IoT Device to ADAS system

Reports

Jay Jonggoo Kang

Contents:

1. basicPubsub

1-1. TimeStamp():
Returns date and time in the specific time zone (default: Chicago)
Ex)
"time_stamp": "2018-07-17 16:09:56"

```
def TimeStamp(timeZone = 'America/Chicago'):
        # return local time (default = Chicago)
        tzone = datetime.now(timezone(timeZone))
        dtime = tzone.strftime(ts)
        return dtime
```

1-2. KeyInfo():
Returns TimeStamp(), Camera_id, and Event (required new gps kit)
Ex)
"trip_start": "2018-07-17 16:09:56"
"camera_id": "raspberry1"
"event": " "

```
def KeyInfo(camID='raspberry'):
        # define 'message' dict and assign 'trip_start' and 'camera_id'
        message = {}
        message['trip_start'] = TimeStamp()
        message['camera_id'] = camID
        message['event'] = ' '
        return message
```

1-3. ButtonData():
Returns Boolean data if the switch is pushed

```
def ButtonData():
        # get the button value as True/False
        button = Button(25)
        button = Button(25, debounce=1.0)
        buttonPress = button.is_pressed()
        return buttonPress
```

Ex)

```
        # Button
        message['emergencyCall'] = ButtonData()
```

"emergencyCall": false

1-4. SensorData():

Returns sensor data in a list for one second

```
def SensorData(stopEvent, timeFinish, q):
        global msg # collect the sensor data
        while not stopEvent.is_set():
                while time.time() < timeFinish:

                        # assign the sensor data (Gyroscope, Accelermeter,and Magnetometer)
                        pitch, roll, yaw = sense.get_orientation().values()
                        ax, ay, az = sense.get_accelerometer_raw().values()
                        mx, my, mz = sense.get_compass_raw().values()

                        msg['gx'].append(pitch)
                        msg['gy'].append(roll)
                        msg['gz'].append(yaw)
                        msg['ax'].append(ax)
                        msg['ay'].append(ay)
                        msg['az'].append(az)
                        msg['mx'].append(mx)
                        msg['my'].append(my)
                        msg['mz'].append(mz)
                        q.put(msg)

                        if stopEvent.is_set():
                                break
                        if time.time() > timeFinish:
                                stopEvent.set()
                                #print("STOP: Stopping as you wish.")
```

EX)

"ax": [0, 0.8937194347381592, 0.8944478631019592],
"ay": [0, -0.41419023275375366, -0.4173990786075592],
"az": [0, 0.063848577439785, 0.06336117535829544],
"gz": [25.014015674682813, 25.013675872771607, 24.999001211339834],
"gy": [85.68756449650469, 85.68783770407148, 85.70231770511191],
"gx": [180.16615983317175, 180.1631408895586, 180.15678881363047]
"mx": [24.493427276611328, 24.493427276611328, 24.493427276611328],
"my": [-58.12144470214844, -58.12144470214844, -58.12144470214844],
"mz": [1.7309249639511108, 1.7309249639511108, 1.7309249639511108]

1-5. GeoFence():

Returns Boolean data if the device in in the polygon are that you set up

```
def GeoFence(location, zone):
    polygon = Polygon(zone)
    lat = float(location.split(',')[0])
    long = float(location.split(',')[1])
    point = polygon.contains(Point(lat, long))
    return point
```

EX)

"aaN_geo": true

1-6. GPSReader():

Returns “speed”, “location”, “satellites”, “hdop”, “heading”

```
def GPSReader(line):
        # assign the GPS data
        if line.find('RMC') > 0:
                data = pynmea2.parse(line)
                message['time_stamp'] = data.timestamp
                message['speed'] = data.spd_over_grnd
        message['location'] = LocaFormat(data.lat) + "," + LocaFormat(data.lon)
        if line.find('GGA') > 0:
                data = pynmea2.parse(line)
                message['satellites'] = data.num_sats
        if line.find('GSA') > 0:
                data = pynmea2.parse(line)
                #print('GPGSA_line : ', data) # check the gps data line
                message['hdop'] = data.hdop
        if line.find('GSV') > 0:
        data = pynmea2.parse(line)
                message['heading'] = data.azimuth_1
                #print('GPGSV_line : ', data) # check the gps data line
        return message
```

