Zanzito
User Manual

Ver. 1.2 – 23/07/2017
Author: Gianluca Barbaro

Contents

1. Introduction

Zanzito is an application that acts as a bridge between your Android phone and an MQTT server. MQTT is a lightweight means of communication between systems, such as your home automation system and the various sensors and hardware devices in your home. The MQTT protocol is based on the principle of publishing messages and subscribing to topics, or "pub/sub". Multiple clients connect to a broker and subscribe to topics of interest. Clients connect to the broker and publish messages to topics.

With Zanzito, your Android phone becomes a fully integrated home automation appliance, but other uses are possible, MQTT being an open and very flexible system.

With Zanzito, you can send to your server/home automation system information from your Android phone:

- Your current location;
- The values of the hardware sensors in your phone (i.e. battery level, ambient temperature, light level and so on);
- Pictures from your phone (manually or at a given interval).
- Any other information you want through MQTT publishing to your own custom topics ("Activities"). For example, you can send a message to switch a light on, or activate your home alarm system. You can associate a custom voice command to any custom topic and activate it by voice.

You can also receive:

- Text messages, pictures and notifications;
- Audible alarms;
- SMS text messages to be sent through your phone.
- Any other information (including pictures) through MQTT subscriptions to your own custom topics ("Reports").

2. First run

After installation, Zanzito has default settings that allow your to start the connection straightaway.

However, be aware that the default MQTT server is a PUBLIC server and it is intended for testing only. All your information will be publicly visible to anyone.

Therefore, it is better to first configure the connection to your private MQTT server as described below in "Setting up an MQTT Connection". This directs all information captured by Zanzito to your private server and allows verifying basic connectivity and functionality.

Preventing possible third party capture of information sent over the Internet requires configuration on the backend, as described in the Technical Addenda sections "SSL/TLS Connection" and "Connecting to the MQTT server through Orbot".

3. Zanzito Settings

This section describes all the Zanzito settings and their default values.

From *Menu->Preferences* you have access to Zanzito settings screen. Under the *BASIC* section, you can:

- Setup your optional Device name;
- Configure your MQTT connection.

The Device name is used to build the main MQTT topic that Zanzito uses with the server. Its default value is the Android device ID.

3.1 Setting up an MQTT connection

By accessing the *MQTT Connection* preference screen, you can setup:

- The main server address (defaults to `test.mosquitto.org`).
  Do not add any prefix like `tcp://` or similar;
- The server port (defaults to `1883`);

Under "Security":

- Your optional username;
- Your optional password;
- Activation of SSL/TLS Connection
- Importing a CA certificate for your MQTT server;
- Deletion of a previously imported CA certificate.

SSL Connection requires the CA that released the server certificate to be recognized by your device. See the Technical Addenda sections "SSL/TLS Connection".

Under "Home host":

- Your home WiFi SSID(s)
- "Home" server address
- "Home" server port number
- "Home" server SSL/TLS connection

You can insert any number of SSIDs for your home WiFi network(s) (its name), each on a new line. If you inserted at least one SSID, Zanzito will auto-detect it and switch automatically to the alternate address indicated in "Home host", using the indicated port and SSL/TLS preference.

When the home WiFi is left, Zanzito will switch back to the main address.

If "Home host" and/or "Port" are left empty, the main MQTT host address and/or port number will be used.

Remember to select the right option for "SSL/TLS connection".

Under the section *Update Modes* of the *MQTT Connection* preference screen, you can decide the way Zanzito sends its updates to the server:

- Instantly, as soon as a sensor value changes or the location has changed;
- At regular intervals (*heartbeat*). Every X seconds (a value you can setup in the *Interval* field) Zanzito sends a series of messages containing the current values of all the selected sensors plus the current location;

Zanzito can send an update *On demand*: when it receives a message under the `zanzito/<device-name>/sendheartbeat` topic, it sends an heartbeat. You can also send a manual update by selecting "Send report" from the main menu drawer.

All Zanzito messages are published under a topic starting with `zanzito/`, so by subscribing to the `zanzito/#` topic on your MQTT server, you can see all the messages Zanzito is sending.

After the *BASIC* section, Zanzito preferences are divided into two main branches: *OUTBOUND* (messages that Zanzito sends to the server) and *INBOUND* (messages that are received from the server).

3.2 Prefs: OUTBOUND

The three main categories of outbound information are:

- Location;
- Sensors;
- Photos.

By accessing the ***Location*** preferences screen, you can:

- Activate the Location tracking activity;

When active, your current location is sent (manually or automatically) to your MQTT server with a message like:

```
zanzito/<device-name>/location
{"longitude":<longitude>,"latitude":<latitude>,"altitude":<altitude>,"gps_accurac
y":<accuracy in meters>, "battery_level":<battery-level>,"tst":<unix timestamp>}
```

`battery_level` is sent only if it is selected in the Sensors list.

- Use high precision location (when active, it uses GPS information so the battery gets drained faster);

- Determine the minimum number of meters of displacement that triggers a location updated. The default value is **10** meters. Be aware that very low values could cause more battery draining.

- Activate **FIND** integration (http://www.internalpositioning.com/). When active, current WiFi fingerprints will be sent through MQTT messages in order to fine-track your position (for example, the room you are in).

It is necessary to have a local installation of the FIND server connected to your local MQTT server (the same used with Zanzito). Please refer to FIND documentation for further information. (To install a private, local FIND server, see here).

You can have FIND tracking active only when connected to your Home WiFi network by selecting the corresponding option.

