TITLE タイトル

SUBTITLE サブタイトル

著者 1　　著者 2

提出日: \today

概要

アブストラクトの中身 abstract の中身

目次

第 1 章　contents コンテンツ

1-1　about run pdf 生成

```
$ docker run -it --rm --volume "$(pwd):/data" manned2665/pandoclatex311 -d
    pdf-defaults.yaml -M listings
```

1-2　about 書き方

斜体だよ *syataidayo*

太字だよ HUTOJIDAYO

太字斜体だよ *HUTOJIsyataidayo*

contents contents.

本文だよ

URL だよ

はみ出す↓

https://aaaaaaaaaaaaaaaaaaaa-aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

aaaaaaaaaaaaaaaaaaaa-aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

はみ出す↓（欧文で単語区切りが無いため）

aaaaaaaaaaaaaaaaaaaa-aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

はみ出さない↓（欧文で単語区切りがあるため自動改行）

LaTeX is a document markup language that is particularly well suited for the publication of mathematical and scientific articles.

Pandoc is a free-software document converter, widely used as a writing tool (especially by scholars)[2] and as a basis for publishing workflows.[3] It was created by John MacFarlane, a philosophy professor at the University of California, Berkeley.

はみださない↓（和文）

あああ

さ
き

1-3　about fig 図

図 1　すごい図[*2]

すごい図. 1 を貼り付けたよ。
図はスペースいらない

1-4　about table 表

表 1: AND

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

This is 表. 1.
表はスペースが**必要**

1-5　about equations 式

$$a^2 + b^2 = c^2 \tag{1}$$

This is 式. 1 .
式はスペースいらない
$\alpha, \beta, \gamma, \delta, \Delta, \varepsilon, \theta, \lambda, \mu, \nu, \pi, \rho, \sigma, \Sigma, \tau, \phi, \omega$

$$\frac{\partial f}{\partial y}\frac{df}{dx}$$

1-6　about commemts コメント注釈

This is comment[*3].

[*2] 図のタイトルにも注釈をつけられる
[*3] This is comment.

1-7　about code block コードブロック

python コードをコード. 1 に示す。

コード 1　nyancam.py

```
"""
にゃんCAM

更新履歴
2021/09/26 : rev0 new
2021/09/29 : rev1 撮影時のみカメラ起動、日本語化
2021/10/08 : rev2 リトライ処理実装（おもにwifiが無くなったことを想定）
2021/10/09 : rev3 カメラ2台対応（1台用とは別にする
    1台用はrev2）,カメラ自動認識(1台でも認識はするがエラーになる。あくまで2台用)
2021/10/11 : rev4 カメラ2台用
    リトライ処理削除(リトライはsystemdのRestart=alwaysが行うため)
2021/10/13 : rev6 カメラ2台用（ベース：rev4） 起動時にまず撮影
2021/10/13 : rev7 いじり respをそとに resp間にsleep(resp_time)
    送る間隔が早すぎてエラーになってたかも(ネットが低速で画像が送れない)


自動起動
sudo nano /lib/systemd/system/nyancam.service
sudo systemctl start nyancam.service
sudo systemctl stop nyancam.service
sudo systemctl enable nyancam.service
sudo systemctl disable nyancam.service

/etc/systemd/system.conf
#DefaultStartLimitInterval=10s
#DefaultStartLimitBurst=5
systemdは10秒の間に5回まで再起動が行われ、それを超えると再起動をやめてしまう
ので、そうならないように起動時にtime.sleep(initwait)する
initwait = 3
10秒では3秒*3回=9秒より、3回しか再起動できないため、再起動しつづける動きとなる

再起動 1日4回 0,6,12,18時
systemdで実装
saikidou.service
saikidou.timer ← enable


機能
・ラズパイ起動時に自動起動、撮影開始
 ・プログラムが失敗（wifi消失などによる）したときはsystemdにより自動再起動する
    再試行中はカメラの青LEDが点灯（接続カメラチェックによる）するためわかる
```

```
    再試行周期はinitwait+5秒くらい
・短押し : テスト撮影（カメラ設置場所決定のため）
・長押し（５秒以上） : shutdown

エラー時（wifi消失）
wifi消失によりエラーとなるのはAPIで通信する
resp = requests.post(api,headers = headers,data = data)
がある以下3箇所
1.### にゃんCAM 起動 ###
2.ボタン短押し（テスト撮影）
3.定期撮影（30分ごととか）

エラーパターン1 : 最初からwifiがない
1.でエラーになる

エラーパターン2 : 途中でwifiがなくなる
1.はきっと通過している
2.や3.でエラーとなる
wifiが復帰したら、最初からやり直しなのでLINEで### にゃんCAM 起動
    ###がまた来る

"""
import cv2,time,requests,os
import RPi.GPIO as GPIO
import datetime
import sys

initwait = 3
time.sleep(initwait)

######## Variables ########
rev = 7 # プログラムレビジョン
token = 'あなたのトークン' # LINE Notifyトークン

init = True # 初回のみフラグ
Snapshot_period = 1800 # 撮影周期 sec 30分=60秒*30=1800秒
Longpush_period = 5 # 長押し時間 sec 5秒
shutter_time = 5 # 露光？時間 sec elecomピンクが性能悪いので5秒
resp_time = 1 # カメラ2台 resp間ウエイト（画像が重い）

# リトライ関連
# デバッグ時は短く設定 エラー行数が出ないのが惜しい
MAXTRY = 0 # トライ回数 20回
Try_period = 1 # トライ周期 sec 1分
Trycnt = 0 # トライカウント

