Broadcast Drone Remote ID

Project overview

This is a DEMO project that must show the technical capabilities to broadcast a remote ID to a user device (smartphone) via BT/Wi-Fi and to publish (via GPRS/GSM connection) the drone position to a remote server.
<u>This is not the final product</u>

Author(s):

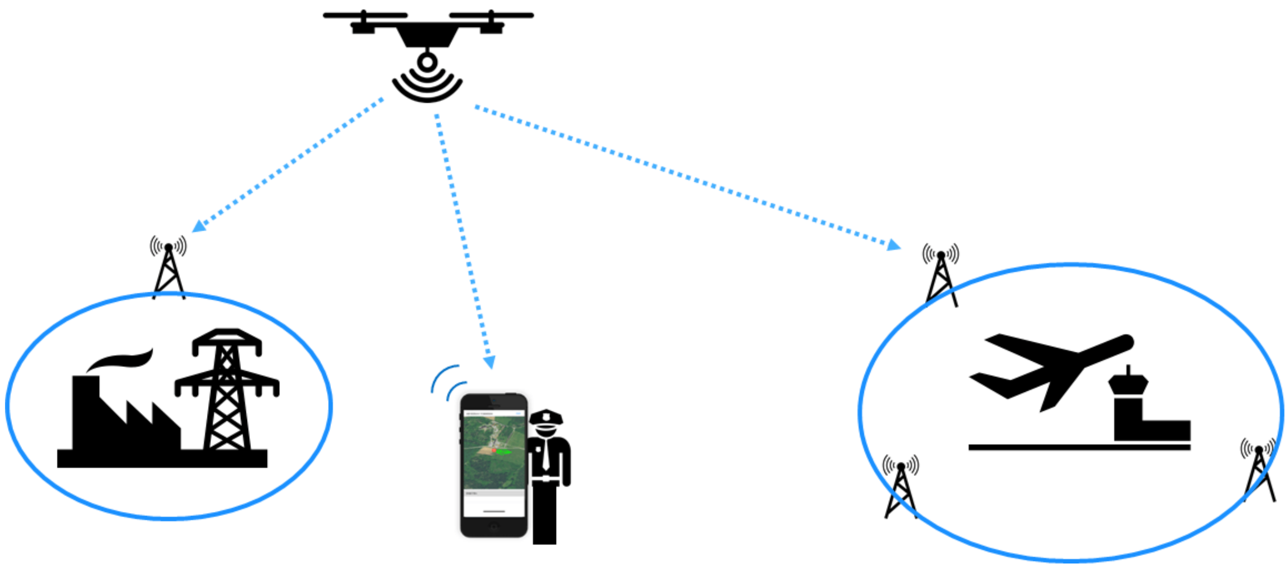
Contributor(s):

Revisions:

1.2 First Draft Candidate **10/02/2020**

What is Drone Remote ID?
Drone Remote ID is the concept of a system that identifies drones operating in the national airspace.
It's a digital license plate for drones.

Broadcast Drone Remote ID
Broadcast Drone Remote ID is a useful method of identifying drones operating nearby
Having a drone broadcast a signal to a receiver on the ground may be a solution for some mission types operating in a small area.
Such advertisements can be used by the general public, law enforcement, critical infrastructure managers, ATC, or other drones to give better situation awareness of the airspace around them.