To enable FIND in Zanzito, it is necessary to indicate a FIND group name (if not existing, it will be automatically created at the first Zanzito's update).

You can also *learn* the fingerprint for your current location and send it to the FIND server (through MQTT). Just tap on "Learn" and insert a location name. The learning process will take place in the background: you can see its progress through the corresponding Android notification. Touch the notification to stop the learning process (for better results, let it run for a few minutes without moving the device).

- Activate **Owntracks** (http://owntracks.org/) emulation. When active, all location updates Zanzito sends to the server are formatted according to the current Owntracks specifications. This is useful if your home automation system already has an Owntracks tracking module. If *Battery level* is selected in the Sensors list, this information will be included in Owntracks emulation updates. If you are using Home Assistant, starting from Home Assistant vers. 0.43 Owntracks emulation is not necessary anymore;

- Setup your custom Owntracks Device ID and Tracker ID. If left empty, this information will be retrieved from the *Device name* you setup in the *MQTT Connection* section.

- Share your current location with other Zanzito users connected to the same MQTT server. When enabled, it is possible to see on the map the location of all other users that are sharing their position.

When this option is enabled, at every scheduled or manual location update a message is sent in the form:

```
zanzito/shared_locations/<device-name> {"user":"<device-
name>","longitude":<longitude>,"latitude":<latitude>,"altitude":<altitude>,"gps_a
ccuracy":<accuracy>,"tst":<timestamp>}
```

It is possible to determine who sees who by configuring some ACL permissions on the MQTT server. See below in the “Technical addenda” section.

By accessing the ***Sensors*** preferences screen, you are presented with a list of hardware sensors provided by your phone. Simply select those you want tracked.
The information from the sensors are sent to your MQTT server with a message like the following:
`zanzito/<device-name>/<sensor-name> <value>`

Under the ***Photos*** section you can:

- Enable/disable camera functionalities;
- Activate/deactivate the automatic sending of pictures;
  In general, this Zanzito's functionality is particularly useful in case you have a fixed Android device (phone or tablet) attached to a wall. If your home automation system is properly configured, with Zanzito your Android device can became a surveillance camera.
- Decide the interval (in seconds) at which send pictures;
- Activate a shutter click sound upon manually taking a picture;
- Activate/deactivate pictures sharing;
- Choose the camera to take pictures.
- Enable/disable Autofocus.
  This option is available only on devices that support camera autofocusing. On some hardware, this option could give errors or prevent the camera from working. In case of problems, try disabling autofocusing.

All pictures are sent to your MQTT server with a message like:
`zanzito/<device-name>/picture <binary data>`

Technical tip: pictures are sent in the jpg format as a binary payload of an MQTT message. Your backend system must be able to extract the binary data from the payload and save it as a regular file. A simple python example script is shown below. Starting from vers. 0.43, Home Assistant can receive the pictures sent by Zanzito directly, see the Technical Addenda below.

When *Share photos* is selected, all pictures taken by Zanzito (manually or automatically) are also sent with a second message to:
`zanzito/shared_locations/<device-name> <binary data>`

All shared pictures are shown in the *Pictures* gallery. If you want to decide who sees what, you need to set up some ACL permissions on your MQTT server. See below (*Technical Addenda*).

Zanzito automatically selects the best available resolution for the selected camera.

3.3 Prefs: INBOUND

The ***Inbound*** preferences section allows you to control the incoming information and commands that Zanzito accepts. You have:

• *Push notifications*. These are text or picture messages from your backend system: they are shown in the log and, if active, in a standard Android notification (you can choose a notification sound from among those available);
Text notifications are received under the topic `zanzito/<your-device-name>/notification`, while picture (binary) notifications are received under the topic
`zanzito/<your-device-name>/photonotification`.
If TTS is enabled under *Advanced – Voice* preferences screen and you enable **Read notifications**, all received push notifications will be read aloud in your device's default language.

• *Send SMS*. When active, Zanzito uses the SMS messaging capabilities of your phone to send actual SMS messages;
To request the sending of an SMS message, your backend server must publish an MQTT message in the form:

```
zanzito/<your-device-name>/sendsms/<phone-number> <text>
```

You can decide whether or not a standard Android notification is shown upon receiving an SMS message request and the corresponding notification sound;

• *Play alarm*. If activated, when Zanzito receives an MQTT message in the form:

```
zanzito/<your-device-name>/alarm/play <message>
```

it plays the selected alarm sound indefinitely, until stopped.
A standard Android notification gets always fired and it will show the `<message>`: touch it to stop the alarm sound.
When the alarm sound is stopped, Zanzito sends a message like:

```
zanzito/<your-device-name>/alarm ack
```

To stop the alarm you can also have the backend send a message like:

```
zanzito/<your-device-name>/alarm/stop <empty-payload>
```

In the preferences you can indicate whether to override an eventually active “silent” mode or not.

3.4 Prefs: ADVANCED

The ***Advanced*** preferences section allows you to control some more settings in Zanzito. First you have **Voice** preferences.

3.4.1 Prefs: VOICE - TTS

Zanzito offers a *Text to speech* (TTS) functionality based on Google TTS. You can enable it at application level by selecting the corresponding preference item under A*dvanced – Voice*. When enabled, you can use TTS for:

- **Push notifications** - if enabled also in the corresponding preferences section;
- **Custom Reports** - i.e. subscriptions, when enabled in each custom report;
- **Vocal feedback for vocal commands** – If enabled, every vocal command will be followed by a spoken feedback by Zanzito.

