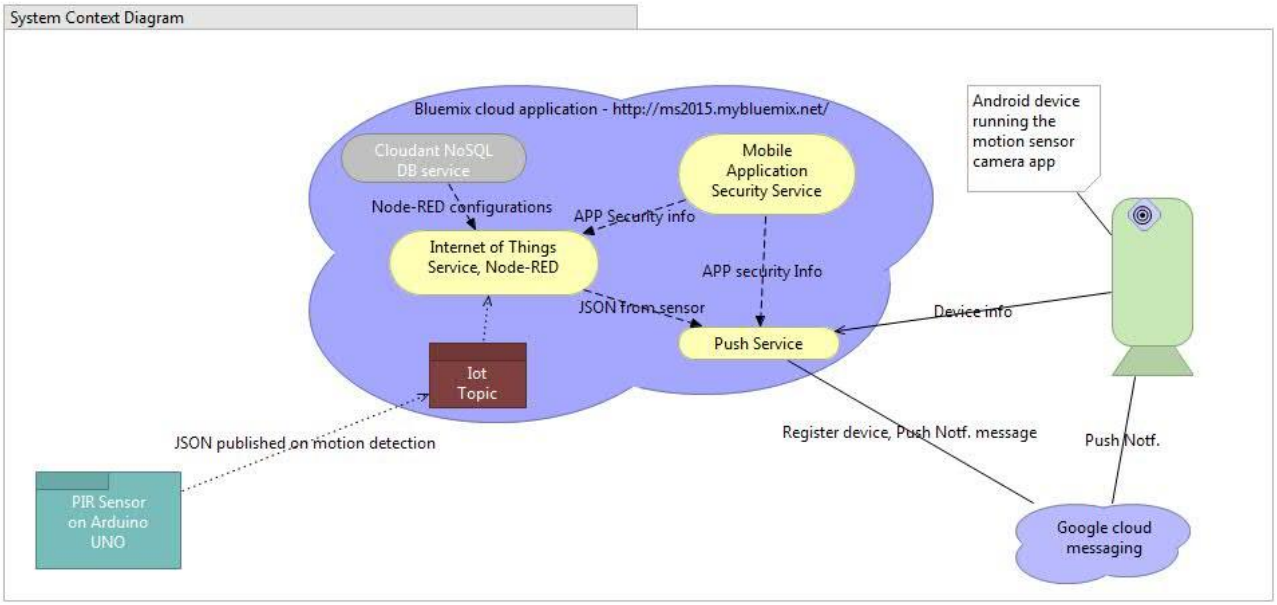
Build a motion sensor android camera with Arduino UNO, PIR sensor, Bluemix Iot foundation and Push services

There is this garden bird feeder hanging outside my window and whoever (birds, squirrels, rodents!) visits it does it in my absence. Always! How I wished I could in some way know my visitors and that led to the creation of this fun project.

The Iot app I am going to explain in the next few pages was primarily developed to capture photographs of birds visiting the garden bird feeder. The solution comprises of a PIR sensor which detects any motion when birds land on the feeder, sends a push notification to an android app to activate the phone's camera shutter and click an image of the bird. The photograph of the bird is stored in the same phone. We could however enhance the app and extend its usage in other ways such as a security camera.

The birds eye view



There are 3 parts to the Iot App

1. The motion sensor
2. The backend application on Bluemix cloud platform
3. The Android camera app

Throughout the following sections, I will be referring to the source code files made available at the following github location:

https://github.com/evanjas77/MotionSensorCamera

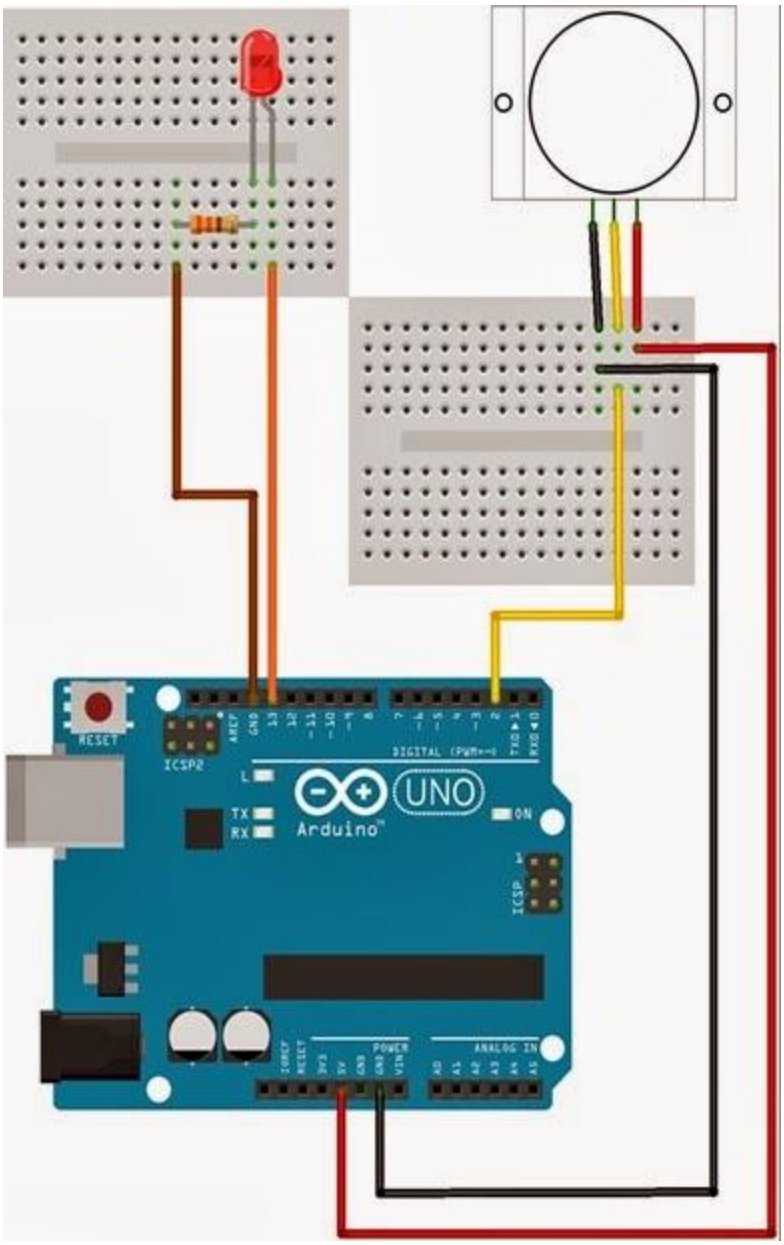
The motion sensor

Things you will need to build the Iot based motion sensor:

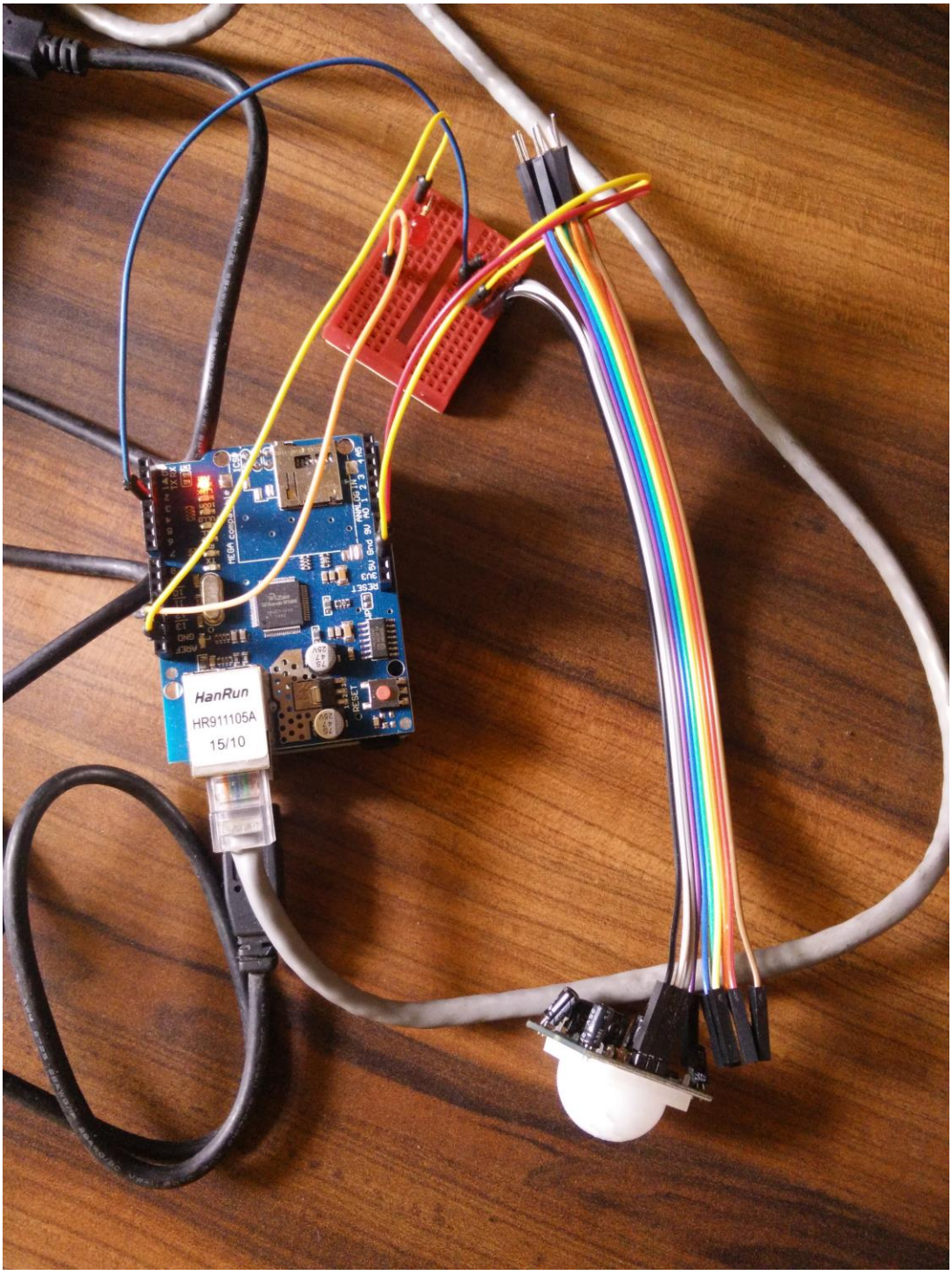
1. Arduino UNO (I used the R2 version)