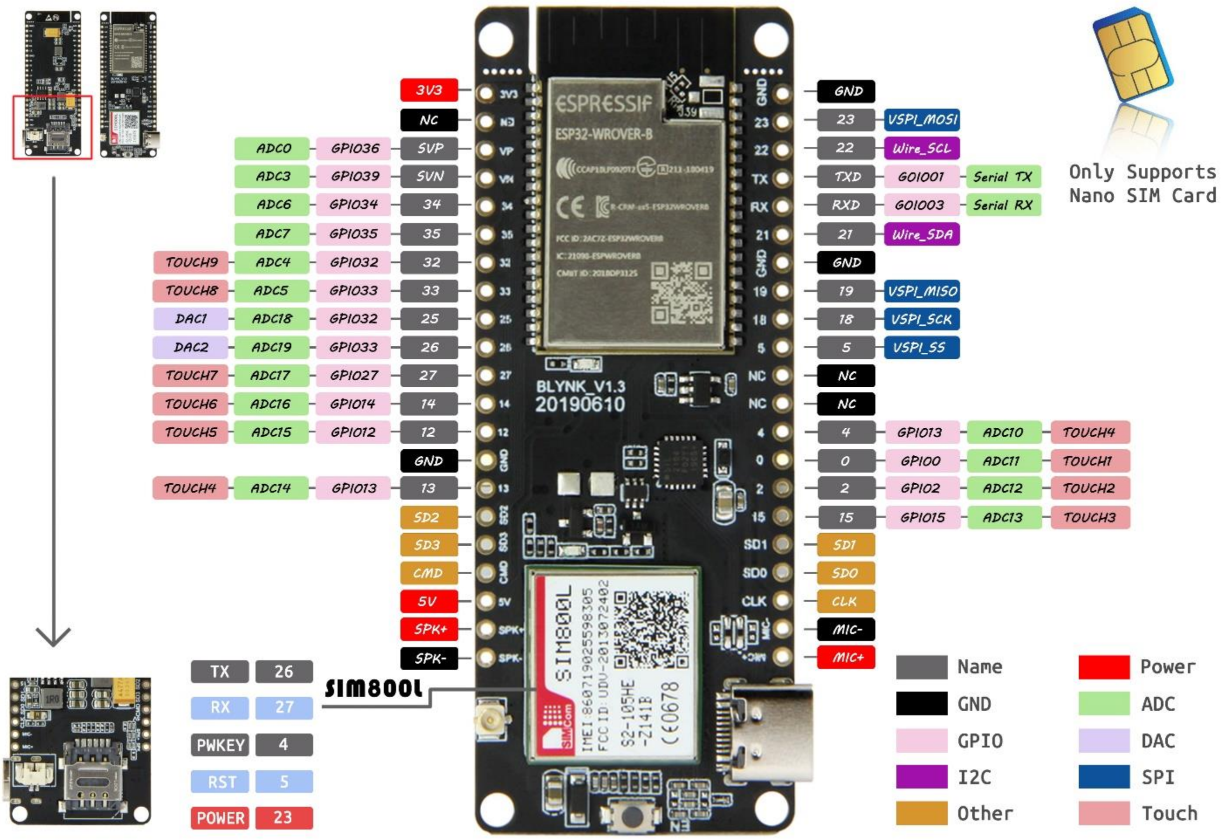
Broadcast Remote Drone ID sample scenario

What we will use for testing Drone Remote ID
Open Drone ID is a project to provide a low cost and reliable "beacon" capability for drones so that they can be identified when within range of a receiver.
The current specification is based on Bluetooth* Legacy Broadcasts Packets (Advertisements), the new Bluetooth 5 (long range) Advertising Extensions, and Wi-Fi implementation advertisements based on Neighbor Awareness Network protocol.

Components of Drone Remote ID

The main components of a Drone Remote ID device are the following:

- ESP32 card with GSM/GPRS, Wi-Fi and BT (TTGO T-Call board **https://bit.ly/2SzPIYC**)
- On-Board GPS (NMEA) (M8Q uBlox GPS **https://bit.ly/2UBA5mj**)
- On-Board battery (1S 350mAh LiPo battery)
- Power On button



ESP32 SIM Card SIM800L T-Call V1.3

The device must operate without any interconnection with the host drone (i.e. no telemetry data coming from the drone etc.)

The information (messages) sent is divided into static and dynamic data where the static data is broadcast less frequently than dynamic data. These messages are "connectionless advertisements" that do not require any acknowledgement from the receiver.