You can also use Zanzito as a MQTT TTS device. If you enable ***Say MQTT texts*** in the preferences, any text sent as payload to the topic `zanzito/<device-name>/say` will be read aloud in your device's default language.

By clicking on ***Audio stream for TTS***, you can select the audio channel the TTS utterances will be played on. The default value is "Notification". Be aware that, if you select "Multimedia", TTS will be muted if Continuous Voice Recognition is active.

3.4.2 Prefs: VOICE Commands

By activating ***Enable voice commands*** under *Preferences->Advanced - Voice*, you can activate any custom Activity (i.e. MQTT message publishing) by simply giving the corresponding voice command. You can start the speech recognizer by clicking on the microphone button or long-pressing the hardware back button in Zanzito's main screen, or by selecting "Voice command" in the main Zanzito's notification message.

You can associate to each custom outgoing topic ("Activity") your own voice command by directly dictating it in the custom activity edit screen: just touch the text field.

You can optionally *dictate the payload*: when enabled in the single Activity, everything you say after the voice command will be sent as payload.

When **Vocal confirmation** is selected, after each vocal command Zanzito will give a vocal feedback by saying the Activity name, if recognized, or "Command not found" if not recognized. You can customize the negative answer by inserting a phrase or a list of phrases (one per text line) in the preference field ***Command not found****.*

In the *Logs* you can see what Zanzito is sending after a vocal command.

3.4.3 Prefs: Continuous listening and activation keyword

Zanzito offers the possibility to continuously listen to what you say and being activated by a specific keyword you can define yourself (*keyword spotting*),

This function is **EXPERIMENTAL** and it is not guaranteed to be working on your device: in case of problems just deactivate it.

Continuous listening relies on Google Speech recognizer: the easiest way to have it available on your device is to install the **Google app** from Google Play Store and run it at least once to set it up.

You should also be sure that the Google recognizer is the default recognizer on your device by setting it in your device's settings.

When this functionality is active, the "Multimedia" audio stream of your device will be muted.

Continuous listening is very heavy on battery usage, we advice to check **Only when plugged in**: when on battery, this functionality will be automatically disabled until you re-connect a power source.

To set up your vocal activation keyword, just touch **Keyword** in the preferences and say the word.

To impart a vocal command, say:

> *Please <keyword>, <activity-vocal-command> <optional-payload-value>*

The word "Please" can be anything else, as long as it is not too short.

You can troubleshoot this functionality by opening the *Logs* screen: you can there see what Zanzito is actually understanding and transcribing from your utterances.

3.4.4 Prefs: ADVANCED – MORE

Under *Preferences->Advanced* – ***More*** settings you have:

- Immediately connect to the MQTT server after being launched;
- Start automatically at system boot or not;
- Remote administration (see Technical Addenda below);
- Show Admin module (see Technical Addenda below);
- Parental control.

Prefs: Parental control

When you enable Parental control in Zanzito and setup a password, you will be prompted for the password before accessing the preferences and/or stopping the connection. Zanzito has a very basic password implementation, only meant to prevent kids from messing around with the settings and from stopping the connection: it is not a hacker proof security measure.

Also consider that, if you forget your password, the only way to reset it is to uninstall and reinstall Zanzito. In that case, all the previous settings will be lost.

4. Main Screen

In Zanzito's main screen you have an image button to start/stop the connection to the MQTT server, two visual indicators that report the current status of the MQTT connection and the Location service, and microphone button to start the speech recognizer and give a vocal command.

5. Menu/Drawer

- **Activities**. A customizable list of topic publications. You can add an element by touching the "+" sign or by importing it from the list of all your current topics on your MQTT server (see the Technical Addenda below). Insert the requested information and go back to the list: by touching the "arrow" button of an element, the message gets sent. By long-touching an element or by touching the menu button of the element, you have access to a contextual menu where you can edit, clone, delete and get info about the element.

  In the topic field you can use a placeholder for your device name: just put `%devicename%` anywhere in the topic and it will be substituted by the actual device name at run time.

  If Voice commands are enabled in the preferences, you can add the voice command associated to each custom topic publication, by directly dictating it in each custom topic edit screen: just touch the text field.

  By selecting *Get payload from vocal command*, Zanzito will send as payload everything you say after the Activity's voice command. If you say nothing, the default payload will be sent. In the *Logs* you can see what Zanzito sends.

  By selecting *Send on shake*, every time you shake your phone, the corresponding activity will be sent. Be aware that when there's an Activity associated to the "shaking" movement, the default accelerometer of your device will be active: this will drain your battery faster.

- **Reports**. A customizable list of topic subscriptions. You can add an element by touching the "+" sign or by importing it from the list of all your current topics on your MQTT server (see the Technical Addenda below). Insert the requested information and go back to the list: by touching the "i" button of an element, you can see some information about it. By long-touching an element or by touching the menu button of the element, you have access to a contextual menu where you can edit, clone, delete and get info about the element. You can subscribe to a picture topic, see the corresponding section under the Technical addenda section below. If TTS is generally enabled and you select *Read message*, the content of the custom report will be read in your device's default language.

• **Admin**. Gives access to Zanzito's remote administration module. From here you have access to a topic discovery screen that presents a list of all current topics on your MQTT broker. You can also see a list of all Zanzito's devices that ever connected to your MQTT server. By long-pressing on a device name, you can delete it from the server or manage it remotely.

By touching a device name you have access to an administration screen fro that device, where you can manage its preferences, activities, reports, see its position on the map, see its logs and send a few remote commands.

In "Demo mode" you can manage remotely your own current device only. Please refer to the Licensing section below.