2. Arduino Ethernet/WiFi shield
3. A PIR motion sensor (HC-SR501)
4. An LED and 330 ohm resistor
5. Mini breadboard
6. Male-female and female-female breadboard jumper wires
7. USB cable type A/B
8. Ethernet cable
9. Arduino 1.6.x sketch IDE

Circuit sketch without Ethernet shield



Before you proceed to introduce the Ethernet/WiFi shield, please use the “File>Examples>01. Basics>Blink” sketch and PIR sensor sketch (http://playground.arduino.cc/Code/PIRsense) to test the board and the sensor. Once the Ethernet shield is introduced, test the shield using “Examples>Ethernet>Web Server” sketch. You will need to update the IP address and MAC address in the Web server sketch before uploading it to Arduino. Once Ethernet shield is introduced, pin 2 is used by the shield and is unavailable. Hence shift the output of PIR sensor to pin 3. Here is my version of the circuit:



Upload the PIRSensor_Arduino_MQTT_Bluemix.ino sketch (from here) to the device. Let's go over the sketch and see how we can get it to work.

The following set of declarations in the sketch need modification so that the device gets a unique identity in the network and post the motion detection information to the cloud.

```
// Update these with values suitable for your network.
byte mac[]    = {  0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xEF};   ---- If mac address is not available at the back of the shield. make up your own.
String macString = "deedbafefeef";
char servername[] = "[redacted].messaging.internetofthings.ibmcloud.com";   ---- specify the device id. If you need to perform a quick test, connect to the quicks
                                                                                 DeviceId is configured in the Bluemix Iot service to be explained later.
//char servername[] = "messaging.quickstart.internetofthings.ibmcloud.com";
byte ip[]     = { 192, 168, 1, 49 };   ---- Assign an unused local ip address.
String clientName = String("d:[redacted]:") + macString;   ---- d:<deviceId>:<deviceType> - Device type is also configured in the Bluemix Iot s
//String clientName = String("d:quickstart:arduino:") + macstr;
String topicName = String("iot-2/evt/status/fmt/json");
String usernameStr = "use-token-auth";
String passwordStr = "[redacted]";   ---- password received when device is configured in Bluemix IoT service
```

Most of the values to be filled in the above code are dependent on the Bluemix Iot service configuration which we will take a look in detail in section 2. For now let us proceed with the Arduino sketch to understand how the data read from pin 3 (output of PIR sensor) is transformed into a JSON object and published to the topic “iot-2/evt/<event-id>/fmt/<data-format>”.

The setup() method code initializes the Ethernet shield and serial monitor, assigns pin 3 as INPUT pin (since this is the pin at which we receive the PIR sensor output) and pin 13 as output pin (this is where the LED is connected which glows ON when motion is detected).

The code in loop() method attempts to connect to the Iot foundation using the information provided in the clientStr, username and password. In order to get hold of these values, we will need to configure the Iot service in bluemix (to be taken up in section 2).

```
client.connect(clientStr,username,password);
```

The loop() method further constructs the JSON object when the PIR pin reads a HIGH (indicating that a motion is detected) and publishes it to the Iot topic “iot-2/evt/<event-id>/fmt/<data-format>” where event-id=”status” and data-format=”json”.

```
if (client.connected() ) {
  pubresult = client.publish(topicStr,jsonStr);
}
```

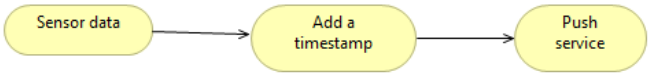
This is how the JSON object looks like

```
{ "d": { "myName": "Arduino PIRSensor", "motionDetectedAt": "51s"}}
```

Now let us move on to Bluemix cloud platform where the above JSON is received by the backend application on the cloud and see how the information is processed further.