EX)

"speed": null, (float in Km/h)

"location": "0.0,0.0"

“satellites": "00"

“hdop”: 200

“heading” : null. (0 to 359 in degree)

1-6. JsonSchema()

Returns Error if data fails to the json schema

```
def JsonSchema(message, schema):
    try: validate(message, schema)
    except jsonschema.exceptions.ValidationError as ve:
        print("Schema ERROR: ", ve)
```

1-7. CameraVideo():

Commands to start camera-recording and to stop the recording when the trip is over

```
def CameraVideo(cdtime):
        os.system("avconv -f video4linux2 -r 10 -s 1280x960 -i /dev/video0 \
            /home/pi/aws-iot-device-sdk-python/samples/basicPubSub/Camera/Rpi_video_{}.avi".format(cdtime))
        print("Camera recording start")
```

1-8. PweroffEvent():

Returns messages of “Power off” or “Power on” when the device is off or on

```
def PweroffEvent(signal, frame):
    print("Ctrl+C received: Power Off")
    global basicInfo
    message = basicInfo.copy()
    message['event'] = "Power Off"
    print(message)

    # upload 'Power Off' message to 'S3'
    messageJson = json.dumps([message])
    myAWSIoTMQTTClient.publish(topic, messageJson, 1)
    if args.mode == 'publish':
        print('Published topic %s: %s\n' % (topic, messageJson))

    time.sleep(1)
    sys.exit(0)
```

1-9. getVideoList():

Uploads the list of video files on S3 when the subscribe topic requests

```
def getVideoList(videotopic):
    print("Uploading list")
    session = boto3.Session(aws_access_key_id = AWS_ACCESS ,aws_secret_access_key = AWS_SECRET)
    client = session.client('s3')

    directory = os.popen('pwd').read().rstrip() + '/Camera' + '/'
    filelists = [os.path.basename(x) for x in glob.glob(str(directory) + '*.avi')]
    filename = "RPiVideoList.txt"
    file = open(directory + filename, "wb")
    for f in filelists:
        file.write(f + ',')
    file.close()

    # upload the list
    client.upload_file(directory + filename, bucketName, filename)

    print('File name: %s, Bucket name: %s' %(filename, bucketName))
```

EX)

```
RPiVideoList.txt
Rpi_video_20180716_142851.avi
Rpi_video_2018-07-16.avi
Rpi_video_3.avi
Rpi_video_20180716_140337.avi
Rpi_video_20180716135917.avi
Rpi_video_2.avi
Rpi_video_20180717_121111.avi
Rpi_video_20180717_121743.avi
Rpi_video_20180716_144034.avi
Rpi_video_TEST.avi
Rpi_video_20180717_120721.avi
Rpi_video_20180717_122959.avi
Rpi_video_1.avi
Rpi_video_20180717_123443.avi
Rpi_video_20180717_122740.avi
Rpi_video_20180717_123332.avi
Rpi_video_20180717_122609.avi
Rpi_video_20180717_123604.avi
Rpi_video_%Y%m%d_%H%M%S.avi
```

1-10. uploadVideo()

Uploads the selected video file on S3 as requested.

