Camera Security System with Raspberry Pi Minicomputer Documentation

Release 0.1

Marek Vitula

May 21, 2018

This package provides camera security system functionality for the Raspberry Pi minicomputer. It uses Raspberry Pi NoIR camera for the motion detection and also PIR sensor if available.

Capturing is handled by the picamera package and image processing is handled by the Pillow package.

Captured images are sent to the Google Drive storage.

Package itself consist of four submodules - config, filemanipulation, motion and telemetry. Each of them has several classes.

CHAPTER

ONE

CONFIG MODULE

Config modules is used for storing of the application configuration internally. It consist of two classes - BaseConfig and UserConfig. UserConfig inherits from the BaseConfig class.

UserConfig class adds variables which can be loaded from the external JSON file.

class `rpicameramon.config.`**BaseConfig**
Bases: `object`

Basic configuration which cannot be changed online

imageDir
str – default image directory

imagePath
str – path to the directory + imageDir

imageWidth
int – width of taken pictures in pixels

imageHeight
int – height of taken pictures in pixels

imageVFlip
bool – flips image vertically if true

imageHFlip
bool – flips image Horizontally if true

imagePreview
bool – opens window with picture if true

imageQuality
int – JPEG quality 0...100

vecMagnitude
int – minimum magnitude of a motion vector

vecCount
int – minimal number of motion vectors

cameraFPS
int – framerate of the camera

confCheckTime
int – how long does it take between conf file checks in sec

PIRpin
int – GPIO pin (BCM) where is PIR sensor connected

LEDpin
int – GPIO pin (BCM) where is LED connected

pictureFolderID
str – Google Drive ID of folder with pictures

confFileID
str – Google Drive ID of JSON configuration files

dashboardFileID
str – Google Drive ID of dashboard panel

logRange
str – Range in sheets for telemetry messages

msgRange
str – Range in sheets for log messages

OAuthJSON
str – Name of the JSON file for Google OAuth authentication

fileUploadSleep
int – Sleep time in seconds for file uploader thread

batchUploadWindow
int – Allowed hour for file uploads in batch mode

GSMport
str – Device address of a GSM Module

GSMbaud
int – Baud rate for communication with the GSM module

GSMPIN
int – PIN for unlocking the SIM card

class `rpicameramon.config.`**UserConfig**
Bases: *rpicameramon.config.BaseConfig*

Extends BaseConfig class with the user defined configuration. This configuration can be updated from the external JSON.

mode
str – default runtime mode (realtime, interval, batch, ondemand)

echo
boolean – echo mode

interval
int – default interval time in seconds for interval mode

storage
str – gdrive or dropbox. Dropbox is only experimental

usePIR
boolean – use PIR sensor

SMSNotification
boolean – SMS SMSNotification

SMSControl
boolean – Allow control via SMS messages

authorizedNumber
int – Authorized number for control and notifications

classmethod load_config(*conf*)
This method loads configuration to UserConfig class from the JSON file.

Raises `KeyError` – If JSON cannot be parsed

CHAPTER

TWO

FILEMANIPULATION MODULE

This module consist of classes for the manipulation with files in Google Drive.

class rpicameramon.filemanipulation.**ConfFileDownloader**(*group=None*, *target=None*, *name=None*, *filename=None*, ***, *daemon=None*)

Bases: threading.Thread

ConfFileDownloader is a subclass of a Thread class First it loads the initial JSON configuration from Google drive and then it check if the configuration was changed. If yes, it loads it.

filename
string – file ID of JSON configuration file on Google drive

run()
This method runs when ConfFileDownloader thread is started. First it initialize connection with Google Drive and starts loop for checking if JSON configuration file on Google Drive was modified. If yes, It downloads it and calls the method for loading it into UserConfig class.

File is checked in intervals specified in confCheckTime variable.

auth()
Function for Google Drive authentication. It loads credentials from the file "mycreds.txt".

Returns Drive object

Return type drive

download_file(*drive*)
This function downloads specified file from the Google Drive.

Parameters **drive** (*obj*) – Google Drive object

Returns Content of the file as string

Return type str

get_timestamp(*drive*)
This function fetches date of the last modification of specified file.

Returns Timestamp of last modification of the file

Return type timestamp

parse_json(*jsonfile*)

load_config(*configdata*)

class rpicameramon.filemanipulation.**FileUploader**(*group=None*, *target=None*, *name=None*, *storage=None*, ***, *daemon=None*, *q=None*)

Bases: threading.Thread

This class provides a File uploader thread for uploading captured photos. It supports google drive and dropbox (experimental)

storage
str – storage type (gdrive or dropbox)

q
Queue obj – queue with filenames of pictures to be uploaded

dropbox_is_init
boolean – True if Dropbox is initialized

gdrive_is_init
boolean – True if Google Drive is initialized

run()

gdrive_init()

gdrive_upload(*filename*, *drive*)

dropbox_init()

dropbox_upload(*filename*, *dbx*)

rpicameramon.filemanipulation.**stopwatch**(*message*)

CHAPTER

THREE

MOTION MODULE

This module has functionality for the motion detection from the scene and also from the PIR sensor and also SMS handler. Optical flow method is used for the motion detection.

class rpicameramon.motion.**MotionAnalysis**(*camera*, *handler*)
Bases: picamera.array.PiMotionAnalysis

MotionAnalysis class extends PiMotionAnalysis class.

