MotionSystems

SOFTWARE

ForceSeatDI

v.1.1

2023.12.06

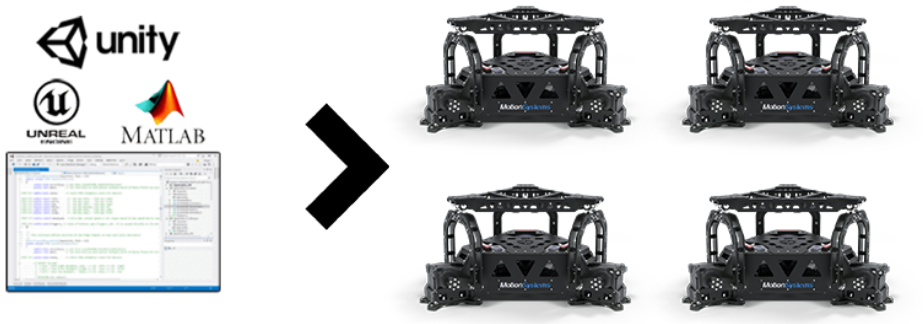
Miedziana 7 Street ■ 55-003 Nadolice Wielkie, Poland
www.motionsystems.eu ■ contact@motionsystems.eu

Contents

1 General information

1.1 Introduction

ForceSeatDI (Direct Interface) is a lower level interface than ForceSeatMI. It controls hardware directly and ForceSeatPM is not required at all. All error handling and status checking have to be performed by the application. This interface allows to control more than one motion platform from the same PC and allows to create complex but fully synchronized movements of multiple motion platforms.



1.2 ForceSeatMI vs ForceSeatDI

ForceSeatMI (Motion Interface) is a programming interface that allows to add a motion platform support to any application or a game. The ForceSeatMI does not control hardware directly – it sends all data to ForceSeatPM. This approach delegates the responsibility of transforming telemetry data to a real motion from the application to ForceSeatPM. It means that application developers do not have to worry about things like platform disconnections, transmission errors, thermal protection warnings or signal filtering.

Direct Interface is less complex than ForceSeatMI. It manages hardware directly, thus ForceSeatPM is not essential. Checking errors or status is a duty of the application. It allows to control multiple platforms from the same computer. Thanks to such a solution it is possible to build complex and fully synchronized movements of multiple motion platforms at once.

WARNING

ForceSeatDI should be used only in very specific applications. For all other applications, ForceSeatMI is recommended.

1.3 Features comparison

	ForceSeatMI	ForceSeatDI
C/C++	■	■
C#	■	■
Python	■	■
Unity 3D	■ (only Windows PC target platform)	■ (only Windows PC target platform)
Unreal Engine	■ (only Windows PC target platform)	■ (only Windows PC target platform)
Matlab/Simulink	■	■
Microsoft Windows	■	■
Linux (Ubuntu 16.04.3 LTS Desktop x64)	□	■
Raspberry Pi 3 (armv7l 4.9.35))	□	■
Raspberry Pi 4 64-bit (armv72 6.1.5))	□	■
Gear VR platforms (e.g. Oculus Go, Samsung Gear VR)	□	□
Multiple platforms from single PC over USB	■ (same data sent to all platforms)	■ (separated control of each platform)
Multiple platforms from single PC over Ethernet	□	■
Easy error handling	■ (by ForceSeatPM)	□ (by the application)
Diagnostic features	■ (by ForceSeatPM)	□ (by the application)
Requires ForceSeatPM	■	□
Telemetry mode & scripting engine	■	□
Motion profile selection by the user	■	□
Inverse kinematics	■	■
Fordward kinematics	■	■
Motion compensation for VR	■ (by VR HeadWay in ForceSeatPM)	□ (by the application, e.g. camera position correction)
Licensing	per motion platform (license stored on PC)	per motion platform (license stored on motion controller)

1.4 Documentation

Idea behind ForceSeatDI and its API structure is similar to ForceSeatMI and at this moment is it not described separately in any document. Please refer to ForceSeatMI documentation to get better understanding how the motion platform control works and examine examples delivered in ForceSeatDI SDK archive to see implementation details.