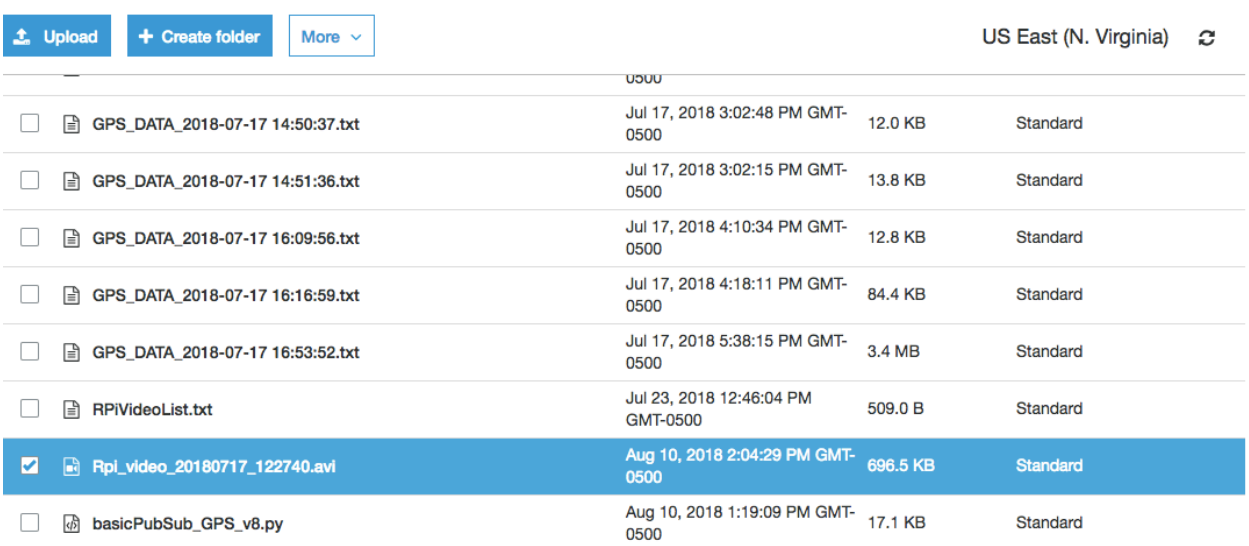
```python
def uploadVideo(videotopic, videoFile):
    print("Uploading video file: ", videoFile)
    session = boto3.Session(aws_access_key_id = AWS_ACCESS ,aws_secret_access_key = AWS_SECRET)
    client = session.client('s3')

    directory = os.popen('pwd').read().rstrip() + '/Camera' + '/'
    filenames = [os.path.basename(x) for x in glob.glob(str(directory) + '*{}*.avi'.format(videoFile))]

    for f in filenames:
        client.upload_file(directory+f, bucketName, f)

        print('File name: %s, Bucket name: %s' %(f,bucketName))
```

EX)



2. basicShadow

```
def customShadowCallback_Update(payload, responseStatus, token):
    # payload is a JSON string ready to be parsed using json.loads(...)
    # in both Py2.x and Py3.x
    if responseStatus == "timeout":
        print("Update request " + token + " time out!")
    if responseStatus == "accepted":
        payloadDict = json.loads(payload)
        print("~~~~~~~~~~~~~~~~~~~~~~~~")
        print("Update request with token: " + token + " accepted!")
        print("temperature: " + str(payloadDict["state"]["reported"]["temperature"]))
        print("~~~~~~~~~~~~~~~~~~~~~~~~\n\n")
    if responseStatus == "rejected":
        print("Update request " + token + " rejected!")

# Loop forever
import random
updated = False
while True:
    if not updated:
        updated = True
        ######## updated callbac function ########
        JSONPayload = '{"state":{"reported":{"temperature": "%d"}}}' % random.randint(10,40)
        deviceShadowHandler.shadowUpdate(JSONPayload, customShadowCallback_Update, 5)

    time.sleep(1)
```

II. After commanding, AWS subscribes the outputs in three different directories.

II-1. $aws/things/Bot/shadow/update

: this shows the outputs that has been updated

$aws/things/Bot/shadow/update Jun 12, 2018 2:59:28 PM -0500 Export Hide


```
{
  "state": {
    "reported": {
      "temperature": "19"
    }
  },
  "clientToken": "c3aa24b7-f0c9-4852-bac8-a7fdc6cc8243"
}
```


II-2. $aws/things/Bot/shadow/documents

: this shows that the two different outputs that before updating and after updating