The backend application on the Bluemix cloud platform

The app created on Bluemix platform is initially created using “Internet of Things Foundation Starter” Boilerplate. This allows us to use Node-RED flow editor which enables wiring of data flowing to/from IoT devices with ease. In the app we are currently building, we need to find a way to use the motion sensor data published by Arduino and direct it to the android device which clicks the photograph. Using Node-RED, we can create a simple flow to achieve this feature



Each of the nodes in the above flow have to be supported by services configured in Bluemix and supplied with appropriate security level

information in order to connect to respective systems to read/write the data. Let us build the application and look at what services can support the above Node-RED flow.

Things you will need:

1. A Bluemix account

2. A gmail account and a GCM sender ID and API Key from Google.

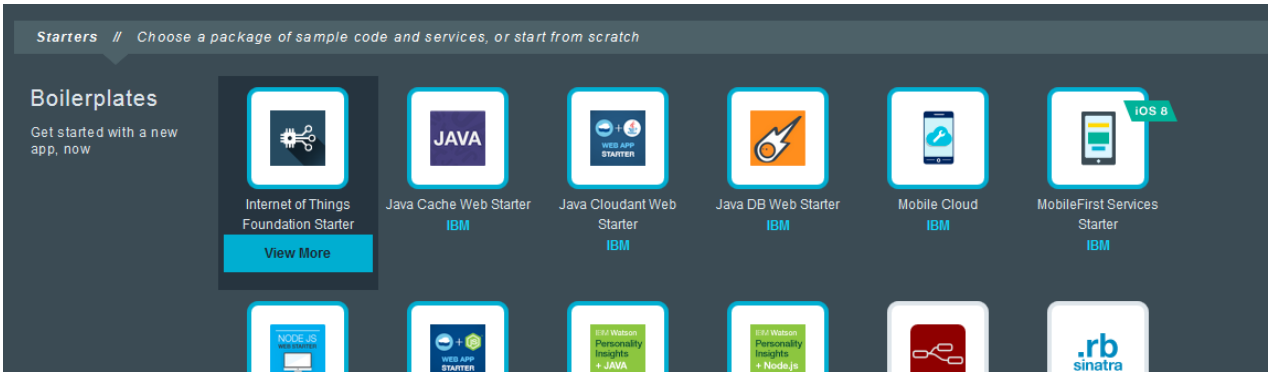
Please refer to the system context diagram to understand where GCM fits in the complete solution.

Get the GCM sender ID and API Key from Google:

1. Open the Google Developers Console.
2. Click CREATE PROJECT, enter a Project name and click Create.
3. Note down the Project Number from the top of the page. This is your GCM Sender Id (Google API Project Number).
4. Click APIs & auth on the left of the page.
5. Turn ON Google Cloud Messaging for Android.
6. Under APIs & auth, click Credentials.
7. Click CREATE NEW KEY under the Public API access section.
8. Click Server key.
9. Click Create.
10. Note down the API key from the Public API access section. This is your Sender Auth Token (GCM API Key).

Create a new Bluemix application

Select Catalog>Boilerplates>Internet of Things Foundation Starter, provide a unique name for the app (Eg: ms2015) and click “Create”



Navigate back to “Dashboard” and observe the “Services” section of the application you just created. There is a “Cloudant NoSQL DB” shared service which would have got automatically added to your app. We will not be explicitly using this DB service for the app we are building currently, but this service will be used by node-RED for its own storage requirements. Now let us add a few services to our app which are required for our design.