WARNING

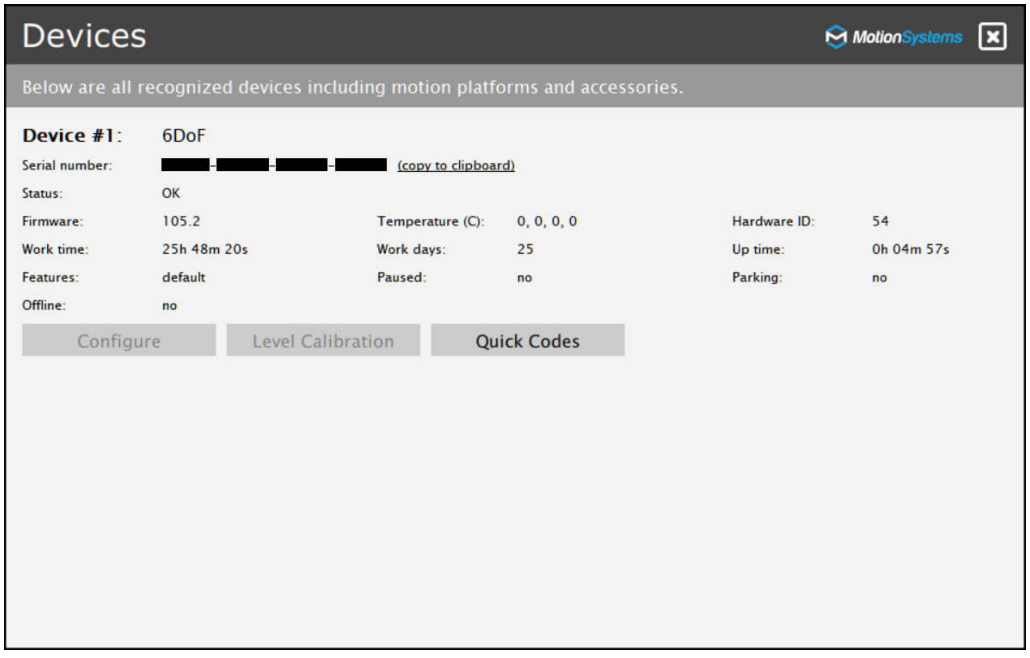
ForceSeatDI should be used only in very specific applications. For all other applications, ForceSeatMI is recommended.

How to activate the license

2

ForceSeatDI license is kept in the motion controller so the activation process is different than for ForceSeatMI. Once you receive the activation code do following steps:

1. Install ForceSeatPM
2. Open ForceSeatPM main window
3. Click **Tools and Diagnostic** and then **Devices**
4. Click **Quick Codes**
5. Enter the activation code. The verification might take a while.
6. Once finish, close the Quick Codes window.
7. Turn off the device
8. Turn on the device
9. In Devices window you should see **FSDI** displayed in **Features** field



3 Cross compiling for Raspberry Pi

3.1 Introduction

This short tutorial shows the idea of using VisualGDB to build one of our examples for Raspberry Pi. This section does not describe the full tool chain and build system configuration, it only indicates a few important key points that should help you to configure your project:

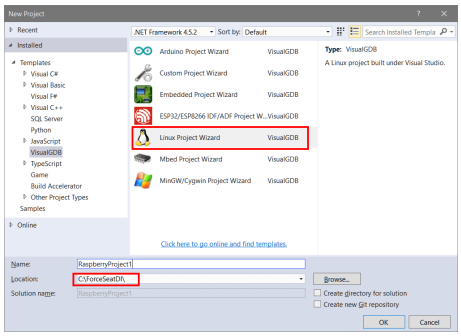
- preprocessor directives
- copying .so file to destination board
- debugging the program as root.