$aws/things/Bot/shadow/update/documents Jun 12, 2018 2:59:28 PM -0500 Export Hide


```
{
  "previous": {
    "state": {
      "reported": {
        "temperature": "18"
      }
    },
    "metadata": {
      "reported": {
        "temperature": {
          "timestamp": 1528833271
        }
      }
    },
    "version": 7
  },
  "current": {
    "state": {
      "reported": {
        "temperature": "19"
      }
    },
    "metadata": {
      "reported": {
        "temperature": {
          "timestamp": 1528833568
        }
      }
    },
    "version": 8
  },
  "timestamp": 1528833568,
  "clientToken": "c3aa24b7-f0c9-4852-bac8-a7fdc6cc8243"
}
```

II-3. $aws/things/Bot/shadow/accepted

: this shows the outputs that finally updated

$aws/things/Bot/shadow/update/accepted Jun 12, 2018 2:59:28 PM -0500 Export Hide


```
{
  "state": {
    "reported": {
      "temperature": "19"
    }
  },
  "metadata": {
    "reported": {
      "temperature": {
        "timestamp": 1528833568
      }
    }
  },
  "version": 8,
  "timestamp": 1528833568,
  "clientToken": "c3aa24b7-f0c9-4852-bac8-a7fdc6cc8243"
}
```


III. Change the setting on AWS

III-1. Original setting

Shadow Document

Delete **Edit**

Last update: Jun 12, 2018 2:54:58 PM -0500

Shadow state:


```
1  {
2    "desired": {
3      "volume": 3,
4      "track": 4,
5      "blinkColor": "green",
6      "Color": "red",
7      "temp_conversion": 2,
8      "property": "prope"
9    },
10   "delta": {
11     "volume": 3,
12     "track": 4,
13     "blinkColor": "green",
14     "Color": "red",
15     "temp_conversion": 2,
16     "property": "prope"
17   }
18 }
```

III-2. Changing the setting: "temp_conversion": 2 -> 6)

Shadow Document

Delete **Edit**

Last update: Jun 12, 2018 3:02:41 PM -0500

Shadow state:


```
{
  "desired": {
    "volume": 3,
    "track": 4,
    "blinkColor": "green",
    "Color": "red",
    "temp_conversion": 6,
    "property": "prope"
  },
  "delta": {
    "volume": 3,
    "track": 4,
    "blinkColor": "green",
    "Color": "red",
    "temp_conversion": 6,
    "property": "prope"
  }
}
```


IV. After changing the settings, the new outputs were subscribed in three different directories: *Three different subscripts are shown.*

IV-1. $ aws/thigs/RasberryPi_1/shadow/update/accepted

: this shows updated setting status (*"temp_conversion" : 6*)

$aws/things/RasberryPi_1/shadow/update/acc... Jun 12, 2018 3:02:41 PM -0500 Export Hide


```
{
  "state": {
    "desired": {
      "volume": 3,
      "track": 4,
      "blinkColor": "green",
      "Color": "red",
      "temp_conversion": 6,
      "property": "prope"
    }
  },
  "metadata": {
    "desired": {
      "volume": {
        "timestamp": 1528833761
      },
      "track": {
        "timestamp": 1528833761
      },
      "blinkColor": {
        "timestamp": 1528833761
      },
      "Color": {
        "timestamp": 1528833761
      "Color": {
        "timestamp": 1528833761
      },
      "temp_conversion": {
        "timestamp": 1528833761
      },
      "property": {
        "timestamp": 1528833761
      }
    }
  },
  "version": 17,
  "timestamp": 1528833761
}
```

IV-2. $ aws/thigs/RasberryPi_1/shadow/update/delta
: this shows the output that has been changed

$aws/things/RasberryPi_1/shadow/update/delta Jun 12, 2018 3:02:41 PM -0500 Export Hide


```
{
  "version": 17,
  "timestamp": 1528833761,
  "state": {
    "volume": 3,
    "track": 4,
    "blinkColor": "green",
    "Color": "red",
    "temp_conversion": 6,
    "property": "prope"
  },
  "metadata": {
    "volume": {
      "timestamp": 1528833761
    },
    "track": {
      "timestamp": 1528833761
    },
    "blinkColor": {
      "timestamp": 1528833761
    },
    "Color": {
      "timestamp": 1528833761
    },
    "temp_conversion": {
      "timestamp": 1528833761
    },
    "property": {
      "timestamp": 1528833761
    }
  }
}
```