Please refer to the Technical Addenda below for further information.

• **Map**. Shows a map with markers for your current position, your last saved position, the position of all other Zanzito users connected to the same MQTT server (who are sharing their position and whose location you are allowed to see).

Zanzito's map view automatically adjusts the zoom level to show all currently available users. You can tap on a marker and manually adjust the zoom level: Zanzito will follow the user keeping the current zoom level. If you want to keep your phone awake, select the corresponding menu item.

You can go back to automatic zooming by selecting the "Reload" menu item.

• **Gallery**. Shows a gallery of all shared pictures available on your MQTT server. Be aware that the update and downloading of pictures are active only as long as you are on the *Gallery* screen.

By selecting a single picture from the gallery, you can manually zoom, share it through other apps on your device and erase it. The deletion of a pictures impacts only the storage on the local device: as long as your MQTT server stores it, it will be downloaded again to your device (shared pictures are retained by the MQTT server) .

By long-touching an element you have access to a contextual menu where you can show or delete the picture.

• **Send report**. Force-sends an update (*heartbeat)* of all the currently available informations (the connection must be active);

• **Send photo**. Takes a picture and sends it to the server (automatic sending does not have to be enabled). After being sent, the pictures is shown fullscreen. Zanzito also accepts and sends photos shared by other Android apps.

• **Bookmark position**. Saves your current position. This information will be only saved on your phone;

• **View logs.** Accesses the Log screen;

• **Buy license**. Upgrade Zanzito's license level. See the *Licensing* section below.

• **Preferences**. Accesses the Preferences screen;

• **User manual**. Shows Zanzito's User manual (this manual). A third party pdf viewer is required.

• **About Zanzito**. Accesses the Info about the application.

• **Quit Zanzito**. Exits completely from Zanzito.

6. Licensing

Once downloaded and installed, Zanzito will work in "demo" mode: it will connect to your MQTT server for each first 120 minutes every four hours. After the four hours have passed, you will have to restart the connection manually.

In demo mode you can add only one custom *Activity* and one custom *Report* and the Admin module will be restricted to your own current device.

There are two licensing levels available:

– **Base**. It unlocks all current Zanzito's functionalities except the *Admin module*.
  This license is not upgradeable and it is not available for Family library.

– **Full**. It unlocks all Zanzito's functionalities, forever.
  This license is available to all your family members.
  It requires a licensing app to be purchased and installed from the Google Play store.

You can buy a license from within the app by clicking on the "Buy license" button in the Main screen, or by selecting *Buy license* from the main menu drawer.

7. Quitting Zanzito

To completely exit from Zanzito, from the main menu drawer select "Quit Zanzito" and confirm exit.

If you "swipe-kill" Zanzito from the list of currently open application, the connection to the server will remain active and so will be the automatic tracking. By touching Zanzito's main notification you can bring the app back to front.

8. Android permissions

Zanzito requires the following main Android permissions:

- EXTERNAL_STORAGE. Required to share photos to and from other Android apps and to visualize Zanzito's User manual;
- LOCATION. Required for location tracking;
- CAMERA. Required for camera related functionalities;
- SMS. Required for the SMS functionality;
- MICROPHONE. Required for the use of Vocal commands.

Starting from Android vers. 6.0 you can decide which one of the permissions grant to Zanzito. If not granted, the related functions will not work.

9. Privacy

Zanzito's author(s) cannot see your informations, nor are they in any way connected to your device. All the informations that Zanzito collects is sent only to the MQTT server of your choice. Be aware of all the possible security and privacy implications: you must ensure that the connection between your device and your MQTT server is secure and that you have properly configured the users that have access to the server and their permissions.

Do not use public MQTT servers unless you know what you're doing: they are "public" so that anyone can see your location, your pictures and/or any other information exchanged through Zanzito.

Also be aware that according to your country's laws, unauthorized installation of Zanzito on someone else's device may be a felony. In addition Zanzito's terms of use prohibit installing Zanzito on someone else's device without their consent.

In summary:

- The author(s) of Zanzito does not possess any information about the customer except what is given in the Google Play store payment details. That data will not be used by the author except for accounting and tax purposes, where required by the relevant tax authorities.

- Zanzito itself sends data only to the MQTT server indicated by the user in the preferences. *Zanzito has no control over the way Google and your device vendor handle your data while accessing their services, such as Map, Location and Vocal recognition.*

9.1 Zanzito's Persistent notification

Zanzito will always present a notification that indicates its current status and that is not dismissible.

Attached to the notification, there are some buttons by which you can start the connection, give a vocal command, bookmark your current position and take a picture. If other notifications are present, you might need to touch it with one or two fingers and drag down in order to see the buttons.

Note: Zanzito's main Android notification has to be persistent for compliancy to Google Play store rules.

Technical Addenda

A1. Zanzito status messages

Every time it connects to an MQTT server, and at each *heartbeat*, Zanzito sends a

```
zanzito/<device-name>/status 1
```

message. When it disconnects, it sends a “0” payload.

Similarly, for notifications:

```
zanzito/<device-name>/notification 1
```

when enabled,

```
zanzito/<device-name>/notification 0
```

when not enabled

SMS sending:

```
zanzito/<device-name>/smssend 1
```

when enabled,

```
zanzito/<device-name>/smssend 0
```

when not enabled

Alarm sound playing requests:

```
zanzito/<device-name>/alarm 1
```

when enabled

```
zanzito/<device-name>/alarm 0
```

when not enabled.