For a more detailed description of how to use and configure VisualGDB, please refer to VisualGDB documentation.

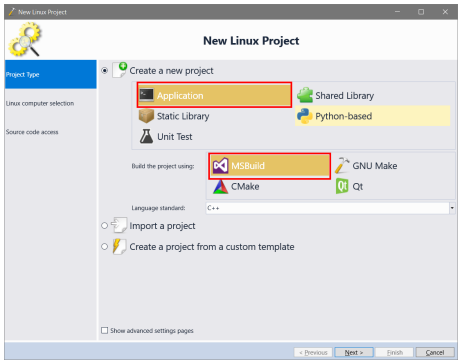
> **TIP**
>
> VisualGDB is not mandatory for ForceSeatDI to work on Raspberry Pi. You can compile ForceSeatDI using any other build system.

3.2 New project

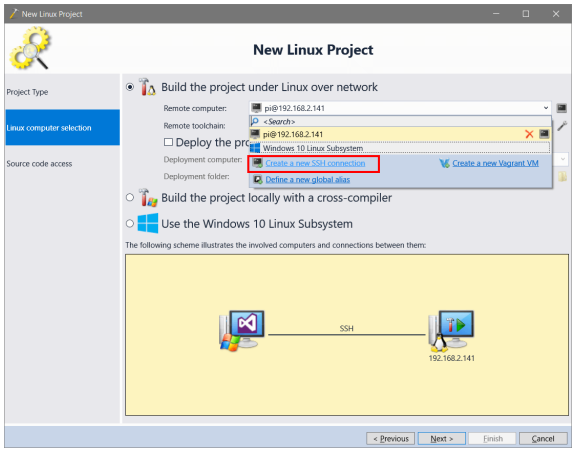
Start Visual Studio and create a new Linux Project. Make sure to create it in the directory without spaces in the name, e.g. **C:\ForceSeatDI**

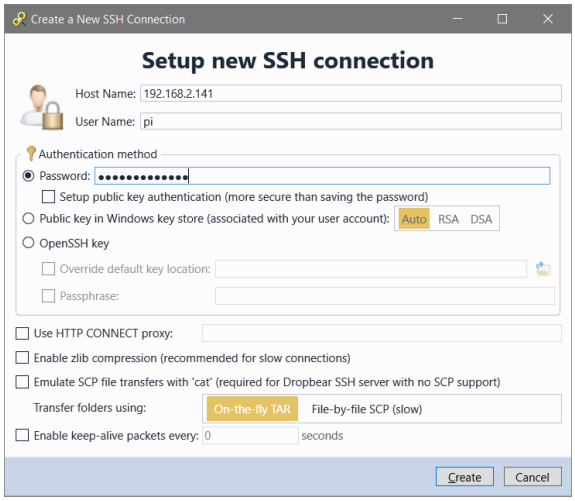


Choose **Application** as project type and **MSBuild** as tool chain.

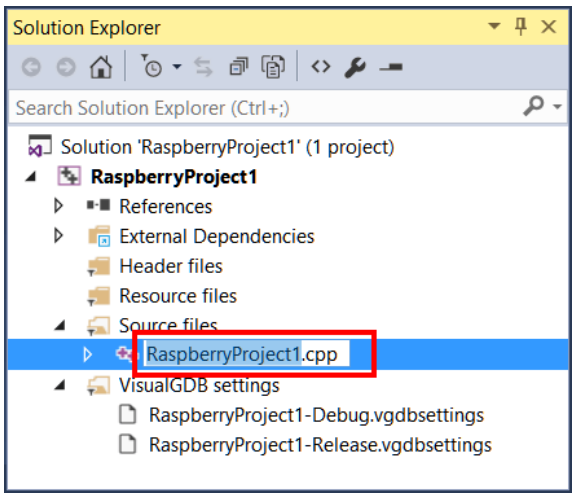


Choose building under Raspberry Pi over network. In order to do it, create **new SSH connection**, then fill IP address and credentials. Make sure that your Raspberry Pi has **libusb** and compiler installed (**check ForceSeatDI readme.txt for details**).





