YOST LABS
630 Second Street
Portsmouth Ohio 45662, USA

Phone: 740.876.4936
info@yostlabs.com
yostlabs.com

AN2016.04

Using the Prio API for wired recording with a PrioVR™ Hub

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 to the host computer via USB (shown in Figure 1).

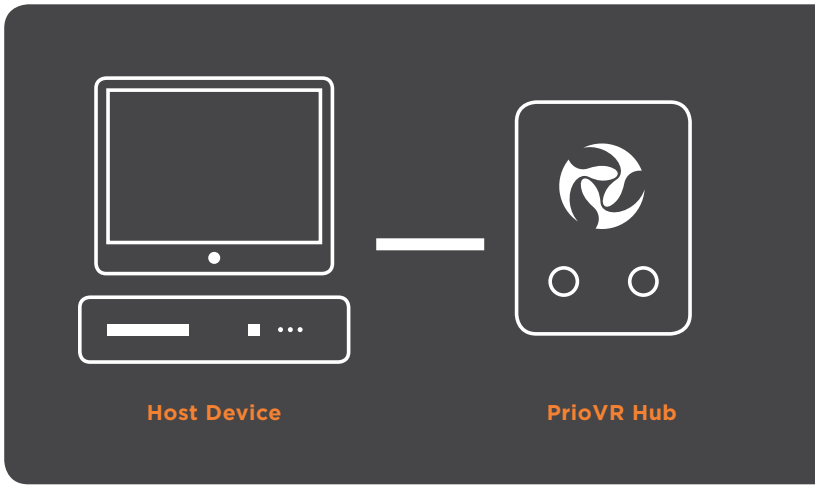

Figure 1

The purpose of this document is to provide an overview of recording data from a PrioVR Suit using the PrioVR Hub. The example code below illustrates the use of the Prio API to read the global orientations from the PrioVR Suit and record them to a text file.

Description

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

1. Find the PrioVR Hub connected to the computer
2. Setup the recording options
3. Start recording the data
4. Stop recording the data
5. Output data to a file
6. Deinitialize the Prio API to allow for later use.

Step 1: Creating the PrioVR Hub

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 Hub
prio_device_id hub_device;

// The communication port of a PrioVR Hub
prio_ComPort port;
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 Hubs
prio_findPorts(PRIO_HUB);

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

// Create a PrioVR Hub object from the communication port
prio_createHub(port.port_name &hub_device);
```

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

After creating these variables, the code finds and creates a PrioVR Hub connected to the host PC at the given offset in the communication ports (com_offset), 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 2: Setup Recording

With the PrioVR Hub created and its hub found, it is now time to give the PrioVR Hub instructions on how many packets to record and whether or not to wrap the recording.

```
// Setup to record the first 1000 packets
prio_hub_setupRecordingOptions(&hub_device, 1000, false);
```

Step 3: Start Recording

With the recording options setup, it is now time to specify what data to gather and to instruct the PrioVR Hub to begin gathering that data from the PrioVR suit.

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

// Wait to get 1000 packets
printf("Press enter when you are finished recording 1000 packets");
getchar();
```

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 recorded simultaneously.

Step 4: Stop Recording

Now that the PrioVR Hub has collected the data, it is time to stop recording.

```
// Stops the recording of the PrioVR Hub
prio_hub_stopRecordingWired(hub_device);
```

Step 5: Output the data

With the data recorded and recording stopped, it is now time to output the recorded data to a file. To accomplish this, the program:

- Finds the count of recorded packets
- Gathers the data from the Prio API
- Outputs the data to a file.

```
// Gather the actual number of recorded packets
U32 packet_count = 0;
prio_hub_getLengthOfRecordedSamples(hub_device, &packet_count);

// Create buffers to hold the header data, and packet data for 1000 packets sized for 19 sensors
PrioHeader headerData[1000];
U8 packetData[304000];

// Grab the data from the PrioAPI
prio_hub_getRecordedSamples(hub_device, headerData, packetData, packet_count);

// Create a file pointer and file
FILE *fp;
fp = fopen("PrioRecordedData.txt", "w+");
```

```
// Iterate over every packet and write the packet to the file
for (int i = 0; i < (int)packet_count; i++) {

    // Get the packet header data for the current index
    PrioHeader tmpHeader = headerData[i];
    fprintf(fp, “====Header Data at Index%d====\n”, i);
    fprintf(fp, “TimeStamp: %d\n”,(int)tmpHeader.SensorTimestamp);
    fprintf(fp, “Battery Level: %d\n”,(int)tmpHeader.BatteryLevel);
    fprintf(fp, “Joystick 1 X axis: %d\n”, (int)tmpHeader.Joystick1.x_Axis);
    fprintf(fp, “Joystick 1 Y axis: %d\n”, (int)tmpHeader.Joystick1.y_Axis);
    fprintf(fp, “Joystick 1 Trigger: %d\n”, (int)tmpHeader.Joystick1.Trigger);
    fprintf(fp, “Joystick 1 Button State: %d\n”,(int)tmpHeader.Joystick1.ButtonState);
    fprintf(fp, “Joystick 2 X axis: %d\n”, (int)tmpHeader.Joystick2.x_Axis);
    fprintf(fp, “Joystick 2 Y axis: %d\n”, (int)tmpHeader.Joystick2.y_Axis);
    fprintf(fp, “Joystick 2 Trigger: %d\n”, (int)tmpHeader.Joystick2.Trigger);
    fprintf(fp, “Joystick 2 Button State: %d\n”,(int)tmpHeader.Joystick2.ButtonState);
    fprintf(fp, “=======================================\n”);
    fprintf(fp, “====Sensor Data====\n”);

    for (int j = 0; j < 19; j++) {
        // Get the current sensor quaterion
        float tmp[4];
        memcpy(tmp, packetData + (i * 32) + j * 16, 16);
        fprintf(fp, “(%f,%f,%f,%f)\n”, tmp[0], tmp[1], tmp[2], tmp[3]);
    }

    fprintf(fp, “====End of Sensor Data===\n”);

}

// Close the file
fclose(fp);
```

