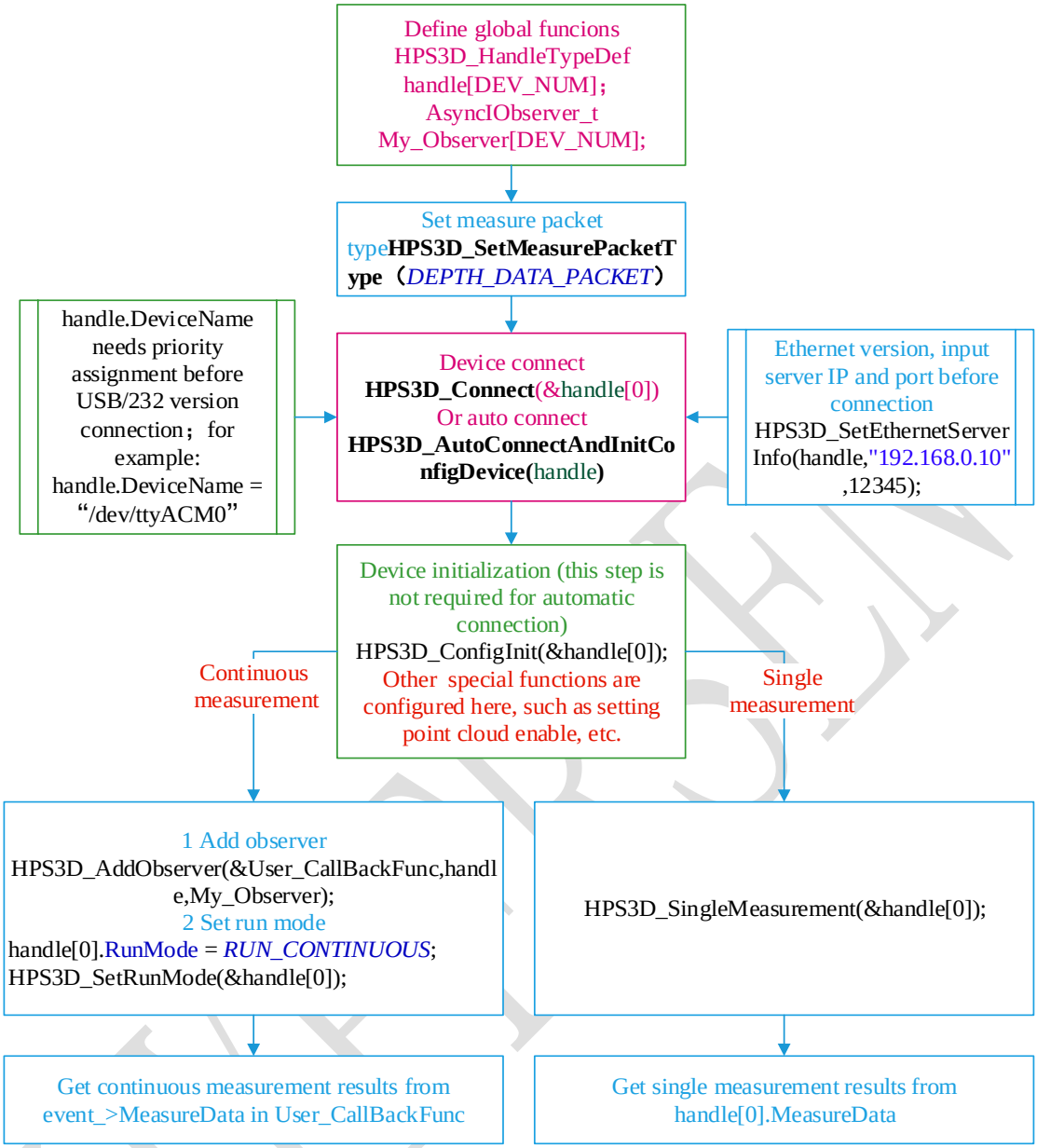
HPS-3D160 LiDAR SDK QUICK START GUIDE

1、SDK introduction

The SDK provides the application interface of the HPS3D160 Solid-State LiDAR, which is currently available on the Linux platform, the Windows platform, the ROS platform, and most of the microcontrollers that do not run the operating system; the SDK is a secondary development kit tool, and the interface provided includes most of the operating instructions of the HPS3D160 Solid-State LiDAR developed by our company. More details please refer to HPS-3D160 LiDAR user guide(HPS3D-RM002)。

Currently the SDK supports USB, RS232, and Ethernet versions. USB version communicate through virtual serial port. Ethernet version uses TCP/IP protocol and uses socket to transmit data.

