Contents

1 Introduction

Welcome to the AR.Drone 2.0 Software Development Kit !

The AR.Drone 2.0 product and the provided host interface example have innovative and excit-

This SDK includes :

- this document explaining how to use the SDK, and describes the drone communications protocols;
-

2 AR.Drone 2.0 Overview

2.1 Introduction to quadrotor UAV4.34ewUAV4.34ew151c-(832 Drone)-250(2.0

(a) Throttle

(b) Roll

(c) Pitch

(a) Indoor (b) Outdoor

Figure 2.2: Drone hulls

2.2

(a) Ultrasound sensor

2.8 Video streaming, tags and roundel detection

The frontal camera is a CMOS sensor with a 90 degrees angle lens.

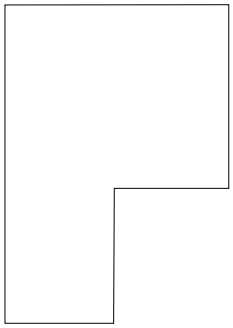
The AR.Drone automatically encodes and streams the incoming images to the host device.

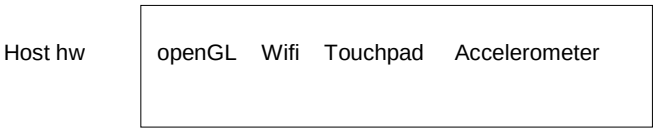
- the drone own IP address plus a number between 1 and 4 (for AR.Drone 2.0 and AR.Drone 1.0 after 1.1.3 version)

5. the client device can start sending requests the AR.Drone IP address and its services ports.

2.10 Communication services between the AR.Drone 2.0 and a client device

Cameras





3.2 The AR.Drone 2.0 Library

  - **VPOS** : multiplatform (Linux / Windows / Parrot proprietary platforms) wrappers for system-level functions (memory allocation, thread management, etc.)
  - **VPCOM** : multiplatform wrappers for communication functions (over Wifi, Bluetooth, etc.)
  - **VPAPI** : helpers to manage video pipelines and threads

Let's now detail the `ARDroneTool` part :

- *ardrone_tool.c* : contains a ready-to-use *ardrone_tool_main* C function which initialises the Wifi network and initiates all the communications with the drone
- **UI** : contains a ready-to-use gamepad management code
- **AT**

- a *control* thread which handles requests from other threads for sending reliable commands from the drone, and automatically checks for the drone acknowledgements.
-

4 ARDroneLIB

-

5 Creating an application with ARDroneTool

The `ARDroneTool` library includes all the code needed to start your application. All you have

5.2 Customizing the client initialization

Listing 5.1: Application initialization with ARDroneLIB

```
int ardrone_tool_main(int argc, char *
```

Listing 5.3: Declare a *navdata* management function

```
BEGIN_NAVDATA_HANDLER_TABLE //Mandatory
  NAVDATA_HANDLER_TABLE_ENTRY(navdata_ihm_init, navdata_ihm_process,
      navdata_ihm_release,
NULL)
END_NAVDATA_HANDLER_TABLE //Mandatory
//Definition for init, process and release functions.
C_RESULT navdata_ihm_init( mobile_config_t* cfg )
{ ... }

C_RESULT navdata_ihm_process( const navdata_unpacked_t* const pnd )
{ ... }

C_RESULT navdata_ihm_release( void )
{ ... }
```

Listing 5.4: Example of *navdata* management function

```
/* Receving navdata during the event loop */
inline C_RESULT demo_navdata_client_process( const navdata_unpacked_t* const
    navdata )
{
  const navdata_demo_t* const nd = &navdata->navdata_demo;

  printf("Navdata for flight demonstrations\n");

  printf("Control state : %s\n",ctrl_state_str(nd->ctrl_state));
  printf("Battery level : %i/100\n",nd->vbat_flying_percentage);
  printf("Orientation   : [Theta] %f  [Phi] %f  [Psi] %f\n",nd->theta,nd->phi,nd->
      psi);
  printf("Altitude      : %ig0G00.30rg00.30RG[-600(%)]TJ0g0G00.30rg00.30RG[(i)]TJ0g0G00.30rg00.30RG
   )
```

5.4

5.6 Managing the video stream

6 AT Commands

AT commands are text strings sent to the drone to control its actions.

Those strings are generated by the `ARDroneLIB` and `ARDroneTool` libraries, provided in the SDK. Most developers should not have to deal with them. Advanced developers who would like to rewrite their own AR.Drone middle ware can nevertheless send directly those commands to the drone inside UDP packets on port UDP-5556, from their local UDP-port 5556 (using the same port numbers on both sides of the UDP / IP link is a requirement in the current SDK).

