一、 关于 SDK 目录

--akeeta 雅观的业务文件

--inc 业务 API 头文件

ya_app_main.h：包含启动 API，以及启动函数所需要的参数配置等；

ya_mode_api.h：包含模块模式设置 API，例如进入 smart cofig，进入热点状态等；

ya_thing_data_api.h：包含数据处理 API，例如上报数据到云端，发送数据到串口等；

--hal/inc 适配层 API 头文件

ya_hal_flash.h：flash 操作接口

ya_hal_net.h.h：网络 API 接口；

ya_hal_os.h：系统 API 接口；

ya_hal_timer.h：时间 timer 的 API 接口；

ya_hal_wdt.h：看门狗 API 接口；

ya_hal_wlan.h：wifi 的 API 接口；

--- lib_akeeta.a 包含雅观所有业务的库文件。

--io 关于芯片 io 的配置文件

ya_hal_button.h：gpio 通用的输入配置文件；

ya_hal_gpio.h：gpio 通用的输出配置文件；

ya_hal_pwm.h：pwm 通用的配置文件；

ya_hal_uart.h：串口 API 函数。

--user 用户需要实现的 API 文件

ya_api_thing_uer_define.h：用户需要实现的 API 头文件。

--test_example 测试用例文件夹

uart：串口的测试用例；

outlet：outlet 产品的测试用例。

二 、重要 API 说明

1 ya_app_main.h

1）CLOUD_T

```
typedef enum
{
```

```
    /* the cloud is oversea */
    AKEETA_US = 0,
    /* the cloud is in China */
    AKEETA_CN,

    UNKNOWN_CLOUD_TYPE = 0xFF,
}CLOUD_T;
```

使用说明：云端类型枚举：AKEETA_US 表示雅观海外云端；AKEETA_CN 表示雅观国内云端。

2）ya_init_module_mode

```
typedef enum{
    /* AP mode. The module will start AP/router with ssid is "Argrace_xxxxx". The "xxxx" is from the last two bytes of mac address.
      Note this mode only used in factory state. When the module has obj router ssid and pwd, it will connect the router automatically*/
    AP_MODE = 0,

    /* sniffer mode.
    Note this mode only used in factory state. When the module has obj router ssid and pwd, it will connect the router automatically*/
    SNIFFER_MODE,

    CONNECT_MODE,

    IDLE_MODE,
}ya_init_module_mode;

/* The product version length, the product version must meet the format "xx.xxx.xxx.xxx". */
#define     VERSION_LEN   14
```

使用说明：模组出厂初始状态枚举：AP 模式、sniffer 模式以及空闲模式。

3）ya_app_main_para_t

```
typedef struct
{
    /* cloud    type */
    CLOUD_T                 cloud_type;

    /* intial mode when module is in factory state */
    ya_init_module_mode     ya_init_mode;

    /* sniffer timeout, we suggest 5 minutes or less. */
    uint32_t                sniffer_timeout;

    /* AP timeout, we suggest 5 minutes or less. */
    uint32_t                ap_timeout;

    /* The product version and it must meet the format "xx.xxx.xxx.xxx". The last three chars is from "000" to "999"    */
    char                    cur_version[VERSION_LEN + 1];

    /* enable_factory_uart is used to enble the akeeta uart. akeeta uart is used to write and read licenses.
    If the modlue has license and user has itself uart protocol, user should disable it. */
    bool                    enable_factory_uart;
}ya_app_main_para_t;
```

使用说明：启动雅观业务所需要的参数。

4）ya_test_akeeta_cn_with_obj_cert

```
/**
 * @brief This function is used to set the user obj akeeta CN cert for test. If using the Akeeta cert tools to set the cert, there is no need to use this function.
 *
 * @return 0: sucess, -1: failed
 */
extern int ya_test_akeeta_cn_with_obj_cert(char *productkey, char *deviceid, char *secret);
```

使用说明：第三种方式写入国内证书需要调用的 API。

5）ya_test_akeeta_us_with_obj_cert

```
/**
 * @brief This function is used to set the user obj akeeta US cert for test. If using the Akeeta cert tools to set the cert, there is no need to use this function.
 *
 * @return 0: sucess, -1: failed
 */
extern int ya_test_akeeta_us_with_obj_cert(char *certificateBuff, char *privateKeyBuff, char *client_id,    char *thing_type);
```

使用说明：上述第三种方式写入国外证书需要调用的 API。

6）ya_start_app_main

```
/**
 * @brief This function is the one used to start the main app.
 *
 * @return 0: sucess, -1: failed
 */
extern void ya_start_app_main(ya_app_main_para_t ya_app_main_para);
```