Step 6: Finish up

The Prio API needs to be shut down before the program ends.

```
// 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 Hub
    prio_device_id hub_device;

    // The communication port of a PrioVR Hub
    prio_ComPort port;
    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 Hubs
    prio_findPorts(PRIO_HUB);

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

    // Create a PrioVR Hub object from the communication port
    prio_createHub(port.port_name & hub_device);

    // Setup to record the first 1000 packets
    prio_hub_setupRecordingOptions( & hub_device, 1000, false);

    // This starts recording untared orientations, at an interval of 10000 mircoseconds forever
    prio_hub_startRecordingWired(hub_device, PRIO_STREAM_UNTARED_ORIENTATION_AS_QUATERNION, 10000,
    PRIO_STREAM_DURATION_INFINITE);

    // Wait to get 1000 packets
    printf(“Press enter when you are finished recording 1000 packets”);
    getchar();

    // Stops the recording of the PrioVR Hub
    prio_hub_stopRecordingWired(hub_device);

    // Gather the actual number of recorded packets
    U32 packet_count = 0;
    prio_hub_getLengthOfRecordedSamples(hub_device, & packet_count);

    // Create buffers to hold the header data, and packet data for 1000 packets sized for 19 sensors
    PrioHeader headerData[1000];
    U8 packetData[304000];

    // Grab the data from the PrioAPI
    prio_hub_getRecordedSamples(hub_device, headerData, packetData, packet_count);

    // Create a file pointer and file
    FILE * fp;
    fp = fopen(“PrioRecordedData.txt”, “w + “);
```

Complete Example continued...

```
    // Iterate over every packet and write the packet to the file
    for (int i = 0; i < (int) packet_count; i++) {

        // Get the packet header data for the current index
        PrioHeader tmpHeader = headerData[i];
        fprintf(fp, " === = Header Data at Index % d === = \n", i);
        fprintf(fp, "TimeStamp: % d\ n", (int) tmpHeader.SensorTimestamp);
        fprintf(fp, "Battery Level: % d\ n", (int) tmpHeader.BatteryLevel);
        fprintf(fp, "Joystick 1 X axis: % d\ n", (int) tmpHeader.Joystick1.x_Axis);
        fprintf(fp, "Joystick 1 Y axis: % d\ n", (int) tmpHeader.Joystick1.y_Axis);
        fprintf(fp, "Joystick 1 Trigger: % d\ n", (int) tmpHeader.Joystick1.Trigger);
        fprintf(fp, "Joystick 1 Button State: % d\ n", (int) tmpHeader.Joystick1.ButtonState);
        fprintf(fp, "Joystick 2 X axis: % d\ n", (int) tmpHeader.Joystick2.x_Axis);
        fprintf(fp, "Joystick 2 Y axis: % d\ n", (int) tmpHeader.Joystick2.y_Axis);
        fprintf(fp, "Joystick 2 Trigger: % d\ n", (int) tmpHeader.Joystick2.Trigger);
        fprintf(fp, "Joystick 2 Button State: % d\ n", (int) tmpHeader.Joystick2.ButtonState);
        fprintf(fp, " === === === === === === === === === === === === = \n");
        fprintf(fp, " === = Sensor Data === = \n");

        for (int j = 0; j < 19; j++) {
            // Get the current sensor quaterion
            float tmp[4];
            memcpy(tmp, packetData + (i * 304) + j * 16, 16);
            fprintf(fp, " ( % f, % f, % f, % f)\ n", tmp[0], tmp[1], tmp[2], tmp[3]);
        }

        fprintf(fp, " === = End of Sensor Data === \n");

    }

    // Close the file
    fclose(fp);

    // 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.