Note : According to tests, a satisfying control of the AR.Drone 2.0 is reached by sending the AT-commands every 30 ms for smooth drone movements. To prevent the drone from considering the WIFI connection as lost, two consecutive commands must be sent within less than 2 seconds.

- Always send 1 as the sequence number of the first sent command.

- Always send commands with an increasing sequence number. If several software threads send commands to the drone, generating the sequence number and sending UDP packets should be done by a single dedicated function protected by a mutual exclusion mechanism.

6.3 Floating-point parameters

Let’s see an example of using a *float*

The drone translation speed in the horizontal plane depends on the environment and cannot be determined. With roll or pitch values set to 0, the drone will stay horizontal but continue sliding in the air because of its inertia. Only the air resistance will then make it stop.

The vertical speed (aka. "gaz") argument is a percentage of the maximum vertical speed as defined here. A positive value makes the drone rise in the air. A negative value makes it go down.

The angular speed argument is a percentage of the maximum angular speed as defined here. A positive value makes the drone spin right; a negative value makes it spin left.

The psi argument is a normalized psi angle from north provided by magnetometer sensor as defined here. An angle value of 0 means that the controller is facing north. A positive value means that the controller is oriented to the east and a negative value is orienting to the west. 1 and -1 value are the same orientation. (only for AT*PCMD_MAG)

The psi accuracy argument is an accuracy of the magnetometer sensor. This value represents the maximum deviation of where the magnetic heading may differ from the actual geomagnetic heading in degrees. Negative values indicates the invalid heading. (only for AT*PCMD_MAG)

Description :
This command asks the drone to calibrate the drone magnetometer. This command *MUST* be sent when the AR.Drone is flying.

When receiving this command, the drone will automatically calibrate its magnetometer by spinning around itself for a few time.

AT*CONFIG ___

7 Incoming data streams

The drone provides its clients with two main data streams : the navigation data (aka. *navdata*) stream, and the video stream.

This chapter explains their format. This is useful for developers writing their own middleware. Developers using `ARDroneTool` can skip this part and directly access these data from the

All the blocks share this common structure :

Listing 7.1: Navdata option structure

```
typedef struct _navdata_option_t {
uint16_t tag; /* Tag for a specific option */
uint16_t size; /* Length of the struct */
uint8_t data[]; /* Structure complete with the special tag */
} navdata_option_t;
```

The most important *options* are *navdata_demo_t*, *navdata_cks_t*, *navdata_host_angles_t* and *navdata_vision_detect_t*. Their content cc4 Tf -2S3r-2S3rSounducture, mainly *navdata_common.h*TnS2th

7.1.3 Augmented reality data stream

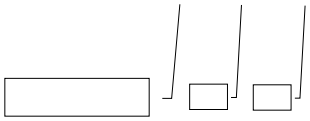
In the previously described NavData, there are informations about vision-detected tags. The goal is to permit to the host to add some functionalities, like augmented reality features. The principle is that the AR.Drone sends informations on recognized pre-defined tags, like type and position.

7.2 The AR.Drone 1.0 video stream

Two codecs are available UVLC (MJPEG-like) and P264 (H.264-like).

UVLC features :

colorspace	YUV 4:2:0
transform	8x8 dct
entropy coding	



Each macroblock contains informations of a 16x16 image, in YC_BC_R format, type 4:2:0.

(see http://en.wikipedia.org/wiki/YCbCr,
http://en.wikipedia.org/wiki/Chroma_subsampling,
http://www.ripp-it.com/heita9sys/motit.c5(420it.com146it.comlettreit.c5(tousit.com

The motion vector has a pixelic precision for luma component and half pixel precision for chroma component due to chroma subsampling. Therefore Chroma needs to be interpolated to access sub pixels (refer to h.264 specification).
Today, P264 doesn't allow macroblock fragmentation for motion estimation. Only one motion vector is computed for the entire 16x16 macroblock. The reference frame is always the previous encoded / decoded frame.

7.2.3.3 Residual data

7.2.4 Specific block entropy-encoding

The proprietary format used to encode blocks is based on a mix of RLE and Huffman coding (cf. http://en.wikipedia.org/wiki/Run-length_encoding and http://en.wikipedia.org/wiki/Huffman_coding).