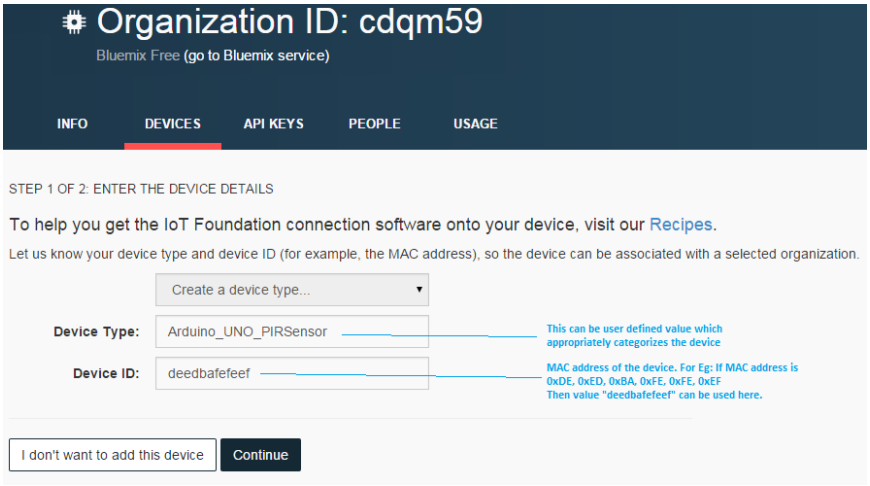
1. Internet of things service -

This service will allow us to add device information, generate api keys and add person information which will help the Iot devices (Arduino in this case) to uniquely identify themselves and securely connect to the Iot topic (iot-2/evt/<event-id>/fmt/<data-format>) to publish the sensor data using MQTT protocol. Let us first configure the service and I will follow-up with an example of how the client device can identify itself and connect to the topic provided by the above service

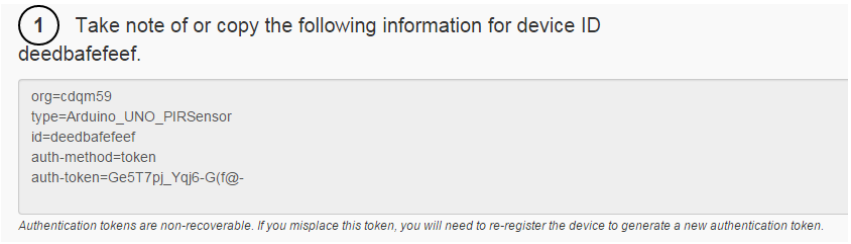
Navigate to Catalog>Internet of Things section and add the “Internet of things” service. Bind it to ms2015 app and click “Create”. Click “Restage” if asked for restaging the application. Navigate back to “Dashboard” and observe the service added to your app under “services” section. You should be able to see the “Internet of Things” service. Click on the service. This will navigate to a page where you will find the “Launch dashboard” button to add your sensor device information. Click on the button and you will land in a page with url https://<Org_ID>.internetofthings.ibmcloud.com/dashboard/#/info

For Example: https://cdqm59.internetofthings.ibmcloud.com/dashboard/#/info where OrganizationID = cdqm59

Navigate to the DEVICES tab and click on “Add Device”. Fill the details illustrated in the screenshot below and click “Continue”.



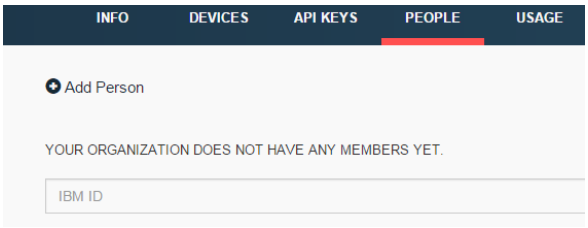
You will receive some device information which needs to be copied and stored elsewhere since the data is non-recoverable past this page.



Now you have got all you need to connect the Arduino device to Bluemix. Before we move on to the next service lets quickly go back and change our Arduino sketch (PIRSensor_Arduino_MQTT_Bluemix.ino) to update the above device information. This is how it looks now.

```
// Update these with values suitable for your network.
byte mac[]    = {  0xDE, 0xED, 0xBA, 0xFE, 0xFE, 0xEF};
String macString = "deedbafefeef";
char servername[] = "cdqm59.messaging.internetofthings.ibmcloud.com";
//char servername[] = "messaging.quickstart.internetofthings.ibmcloud.com";
byte ip[]     = { 192, 168, 1, 49 };
String clientName = String("d:cdqm59:Arduino_UNO_PIRSensor:") + macString;
//String clientName = String("d:quickstart:arduino:") + macstr;
String topicName = String("iot-2/evt/status/fmt/json");
String usernameStr = "use-token-auth";
String passwordStr = "Ge5T7pj_Yqj6-G(f@-";
```

Compile and upload the sketch to Arduino. We are just a few steps away from testing the device integration with Bluemix ☺ Oh! And before you proceed, do not forget to add you IBM ID into the “People” tab in the current page



2. Mobile Application Security Service

Navigate back to Bluemix dashboard and click on to Catalog>Mobile section and add the “Mobile Application Security” service. Bind it to ms2015 app and click “Create”. Click “Restage” if asked for restaging the application. Navigate back to “Dashboard” and observe the service added to your app under “services” section. You should be able to see the “Mobile Application Security” service. Click on the service. There will be an “Application ID” and “Application Secret” displayed in the page. Make a note of them since they will be useful in the Android app which I will detail in the next section. Ensure that you uncheck the “Enable google sign-in” checkbox and save.

