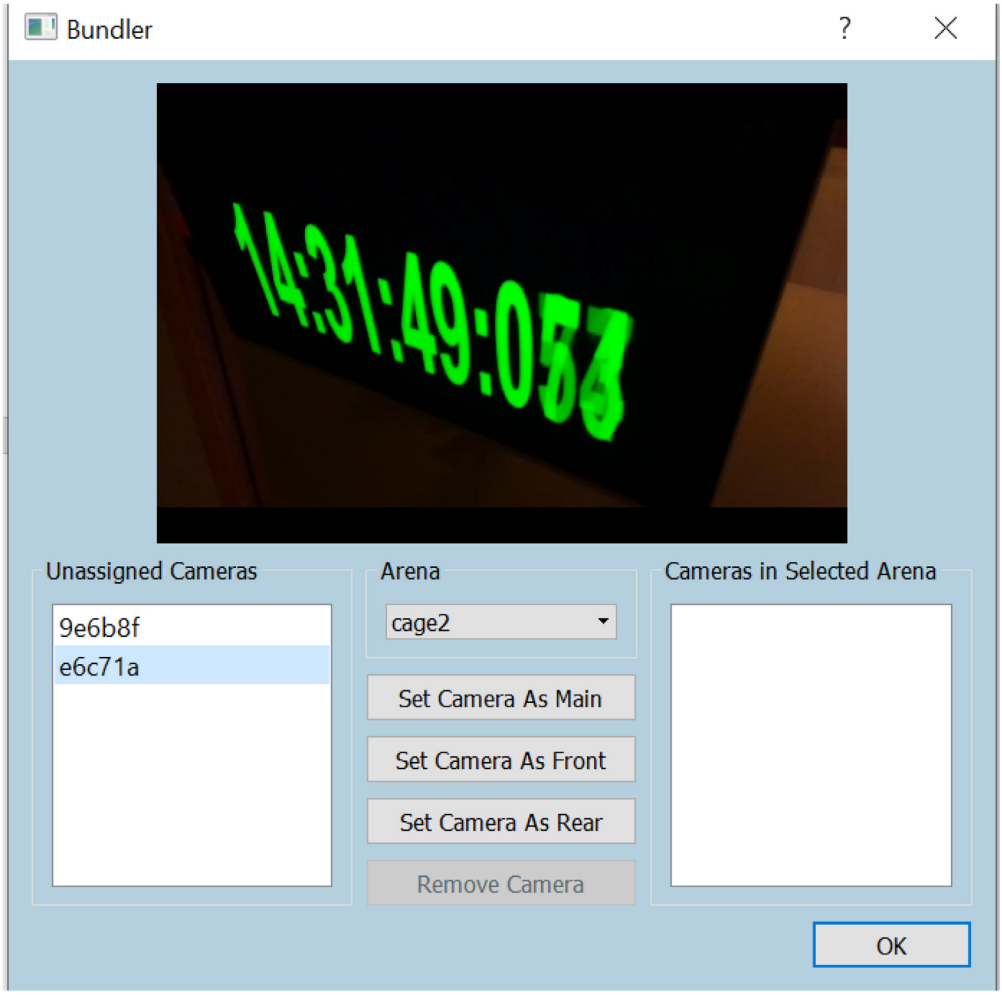

Fig. 6. Camera stream preview on bundler GUI.

transgenic Drosophila, understanding molecular bases of disease, and assessing efficacy of new drugs, ultimately accelerating development of novel therapeutic treatments and targeted medicine. Currently employed methods can monitor several flies in a single vial or a height-limited flat volume, both of which are relatively low-throughput and unsuitable for scaling up to support large or multiplex studies. Therefore, based on the DVR described in this work, a novel monitoring system focused on advancing high-throughput automated activity assessment of Drosophila was designed and implemented.

The mechanical construction of the system, named the Monitoring Unit for Fruit Fly Imaging in Ninety-six well plates (MUFFIN), is described in [13]. At the core of the system's operation is an array of 24 RPi cameras placed in an arrangement to monitor 96 separately housed fruit flies (i.e., four fruit flies per camera). A 24-port Ethernet switch and a five-port Ethernet switch were used, as described in Section 5.2, to send the video data to an external computer. The DVR was used to capture high resolution (720p) and frame rate (30 fps) video from all 24 cameras for multiple 24-hour experiments. The acquired video was analyzed to track fly position and enable detection of behaviors of interest. The use of the DVR allowed fully automated quantitative analysis of the progression of behavioral patterns associated with chemical treatments, disease models, and other stimulus. Further details on system construction and experimental results can be found in [13].