MQTT TTS:

```
zanzito/<device-name>/say 1
```

when enabled

```
zanzito/<device-name>/say 0
```

when not enabled.

Finally, it sends a message containing the current version of Zanzito:

```
zanzito/<device-name>/version <your-zanzito's-current-version>
```

You can use these status messages on your backend to keep track of each device status and currently enabled capabilities.

A2. SSL/TLS Connection

If you have configured your router so that your MQTT broker is accessible from outside your local network, you can put your public address into the preferences and Zanzito will connect to it. However we do not recommend this unless you have an encrypted connection.

Zanzito supports SSL/TLS connections to the MQTT server, but no client authentication with certificates (you must use usernames and passwords). To enable encrypted connections to your MQTT broker, your Android device must first recognize the Certificate Authority (CA) that signed the certificate your server is using.

If the server certificate is signed by a recognized authority, Zanzito should work straightaway.

If the certificate is self signed, it is necessary to load the CA certificate (usually `ca.crt`) to your device (through email attachment, USB copy or any other means). Then you can:

– launch the certificate file through a File browsing app and accept it into you system (Android will complain a bit about the fact that the certificate is self signed: as long as you know where it comes from, there's no real risk);

- or, alternatively, load the CA certificate directly from Zanzito (under *Preferences->MQTT Connection->Security*).

Once the self signed CA certificate is imported by your device, enable "SSL/TLS connection" in Zanzito->Preferences->MQTT Connection->Security and change the port according to your server configuration.

There are many tutorials on net on how to configure SSL/TLS for an MQTT broker. For mosquitto, the following article is a good starting point:

https://primalcortex.wordpress.com/2016/03/31/mqtt-mosquitto-broker-with-ssltls-transport-security/

A3. Connecting to the MQTT server through Orbot

As an alternative to SSL/TLS connection, you can install and setup a Tor server (https://www.torproject.org/) on your backend, configure a hidden service that points to your local MQTT server and use Orbot (https://guardianproject.info/apps/orbot/) on your phone to connect from the outside.

For example, after installing Tor on your server, you can add a hidden service to the local `torrc` file:

```
HiddenServiceDir /var/lib/tor/mosquitto/
HiddenServicePort 1883 127.0.0.1:1883
HiddenServiceAuthorizeClient stealth mqtt1
```

After restarting Tor, you can read the newly generated authentication cookie from the Tor-generated hostname file, for example with:

```
$ sudo more /var/lib/tor/mosquitto/hostname
```

It would be something like:

```
abcdef1234567890.onion ABCDEF1122334455667789 # client: mqtt1
```

For Orbot, add the previous line in *Orbot -> Menu -> Settings* to the "Torrc Custom Config" entry. Restart Orbot, activate "Apps VPN Mode" and select Zanzito from the list of apps.

In Zanzito, it will be sufficient to enter the `.onion` address in the MQTT broker address preference, such as: `abcdef1234567890.onion`

The dual host configuration in Zanzito ("Home host") will not work if Orbot is enabled (Tor does not allow connections to local addresses).

A4. Subscription to custom topics (Reports)

Zanzito supports five types of subscribed topics:

- **Generic text**. It simply returns the text contained in the payload of the message.
- **Alarm**. Whenever a message is received for the subscribed topic, the selected alarm sound is played until dismissed from the relative notification.
- **Picture**. Subscriptions to topics that publish binary payloads containing an image. An image preview will be shown in the Android expanded notification (if enabled).
- **Notification**. Subscriptions of this type can simply receive and show a text payload. However, it is also possible to receive a JSON formatted message containing the keys `title` and `message`. The value of `title` will be used as title of the Android notification.
- **JSON**. Subscriptions to JSON formatted messages. You can insert a name in *key* field and the corresponding value will be extracted upon reception of a message.

A5. Extracting a picture jpeg file from an MQTT message

The following is a simple python script that will subscribe to a list of Zanzito picture topics, extract and save an image file when one is received. Note that Home Assistant doesn't need this script.

```
#!/usr/bin/python3.5
import paho.mqtt.client as mqtt
import io
from PIL import Image

base_dir = "path-to-images-directory"

mqttServerAddress = "<server-address>"
mqttServerPort = <server-port>
mqttUsername = "<mqtt-username>"
mqttPassword = "<mqtt-password>"

# List of Zanzito's device ids to subscribe to
clients = {"<device-name-1>", "<device-name-2>"}

def on_connect(client, userdata, rc):
  for aClient in clients:
    topic = "zanzito/" + aClient + "/picture"
    client.subscribe(topic)

def on_message(client, userdata, msg):
  device_id = msg.topic.split("/")[1]
  file_name = base_dir + device_id + ".jpg"
  Image.open(io.BytesIO(msg.payload)).save(file_name)

client = mqtt.Client()
client.username_pw_set(username=mqttUsername,password=mqttPassword)
client.on_connect = on_connect
client.on_message = on_message

client.connect(mqttServerAddress, mqttServerPort, 60)

client.loop_forever()
```

A6. Sending a picture jpeg file through an MQTT message

The following is a simple python script that will send a jpeg image file to a topic. The command line syntax to call this script is:

```
mqtt_send_file.py <topic> <image-file-path>
```

The script mqtt_send_file.py:

```
#!/usr/bin/python3.5
# -*- coding: utf-8 -*-
import paho.mqtt.client as mqtt
import io
import sys, getopt

# MQTT parameters
mqttServerAddress = "<server-address>"
mqttServerPort = <server-port>
mqttUsername = "<mqtt-username>"
mqttPassword = "<mqtt-password>"
mqttQos = 0
mqttRetained = False

# Reads command line arguments
topic = sys.argv[1]
imageFilePath = sys.argv[2]

# Connects to the MQTT broker
client = mqtt.Client()
client.username_pw_set(username=mqttUsername,password=mqttPassword)
client.connect(mqttServerAddress, mqttServerPort, 60)

# Reads the file
with open(imageFilePath, "rb") as imageFile:
  myFile = imageFile.read()
  data = bytearray(myFile)

# Publishes it
client.publish(topic, data, mqttQos, mqttRetained)
```

Be aware that in the previous script there's no control over the passed arguments: if they are not in the right format, the script will crash.