```

```
91  ######## 接続カメラチェック ########
92  true_cam = [] # 配列用意
93  for camera_num in range(10): #
        カメラ番号を0～9まで変えて、COM_PORTに認識されているカメラを探す
94      cap = cv2.VideoCapture(camera_num)
95      ret, frame = cap.read()
96      if ret == True: #
            capture.read()に画像が格納されていたら=画像が取得できたら=カメラが接続されていた
97          true_cam.append(camera_num)
98          #print('camera number {} Find!'.format(camera_num))
99      else: # ここでワーニング
100         #print('camera number {} None'.format(camera_num))
101         pass
102 print('接続されているカメラは {} 台です'.format(len(true_cam)))
103
104 for i in range(len(true_cam)): # カメラ番号調べ
105
106     print('カメラ{} : {} 番'.format(i,true_cam[i]))
107
108 CamNum0 = true_cam[0] # 接続されたカメラの番号
109 CamNum1 = true_cam[1]
110 print('CamNum0 : {}'.format(CamNum0))
111 print('CamNum1 : {}'.format(CamNum1))
112 ##################################
113
114 # Setup working directory
115 os.chdir('/home/USERNAME/Desktop')
116
117 # Setup GPIO
118 button = 26 # VDD(3.3V) - button - GPIO26
119 GPIO.setmode(GPIO.BCM)
120 GPIO.setup(button,GPIO.IN,pull_up_down = GPIO.PUD_DOWN) # pull down
121 button_old = time.time()
122 trig = False
123 val = False
124 long = False
125 flag = False
126
127 # Setup LINE Notify
128 api = 'https://notify-api.line.me/api/notify'
129 headers = {'Authorization': 'Bearer' + ' ' + token}
130 filename0 = 'cat0.jpeg'
131 images0 = '/home/USERNAME/Desktop/' + filename0
132 filename1 = 'cat1.jpeg'
133 images1 = '/home/USERNAME/Desktop/' + filename1
134
```

```
135 # Setup OpenCV
136 windowsize = (1024,768) # 4:3 LINE Notifyの送信可能な最大解像度
137 ImageNum = 0
138 old = time.time() # 初回時間
139
140
141 try:
142     while(True): # にゃんCAM main loop
143
144         if init == True: # 初回のみ
145             message = '\n\n### にゃんCAM 起動
                ###\nrev:{}\n\n{}分ごとに撮影しまーす! \n接続カメラ : {}
                台'.format(rev,Snapshot_period/60,len(true_cam))
146             data = {'message': message}
147             resp = requests.post(api,headers = headers,data = data)
148
149             print(message)
150             print(datetime.datetime.now())
151
152             # 初回にまず撮影する
153             shutter_t0 = time.time()
154             cap0 = cv2.VideoCapture(CamNum0)
155             cap1 = cv2.VideoCapture(CamNum1)
156
157             while True:
158                 ret, frame0 = cap0.read() # capture(too slow!!)
159                 ret, frame1 = cap1.read() # capture(too slow!!)
160                 shutter_t1 = time.time()
161
162                 if(shutter_t1 - shutter_t0) > shutter_time: # shutter time
163
164                     frame0 = cv2.resize(frame0, windowsize) # resize the
                        window
165                     frame1 = cv2.resize(frame1, windowsize) # resize the
                        window
166
167                     cv2.imwrite(filename0,frame0) # save
168                     cv2.imwrite(filename1,frame1) # save
169
170                     cap0.release()
171                     cap1.release()
172                     break # break while loop
173
174             ratelimit_image = resp.headers.get("X-RateLimit-ImageLimit") #
                max image upload at 1hour
175             ratelimit_image_remaining =
```

```
                resp.headers.get("X-RateLimit-ImageRemaining") # image
                upload remaining

            # for LINE Notify
            # カメラ0
            message0 = '\nカメラ0\nうp残 : {} /
                {}'.format(ratelimit_image_remaining,ratelimit_image)
            data0 = {'message': message0}
            files0 = {'imageFile': open(images0,'rb')}
            requests.post(api,headers = headers,data = data0,files = files0)

            time.sleep(resp_time)

            # カメラ1
            message1 = '\nカメラ1'
            data1 = {'message': message1}
            files1 = {'imageFile': open(images1,'rb')}
            requests.post(api,headers = headers,data = data1,files = files1)