2、SDK easy operation procedure



1) Define global parameters：

```
/*handle*/
typedef struct
{
    char *DeviceName;    /*R/W Names of all currently connectable devices (automatic filtering)*/
    uint32_t DeviceFd;    /*R    File descriptor to store the current connected device */
    uint8_t DeviceAddr;   /*R    Device address (also the frame ID) to store current connected device*/
    uint8_t ConnectionNumber;                /*R    Device connection number*/
    HPS3D_SynchronousTypedef SyncMode; /*R    Synchronous or asynchronous mode*/
    RunModeTypeDef RunMode;              /*R/W Run mode*/
    MeasureDataTypeDef MeasureData;      /*R    Synchronously measure data, */
```

```
    RetPacketTypedef RetPacketType;      /*R    Synchronous measure return packet type*/
    OutPacketTypeDef OutputPacketType;   /*R    Output data packet type*/
    bool ConnectStatus;                  /*R    Connect status*/
    uint8_t RoiNumber;                   /*R    Save ROI number the current device supported*/
    uint8_t ThresholdNumber;             /*R    Save threshold number the current device supported*/
    uint8_t ViewAngleHorizontal;         /*R    Horizontal view of angle*/
    uint8_t ViewAngleVertical;           /*R    Vertical view of angle*/
    struct sockaddr_in ServerAddr;       /*R/W Server IP address and port number*/
    TransportTypeDef TransportType;      /*R    Current transport type*/
}HPS3D_HandleTypeDef;
```

Note: R means only readable and unwritable; RW means readable and writable.

- HPS3D_HandleTypeDef handle[DEV_NUM];

Save the basic parameters of the connected device in this structure array to facilitate whole parameters access.

- AsyncIObserver_t My_Observer[DEV_NUM];

  Defining an observer event related structure for configuring the relevant parameters of the event notification by the observer.

  The observer mode is a notification callback event: the current callback function is notified when the current event is the same as the observer subscription event; the callback function is defined by the user according to the format

2) Set measure data packet type:

```
/* Set measure data packet type */
typedef enum
{
    DEPTH_DATA_PACKET = 0x0,    /*Depth data packet*/
    ROI_DATA_PACKET,            /*ROI data packet*/
    OBSTACLE_PACKE              /*Obstacle data packet*/
}MeasurePacketTypeDef;
```

The measure data packet includes the above three types, the default is a deep data packet, and the deep data packet includes the point cloud data at the same time；

- HPS3D_SetMeasurePacketType(*DEPTH_DATA_PACKET*); default type is depth data type.

3) Device connection.

Device connection includes manual connection and auto connection:

Manually connect is HPS3D_Connect(&handle); finish the following steps before performing this step:

- **USB or RS232 connection:** Need to get the corresponding device name and assign its name to the DeviceName in the handle. Note that you need to manually modify the device permissions to access in Linux platform.
- **Ethernet connection：** Input server IP and port number to get the corresponding network socket, ie HPS3D_SetEthernetServerInfo(&handle[0],"192.168.0.10",12345). Call this interface to set the server IP, and the obtained network socket will be saved in DeviceFd in the handle

- After the connection is completed, start device initialization according to step 4).

Auto connection is HPS3D_AutoConnectAndInitConfigDevice(handle); Selecting this connection method will automatically connect the currently connectable device and run initial configuration. The return value is the number of successful connections. If you choose this mode, you do not need to perform step4) device initialization.

