Prepared	Title		
Stephane Piskorski Nicolas Brulez	AR.Drone Developer Guide		
Approved	Date February 24, 2011	Revision SDK 1.6	File

Notations used in this document :

```
$  This is a Linux shell command line (the dollar sign
represents the shell prompt and should not be typed)
This is a Linux shell output (do not type this)
```

Contents

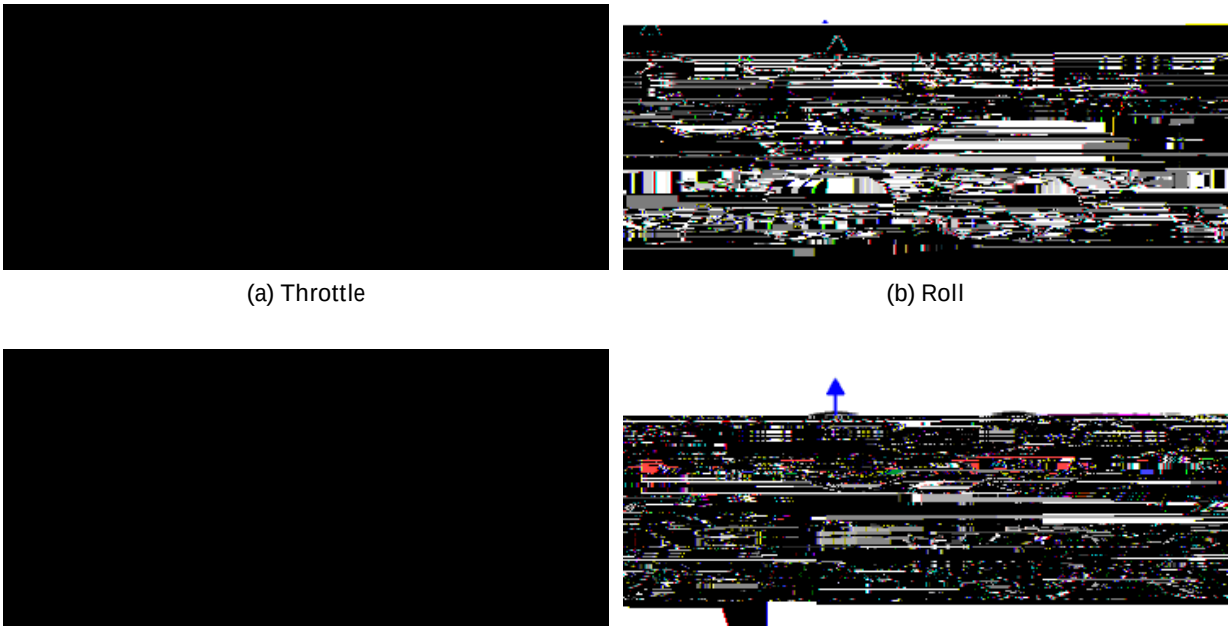
1 Introduction

Welcome to the AR.Drone Software Development Kit !

The AR.Drone product and the provided host interface example have innovative and exciting features such as:

- intuitive touch and tilt flight controls
- live video streaming and photo shooting
- updated Euler angles of the AR Drone
- embedded tag detection for augmented reality games

The AR.Drone SDK allows third party developers to develop and distribute new games based on AR.Drone product for Wifi, motion sensing mobile devices like game consoles, the Apple iPhone, iPod touch, the Sony PSP, personal computers or Android phones.



(a) Throttle (b) Roll

(c) Pitch (d) Yaw

Figure 2.1: Drone movements

Manoeuvres are obtained by changing pitch, roll and yaw angles of the AR.Drone .

Varying left and right rotors speeds the opposite way yields roll movement. This allows to go forth and back.