After wizard creates project for you, rename **RaspberryProject1.cpp** to **Main.cpp**



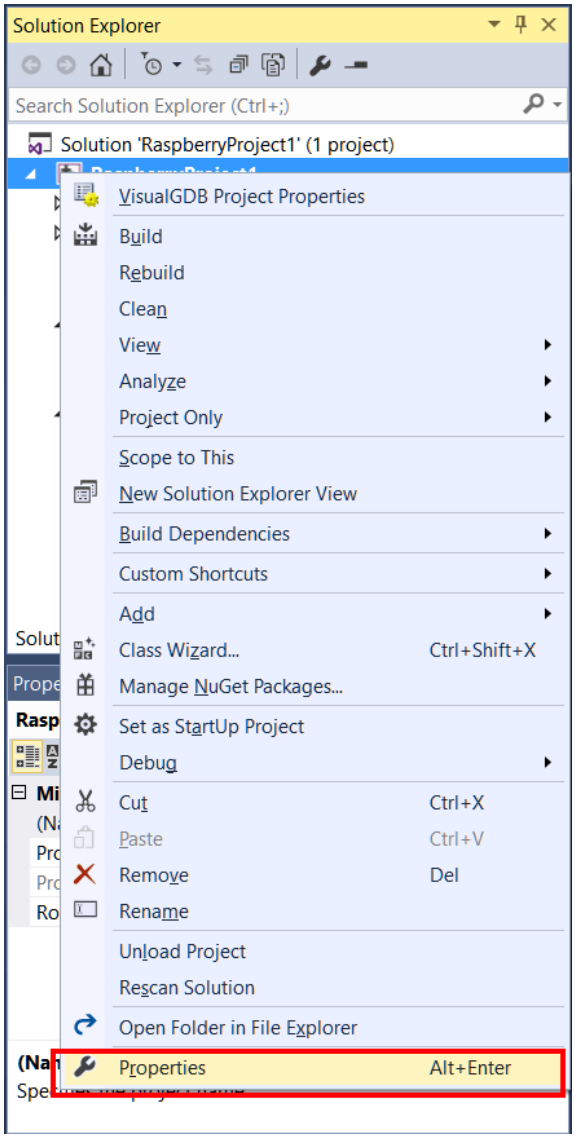
Copy following files from ForceSeatDI archive **code** directory to your project directory:

1. ForceSeatDI.h
2. ForceSeatDI_Defines.h
3. ForceSeatDI_Functions.h
4. ForceSeatDI_Loader.c
5. ForceSeatDI_Structs.cs

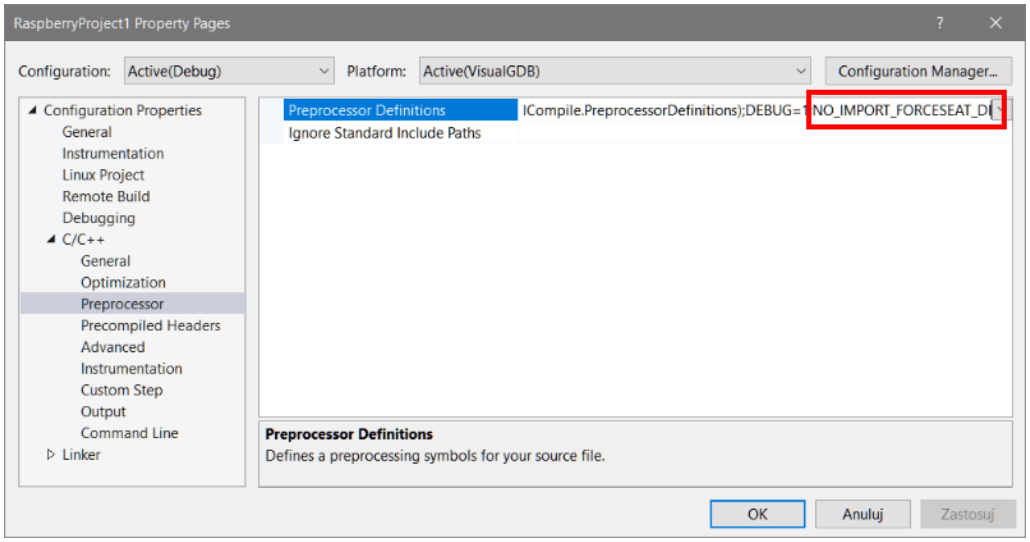
Add **ForceSeatDI_Loader.c** to your project and copy **Main.cpp** from **FastPos_CPP_Linux** example (overwrite the Main.cpp in your project).