A7. ACL permissions with mosquitto

If you use the location or pictures sharing functions, any Zanzito device will be posting its location/pictures to any other device/user.

If you want to avoid this, you can add some permission control on your MQTT server. For example, using a mosquitto server (https://mosquitto.org/), you first add an acl file reference in the general configuration (usually `/etc/mosquitto/mosquitto.conf`) by adding the following line:

```
acl_file /etc/mosquitto/local.acl
```

then, in the `/etc/mosquitto/local.acl` you first add a line like:

```
pattern readwrite owntracks/%u/#
```

that allows all Zanzito devices to subscribe and publish to the eventual Owntracks emulation topic. Then you can specify an administrative role, with the following:

```
user myadministrator
topic #
```

which means the user "`myadministrator`" can subscribe and publish to any topic, thus see any other user's location/picture.

And some "regular" users:

```
user normal_user
topic readwrite zanzito/<normal-user-device-name>/#
topic readwrite zanzito/shared_locations/<normal-user-device-name>
topic readwrite zanzito/shared_pictures/<normal-user-device-name>
topic read zanzito/shared_locations/Another-Device
topic read zanzito/shared_pictures/Another-Device
```

that allow "normal_user" to subscribe and publish to his/her own topics and to see only the shared location and the shared pictures of "`Another-Device`": all other devices will be hidden to her/him.

Mosquitto must be restarted for the changes to take effect.

If you are using a different MQTT server, please refer to their documentation to obtain similar results.

A8. Admin module

This module, accessible thorough the corresponding main menu drawer item, gives access to a list of all Zanzito devices that ever connected to your MQTT server.

> *Please understand that the Admin module works in* real time*, meaning Zanzito has to be connected to your server, as well as your other devices.*
>
> *Also,* upgrade *all your Zanzito's devices to the last version otherwise some functionalities might not work.*
>
> *Finally,* Remote administration *has to enabled on remote devices.*

From the devices list, by selecting an item, you have access to a device's overview where you can manage remotely the device itself.

By long-pressing on a device item you are presented with a contextual menu where you can either manage the device (same as selecting it) or delete it. The deletion of a device will erase all retained topic messages on your MQTT server related to that device.

If Zanzito is in "Demo mode" or you purchased a "Base" license, you will be able to manage remotely your current device only, for demo purposes. To admin all your devices a "Full" license is required: please refer to the "Licensing" section for further informations.

A8.1 Current topics Discovery

If from the Admin module's main screen you touch the "Discover current topics" button (or from your Activities/Reports screens), you have access to a list of all current topics on your MQTT broker. Please, allow for some time for the list to be fully loaded (not all topics are retained, so in some cases you have to wait until something is published on a topic, otherwise it won't be visible).

By long-pressing on an item or by touching the menu button, you have access to a contextual menu from where you can import the selected topic into your Activities or Reports, or you can delete the topic from your server.

While this screen is active, the list is continuously updated and all normal functions in Zanzito are suspended. You can pause/resume the list updating by pressing the corresponding button on the top-right of the screen.

A8.2 Device overview

Please, allow some moments for the device's informations to be retrieved from the server.

If the device is online and has "Remote administration" enabled, you will have access to its:

- On line status;
- Battery level;
- Current position on map;
- Information about the device (such as screen status, current foreground application etc., see below);
- Installed Zanzito's version;
- Preferences;
- Activities;
- Reports.

You can also send a few remote commands. At the moment, the following commands are available:

- **Restart**. Restarts Zanzito;
- **Reconnect**. Forces a reconnection to the MQTT server;

– **Take picture**. Forces a picture taking. In order to see the remotely taken picture, you either go to your *Gallery* screen (if the remote device has picture sharing enabled), or setup a custom Report of type picture with the topic `zanzito/<device-name>/picture`
– **Send Heartbeat**. Forces a full report to be sent to the server;
– **Set Bookmark**. Saves the remote current position;
– **Play Alarm**. Plays an alarm on the remote device (*Play alarm* has to enabled in its preferences).
– **Stop Alarm**. Stops any currently playing alarm on the remote device.

Other commands will be added in the future: please propose your suggestions here.

A8.3 Device detailed infos

When *Remote* administration is enabled in *Preferences->More settings*, some additional information are sent at every heartbeat (scheduled or manual) under the topic:

```
zanzito/<device-name>/device_info
```

containing a JSON formatted payload with the following information:

Key	Value	Description
`time`	Long number	UNIX timestamp of the current update
`device_info`	text	Manufacturer, model and Android version of the device
`battery_charging`	boolean	Is the battery charging?
`charge_type`	text	Type of current charge. Possible values are: *AC*, *USB*, *Wireless*, *None*, *N/A*
`current_foreground_app`	text	Name of the current foreground application. Please note that, if the screen is locked, the name of the last app is given.
`screen_on`	boolean	Is the screen on?
`screen_locked`	boolean	Is the screen locked?
`screen_orientation`	text	Current screen orientation. Possible values are: *Portrait*, *Landscape*, *N/A*
`current_wifi`	text	Name (SSID) of the current WiFi network
`current_operator`	text	Name of the current mobile operator

Zanzito reads the above values at each heartbeat, so there might be a delay before they are sent to your server: it depends on how you configured Zanzito.