(a) Ultrasound sensor (b) Camera

Figure 2.3: Drone Sensors

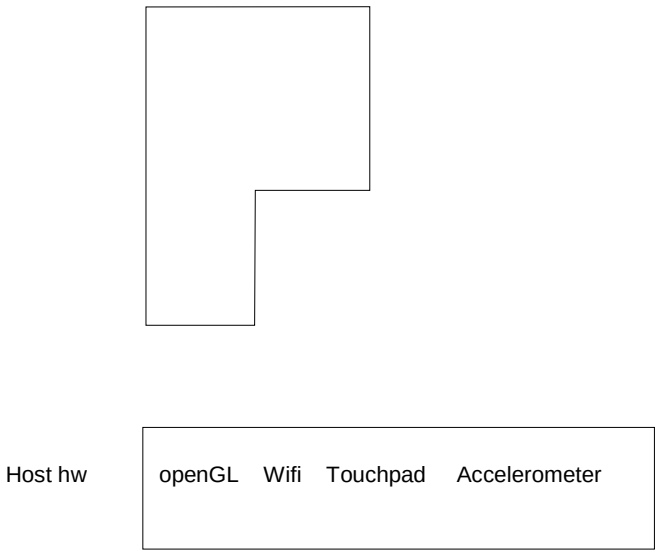
2.4 LiPo batteries

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

Controlling the AR.Drone is done through 3 main communication services.

Controllingcnfigurling theone isone

3 AR.Drone SDK Overview



3.2 The AR.Drone Library

The AR.Drone Library is currently provided as an open-source library with high level APIs to access the drone.

Let's review its content :

- **S**

Let's now detail the `ARDroneTool` part :

- *ardrone_tool.c* : contains a ready-to-use *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** : contains all the functions you can call to actually control the drone. Most of them directly refer to an AT command which is then automatically built with the right syntax

The flags argument is a bitfiels containing the following informations :

- Bit 0 : when Bit0=0 the drone will enter the *hovering* mode, i.e. try to stay on top of a fixed point on the ground, else it will follow the commands set as parameters.
- Bit 1 : when Bit1=1 AND CONTROL:control_level configuration Bit1=1, the new Com-
  bipsa

5 Creating an application with ARDroneTool

The `ARDroneTool` library includes all the code needed to start your application. All you have to do is writing your application specific code, and compile it along with the `ARDroneLIB` library to get a fully functional drone client application which will connect to the drone and

5.2 Customizing the client initialization

As is true for every C-based application, the initial entry point for every AR.Drone application

Once these functions are written, you must register your to `ARDroneTool` by using *ardrone_tool_input_add* function which takes a structure parameter *input_device_t* defined in *<ardrone_tool/UI/ardrone_input.h>*. This structure holds the pointers to the three above-mentioned functions.

Structure *input_device_t*

6

6

- Always send

6.6 Commands description

Note :
The names "start" and "select" come from previous versions of the SDK when take-off and landing were directly managed by pressing the select and start buttons of a game pad.

Example :
The following commands sent in a standalone UDP packet will send an emergency signal :

```
AT*REF=1
```

AT*FTRIM .

AT*CONFIG _

All the blocks share this common structure :

Listing 7.1: Navdata option structure

7.1.3 Augmented reality data stream

7.2.1.4 Picture resolution (PRESOLUTION) (3 bits)

Picture resolution which is used in combination with the picture format (3 bits)

- 000 : forbidden
- 001 : for CIF it means sub-QCIF
- 010 : for CIF it means QCIF
- 011 : for CIF it means CIF
- 100 : for CIF it means 4-CIF
- 101 : for CIF it means 16-CIF

7.2.1.5 Picture type (PTYPE) (3 bits)

Picture type:

- 000 : INTRA picture
- 001 : INTER picture

7.2.1.6 Picture quantizer (PQUANT) (5 bits)

The PQUANT code is a 5-bits-long word. The quantizer reference for the picture that range from 1 to 31.

7.2.1.7 Picture frame (PFRAME) (32 bits)

The frame number (32 bits).

7.2.1.8 Layer groups obr

- Bit 4 : 1 means there is non dc coefficients for block cb.
- Bit 5 : 1 means there is non dc coefficients for block cr.
- Bit 6 : 1 means there is a quantization value following this code.
- Bit 7 : Always 1 to avoid a z310(1)to

-

    - 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).

8.2.2 From the Control Engine for iPhone

Using the Control Engine for iPhone, the ARDrone instance of your application can accept messages to control some parts of the configuration.

