Setup & Operation

Table of Contents

1. Introduction

The following document contains information about sending multimedia files to and from the Takano platform. The platform can store those files and send them attached to emails to registered users. The supported file formats are .jpg, .jpeg, .gif, .png, .wmw, .avi, .mp3, .wav, .m4a, .mp4 and .h264. Before delving into the examples, it should be noted that the hardware linked to the platform should be able to handle multimedia files i.e. record, save and send those files to the internet. For that, we have chosen the Raspberry Pi 3 Model B+ (www.raspberrypi.org) to run our examples since it is a relatively cheap single board computer and enjoying a large support community worldwide, even though it is not the latest model . It can accommodate a pcb-insertable camera to directly take photos or videos. The examples can of course be ported to other hardware as well with minimum tweaking, especially that they are written in a popular open-source Software, Python (using the Thonny IDE that is pre-installed with the Raspbian Stretch OS), that runs on almost all operating systems. It is assumed here that the user has already registered on the Takano platform and obtained a hardware registration key for the hardware in question, the Rasbperry Pi.

2. Uploading a File to Takano

In this example, the Raspberry Pi will simply upload a file to the Takano platform. The platform will save the file on its server and notify registered users by an email with the file attached to it. Please note that if the file size is bigger than 1.1Mbytes, it will be rejected by the server, and that code indentation must be respected as it is part of program logic flow in Python. The file name must be the hardware registration key obtained from the Takano platform during registration. The extension can only be one of the following: .jpg, .jpeg, .gif, .png, .wmw, .avi, .mp3, .m4a, .mp4 and .h264.
The code goes below:

```
import os
import requests
#the file name to be uploaded
#change it to your hardware registration key!
file_to_upload = 'uwq4ux0wxrz4.jpg' #put the file in the same directory as this Python script
#supported file extensions: jpg, jpeg, gif, png, wmw, avi, ,wav, mp3, m4a, h264 & mp4
url = 'http://www.iotsprint.com/upload.php'
with open(file_to_upload, 'rb') as img:
    name_img= os.path.basename(file_to_upload)
    b = os.path.getsize(file_to_upload) / 1000 #convert from bytes to Kbytes
    if b > 1100: #terminate script if file size > 1100 Kb
        print ("Aborting- File too large: ", b, " Kb")
        quit()

    files= {'fileToUpload': (name_img,img,'multipart/form-data',{'Expires': '0'}) }
    with requests.Session() as s:
        r = s.post(url,files=files)
        print(r.status_code)
```

Running the above script will save the file 'uwq4ux0wxrz4.jpg' to the Takano server. Running it again with a file named 'uwq4ux0wxrz4.mp4' will override the .jpg file. In other words, each hardware registration key is allowed one multimedia file at a time. The user will of course receive another email with the file 'uwq4ux0wxrz4.mp4' attached to it.

3. Taking a Photo and Uploading it

In this example the Raspberry Pi will take a photo using the attached camera (Rev 1.3, 5MP, 1080p), save the image (.jpg) locally, and then upload it to the Takano platform. Registered users will receive it by email.

The Python program goes below:

```
import picamera
import time

hardware_reg_key = "uwq4ux0wxrz4"

#open cam
cam = picamera.PiCamera()
#take shot with 10% quality to avoid having a large size image
#otherwise it will be rejected by server
#you can try to change image quality, depending on cam used
#this example was done on a Raspberry PI Camera Rev 1.3 (5MP, 1080p)
#you might need to further reduce quality in case of a higher camera resolution
cam.capture("/home/pi/Downloads/"+hardware_reg_key+".jpg",quality=10)
#close cam when done to release its resourses and avoid GPU memory leak
cam.close()
print('picture taken') #print it to console

image_to_upload = hardware_reg_key+".jpg" #image is in directory of this Python script
url = 'http://www.iotsprint.com/upload.php'
with open(image_to_upload, 'rb') as img:
    name_img= os.path.basename(image_to_upload)
    b = os.path.getsize(image_to_upload) / 1000 #convert from bytes to Kbytes
    if b > 1100: #terminate script if file size > 1100 Kb
        print ("Aborting- File too large: ", b, " Kb")
        quit()

    files= {'fileToUpload': (name_img,img,'multipart/form-data',{'Expires': '0'}) }
    with requests.Session() as s:
        r = s.post(url,files=files)
        print(r.status_code) include <LiquidCrystal.h> //needed for LCD control
```