Broadcasted Data (at least)

Static data

Drone ID format: ABCDF1FG916003A12345 (example)
Pilot ID format: ABCDEFGHIJKLMNOPQRST (example)

The static data are hardcoded in the firmware

Dynamic data

Operator data (gathered from user's smartphone)

Timestamp
Latitude
Longitude

Drone data (gathered from GPS connected to ESP32 UART)

TimeStamp
Latitude
Longitude
Altitude
Ground Speed

Note: A simple I2C sensor like BME680 can be used to get real time data like ABS pressure and so on (optional)

MANDATORY

- Broadcast Remote Drone ID via BT

OPTIONAL (it depends by the ESP32 Wi-Fi chip capabilities)

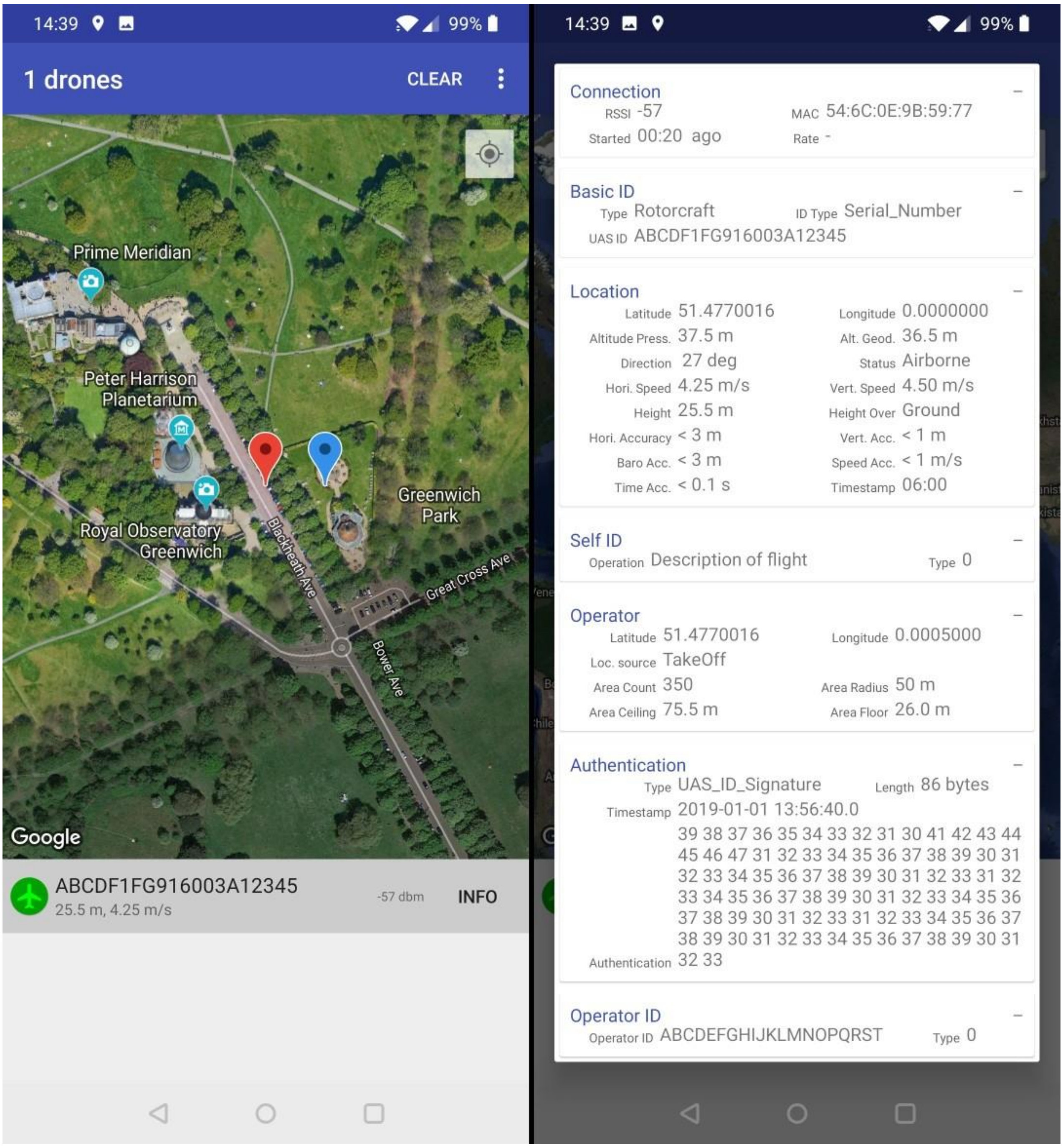
- Broadcast Remote Drone ID via Wi-Fi

For reference about OpenDroneID pls check https://github.com/opendroneid/specs/

User's smartphone testing (Mandatory)