3. Json Schema

```
{"required":["camera_id", "ax", "ay", "az", "gx", "gy", "gz", "mx", "my", "mz",
                        "event", "satellites", "speed", "time_stamp", "altitude", "trip_start",
                        "distance", "emergencyCall", "gps_qual"],
            "type": "object",

            "properties": {
                "ax":{
                    "type": "array",
                    "itmes":{
                        "type": "number"
                    }
                },
                "ay":{
                    "type": "array",
                    "itmes":{
                        "type": "number"

                    }
                },
                "az":{
                    "type": "array",
                    "itmes":{
                        "type": "number"

                    }
                },
                "gx":{
                    "type": "array",
                    "itmes":{
                        "type": "number"
                    }
                },
                "gy":{
                    "type": "array",
                    "itmes":{
                        "type": "number"
```

```
                }
            },
            "gz":{
                "type": "array",
                "itmes":{
                    "type": "number"

                }
            },
            "mx":{
                "type": "array",
                "itmes":{
                    "type": "number"
                }
            },
            "my":{
                "type": "array",
                "itmes":{
                    "type": "number"

                }
            },
            "mz":{
                "type": "array",
                "itmes":{
                    "type": "number"

                }
            },
            "location":{
                "type": "string",
            },
            "satellites":{
                "type": "string",
            },
            "speed":{
                "type": "number",
            },
            "time_stamp":{
                "type": "string",
            },
            "altitude":{
                "type": "number",
            },
            "trip_start":{
                "type": "string",
            },
            "distance":{
                "type": "number",
            },
            "emergencyCall":{
                "type": "boolean",
            },
            "gps_qual":{
                "type": "number",
            }
        },
        "$schema": "http://json-schema.org/draft-07/schema#",
        "title": "CarVi Data",
        "type": "object"
}
```

4. Job OTA

4-1. DownloadFile():

Download the file from a S3 bucket

```
def DownloadFile(directory, KEY, BUCKET_NAME, FILE_NAME):
    if directory: os.chdir(directory)
    else: directory = os.popen('pwd').read().rstrip() + '/'

    try:
        s3.Bucket(BUCKET_NAME).download_file(KEY, FILE_NAME)
    jobsMsgProc.processJobs()
    SenseLight()
    print("++++++++++++++++++++++++++++++++++++++++++++++++++")
        print("File downloaded")
        print("File: %s" %(directory+FILE_NAME))
    print("++++++++++++++++++++++++++++++++++++++++++++++++++")
    except botocore.exceptions.ClientError as e:
        if e.response['Error']['Code'] == "404":
            print("The object does not exist.")
        else:
            raise
```

4-2. GetJobInfo():

Returns job information and job document

```
def GetJobInfo(JOBID):
        # get the job and job document information
        info_job = iot.describe_job(jobId=JOBID) # job info
        info_job_doc = iot.get_job_document(jobId=JOBID)
        job_doc_dict = ast.literal_eval(info_job_doc['document']) # job doc info

        print("============================Job Information============================")
        print("Job name: %s" %JOBID)
        print("Target: %s" %TARGET)
        print("Status: %s" %info_job['job']['status'])
        print("Document Sourse: \n%s" %info_job['documentSource'])
        print("Document:")
        print(job_doc_dict)

        return info_job, job_doc_dict
```

4-3. MainFunction()