            # for debug
            print(message0)
            print(message1)

            init = False # 初回のみより、initのTrueへの復帰は不要

        debug0 = time.time() # for measure time of main loop

        # button detect
        button_t0 = time.time() # now

        # snapshot
        t0 = time.time() # now

        debug1 = time.time() # for debug

        # button detect
        if button_t0 - button_old > 50/1000: #
            50msecごと（以上）にボタンの状態を監視

            button_state = GPIO.input(button) # check button state

            if button_state == 1: # button pushed
                if trig == False: # first pushed
                    firstpush = time.time() # for debug
                    trig = True
                val = True
```

```
218
219             else: # button released
220                 if val == True and long == False: # short pushed( < 5sec )
                        : Test shot
221                     shortpush = time.time() # for debug
222                     shutter_t0 = time.time()
223                     cap0 = cv2.VideoCapture(CamNum0)
224                     cap1 = cv2.VideoCapture(CamNum1)
225
226                     while True:
227                         ret, frame0 = cap0.read() # capture(too slow!!)
228                         ret, frame1 = cap1.read() # capture(too slow!!)
229                         shutter_t1 = time.time()
230
231                         if(shutter_t1 - shutter_t0) > shutter_time: #
                                shutter time
232
233                             frame0 = cv2.resize(frame0, windowsize) #
                                    resize the window
234                             frame1 = cv2.resize(frame1, windowsize) #
                                    resize the window
235
236                             cv2.imwrite(filename0,frame0) # save
237                             cv2.imwrite(filename1,frame1) # save
238
239                             cap0.release()
240                             cap1.release()
241                             break # break while loop
242
243                     ratelimit_image =
                            resp.headers.get("X-RateLimit-ImageLimit") # max
                            image upload at 1hour
244                     ratelimit_image_remaining =
                            resp.headers.get("X-RateLimit-ImageRemaining") #
                            image upload remaining
245
246                     # for LINE Notify
247                     # カメラ0
248                     # message0 = '\nカメラ0\nテスト撮影\n短押し : {}
                            sec\nうp残 : {} / {}\nエラー時リトライ回数 : {} /
                            {}\nリトライ周期 : {} sec'.format(round((shortpush
                            -
                            firstpush),2),ratelimit_image_remaining,ratelimit_image,Trycnt,M
249                     message0 = '\nカメラ0\nテスト撮影\n短押し : {}
                            sec\nうp残 : {} / {}'.format(round((shortpush -
                            firstpush),2),ratelimit_image_remaining,ratelimit_image)
```

```
250                     data0 = {'message': message0}
251                     files0 = {'imageFile': open(images0,'rb')}
252                     requests.post(api,headers = headers,data = data0,files
                            = files0)
253
254                     time.sleep(resp_time)
255
256                     # カメラ1
257                     message1 = '\nカメラ1\nテスト撮影'
258                     data1 = {'message': message1}
259                     files1 = {'imageFile': open(images1,'rb')}
260                     requests.post(api,headers = headers,data = data1,files
                            = files1)
261
262                     # for debug
263                     print(message0)
264                     print(message1)
265                     print('main loop : {} sec'.format(debug1 - debug0))
266
267                 val = False