使用说明：雅观业务的启动函数。

2 ya_mode_api.h

1）ya_app_status_t

```
typedef enum
{
    YA_APP_IDLE                         = 0x00,
```

```
    YA_APP_SNIFFER_START,
    YA_APP_SNIFFER,

    YA_APP_AP_START,
    YA_APP_AP_STATE,

    YA_APP_TO_CONNECT,
    YA_APP_CONNECTING,
    YA_APP_CONNECTED,
}ya_app_status_t;
```

使用说明：模组状态枚举：AP 状态、sniffer 状态以及连接路由状态。

2）ya_get_app_status

```
/**
 * @brief This function is used to get the module state.
 *
 * @return 0: sucess, -1: failed
 */
extern ya_app_status_t ya_get_app_status(void);
```

使用说明：查询模组状态。

3）ya_reset_module_to_factory

```
/**
 * @brief This function is used to reset the module to factory mode. The module will erase all flash data and reboot.
 *
 * @return 0: sucess, -1: failed
 */
extern int32_t ya_reset_module_to_factory(void);
```

使用说明：恢复模组为出厂配置。

3）ya_set_toggle_mode

```
/**
 * @brief This function is used to set the module to toggle mode
 *
 * @param change_with_power_on: 1: when power on,  choose the toggle mode, 0: when normal running, choose the toggle mode and reboot.
 * @return 0: sucess, -1: failed
 */
extern int32_t ya_set_toggle_mode(uint8_t change_with_power_on);
```

使用说明：设置模组为另外一种模式，假如模组当前为 smart config 模式，则模组切换为 AP 模式；假如模组为 AP 模式，则模组切换为 sniffer 模式。参数 change_with_power_on 表示模组是否在上电的状态进行切换。

4）ya_set_sniffer_mode

```
/**
 * @brief This function is used to set the module to sniffer mode. The module will erase all flash data and reboot.
 *
 * @return 0: sucess, -1: failed
 */
extern int32_t ya_set_sniffer_mode(void);
```

使用说明：设置模组为 smart config 模式。

5）ya_set_ap_mode

```
/**
 * @brief This function is used to set the module to AP mode. The module will erase all flash data and reboot.
 *
 * @return 0: sucess, -1: failed
 */
```

```
extern int32_t ya_set_ap_mode(void);
```

使用说明：设置模组为 ap 模式，热点的名称是“Argrace_XXXX”，其中“XXXX”表示设备 MAC 地址的最后两个字节。

3 ya_thing_data_api.h

1）ya_thing_report_to_cloud

```
/**
 * @brief This function is used to handle the translate from thing mode to cloud json.
 *
 * @return 0: sucess, -1: failed
 */
extern int32_t ya_thing_report_to_cloud(uint8_t *buf, uint16_t len);
```

使用说明：上报数据到云端。

2）ya_put_data_uart_into_queue

```
/**
 * @brief This function is used to put data into queue, and wait the task to send it
 *
 * @return 0: sucess, -1: failed
 */
extern void ya_put_data_uart_into_queue(uint8_t *data, uint16_t data_len);
```

使用说明：将数据发送到串口队列中。

三、第三方开发者需要实现的 API 说明

1）ya_thing_handle_downlink_data

```
/**
```

The following funciton is realized by the user according to the thing mode.
**/

```
/**
 * @brief This function is the one used to handle downlink contorl by cloud
 * @param buf: json string
 * @param len: json string len
 */
extern void ya_thing_handle_downlink_data(uint8_t *buf, uint16_t len);
```

使用说明：该 API 给用户处理云端发送下来的数据，由用户实现。

2）ya_thing_handle_cloud_status

```
/**
 * @brief This function is the one used to handle the cloud event and is defined by the user.
 *
 */
typedef enum{
    YA_CLOUD_OFFLINE = 0,
    YA_CLOUD_ONLINE,
}ya_cloud_status_t;
extern void ya_thing_handle_cloud_status(ya_cloud_status_t ya_cloud_status);
```

使用说明：该 API 给用户处理云端在线与离线时的业务处理，由用户实现。

3）ya_thing_handle_router_status_callback

```
/**
 * @brief This function is the one used to handle the router event and and is defined by the user.
 *
 */
typedef enum{
    MODULE_AP = 0,
    MODULE_SNIFFER,
    MODULE_IDLE,
```

```
    MODULE_CONNECTING,
    MODLUE_CONNECT,

    MODULE_UNKNOWN = 0xFF,
}ya_router_status_t;

typedef void (*ya_thing_handle_router_status)(ya_router_status_t ya_app_router_status);

extern void ya_thing_handle_router_status_callback(ya_router_status_t ya_app_router_status);
```

使用说明：当模块有状态改变，模块会通过这 API 回调给用户给，用户结合业务是否需要处理，由用户实现。

4）ya_thing_app_tick

```
/**
 * Whenever app application tick is called, this callback will be called at the end of the app tick execution.
 *
 */
extern void ya_thing_app_tick(void);
```

使用说明：雅观 SDK 会循环调用这个用户自定义的业务函数，用户需要循环调用的函数可以放置在改 API 中。