3.3 Project configuration

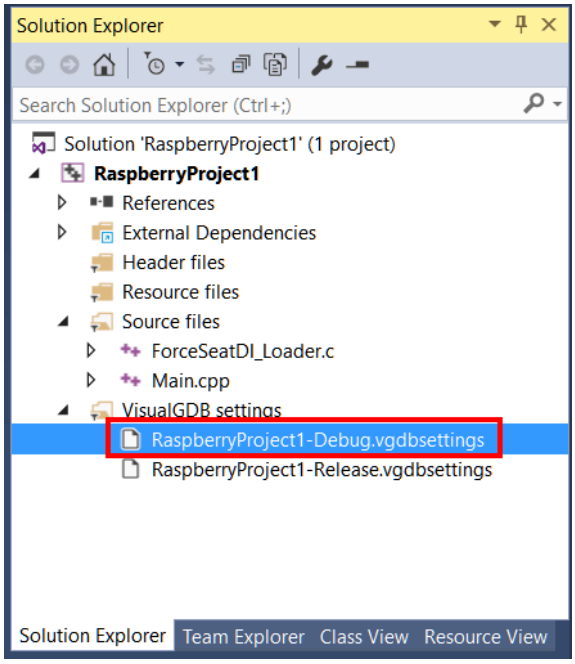
Open the project **Properties**.



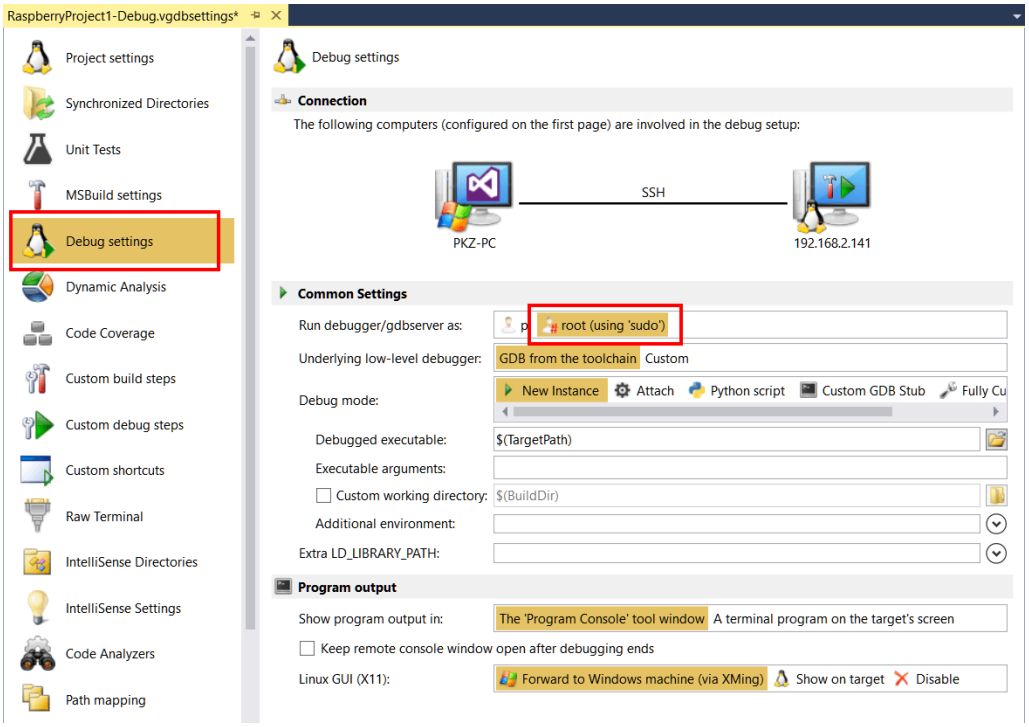
Go to **Preprocessor** configuration and add **NO_IMPORT_FORCESEAT_DI** to the list of definitions.