4. Taking a Video and Uploading it

In this example the Raspberry Pi will take a short video using the attached camera (Rev 1.3, 5MP, 1080p), save it locally (in .h264 format), and then upload it to the Takano platform. Registered users will receive it by email. The .h264 video compression digital standard is automatically generated by the Raspberry camera. Not all media playing Softwares or apps will accept it. If email recipients are unable to play it, they can download the VLC media player, freely available for desktop and mobile phones, in order to view it. If it still looks bumpy when playing, they can use the next python script to convert the .h264 to the more widely used .mp4 format.

The Python program goes below:

```
import os
import requests
import picamera
import time

hardware_reg_key = "uwq4ux0wxrz4"

#open cam
cam = picamera.PiCamera()
filename = hardware_reg_key+".h264"
#reduce resolution and frames per second so that saved file is < 1.1Mbytes
cam.resolution = (640, 480)
#take 24 frames per second, could also be further reduced for longer videos
#or for a higher resolution camera
cam.framerate = 24
#quality between 1 (best) and 40 (worst)
#this example was done on a Raspberry PI Camera Rev 1.3 (5MP, 1080p)
#you might need to further reduce quality in case of a higher camera resolution
cam.start_recording(filename,quality=30)
#record for 10 seconds
cam.wait_recording(10)
#then stop recording
cam.stop_recording()
print('video saved') #print it to console
#close cam when done to release its resourses and avoid GPU memory leak
cam.close()
time.sleep(1) #wait 1 second for save

url = 'http://www.iotsprint.com/upload.php'
file_to_upload = hardware_reg_key+".h264" #file is in directory of this Python script
with open(file_to_upload, 'rb') as img:
   name_img= os.path.basename(file_to_upload)
   b = os.path.getsize(file_to_upload) / 1000 #convert from bytes to Kbytes
   if b > 1100: #terminate script if file size > 1100 Kb
      print ("Aborting- File too large: ", b, " Kb")
      quit()
```

```
files= {'fileToUpload': (name_img,img,'multipart/form-data',{'Expires': '0'}) }
with requests.Session() as s:
    r = s.post(url,files=files)
    print(r.status_code)
```

If you want to convert the .h264 to the more widely used .mp4 video standard, the Python program below does that. The same logic as above code is kept. But note that the video conversion routine is heavily time consuming, so the file upload and email could take up to few minutes to complete (depending on video duration and quality), and consume CPU resources.

```
import os
import requests
import picamera
import time
import subprocess as sp

hardware_reg_key = "uwq4ux0wxrz4"

#open cam
cam = picamera.PiCamera()
filename = hardware_reg_key+".h264"
#reduce resolution and frames per second so that saved file is < 1.1Mbytes
cam.resolution = (640, 480)
#take 24 frames per second, but could be reduced for longer videos
#or for a higher resolution camera
cam.framerate = 24
#quality between 1 (best) and 40 (worst)
#this example was done on a Raspberry PI Camera Rev 1.3 (5MP, 1080p)
#you might need to further reduce quality in case of a higher camera resolution
cam.start_recording(filename,quality=30)
#record for 8 seconds
cam.wait_recording(8)
#then stop recording
cam.stop_recording()
print('video saved') #print it to console
#close cam when done to release its resourses and avoid GPU memory leak
cam.close()
time.sleep(1) #wait 1 second for save

url = 'http://www.iotsprint.com/upload.php'
out_file = hardware_reg_key+".mp4"
#convert to .mp4 but using the ffmpeg software
#point to ffmpeg software, specify input file, '-y' is to override the output file
#if it already exists, then specify output file name
#this command is resource-intensive
ffmpeg = sp.Popen(['/usr/bin/ffmpeg', '-i', filename, '-y',out_file], stdout = sp.PIPE, stderr =
sp.STDOUT)
```

```
print('waiting for conversion to complete')
time.sleep(60) #converstion takes time
#you might even need to increase it if longer or higher resolution video
#if received mp4 does not work, then increase this time

with open(out_file, 'rb') as img:
    name_img= os.path.basename(out_file)
    b = os.path.getsize(out_file) / 1000 #convert from bytes to Kbytes
    if b > 1100: #terminate script if file size > 1100 Kb
        print ("Aborting- File too large: ", b, " Kb")
        quit()

    files= {'fileToUpload': (name_img,img,'multipart/form-data',{'Expires': '0'}) }
    with requests.Session() as s:
        r = s.post(url,files=files)
        print(r.status_code)
```