By accessing the *Menu->Admin* module and by selecting a device, you can see the above information for the selected device by tapping on the *Info* button. To trigger an update, you might want to send a *Send heartbeat* command before tapping on *Info*.

A8.4 Remote preferences administration

From a device's admin overview, by clicking on the "Prefs" button you have access to (almost) all remote device's preferences. You can modify any of the fields/checkboxes but consider that:

No change will be applied to the remote device until you press the "Upload" button

You can also press the "Copy" button: all the preferences will be copied into the clipboard. Then, by pressing the "Paste" button into another remote device's preferences screen, you can overwrite all the preferences of the second device, thus "cloning" it from the first one.

Again, pressing the "Upload" button is required for the changes to be applied. Remember to change the "Device name" before uploading the new cloned prefs.

A8.5 Remote Activities and Reports administration

From a device's admin overview, by clicking on the “Activities” or “Reports” buttons, you have access to all remote device's custom topics.
The presented list supports multiple selections and, by clicking on the corresponding menu buttons/items, you can:

- Select all/none;
- Cut the selected topics;
- Copy the selected topics;
- Paste the topics currently present in the clipboard;
- Delete the selected topics.

You also have:

– copy&paste custom topics from/to your local device's Activities/Reports;
– by touching the menu button, you can Edit, Clone, Delete of get info about the corresponding topic, just like you do with your local Activities/Reports;
– by touching the “arrow” button, you can send the corresponding activity to your MQTT server;
– by pressing the “plus” button, you can add a new Activity/Report.

Again, pressing the “Upload” button is required for the changes to be applied.

A9. Remote administration

When enabled in *Preferences->More settings*, this functionality allows you to control Zanzito remotely by the Admin module of another Zanzito's device, or directly via MQTT.

Configuration backup

When *Remote administration* is enabled, every time you change a preference value (including custom topics) and at each *heartbeat*, Zanzito will automatically send to the MQTT server an encoded string containing all configuration values, as a binary payload to the topic `zanzito/<device-name>/lastprefs`

You can extract and save (i.e. copy&paste) this encoded text by subscribing to the same topic with any MQTT tool.

To restore a previously saved configuration:

1. Configure Zanzito to connect to your MQTT server and activate the connection (if not already done);
2. Enable "Remote administration" in Zanzito's preferences;
3. Publish the saved configuration text to the topic `zanzito/<device-name>/restore_prefs`

Zanzito will receive the configuration, apply it and restart the connection with the new values.

If the restored configuration comes from another device, please remember to change the *Device name* as soon as possible.

A9.1 Remote administration: setting the preferences via MQTT

Remote setting of Zanzito's preferences is accomplished by sending a message to the topic:

```
zanzito/<device-name>/set_prefs
```

The payload has to be in JSON format and can contains any number of keys, for example:

```
zanzito/<device-name>/set_prefs
{advanced_parental_control_enabled=true,advanced_parental_control_password="nowaykid"}
```

The following table contains all the preference keys you can set by sending an MQTT message:

Key	Value	Description
`device_name`	text	Name of the device
MQTT		
`host`	text	Broker address
`port`	text	Broker port
`username`	text	Username
`password`	text	Password
`tls`	true/false	SSL/TLS enabled
`home_host`	text	Home broker address
`home_ssid`	text	Home wifi name
`updates_instant`	true/false	Instant updates
`updates_heartbeat`	true/false	Heartbeat enabled
`updates_heartbeat_interval`	text	Heartbeat interval

LOCATION		
`location_active`	true/false	GPS tracking enabled
`location_high_precision`	true/false	High precision GPS enabled
`location_share`	true/false	Sharing enabled
`location_minimum_distance`	text	Minimum number of meters to trigger a location update
`find_enable`	true/false	Find tracking enabled
`find_enable_only_home`	true/false	Find tracking enabled only when at Home
`find_group_name`	text	Find group name
`owntracks_enabled`	true/false	Owntracks emulation enabled
`owntracks_device_id`	text	Owntracks device id
`owntracks_tracker_id`	text	Owntracks tracker id

CAMERA		
`camera_enabled`	true/false	Camera enabled
`camera_auto_take_pictures`	true/false	Automatically send pictures enabled
`camera_auto_take_pictures_interval`	text	Automatically send pictures interval
`camera_shutter_sound`	true/false	Shutter click enabled
`camera_share_pictures`	true/false	Photo sharing enabled
`camera_autofocus`	true/false	Camera autofocusing enabled

INBOUND		
`inbound_notifications_enabled`	true/false	Push notifications enabled
`inbound_photo_notifications_enabled`	true/false	Push photo notifications enabled
`inbound_notifications_tts_enabled`	true/false	TTS of push notification enabled
`inbound_sms_to send_enabled`	true/false	Receiving an SMS text to send enabled
`inbound_alarm_enabled`	true/false	Play an alarm sound enabled
`inbound_alarm_override_silent`	true/false	Override silent mode for Alarm enabled