In conclusion, the DVR presented in this work offers a user-friendly turnkey system for prototyping and scientific research applications. The DVR has one main limitation, namely that it is not intended for operation on an external network, and hence is not remotely accessible. Given that the DVR is open-source, however, one can edit and augment code to circumvent this limitation. The DVR is specific to the Raspberry Pi, and hence offers an economical open-source platform for quick assembly of a cost-effective prototype. Very often, research studies are transient in nature and, therefore, quickly assembling a prototype based on an open-source platform is preferable. Such a prototype can test the feasibility of research ideas and answer key scientific questions.

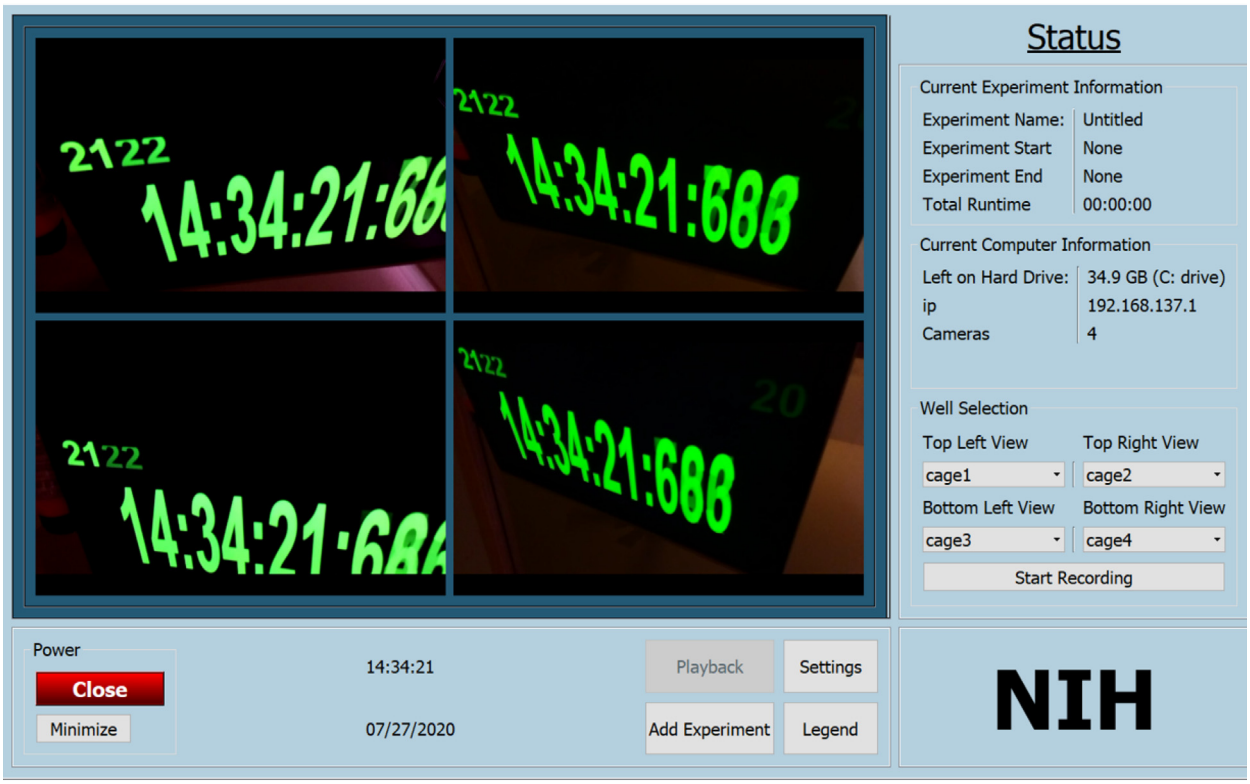

Fig. 7. Four camera preview.