Go to **VisualGDB** settings and open Debug configuration (Release configuration for release build).



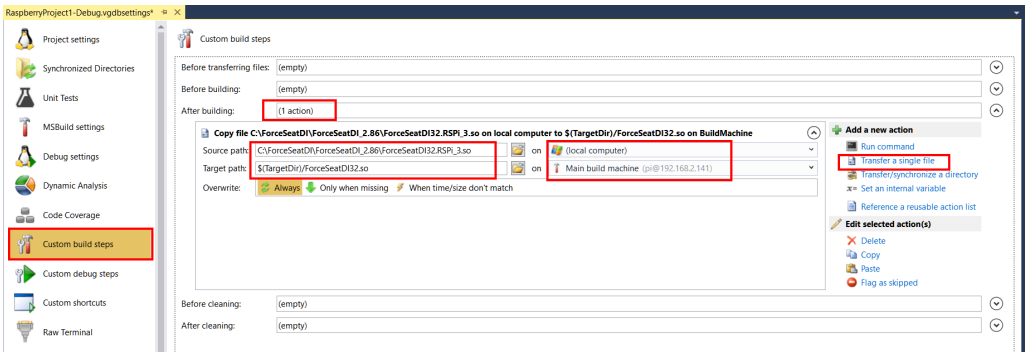
Go to **Debug Settings** and select run as root.



WARNING

If the application is not executed as root, it will not be able to detach the USB device from the kernel and connect to the motion platform.

Go to **Custom build steps** and add new **After build** step – **Transfer a single file**.

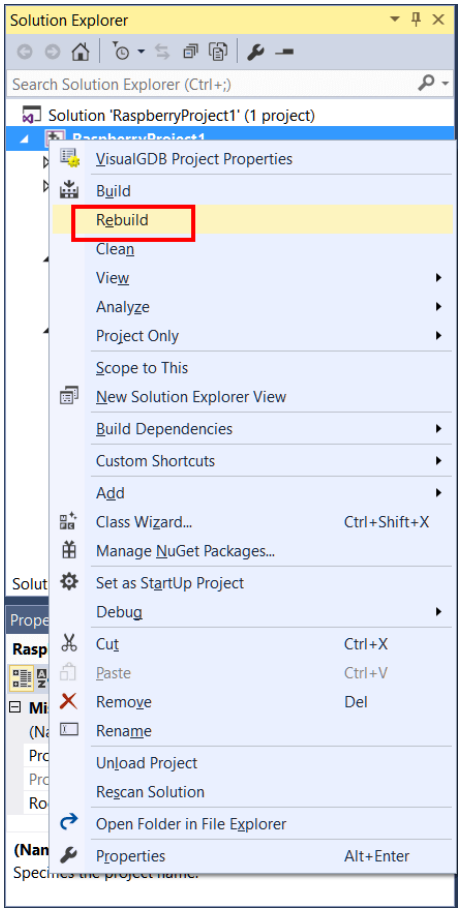


Browse and select source file **ForceSeatDI32.RSPi_3.so** on your PC disk.

Type: $(TargetDir)/ForceSeatDI32.so into **Target path**.

3.4 Build and run

Rebuild the project.



Run the application and check the Output console.