YOST LABS
630 Second Street
Portsmouth Ohio 45662, USA

Phone: 740.876.4936
info@yostlabs.com
yostlabs.com

AN2016.01

Using the Prio API for wireless streaming with a PrioVR™ Basestation

Nick Leyder
Lead Software Engineer, Yost Labs

Introduction

The PrioVR Suit is a full-body suit that tracks motion using 19 PrioVR Sensors and a central PrioVR Hub. The PrioVR Sensors are inertial measurement units that give accurate drift-free absolute orientations of the user's body. The PrioVR Hub collects the data from the PrioVR Sensors, and sends it wirelessly to the PrioVR Basestation. The PrioVR Basestation communicates the received data to the host computer via USB (shown in Figure 1).

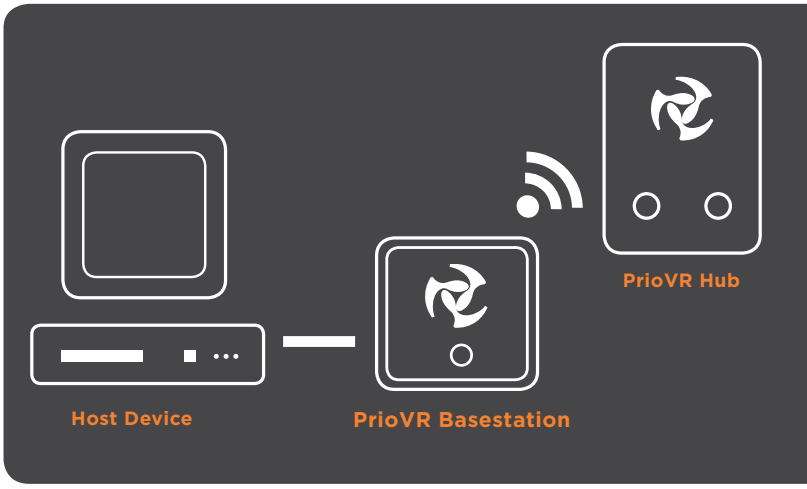

Figure 1

The purpose of this document is to provide an overview of streaming data from a PrioVR Suit using the PrioVR Basestation. The example code below illustrates the use of the Prio API to read the global orientations from the PrioVR Suit and print them to the screen.

Description

In order to achieve wireless streaming, the Prio API must:

1. Find the PrioVR Basestation connected to the computer
2. Find the PrioVR Hub connected to the Basestation
3. Start streaming the data
4. Read the data
5. Stop streaming and deinitialize the Prio API to allow for later use.

Step 1: Creating the PrioVR Basestation

The first step in using a PrioVR suit through the Prio API is to create a PrioVR Basestation to work with.

```
// An unique identifier used by the PrioVR API to identify the PrioVR BaseStation
prio_device_id bs_device;

// An unique identifier used by the PrioVR API to identify the PrioVR Hub
prio_device_id hub_device;

// The communication port of a PrioVR Basestation
prio_ComPort port;
// Allocate a char array of 64 chars for the port name
port.port_name = new char[64];

// An index into the array of communication ports found by the Prio API
uint8_t com_offset = 0;

// Find all of the communication ports that corresponded to PrioVR Basestations
prio_findPorts(PRIO_BS);

// Get the communication port at the index of com_offset
prio_getPort(&port.port_name, com_offset , &port.device_type);

// Create a PrioVR Basestation object from the communication port
prio_createBaseStation(port.port_name &bs_device);
```

This code creates variables to store identifiers for PrioVR Basestation, PrioVR Hub, the communication port of the PrioVR Basestation, and an offset into the found communication ports.

After creating these variables, the code finds and creates a PrioVR Basestation connected to the host PC at the given offset in the communication ports (com_offset), and assigns its identifier to bs_device. If none is present, bs_device will be set to an error value. This identifier will be passed to further functions that reference the PrioVR Basestation.

Step 2: Getting the PrioVR Hub

With the PrioVR Basestation created it is now time to find the paired PrioVR Hub.

```
// Get the PrioVR Hub paired to the PrioVR Basestation
prio_bs_getWirelessHub(bs_device, &hub_device);
```

This code will find a PrioVR Hub that has been paired with the PrioVR Basestation, and assigns its identifier to hub_device. If none is present, hub_device will be set to an error value. This identifier will be passed to further functions that reference the PrioVR Hub.

Step 3: Start Streaming

With the PrioVR Basestation created and it's PrioVR Hub found, it is now time to specify what data to gather and to instruct the PrioVR Basestation to begin gathering that data from the PrioVR suit.

```
// This starts streaming untared orientations, at an interval of 10000 mircoseconds
// forever
prio_bs_startStreaming(bs_device, PRIO_STREAM_UNTARED_ORIENTATION_AS_QUATERNION, 10000,
PRIO_STREAM_DURATION_INFINITE);
```

This code will start streaming untared quaternions. The other options are:

```
1. PRIO_STREAM_TARED_ORIENTATION_AS_QUATERNION
2. PRIO_STREAM_ALL_CORRECTED_COMPONENT_SENSOR_DATA
3. PRIO_STREAM_CORRECTED_GYRO_RATE
4. PRIO_STREAM_CORRECTED_ACCELEROMETER_VECTOR
5. PRIO_STREAM_CORRECTED_MAGNETOMETER_VECTOR
6. PRIO_STREAM_ALL_RAW_COMPONENT_SENSOR_DATA
7. PRIO_STREAM_RAW_GYROSCOPE_RATE
8. PRIO_STREAM_RAW_ACCELEROMETER_DATA
9. PRIO_STREAM_RAW_MAGNETOMETER_DATA
```

Up to 512 bits in any combination of up to 8 options can be streamed simultaneously.

