# Objectives

* Retrieve real time data of the joystick states; button presses and joystick position from a Generic Joystick Controller with USB input.
* Have the states printed to stdout in real time using an executable that can be spawned from a parent process (VREP main client).
* The process must be killable from the parent process as well, to avoid unnecessary hanging and infinite forking, which could damage computer memory.

# API’s and Libraries

## Windows API

Windows has its own API for building window executables. This API is formatted differently to standard C programs, and contains very different properties and types. For example, main is run from [1]:  
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd){

The syntax and libraries from Windows.h, with hidsdi.h allowes direct communication with Human Interface Devices (HID). Windows are generated first, and messages are sent through this thread and the HID, thus to GetRawInputData(), a window must be created to generate the message struct [1] and the lParam argument [2]. Message windows can be created by setting the hWndParent parameter of CreateWindow() to HWND\_MESSAGE; this will create a hidden window that can only be terminated using Task Manager.

The primary problem with the Windows API is that it is separate to the console, thus writing to stdout is ineffective, and the only way to create an output is to write to a file with FILE\* f = fopen(), and fprintf(f, “text”). Setting the linking configuration from Windows to Console will cause the program build to be unsuccessful. Despite these problems, Windows API provides the lowest level of talking directly to the HID drivers, and manipulation of USB data from the controller.

Open-source code for generating a window containing a joystick GUI with real-time button presses is available from [4]. Building RawInputJoystickSource/Raw Input.vcxproj in Visual Studio will generate an executable Raw Input.exe in RawInputJoystickSource/Debug, which can be rum from command prompt. Adding print statements or writing to stdout won’t produce anything on command prompt, however the executable will create a window containing the joystick visualisation. Ridding the source code from the Windows API was unsuccessful as getting the raw input requires the message struct, generated using a window thread. Created text files however will appear in /Debug.

# HTML5 Gamepad API

Code for creating joystick support with browser (HTML5) games is available from [5] with information at [6]. The latter contains instructions for getting a PS4 controller working in Chrome and Firefox. As this isn’t useful for application based programs, it would be a useful alternative for games that are ran using .html.

# Libusb and xusb.c

A library used for retrieving information from USB, with an example executable and solution, xusb.c, which demonstrates this for Generic, PS3 and Xbox joysticks [7]. The library functions have documentation available from [7], however, apart from the xusb example there is limited documentation on how to implement these functions or format code. Thus, manipulating the code for the Generic USB Joystick to work with xusb.c, to see the button inputs is difficult. If the file is ran with “xusb.exe –s vid:pid” the device information and reading input report can be generated, if buttons are being pressed the hexadecimal output will reflect this. It is possible to loop the “Reading Input Report” and print to stdout in the same way as current. The inputs will then have to be mapped to buttons, which can create difficulty with interpreting the state of the joysticks. A PS3 controller code did not work (exited with Input/Output error), as the inputs weren’t printed to terminal, but it is expected that the code format for processing the input, to show which buttons were pressed, will be similar to that for the Generic USB controller. The code can be executed from libusb-1.0.21/examples/bin64, with code kept in libusb-1.0.21/examples/xusb.c.

# Simple Direct Layer (SDL) Source Code

Written in C, a cross platform library can be download for the use of getting data from keyboard, mouse and joysticks, from the SDL website [9]. If the source code is modified it must be marked as so, otherwise it is free to be used and manipulated to repeatedly display input data from devices; originally set to only print device information, however input manipulation functions are provided. The joystick project can be built by loading SDL2-2.0.7/VisualC/SDL into Visual Studio and building from testjoystick.c. This file initialises the joystick and retrieves data from the SDL\_joystick.c file, where it can be executed in a console from SDL2-2.0.7/VisualC/Win32/Debug. The test files are setup to produce a GUI, however the code for this can be commented out, and input manipulation moved to main. Printf can be included by creating a new stream pointing to standard out, and joystick information sent to it (by default, SDL uses a different handler (not stdout) to point information to the command line, and so an external process cannot read this information).