**Sending and Receiving Pictures From a Raspberry Pi via MQTT**

**This will show how to use the built-in camera module on a Raspberry Pi to**

**Take a picture on the Raspberry Pi**

The Pi camera can be controlled using a Python script:

import picamera  
from time import sleep  
  
camera = picamera.PiCamera()  
  
try:   
 camera.start\_preview()  
 sleep(1)  
 camera.capture('image\_test.jpg', resize=(500,281))  
 camera.stop\_preview()  
 pass  
finally:  
 camera.close()

 The purpose of starting the camera preview and the sleep function is to allow time for the camera to warm up, which gives a better image.  The resize of the image greatly reduces its filesize, reducing the time it takes to send and receive the image.  In this example, a 500×281 image is sufficient, but if desired the full 5MP of the Raspberry Pi camera can be used.

**Convert the image to Base64 and publish**

Once the image is taken, encode the image as Base64, then split it into segments to publish.

Converting to Base64:

import base64  
  
def convertImageToBase64():  
 with open("image\_test.jpg", "rb") as image\_file:  
 encoded = base64.b64encode(image\_file.read())  
 return encoded

In this example we will be publishing to the IBM Watson IoT Platform, using our specific device ID and credentials.

 import ibmiotf.device  
  
options = ibmiotf.device.ParseConfigFile("/home/pi/device2.cfg")  
client = ibmiotf.device.Client(options)  
client.connect()

When sending the image, we add some additional fields in order to identify it and make the reconstruction process easier.  We will make use of this function that generates a random string to be used as the picture ID.

import random, string  
  
def randomword(length):  
 return ''.join(random.choice(string.lowercase) for i in range(length))

We then split the data into chunks of size 3000, append some identifying information, then publish.

import math  
  
packet\_size=3000  
  
def publishEncodedImage(encoded):  
   
 end = packet\_size  
 start = 0  
 length = len(encoded)  
 picId = randomword(8)  
 pos = 0  
 no\_of\_packets = math.ceil(length/packet\_size)  
  
   
 while start <= len(encoded):  
 data = {"data": encoded[start:end], "pic\_id":picId, "pos": pos, "size": no\_of\_packets}  
 client.publishEvent("Image-Data",json.JSONEncoder().encode(data))  
 end += packet\_size  
 start += packet\_size  
 pos = pos +1

**Receive and reconstruct image**

In this example, the image will be displayed on a web page in the browser.  A Javascript MQTT client subscribes and receives the image data.  When the data comes in, the image is reconstructed by putting together the chunks in the correct order using the following function.  The image labeled “picture\_to\_show” will show the reconstructed image.

function reconstructBase64String(chunk) {  
 pChunk = JSON.parse(chunk["d"]);  
  
 //creates a new picture object if receiving a new picture, else adds incoming strings to an existing picture   
 if (pictures[pChunk["pic\_id"]]==null) {  
 pictures[pChunk["pic\_id"]] = {"count":0, "total":pChunk["size"], pieces: {}, "pic\_id": pChunk["pic\_id"]};  
  
 pictures[pChunk["pic\_id"]].pieces[pChunk["pos"]] = pChunk["data"];  
  
 }  
  
 else {  
 pictures[pChunk["pic\_id"]].pieces[pChunk["pos"]] = pChunk["data"];  
 pictures[pChunk["pic\_id"]].count += 1;  
  
  
 if (pictures[pChunk["pic\_id"]].count == pictures[pChunk["pic\_id"]].total) {  
 console.log("Image reception compelete");  
 var str\_image="";   
  
 for (var i = 0; i <= pictures[pChunk["pic\_id"]].total; i++)   
 str\_image = str\_image + pictures[pChunk["pic\_id"]].pieces[i];  
  
 //displays image  
 var source = 'data:image/jpeg;base64,'+str\_image;  
 var myImageElement = document.getElementById("picture\_to\_show");  
 myImageElement.href = source;  
 }  
  
 }  
  
}