268                 trig = False
269                 flag = False
270                 long = False
271
272             if val == True: # button pushed
273                 longpush = time.time() # for debug
274                 if (longpush - firstpush) > Longpush_period: # long pushed
                        ( > 5sec ) : shutdown
275                     if flag == False: # once count
276                         print('\n長押し : {} sec'.format(round((longpush -
                                firstpush),2))) # for debug
277                         message0 = '\n\n### にゃんCAM シャットダウン
                                ###\n\nばーい！ '
278                         data = {'message': message0}
279                         requests.post(api,headers = headers,data = data)
280                         print(message0) # for debug
281                         time.sleep(3) # シャットダウンするまで少し待ち
282                         # os.system('sudo reboot')
283                         os.system('sudo shutdown -h now')
284                         flag = True
285                         long = True
286                         trig = False
287
288             button_old = button_t0
289
290         # 定期撮影
```

```
291         if t0 - old > Snapshot_period:
292 
293             shutter_t0 = time.time() # initial time
294             cap0 = cv2.VideoCapture(CamNum0) # VideoCapture
295             cap1 = cv2.VideoCapture(CamNum1) # VideoCapture
296 
297             while True:
298                 ret, frame0 = cap0.read() # capture(too slow!!)
299                 ret, frame1 = cap1.read() # capture(too slow!!)
300                 shutter_t1 = time.time()
301 
302                 if(shutter_t1 - shutter_t0) > shutter_time: # shutter time
303 
304                     frame0 = cv2.resize(frame0, windowsize) # resize the
                            window
305                     frame1 = cv2.resize(frame1, windowsize) # resize the
                            window
306 
307                     cv2.imwrite(filename0,frame0) # save
308                     cv2.imwrite(filename1,frame1) # save
309 
310                     cap0.release()
311                     cap1.release()
312                     break # break while loop
313             ratelimit_image = resp.headers.get("X-RateLimit-ImageLimit") #
                max image upload at 1hour
314             ratelimit_image_remaining =
                resp.headers.get("X-RateLimit-ImageRemaining") # image
                upload remaining
315 
316             #カメラ0
317             files0 = {'imageFile': open(images0,'rb')}
318             # message0 = '\nカメラ0\nImage_{}\nうp残 : {} / {}\n撮影周期 :
                {} 分\nエラー時リトライ回数 : {} / {}\nリトライ周期 : {}
                sec'.format(ImageNum,ratelimit_image_remaining,ratelimit_image,round((t0
319             message0 = '\nカメラ0\nImage_{}\nうp残 : {} / {}\n撮影周期 : {}
                分'.format(ImageNum,ratelimit_image_remaining,ratelimit_image,round((t0-
320             data0 = {'message': message0}
321             requests.post(api,headers = headers,data = data0,files = files0)
322 
323             time.sleep(resp_time)
324 
325             #カメラ1
326             files1 = {'imageFile': open(images1,'rb')}
327             message1 = '\nカメラ1\nImage_{}'.format(ImageNum)
328             data1 = {'message': message1}
```

```
329             requests.post(api,headers = headers,data = data1,files = files1)
330
331             # for debug
332             print(message0)
333             print(message1)
334             print(datetime.datetime.now())
335             print('main loop : {} sec'.format(debug1 - debug0)) #
                    メインループ実行時間
336
337             old = t0
338             ImageNum = ImageNum + 1
339         Trycnt = 0 # プログラムに復帰したらトライカウントをリセット
340
341 except KeyboardInterrupt:
342     print('\nプログラム終了!!')
343     sys.exit()
```

参考文献

{-} をつけるとこのセクションには見出しに通し番号がつかなくなる

- 箇条書き
- 箇条書き
  - ネスト
  - ネスト
  - ネスト