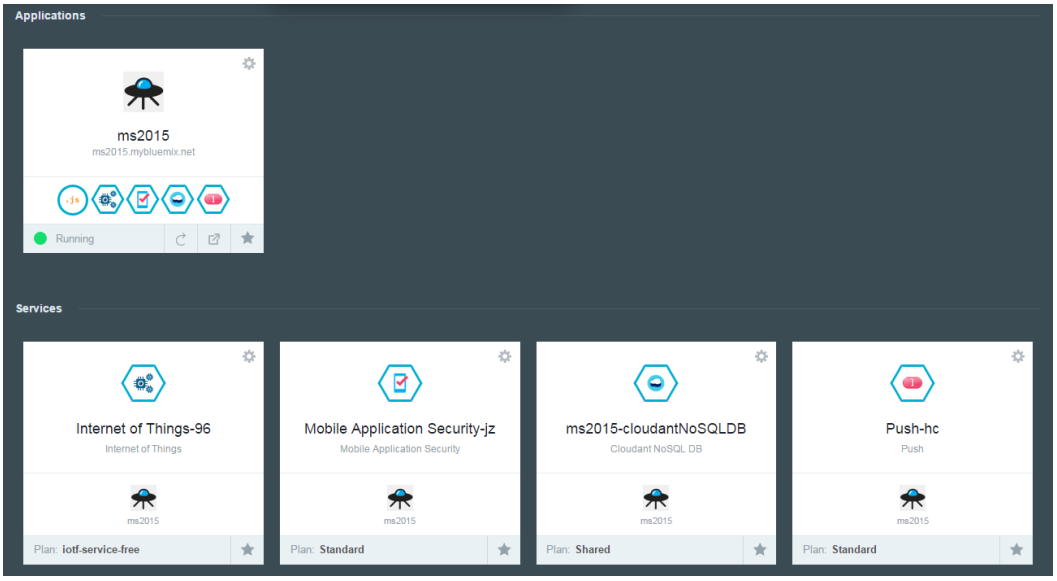
This service enables authentication to our cloud application “ms2015”. Mobile apps interacting with the cloud service need to know the application secret for any operations to be performed. In case of our application, we need this service for the android devices to be able to register themselves and receive Push notifications, which we will see in detail in the immediate next section.

3. Push Service

This service is responsible for sending push notifications to the registered apps, in our case “the Android camera app” using the GCM (Google Cloud Messaging) server. To add this service, navigate to Bluemix dashboard and click on to Catalog>Mobile section and add the “Push” service. Bind it to ms2015 app and click “Create”. Click “Restage” if asked for restaging the application. Navigate back to “Dashboard” and observe the service added to your app under “services” section. You should be able to see the “Push” service. Navigate to the service and under the “Configuration” tab, click “Edit” under the “Google Cloud Messaging” section. Enter the “GCM API Key” and “Google API Project Number” which you obtained in this section under both “Sandbox Configuration” and “Production Configuration” sections. Next click on “Tags” tab and add a NEW TAG “IoT-camera” and save it. This is the tag that the Android app (to be detailed in the next section) will subscribe to in order to receive the push notification.

The “Notification” tab can be used to send a test push notification message to registered devices and the “Registrations” tab lists all the devices which have registered for push notifications.

4. Finally…
   This is how our Bluemix application page looks like

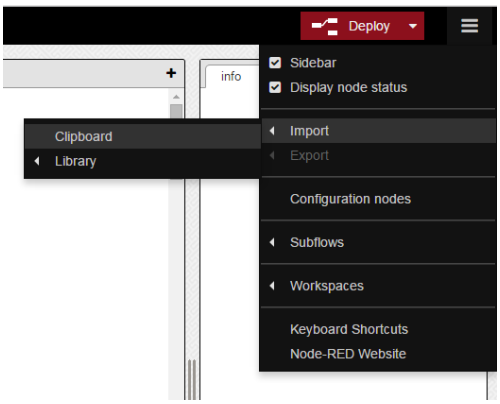


The services configured above will support the application flow which we will be creating using Node-RED flow editor in the next section.