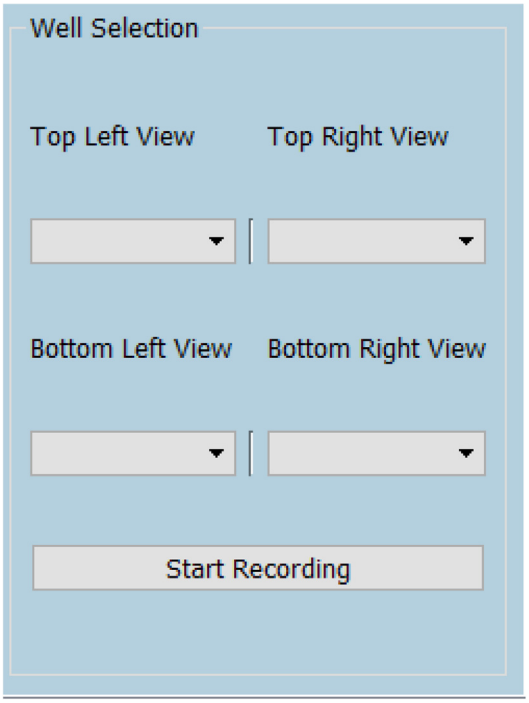

Fig. 8. Camera stream selection section of main GUI.

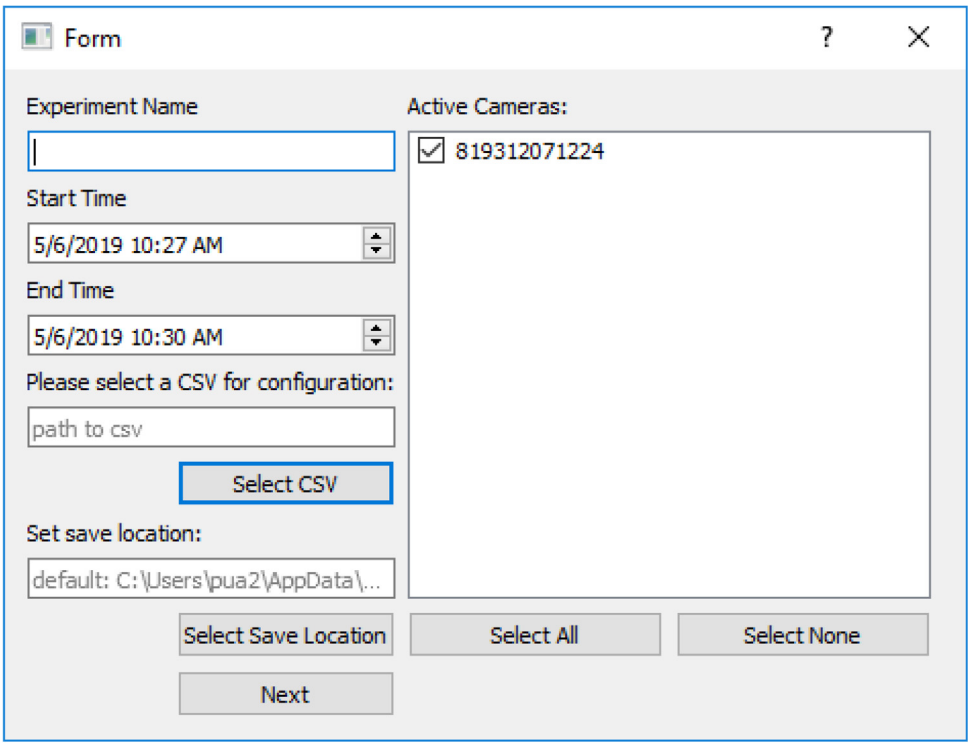

Fig. 9. GUI for defining an experiment.

Table 1
Recording Capabilities of RPi Camera Version 1.x (OV5647).

Mode	Size	Aspect Ratio	Frame rates	FOV	Binning
1	1920 × 1080	16:09	1-30fps	Partial	None
2	2592 × 1944	4:03	1-15fps	Full	None
3	2592 × 1944	4:03	0.1666-1fps	Full	None
4	1296 × 972	4:03	1-42fps	Full	2 × 2
5	1296 × 730	16:09	1-49fps	Full	2 × 2
6	640 × 480	4:03	42.1-60fps	Full	2 × 2 plus skip
7	640 × 480	4:03	60.1-90fps	Full	2 × 2 plus skip

Table 2
Recording Capabilities of RPi Camera Version 2.x (IMX219).