For testing the Broadcast Remote Drone ID, the following OpenDroneID app (for Android only) **must** be used:

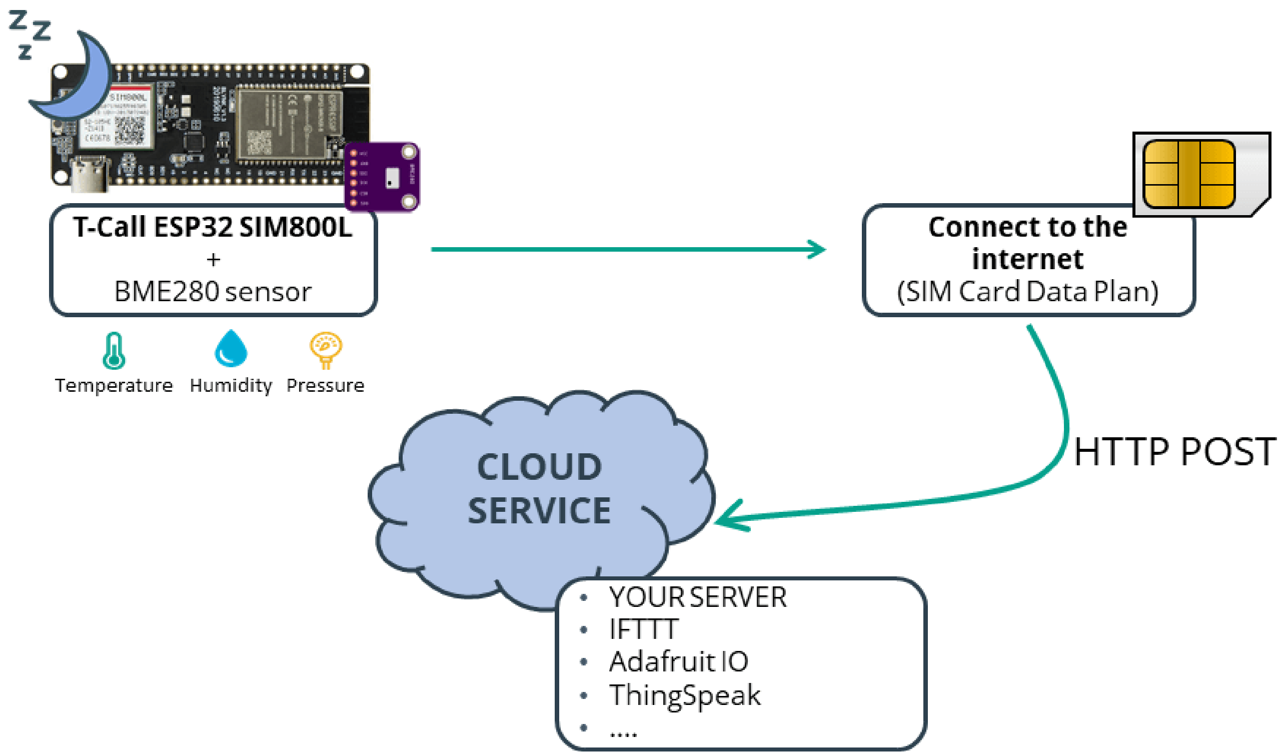
https://github.com/opendroneid/receiver-android



Remote Server

Static and Dynamic data must be transmitted via GPRS/GSM also to a remote server via HTTP post in the following way:

- Authentication URL: URL of the UTM Platform authentication service. Once the user is authenticated, the service will response with a token that will be inserted in the header of the position messages sent afterwards in order to be accepted by the tracking service.
- Username/Password: Remote server user credentials;
- Tracking service URL: URL of the remote server service, which receives the position messages.