Returns job status and version only if in the case of “In Progress”.
Returns job status of Cancel, Complete, and Succeeded.

```
if info_job['job']['status'] != 'IN_PROGRESS':
    jobsMsgProc = JobsMessageProcessor(jobsClient, clientId)
    print("Job is NOT 'IN PROGRESS'")
    print("Current Status: {}".format(info_job['job']['status']))
    print(" ")
else:
#   VersionUpdate(directory, CURR_IMG_FILE)
    print(' ')
    print('Starting to process jobs................')
    print(' ')

    if NEW_ver > CURR_ver :
        print("New Version available!")
        print(' ')

        jobsMsgProc = JobsMessageProcessor(jobsClient, clientId)
        DownloadFile(directory, IMG_KEY, IMG_BUCKET_NAME, 'rasp_img_{}.txt'.format(NEW_ver))
#       jobsMsgProc.completedjob()
#       print(jobsMsgProc.isDone())
        while not jobsMsgProc.isDone():
            print('IN PROCESS...')
            time.sleep(1)

    elif NEW_ver == CURR_ver :
        print("Latest Version!")
        jobsMsgProc = JobsMessageProcessor(jobsClient, clientId)

print(' ')
print('*****************************************************************')
print('Done processing jobs')
print('Stats: ' + json.dumps(jobsMsgProc.getStats()))
print('*****************************************************************')
jobsClient.disconnect()
sensor.clear()
```

5. Trip Analysis

Name	Date Modified	Size
180621_SF.ipynb	Jun 26, 2018 at 10:31 AM	95 KB
Raspberry.ipynb	Jun 26, 2018 at 6:20 PM	7 KB
180622_CHI.ipynb	Jun 27, 2018 at 3:41 PM	31 KB
jsonschema_scratch.ipynb	Jun 27, 2018 at 10:21 PM	4 KB
rasp_scratch.ipynb	Jun 28, 2018 at 2:38 AM	3 KB
CarVi_library.ipynb	Jun 28, 2018 at 11:56 AM	2 KB
180622_SF_standard.ipynb	Jul 1, 2018 at 6:15 PM	36 KB
180629_simonTrip.ipynb	Jul 1, 2018 at 11:05 PM	41 KB
gmaps analysis.ipynb	Jul 2, 2018 at 3:37 PM	19 KB
190702_SFtest.ipynb	Jul 2, 2018 at 4:20 PM	24 KB
s3_new.ipynb	Jul 2, 2018 at 4:57 PM	29 KB
Test_S3.html	Jul 2, 2018 at 5:41 PM	304 KB
TEST_redshift.pdf	Jul 2, 2018 at 5:44 PM	694 KB
Test_S3.pdf	Jul 2, 2018 at 5:45 PM	621 KB
TEST_redshift.html	Jul 2, 2018 at 11:27 PM	324 KB
Untitled.ipynb	Jul 3, 2018 at 4:14 PM	6 KB
CarVi Data Analysis Template.ipynb	Jul 3, 2018 at 4:17 PM	34 KB
get_distance.ipynb	Jul 3, 2018 at 10:29 PM	3 KB
DeadReckoning_scratch.ipynb	Jul 5, 2018 at 11:06 PM	5 KB
TEST_redshift.ipynb	Jul 6, 2018 at 11:42 AM	32 KB
Dead Reckoning.ipynb	Jul 8, 2018 at 9:53 PM	180 KB
PCA_scratch.ipynb	Jul 8, 2018 at 10:10 PM	43 KB
Tripby30.ipynb	Jul 9, 2018 at 1:19 AM	55 KB
tripby15sec_(approved).ipynb	Jul 10, 2018 at 11:22 AM	30 KB
Test_S3.ipynb	Jul 10, 2018 at 1:10 PM	28 KB
GPS_DATA_2018-07-07 19_33_55.json	Jul 10, 2018 at 2:45 PM	68 KB
zero_distance_(approved).ipynb	Jul 11, 2018 at 10:56 AM	42 KB
RaspberryPi Trip Analysis.ipynb	Jul 18, 2018 at 4:07 PM	150 KB
NoGPS_scratch.ipynb	Jul 18, 2018 at 11:37 PM	38 KB
Untitled1.ipynb	Jul 19, 2018 at 2:52 PM	555 bytes
geofensing_scratch.ipynb	Jul 19, 2018 at 6:43 PM	23 KB
speed_limit.ipynb	Jul 20, 2018 at 11:59 AM	15 KB
postman script.json	Jul 20, 2018 at 2:59 PM	20 KB
API_Cron.ipynb	Jul 20, 2018 at 4:25 PM	9 KB