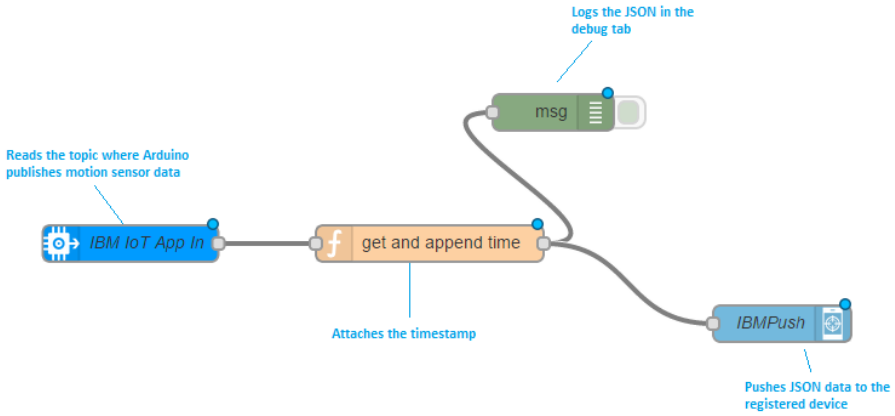
Create the Node-RED flow

Navigate to the application url http://ms2015.mybluemix.net/. The page displayed is the landing page of Node-RED flow editor. This tool will help create our application flow. Click on “Go to your Node-RED flow editor”. You will be navigated to http://ms2015.mybluemix.net/red/. This page displays a default flow for the temperature sensor. Press CTRL-A and Delete Keys to remove the default flow. Click on “Deploy” button in the top right corner of the editor. Observe the “Successfully Deployed” message popping up in green in the top middle of the screen. Now we have a clean slate to work on.

We will begin by importing the flow which I have already created for this project. Copy contents of the file “ArduinoPIRSensor_To_Android_Flow.json” (from here) to clipboard. In the Node-RED flow editor, click on the hamburger icon in the top right corner>Import>Clipboard.

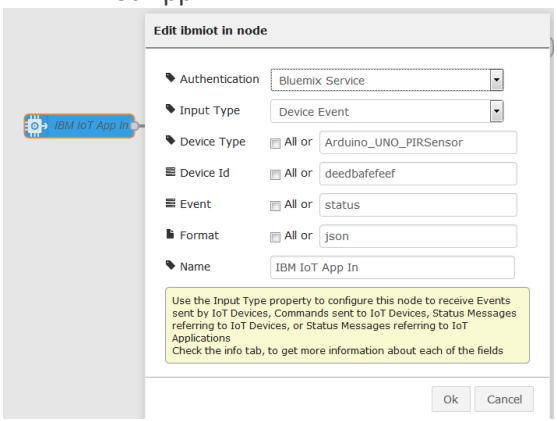


Paste the contents of the clipboard copied earlier and click “ok”. The flow nodes will be dangling on to your mouse pointer, just click them in place on the editor. Here is what we get:

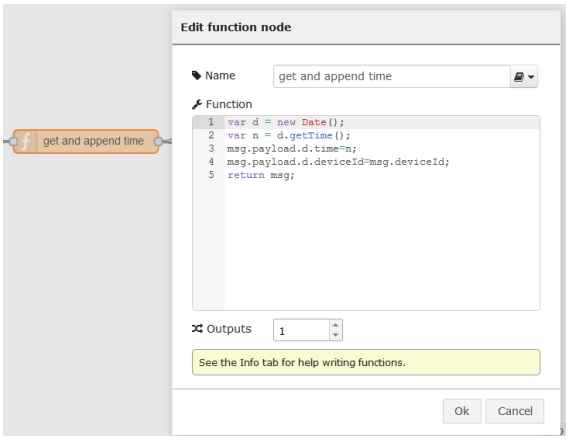


Let’s get into the configurations for each of the above nodes.