4) Device initialization:
   - HPS3D_ConfigInit(handle); Calling this interface will perform a series of initialization operations on the device and dynamically allocate memory space. When removing the device, you need to call HPS3D_RemoveDevice() to release the resource.
   - In automatic connection mode, you do not need to perform this operation.
5) Set measure mode

   There are two types of measurement modes: continuous measurement and single measurement. Continuous measurement is asynchronous notification mode, and the measurement result needs to be monitored by adding observer mode; single measurement is synchronous mode, that is, when there is measurement data or data error, it returns.

- **Continuous measurement:**

```
/* Observer initialization */
My_Observer[0].AsyncEvent = ISubject_Event_DataRecvd; /* Asynchronous notification event for data reception */
My_Observer[0].NotifyEnable = true; /* Enable notification event */
My_Observer[0].ObserverID = 0; /*Observer ID*/

/*Observer callback function, user function
The continuous measurement data return will notify the callback function, and the measurement data is saved in the MeasureData in the event. */
void* User_Func_CallBack(HPS3D_HandleTypeDef *handle,AsyncIObserver_t *event)
{
    …
}

/*Add observer*/
HPS3D_AddObserver(&User_Func,handle,My_Observer);

/*Set continuous measurement mode and start measuring*/
handle[0].RunMode = RUN_CONTINUOUS;
HPS3D_SetRunMode(&handle[0]);
```

- **Single measurement:**

```
HPS3D_SingleMeasurement(&handle[0]);
The single measurement only needs to call the interface. The function is synchronous measurement mode. If the measurement return data is not obtained within the timeout period, it is considered that the measurement is invalid, and it needs to be called again to obtain the measurement result。
```

The measurement result is saved in MeasureData in the handle.

3、SDK FAQ

- The encoding format does not match?

We provide SDK encoding format is UTF-8, if the encoding format error occurs when using the API provided by the SDK, please create a new api.h file in the project, and copy the api.h content provided by the SDK to the new api.h file to solve the compilation error.

- Device connection fail?

When the device connection fails, open the debug mode, export debugging information and analyze the failure reason.

Sample code:

```
/*
 * Debugging use
 * */
void User_Printf(char *str)
{
	printf("%s\n",str);
}
/*enable debug mode*/
HPS3D_SetDebugEnable(true);
HPS3D_SetDebugFunc(&User_Printf);
```

Common fail reasons: The /dev/ttyACM device on the Linux platform does not have permission: manually modify the device permissions sudo chmod 777 /dev/ttyACM0.
Ethernet version connection failure, analyze the errno returned when socket is created to find error reason. Note that the Ethernet setting server IP interface is used to transmit the parameter format.
HPS3D_SetEthernetServerInfo(&handle[0],"192.168.0.10",12345);

- How to get point cloud data?

The conversion of depth data to point cloud has been achieved internally. Get point cloud data by setting point cloud enable.
Before enabling the point cloud data, ensure that the optical compensation enable is enabled, otherwise the correct point cloud result cannot be obtained.

Sample code:

```
HPS3D_SetOpticalEnable(&handle[i],true); /*enable optical compensation*/
HPS3D_SetPointCloudEn(true); /*enable point cloud output*/
```

Note: This operation must be performed after the device initialization is completed; and the point cloud data will also be saved in MeasureData. Please read the description of the

structure in detail.

- How to edit sensor default IP for Ethernet version?

Default IP for Ethernet version is 192.168.0.10, default subnet mask is 255.255.255.0, default gateway is 192.168.0. Modify through client software if it is needed. Please refer to instructions about HPS3D_Client.exe to reset server IP.

Sdk reset server IP, sample code:

```
uint8_t serverIP[4] = {192,168,2,10};
uint8_t netMask[4] = {255,255,255,0};
uint8_t geteway[4] = {192,168,2,1};
HPS3D_ConfigEthernet(&handle,serverIP,netmask,getway);
```

After the reset, the server IP is temporally in effect, it will reset to factory setting when power is restored. if you need to save permanently, you need to save the communication configuration. Please remember the modified IP, otherwise you will not be able to connect again.

```
HPS3D_SaveTransportConf(&handle);
```

Note: After the reset, communication will not be possible. Need to disconnect and connect again using the new IP, and save to communication configuration for permanent effectiveness.