TTGO T-Call remote server sample configuration

Data transmitted (at least) to remote server via GSM/GPRS (HTTP Post)

- Drone ID format: ABCDF1FG916003A12345 (example, hardcoded in firmware)
- Pilot ID format: ABCDEFGHIJKLMNOPQRST (example, hardcoded in firmware)
- Operator data (gathered from user's smartphone)
    - Timestamp
    - Latitude
    - Longitude
- Drone data (gathered from GPS connected to ESP32 UART)
    - TimeStamp
    - Latitude
    - Longitude
    - Altitude
    - Ground Speed

Software and Hardware information

Following you can find some links regarding OpenDroneID specifications, libraries and code samples.

You can find also the TTGO T-Call (ESP32) hardware specifications/features with libraries and code samples

OpenDroneID information

Main Web site: https://www.opendroneid.org/

Main github: https://github.com/opendroneid

OpenDroneID core: https://github.com/opendroneid/opendroneid-core-c

OpenDroneID specs: https://github.com/opendroneid/specs

OpenDroneID receiver (Android): https://github.com/opendroneid/receiver-android

TTGO T-Call information and software

Github URL: https://github.com/Xinyuan-LilyGO/TTGO-T-Call

Sample software: https://github.com/Xinyuan-LilyGO/TTGO-T-Call/tree/master/examples

List of the TTGO T-Call SIM800L ESP32 Board specifications:

- Chipset: ESP32 240MHz Xtensa® dual-core 32-bit LX6 microprocessor
- FLASH: QSPI flash 4MB / PSRAM 8MB
- SRAM: 520 kB SRAM
- Connectivity:
- WiFi 4 802.11 b/g/n (2.4 GHz) up to 150 Mbps
- Bluetooth 4.2 Classic + Bluetooth Low Energy (BLE)
- 2G GSM/GPRS via SIMcom SIM800L module + Nano SIM card slot
- Reset Button
- USB to TTL CP2104
- USB-C port for power and programming
- Interface: pin headers with UART, SPI, SDIO, I2C, PWM, PWM, I2S, IRGPIO, capacitor touch sensor, ADC, DAC
- Power supply:
- USB-C port: 5V 1A
- JST header: 3.7V Lithium battery (500 mA charging current)

Reading GPS NMEA data with ESP32

Following you can find a sample sketch to read NMEA strings from GPS connected to an ESP32's UART and sending location information to a remote web server

https://iotdesignpro.com/projects/nodemcu-esp8266-gps-module-interfacing-to-display-latitude-and-longitude

NMEA Simulator (for testing without real GPS connected to ESP32)

https://chrome.google.com/webstore/detail/nmea-simulator/dfhcgoinjchfcfnnkecjpjcnknlipcll

NMEA sentence generator to mimic vessel position, speed & heading; Wind speed & direction; Water temperature and depth.

NMEASimulator is an easy to use NMEA data sentence generator, ideal for quickly and simply producing a data stream for your connected NMEA devices.

Mimics Vessel movement, GPS fix information as well as environmental information such as Wind speed / direction and Water temperature / depth. It also provides the ability to generate waypoint information by "marking" the current position.

NMEASimulator generates NMEA sentences from seed data which you provide, applies a variance to mimic vessel movement, and then transmits these sentences via TCP/IP to connected devices.

Features:

- Transmit messages over Serial, TCP or Web Socket (for use with browser clients)

- Can operate in both TCP server (to accept client connections) or TCP client (sends data stream to a server) modes.

- Manual control of values and GPS fix status

NMEASimulator generates the following NMEA sentences:

GPGGA: GPS Fix Data

GPGSA: GPS DOP and active satellites

GPGLL: Geographic Position

GPRMC: Recommended Minimum Navigation Information

AIMWV: Wind Speed and Angle

AIMTW: Water Temperature

SDDPT: Water Depth

SDDBT: Water Depth

SDDBK: Water Depth

SDDBS: Water Depth

GPWPL: Waypoint Location

GPZDA: UTC Date and Time

!AIVDO: Own Vessel's Information

!AIVDM: Other Vessel's Information