The array passed to analyse() is organized as (rows, columns) where rows and columns are the number of rows and columns of macro- blocks (16x16 pixel blocks) in the original frames. There is always one extra column of macro-blocks present in motion vector data.

analyse(*a*)

class rpicameramon.motion.**PIRMotionAnalysis**(*pin*, *handler*)
Bases: object

This class provides a handler for PIR sensor It detects if the input pin is in the high state and calls motion_detected handler

Parameters

- **pin** (*number*) – BCM number of GPIO pin where is PIR sensor connected
- **handler** (*obj*) – CaptureHandler object

is_detected()

class rpicameramon.motion.**CaptureHandler**(*camera*, *post_capture_callback=None*, *q=None*)
Bases: object

It provides a handler for capturing the pictures. With the LED Switch functionality.

camera
obj – PiCamera object

callback
str – callback

q
obj – queue for passing captured photos

detected
bool – True if motion is detected

working
bool – True if the picture is saving

i
int – counter of captured photos

echoCounter
int – counter of taken pictures with echo mode

motion_detected()

tick()

This tick method provides a handler for capturing the pictures. It ticks every second after PiMotion.start() was called. If detected is True and method is not processing any capture (That is indicated by variable 'working'), it begins with processing. First, datetime method is called to obtain the actual datetime, then the scene is analyzed with scan_day method which returns true if light conditions appear to be daylight or false if the light level is too low.

If the echo mode is activated

For daylight: exposure compensation is set to 0 <-25;25> exposure mode is set to auto camera shutter speed is set to 0 (auto mode)

For bad light conditions: exposure compensation is set to 25 for maximum brightness exposure mode is set to night preview shutter speed is set to 200000 microseconds which is equal to 0,2s Turn on the LED

scan_day()

This method captures picture as an RGB array and calculates average value of the pixels in the matrix. If the value pixAverage is more than 50 (scene is gray to black). It indicates that the light condition is poor and we can set up night params.

class rpicameramon.motion.**SMSHandler**(*handler*)

Bases: object

handle_sms(*sms*)

class rpicameramon.motion.**PiMotion**(*post_capture_callback=None*, *q=None*)

Bases: object

This class handles the camera and PIR sensor setup. It sets up resolution and framerate and starts capturing the video outputting it to the MotionAnalysis object.

Parameters

- **post_capture_callback** (callback) – not in use
- **q** (*Queue obj*) – queue for captured photos

Raises KeyboardInterrupt – Interrupt the program

start()

CHAPTER

FOUR

TELEMETRY MODULE

This module provides functionality for sending telemetry data to the Google Sheets.

class rpicameramon.telemetry.**GoogleHandler**
Bases: object

get_credentials()
Gets valid user credentials from storage.

If nothing has been stored, or if the stored credentials are invalid, the OAuth2 flow is completed to obtain the new credentials.

Returns Credentials, the obtained credential.

get_sheets_service(*credentials*)
Gets Google Sheets Service v4

Returns Google Sheets service object

get_file_service(*credentials*)
Gets Google Drive Service v3

Retruns: Google Drive service object

add_sheet_line(*service=None*, *line=None*, *spreadsheetId=None*, *rangeName=None*)
Append a line/lines to Google Sheet.

Parameters

- **service** (*obj*) – Google Sheets Service object
- **line** (*nested list*) – matrix of data to be put into Google Sheet
- **spreadsheetId** (*str*) – Id of spreadsheet
- **rangeName** (*str*) – Range in a spreadsheet

upload_file(*service*, *filename*)
Uploads a image/jpeg file to the Google Drive.

Parameters

- **service** (*obj*) – Google Drive Service object
- **filename** (*str*) – full path to the file.

class rpicameramon.telemetry.**LogSender**(*logQueue*, *googleHandler*)
Bases: threading.Thread

LogSender class is a subclass of a Thread class. It provides a functionality to send a captured log to Google Sheets. It should be used together with the logging.handlers.QueueHandler class as a handler for logging. LogSender works only in specified time intervals, dequeues everything in logQueue, sends it to the Google Sheets and goes sleep for the amount of time specified by the variable fileUploadSleep

logQueue
obj – queue.Queue object with logRecord (obj)

enqueued

googleHandler
obj – GoogleHandler instance object

run()

class rpicameramon.telemetry.**TelemetrySender**(*googleHandler*)
Bases: threading.Thread

Telemetry class is a subclass of a Thread class. It provides a functionality for sending a telemetric data to Google Sheets.

Attributes:

run()

getTelemetry()

H

I

L

M

O

P

Q

R

S

T

U

V

W