Step 4: Read the Data

At this point, orientation data will be automatically obtained from the PrioVR Basestation at a regular interval. All that remains is to read the data, and print it to the screen.

```
// For as long as you need suit data for
while(true) {

    // The header data from the packet
    prio_StreamHeaderData header_data;

    // The packer data from the packet, sized for 19 sensors
    U8 packet_data[304];

    // Retrieve the latest packet from the PrioVR Hub
    prio_hub_getLastStreamingPacket(hub_device, &header_data, &packet_data[0]);

    // A float array for storing 19 quaternions
    float packet_quats[76];

    // Copy the packet data into the float array
    memcpy(packet_quats, packet_data, 304);

    // Loop until all of the packet quaternions have been printed
    int i=0;
    do {

        // Printing out the sensor quaternions
        printf("(%f,%f,%f,%f)",packet_quats[i],packet_quats[i+1],packet_quats[i+2],packet_quats[i+3]);

        // Increment the quaternion array by the 4 consumed data
        i += 4;

    } while( i < 76)

}
```

The "prio_StreamHeaderData " structure used here is a container for two uint8s that store the battery level, the state of the PrioVR Hub buttons, and two prio_Joystick that hold the joystick data. The "prio_JoystickData" structure is a container for four uint8s that hold the x_axis, y_axis, trigger state, and button state.

The "packet_data" used here is an array that holds all of the data streamed by the PrioVR Hub for 19 sensors. This data is in an array where a sensor's quaternion data is 16 spots in the array ([0-15],[16-31]...[287-303]). The code then converts that data into a float array for ease of use, and prints the data to the user. A quaternion can be converted to any other major orientation format, including rotation matrix, axis-angle, or Euler angles.

Step 5: Finish Up

Before the program ends, streaming must be stopped and the Prio API must be shut down.

This is accomplished with the functions below.

```
// Stops the streaming of the PrioVR Basestation and PrioVR Hub
prio_bs_stopStreaming(bs_device);
// Deinitialize the Prio API
prio_deinitAPI();
```

Complete Example

```
#include <stdio.h>
#include <time.h>
#include “prio_api_export.h”

#define PRIO_STREAM_DURATION_INFINITE 0xffffffff

void main() {

    // An unique identifier used by the PrioVR API to identify the PrioVR BaseStation
    prio_device_id bs_device;

    // An unique identifier used by the PrioVR API to identify the PrioVR Hub
    prio_device_id hub_device;

    // The communication port of a PrioVR Basestation
    prio_ComPort port;

    // Allocate a char array of 64 chars for the port name
    port.port_name = new char[64];

    // An index into the array of communication ports found by the Prio API
    uint8_t com_offset = 0;

    // Find all of the communication ports that correspond to PrioVR Basestations
    prio_findPorts(PRIO_BS);

    // Get the communication port at the index of com_offset
    prio_getPort(&port.port_name, com_offset , &port.device_type);

    // Create a PrioVR Basestation object from the communication port
    prio_createBaseStation(port.port_name &bs_device);

    // Get the PrioVR Hub paired to the PrioVR Basestation
    prio_bs_getWirelessHub(bs_device, &hub_device);

    // This starts streaming untared orientations, at an interval of 10000 mircoseconds forever
    prio_bs_startStreaming(bs_device, PRIO_STREAM_UNTARED_ORIENTATION_AS_QUATERNION, 10000,
    PRIO_STREAM_DURATION_INFINITE);

    // For as long as you need suit data for
    while(true) {
```

```
        // The header data from the packet
        prio_StreamHeaderData header_data;

        // The packer data from the packet, sized for 19 sensors
        U8 packet_data[304];

        // Retrieve the latest packet from the PrioVR Hub
        prio_hub_getLastStreamingPacket(hub_device, &header_data, &packet_data[0]);

        // A float array for storing 19 quaternions
        float packet_quats[76];

        // Copy the packet data into the float array
        memcpy(packet_quats, packet_data, 304);

        // Loop until all of the packet quaternions have been printed
        int i=0;
        do {

            // Printing out the sensor quaternions
            printf("(%f,%f,%f,%f)",packet_quats[i],packet_quats[i+1],packet_quats[i+2],
            packet_quats[i+3]);

            // Increment the quaternion array by the 4 consumed data
            i += 4;

        } while( i < 76)
    }

    // Stops the streaming of the PrioVR Basestation and PrioVR Hub
    prio_bs_stopStreaming(bs_device);

    // Deinitialize the Prio API
    prio_deinitAPI();

}
```


YOST LABS
630 Second Street
Portsmouth Ohio 45662, USA

Phone: 740.876.4936
info@yostlabs.com
yostlabs.com

Made in USA. Patents: 8498827, 8682610, 9255799, 9354058. Additional patents pending.

About Yost Labs, Inc. We are a fast growing private company based in historic Portsmouth, Ohio. With over a decade of experience in low-latency inertial sensor innovation, we enable motion tracking in many of today's and tomorrow's most exciting products. We make virtual reality interactive. We stabilize drones and navigate autonomous cars. We measure human motion for athletic performance and rehabilitation. We are dedicated to supporting you and your team—providing expert advice and integration consulting for the world's fastest inertial motion sensor technology.

Yost Labs' innovation has been recognized with numerous patents with additional patents pending. Our customers and value-added resellers include the US Navy, US Air Force, NASA, US Army Corps of Engineers and over 1,000 leading technology firms and academic institutions around the world.