Mode	Size	Aspect Ratio	Frame rates	FOV	Binning
1	1920 × 1080	16:09	0.1-30fps	Partial	None
2	3280 × 2464	4:03	0.1-15fps	Full	None
3	3280 × 2464	4:03	0.1-15fps	Full	None
4	1640 × 1232	4:03	0.1-40fps	Full	2 × 2
5	1640 × 922	16:09	0.1-40fps	Full	2 × 2
6	1280 × 720	16:09	40-90fps	Partial	2 × 2
7	640 × 480	4:03	40-200fps[1]	Partial	2 × 2

Fig. 10. A close-up image from the experimental setup shown in Fig. 1. The image highlights the synchronicity of all four RPi cameras. It is clear from the DVR GUI that all four RPi are displaying the same frame counter number (i.e., the '29' shown above the hr:min:sec:msec reading) of the clock being video-recorded by the RPis. The image also shows the latency between the preview and the clock being recorded. Namely, the clock is on frame 4 whereas the preview is on frame 29 of the previous second, amounting to a 5-frame delay or approximately 167 ms preview latency.

This work was funded by the National Institutes of Health Intramural Research Program.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A.I. Dell, J.A. Bender, K. Branson, I.D. Couzin, G.G. de Polavieja, L.P. Noldus, A. Pérez-Escudero, P. Perona, A.D. Straw, M. Wikelski, others, Automated image-based tracking and its application in ecology, Trends Ecol. Evol. 29 (2014) 417–428.
[2] A.A. Robie, K.M. Seagraves, S.R. Egnor, K. Branson, Machine vision methods for analyzing social interactions, J. Exp. Biol. 220 (2017) 25–34.
[3] G.H. Salem, J.U. Dennis, J. Krynitsky, M. Garmendia-Cedillos, K. Swaroop, J.D. Malley, S. Pajevic, L. Abuhatzira, M. Bustin, J.-P. Gillet, SCORHE: A novel and practical approach to video monitoring of laboratory mice housed in vivarium cage racks, Behav. Res. Methods. 47 (2015) 235–250.
[4] W. Hong, A. Kennedy, X.P. Burgos-Artizzu, M. Zelikowsky, S.G. Navonne, P. Perona, D.J. Anderson, Automated measurement of mouse social behaviors using depth sensing, video tracking, and machine learning, Proc. Natl. Acad. Sci. 112 (2015) E5351–E5360.
[5] G.H. Salem, J. Krynitsky, M. Hayes, T.J. Pohida, X. Burgos-Artizzu, Three dimensional pose estimation for laboratory mouse from monocular images, IEEE Trans. Image Process. 28 (2019) 4273–4287, https://doi.org/10.1109/TIP.2019.2908796.
[6] T. Gambo, E.E. Omizegba, A.T. Balewa, A.B. Ba'ams, Real-Time Video Display System in Articulated Vehicle for Road Safety Enhancement, Int. J. Sci. Eng. Appl. 8 (2019) 136–145.
[7] A.F. Symon, N. Hassan, H. Rashid, I.U. Ahmed, S.T. Reza, Design and development of a smart baby monitoring system based on Raspberry Pi and Pi camera, in: 2017 4th Int. Conf. Adv. Electr. Eng. ICAEE, IEEE, 2017: pp. 117–122.
[8] J.A. Grant-Jacob, Y. Xie, B.S. Mackay, M. Praeger, M.D. McDonnell, D.J. Heath, M. Loxham, R.W. Eason, B. Mills, Particle and salinity sensing for the marine environment via deep learning using a Raspberry Pi, Environ. Res. Commun. 1 (2019) 035001.

[9] Norpix, StreamPix, Norpix. (n.d.). https://www.norpix.com/products/streampix/streampix.php.
[10] CrazyPixels, CamDVR, CrazyPixels. (n.d.). http://www.crazypixels.com/.
[11] C. Kaundanya, O. Pathak, A. Nalawade, S. Parode, Smart surveillance system using raspberry Pi and face recognition, Int. J. Adv. Res. Comput. Commun. Eng. 6 (2017).
[12] A.J.K. Jayakumar, S. Muthulakshmi, Raspberry Pi-based surveillance system with IoT, in: Intell. Embed. Syst., Springer, 2018: pp. 173–185.
[13] M.D.L.A. Jaime, S. Karott, G.H. Salem, J. Krynitsky, M. Garmendia, S. Anderson, S. Harbison, T.J. Pohida, B. Oliver, The High-throughput WAFFL system for treating and monitoring individual drosophila melanogaster adults, BioRxiv. (2018), https://doi.org/10.1101/428037.