- Initialize the counter of successive zero-values at zero.
- For each of the remaining 16-bits values of the list:
  - If the current value is zero: For each of the remaining 16-bits values of the 76551alues of thi9a6(For)-2

    - Else: *)* Resulting value (zero-counter) is equal to the direct decimal conversion of the merged read binary values. Ex: if *xxxx* = $1101_{(2)}$ *!* $000001_{(2)}$ + $1101_{(2)}$ = $0000011101_{(2)}$ = $29_{(10)}$
- Add "0" to the output list, as many times indicated by the zero-counter.
- Reading of the non-zero value as explain below:
    - Read the *coarse* pattern part (bit-per-bit, till there is 1 value).
    - On the corresponding line (cf.Figure 15), get the number of complementary bits

55 7.2.5.9

Inter Macroblock :

PARTITION LIST – list of mb subdivision for motion estimation:
Always read as ’000’ because P264 doesn’t support macroblock partition.

MOTION VECTOR LIST – list of motion vector associated to each partition:
There is only one motion vector per macroblock. The vector is not put in the stream directly. A predicted motion vector for the current macroblock is determined with the already transmitted

7.3.4 Latency reduction mecanism

The TCP transmission allow the application to recieve all frames from the live stream, but this can introduce latency.
A latency reduction mecanism is implemented inside the *Video/video_stage_tcp.c* file. This

Note : The legacy (1.6 SDK) *executeCommandIn*

8.3 Multiconfiguration

Starting with firmware 1.6.4, the AR.Drone supports different configurations depending on the application, user and session connected.

8.3.8 Technical details on id generation and descriptions

The `ARDroneTool` generates the different ids the following way :

The application id is generated as a CRC32 of a string composed of the application name and

8.4 General configuration

All the configurations are given accordingly to the *config_keys.h* file order.

In the API examples, the *myCallback* argument can be a valid pointer to a callback function, or a NULL pointer if no callback is needed by the application.

GENERAL:num_version_config _____ *CAT_COMMON | Read only*

Description :
Version of the configuration subsystem (Currently 1).

GENERAL:num_version_mb _____

GENERAL:ardrone_name _

Note :

8.5 Control configuration

CONTROL:accs_offset

CAT_COMMON | Read only

Description :
Parrot internal debug informations. AR.Drone accelerometers offsets.

CONTROL:accs_gains

CAT_COMMON | Read only

Description :
Parrot internal debug informations. AR.Drone accelerometers gains.

CONTROL:pwm_ref_gyros ___

```
float eulerMax = 0.25;
ARDRONE_TOOL_CONFIGURATION_ADDEVENT (euler_angle_max, &eulerMax, myCallback);
```

CONTROL:altitude_max _

Enabling this can cause unexpected behaviour on commercial AR.Drone .

CONTROL:manual_trim _

Note : Oriented Black&White Roundel detection must be activated with the *DETECT:detect_type*

8.7 Nav-board configuration

8.8 Video configuration

VIDEO:camif_fps __

API use example :

API use example :

```
uint32_t newMaxBitrate = 4000;
ARDRONE_TOOL_CONFIGURATION_ADDEVENT (max_bitrate, &newMaxBitrate, myCallback);
```

In all other cases (key set to "FALSE" or no USB key plugged), the record stream is sent to the controlling device, which will be in charge of the actual recording.

AT command example : **AT*CONFIG=605,"video:video_on_usb","TRUE"**

API use example :

```
bool_t recordOnUsb = TRUE;
ARDRONE_TOOL_CONFIGURATION_ADDEVENT (video_on_usb, &recordOnUsb, myCallback);
```

VIDEO:video_file_index ____

8.9 Leds configuration

8.10 Detection configuration

Any other values are either deprecated or in development.

Note : It is advised to enable the multiple detection mode, and then configure the detection needed using the following keys.

Note : The multiple detection mode allow the selection of different detections on each camera. Note that you should NEVER enable two similar detection on both cameras, as this will cause failures in the algorithms.

Note : The Black&White oriented roundel can be downloaded on Parrot website

AT command example : **AT*CONFIG=605,"detect:detect_type","10"**

API use example :

```
CAD_TYPE detectType = CAD_TYPE_MULTIPLE_DETECTION_MODE;
ARDRONE_TOOL_CONFIGURATION_ADDEVENT (detect_type, &detectType, myCallback);
```

8.11 SYSLOG section

This section is a Parrot internal debug section, and should therefore not be used.

8.12 USERBOX section

USERBOX:userbox_cmd -

8.13 GPS section

8.14 CUSTOM section - Multiconfig support

CUSTOM:application_id _____

9

10 Building the Linux

10.4 Run the *Navigation* program

Before running any demo, make sure that your computer is connected to your AR.Drone

You can test the connection with a ping command :

Fly

Now press the button you chose as the *select* button. Press it several times to make the motors

11 Building the android example

The android examples will be provided in later releases of the AR.Drone SDK.

Documentation will be updated with build instructions.