5. Recording a Sound and Uploading it

In this example the Raspberry Pi will record a 10 seconds sound, using a usb-attached microphone, save it locally (in .wav format), and then upload it to the Takano platform. Registered users will receive it by email. The internet is abound with usb microphones compatible with Raspberry Pi, with different prices and quality, and the one we got our hands on for this example (https://www.sunfounder.com/usb-mini-microphone.html) is pictured below. Of course it is not meant to record sounds in high-fidelity, but rather work as a demo.

Running a microphone on the Raspberry Pi requires a bit of setting. This is explained in: http://wiki.sunfounder.cc/index.php?title=To_use_USB_mini_microphone_on_Raspbian
Launch the Python script below using any IDE (Python 3 should already be installed), get the microphone close to your month and talk. After 10 seconds, the .wav file is created in the same directory where the Python script resides. Then this .wav file will automatically be uploaded to the Takano platform, and you should receive it attached in an email.

The Python program goes below:

```
import os
import requests
import time

hardware_reg_key = "uwq4ux0wxrz4"
print("recording sound")
#run the below shell command to record 10s of audio using the attached USB microphone
os.system ("arecord -D plughw:1,0 -d 10 "+hardware_reg_key+".wav")
print("recording saved")
time.sleep(1) #wait 1 second for save

url = 'http://www.iotsprint.com/upload.php'
file_to_upload = hardware_reg_key+".wav" #file is in directory of this Python script
with open(file_to_upload, 'rb') as img:
   name_img= os.path.basename(file_to_upload)
   b = os.path.getsize(file_to_upload) / 1000 #convert from bytes to Kbytes
   if b > 1100: #terminate script if file size > 1100 Kb
      print ("Aborting- File too large: ", b, " Kb")
      quit()

   files= {'fileToUpload': (name_img,img,'multipart/form-data',{'Expires': '0'}) }
   with requests.Session() as s:
      r = s.post(url,files=files)
      print(r.status_code)
```

6. Downloading a File from Takano

In some circumstances, the user might want another hardware to download and play or display the media file that was priorly uploaded to the Takano platform. Machine to machine (M2M) automation is possible with Takano. For that to work, the hardware supposed to download the media file (considered in this example another Raspberry Pi), will have to connect to another php script on the Takano platform to authenticate the hardware registration key (which is the name of the media file to be downloaded). Only after successful authentication (and supposing that a media file has been uploaded before) will download begin.

The Python code goes below:

```
import requests
hardware_reg_key = "uwq4ux0wxrz4" #this key authenticates with the php file
#if a wrong hardware_reg_key is send then no file will be downloaded
url = "http://www.iotsprint.com/download.php?r="+hardware_reg_key
s = requests.Session()
response = s.get(url)
file_bytes =  response.content

r = requests.get(url)
```

```
string_containing_file_name_with_extension = r.headers['Content-Disposition']
print(string_containing_file_name_with_extension)
#look for file type so we can save it with the right extension
dot_pos = string_containing_file_name_with_extension.rfind(".")
file_name_extension  =  string_containing_file_name_with_extension[dot_pos:-1]

#save the file in the same directory where this Python script resides
with open(hardware_reg_key+file_name_extension, "wb") as f:
   f.write(file_bytes) #response.text
   f.close()
```

7. Contact information

For more information or comments, please do not hesitate to contact the Takano technical team by email at: info@iotsprint.com
We are available from 8:00AM (GMT+3) to 6:00PM (GMT+3).