The message is *-(void) executeCommandIn:(ARDRONE_COMMAND_IN)commandId withParameter:(void *)parameter fromSender:(id)sender*.

The first parameter is the ID of the config you want to change. The ID list can be found in the *ARDroneTypes.h*.

The second parameter is the parameter of the command.
As 1.6 SDK, parameter types are the following :
- ARDRONE_COMMAND_ISCLIENT : raw integer casted to (void *).
- ARDRONE_COMMAND_DRONE_ANIM : ARDRONE_ANIMATION_PARAM pointer.
- ARDRONE_COMMAND_DRONE_LED_ANIM : ARDRONE_LED_ANIMATION_PARAM pointer.
- ARDRONE_COMMAND_VIDEO_CHANNEL : raw integer casted to (void *). Values can be found in *ARDroneGeneratedTypes.h* file
- ARDRONE_COMMAND_CAMERA_DETECTION : raw integer casted to (void *). Values can be found in *ARDroneGeneratedTypes.h* file
- ARDRONE_COMMAND_ENEMY_SET_PARAM : ARDRONE_ENEMY_PARAM pointer.
- ARDRONE_COMMAND_ENABLE_COMBINED_YAW : boolean casted to (void *).
- ARDRONE_COMMAND_SET_FLY_MODE : raw integer casted to (void *). Values can be found in *ardrone_api.h* file

The third parameter is currently unused (passing *nil* is ok).

GENERAL:video_enable _

8.4 Control configuration

CONTROL:accs_offset _

CONTROL:control_level _____

setting.

AT command example : **AT*CONFIG=605,"control:control_vz_max","1000"**

API use example :

```
ARDRONE_TOOL_CONFIGURATION_ADDEVENT (control_vz_max, 1000, myCallback);
```

CONTROL:control_yaw .

CONTROL:outdoor_control_vz_max _____

8.5 Network configuration

The UDP socket port where the AT*commands are sent. Defaults to 5556

NETWORK:cmd_port _____

8.6 Nav-board configuration

PIC:ultrasound_freq __

8.7 Video configuration

VIDEO:camif_fps _____

Current implementation supports 4 different channels :
- ARDRONE_VIDEO_CHANNEL_HORI
- ARDRONE_VIDEO_CHANNEL_VERT
- ARDRONE_VIDEO_CHANNEL_LARGE_HORI_SMALL_VERT
- ARDRONE_VIDEO_CHANNEL_LARGE_VERT_SMALL_HORI

AT command example : **AT*CONFIG=605,"video:video_channel","2"**

API use example :

8.9 Detection configuration

DETECT:enemy_colors _____

DETECT:detections_select_h .

```
    ARDRONE_TOOL_CONFIGURATION_ADDEVENT (detections_select_v, TAG_TYPE_-
MASK (TAG_TYPE_ROUNDEL), myCallback);
```

11 Building the Linux Examples

The AR.Drone SDK provides two client application examples for Linux.

The first one, called *Linux SDK Demo*, shows the minimum software required to control an AR.Drone with a gamepad. It should be used as a skeleton for all your Linux projects.

To build the examples, you will need libraries like IW (for wireless management), gtk (to display an graphic interface), and SDL (to easily display the video stream) :

Check the printed messages; it should say a gamepad was found and then initialize a bunch of things :

Now press the button you chose as the *select* button. Press it several times to make the motorsthiswit o24.8893 4

11.6 Run the *Navigation* program

First unpair your drone if you flew with an iPhone before !

Although the demonstration program can configure the WIFI network itself, it is safer for a

12.2 Required settings in the source code before compiling

Operating System related settings

12.4 What to expect when running the example

Before running the example, one of the computer network connections must be connected with the drone. Pinging the drone or connecting to its telnet port MUST work properly. The drone MUST NOT be paired with an iPhone (using the drone with an iPhone prevents the drone from accepting connections from any other device, including a PC, unless the unpair button (below

The example ime95y5

13 Other platforms

13.1 Android example

Please refer to the *<SDK>/Examples/Android/ardrone* directory, and its two files *INSTALL* and *README* to compile the Android example. It was successfully tested (controls and video display) on a rooted Google / HTC Nexus One phone, using NDK r3.