VOICE		
`voice_tts_enabled`	true/false	Text to speech enabled
`voice_tts_over_mqtt_enabled`	true/false	TTS via MQTT enabled
`voice_commands_enabled`	true/false	Voice commands enabled
`voice_commands_feedbacks`	true/false	Voice commands vocal feedbacks enabled
`voice_negative_feedbacks`	text	Voice commands negative feedbacks text(s)
`voice_commands_continuous`	true/false	Voice commands continuous recognition enabled
`voice_commands_continuous_plugged`	true/false	Voice commands continuous recognition only when powered
`voice_commands_keyword`	text	Voice commands activation keyword

ADVANCED PREFERENCES		
`advanced_automatic_connection`	true/false	Automatic connection on zanzito's startup
`advanced_autostart_boot`	true/false	Launch zanzito at boot
`advanced_parental_control_enabled`	true/false	Parental control enabled
`advanced_parental_control_password`	text	Parental control password

A9.2 Remote administration: commands via MQTT

When *Remote administration* is enabled, you can send a few administrative commands to Zanzito, here's the list:

TOPIC	EFFECT
`zanzito/<device-name>/cmd/restart`	restarts Zanzito
`zanzito/<device-name>/cmd/reconnect`	restarts the connection to the MQTT broker
`zanzito/<device-name>/cmd/take_picture`	takes a picture (if camera is enabled and configured)
`zanzito/<device-name>/cmd/send_heartbeat`	sends an heartbeat
`zanzito/<device-name>/cmd/set_bookmark`	saves the current location to a bookmark

`zanzito/<device-name>/cmd/send_logs`	Current logs are sent to:
	`zanzito/<device-name>/log_inbound`
	`zanzito/<device-name>/log_outbound`

A10. Configuration examples with Home Assistant

Home Assistant (https://home-assistant.io/) is an open-source home automation platform running on Python 3.
A preview version of a custom Home Assistant notifier component is available for download and test, please see here.

To integrate a Zanzito device, here are a few configuration examples (HA vers. 0.43 or above required):

```
camera:
  - platform: mqtt
    topic: zanzito/shared_pictures/<device-name>
    name: Zanzito Camera
  - platform: mqtt
    topic: zanzito/<device-name>/picture
    name: Zanzito Camera 2
device_tracker:
  - platform: mqtt_json
    devices:
      my_device: zanzito/<device-name>/location

sensor:
  - platform: mqtt
    name: "Zanzito Battery"
    state_topic: "zanzito/<device-name>/battery_level"
    qos: 0
    unit_of_measurement: "%"

binary_sensor:
  - platform: mqtt
    name: "Zanzito status"
    state_topic: "zanzito/<device-name>/status"
    payload_on: "1"
    payload_off: "0"
    sensor_class: connectivity
```

Script to send a notification message:

```
script:
  zanzito_send_notification:
    alias: Send notification to Zanzito
    sequence:
      service: mqtt.publish
      data_template:
        topic: 'zanzito/{{ dest_id }}/notification'
        payload: '{{ message }}'
        qos: 1
        retain: 0
```

You can call this script with:
```
- service: script.zanzito_send_notification
  data:
    dest_id: '<device-name>'
    message: 'message text'
```

To send an image file with the previous script to a custom defined topic, you first define a:

```
shell_command:
  mqtt_send_file: '<path-to>/mqtt_send_file.py {{ topic }} {{ imageFilePath }}'
```

Then you can call this shell command from any automation with:
```
    action:
      - service: shell_command.mqtt_send_file
        data:
          topic: '<your-topic>'
          imageFilePath: '<path-to-your-picture>.jpg'
```

To play an alarm sound through Zanzito, first you could define an
```
input_boolean:
  zanzito_play_alarm:
    name: Zanzito Alarm
    initial: off
    icon: mdi:alarm-multiple
```

then a `binary_sensor` for the received acknowledge from Zanzito:
```
binary_sensor:
  - platform: mqtt
    name: "Alarm ack Zanzito"
    state_topic: "zanzito/<device-name>/alarm"
    payload_on: "ack"
    payload_off: "1"
```

Then some automations:
```
automation:
  - alias: Zanzito alarm switch goes on
    trigger:
      platform: state
      entity_id: input_boolean.zanzito_play_alarm
      from: 'off'
      to: 'on'
    action:
      - service: script.zanzito_play_alarm
        data:
          dest_id: '<device-name>'
          message: 'Wake up!'

  - alias: Zanzito alarm switch goes off
```

```
    trigger:
      - platform: state
        entity_id: input_boolean.zanzito_play_alarm
        from: 'on'
        to: 'off'
    action:
      - service: script.zanzito_stop_alarm
        data:
          dest_id: '<device-name>'

  - alias: Zanzito Alarm ack
    trigger:
      - platform: state
        entity_id: binary_sensor.alarm_ack_zanzito
        to: 'on'
    action:
      - service: homeassistant.turn_off
        entity_id: input_boolean.zanzito_stop_alarm
```

Finally, a couple of scripts:

```
script:
  zanzito_play_alarm:
    alias: Play alarm on Zanzito device
    sequence:
      service: mqtt.publish
      data_template:
        topic: 'zanzito/{{ dest_id }}/alarm/play'
        payload: '{{ message }}'
        qos: 1
        retain: 0

  zanzito_stop_alarm:
    alias: Stop alarm on Zanzito device
    sequence:
      service: mqtt.publish
      data_template:
        topic: 'zanzito/{{ dest_id }}/alarm/stop'
        payload: '<your-message>'
        qos: 1
        retain: 0
```

This way, you can control the alarm from anywhere in Home Assistant by simply turning on/off `input_boolean.zanzito_play_alarm`