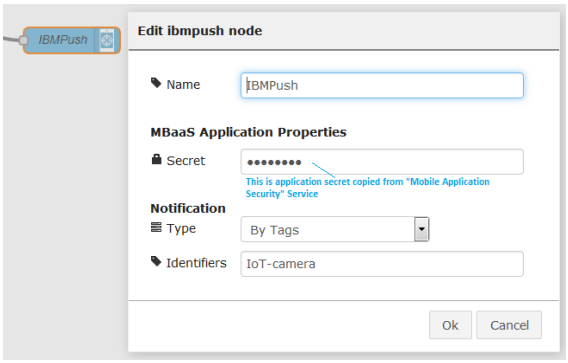
1. IBM Iot App In



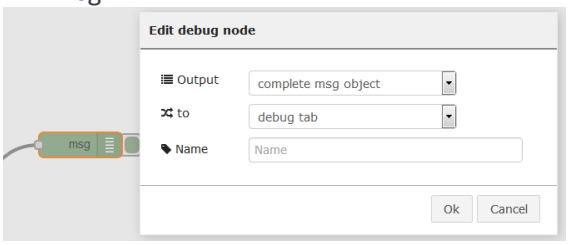
2. Get and append time



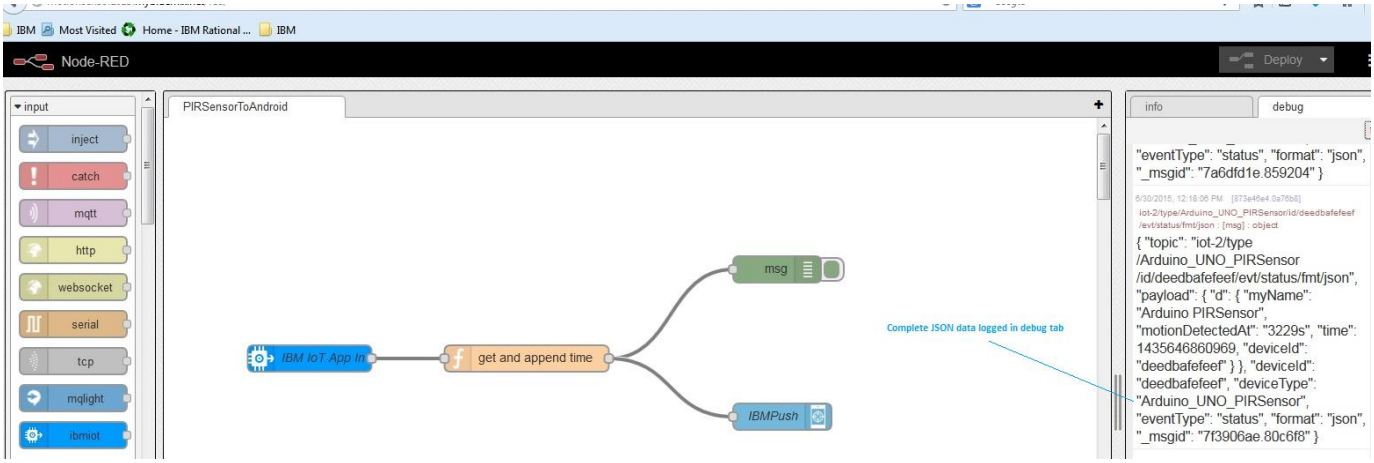
3. IBMPush



4. Msg



Click on “Deploy” button in the top right corner of the editor. Observe the “Successfully Deployed” message popping up in green in the top middle of the screen. Once the device starts publishing messages on motion detection, you will be able to view the JSON data in the debug tab



The Android camera app

The android application is a pretty simple app which performs the following operations

1. Reads the Bluelist.properties. The file can be referred from this location.

   applicationID=APP_ID // This is the Application ID obtained from “Mobile Application Security” service
   applicationSecret=APP_SECRET // This is the Application secret obtained from “Mobile Application Security” service
   applicationRoute=APP_ROUTE //Bluemix application url “ms2015.mybluemix.net”
   deviceAlias=DEVICE_ALIAS //This can be any number which uniquely identifies your Android device. For Eg: IMEI number, MAC address etc
   consumerId=CONSUMER_ID //Preferred value is the email id

2. Registers the device with the Push Notification service.

   Code snippet from MotionSensorCamera.java. The file can be referred from this location.

```
IBMBluemix.initialize(this, props.getProperty(APP_ID),
props.getProperty(APP_SECRET), props.getProperty(APP_ROUTE));
IBMPush push = IBMPush.initializeService();
push.register(props.getProperty(DEVICE_ALIAS),
props.getProperty(CONSUMER_ID)).continueWith({});
```

3. Listens for Push Notifications. Initializes the camera preview and takes a picture using the camera api whenever a Push Notification is received.

   Code snippet from MotionSensorCamera.java. The file can be referred from this location.

```
      IBMPushNotificationListener  notificationListener = new
IBMPushNotificationListener() {
         public void onReceive(final IBMSimplePushNotification
message) {
```

```
                //TODO: Important - Interpret the incoming JSON and
then initiate the camera shutter.
                /*************************INCOMING
JSON****************************************
                 * { "d": { "myName": "Arduino PIRSensor",
"motionDetectedAt": "51s", "time": 1435557391031,
                 * "deviceId": "deedbafefeef" }, "_msgid":
                "b3ae3fb2.4c51c" }

*****************************************************************
***************/
                cameraPreview.postDelayed(new Runnable() {
                    @Override
                    public void run() {
                        mCamera.takePicture(null, null, mPicture);
                    }
                }, 2000);
            }
        };
```

The app has dependency on

- Ibmbluemix.jar
- Ibmpush.jar
- Google Play Services

If you are using Eclipse ADT for development, please download the Android SDK from location http://mbaas-gettingstarted.ng.bluemix.net/android. If you are using Android Studio, then Gradle will resolve dependencies for the above jars automatically. Ensure that “Google Play Services” are installed using the IDE.

I recommend you try out the Quickstart-Push sample application available at the following location first. The Readme file here has pretty much detailed about all the initial setup required to successfully run the app

https://hub.jazz.net/project/mobilecloud/quickstart-push/overview#https://hub.jazz.net/git/mobilecloud%252Fquickstart-push/list/master/quickstart-push-android

That’s it! You are now ready to run the pieces together.

App Demo

https://youtu.be/-ZQ4vnLqHYk

Source code

https://github.com/evanjas77/MotionSensorCamera