User Manual

Pickering Remote Gateway



© Copyright (2026) Pickering Interfaces. All Rights Reserved.

No part of this publication may be reproduced, transmitted, transcribed, translated or stored in any form, or by any means without the written permission of Pickering Interfaces.

Technical details contained within this publication are subject to change without notice.

FM38792

Technical Support

For Technical Support please contact Pickering Interfaces either by phone, the website or via e-mail.

Warranty

All products manufactured by Pickering Interfaces are warranted against defective materials and workmanship for a period of three years, excluding programmable power supplies, from the date of delivery to the original purchaser. Any product found to be defective within this period will, at the discretion of Pickering Interfaces be repaired or replaced.

Products serviced and repaired outside of the warranty period are warranted for ninety days.

Extended warranty and service are available. Please contact Pickering Interfaces by phone, the website or via e-mail.

Environmental Policy

Pickering Interfaces operates under an environmental management system similar to ISO 14001.

Pickering Interfaces strives to fulfill all relevant environmental laws and regulations and reduce wastes and releases to the environment. Pickering Interfaces aims to design and operate products in a way that protects the environment and the health and safety of its employees, customers and the public. Pickering Interfaces endeavors to develop and manufacture products that can be produced, distributed, used and recycled, or disposed of, in a safe and environmentally friendly manner.

Worldwide Technical Support and Product Information

Pickering Interfaces Headquarters

Stephenson Road, Clacton-on-Sea, CO15 4NL United Kingdom
Tel: +44 (0)1255-687900
e-mail: sales@pickeringtest.com

USA

Tel: +1 781 897 1710
e-mail: ussales@pickeringtest.com

France

Tel: +33 9 72 58 77 00
e-mail: frsales@pickeringtest.com

Germany

Tel: +49 89 125 953 160
e-mail: desales@pickeringtest.com

Sweden

Tel: +46 340-69 06 69
e-mail: ndsales@pickeringtest.com

Czech Republic

Tel: +420 558 987 613
e-mail: desales@pickeringtest.com

China

Tel: +86 4008 799 765
e-mail: chinasales@pickeringtest.com

Malaysia

Tel: +60 12 833 7980
e-mail: aseansales@pickeringtest.com

Contents

1. What is gRPC? 1

2. What is proto File? 1
2.1 Example 1

3. Benefits of gRPC for Pickering Software 2

4. What You Need to Install Before Using gRPC 2

5. PIRG Server on Windows 3
5.1 Pre Operation Checks 3
5.2 Hardware Installation 3
5.3 Software Installation 4

6. PIRG Server on Linux 5

7. How to Startup Client on Windows 5
7.1 Python 5
7.2 C++ 6
7.3 C# 6

8. How to Startup Client on Linux 7
8.1 Python 7
8.2 C# 8

9. Installing gRPC on Windows 9
9.1 Method 1: Manual Compilation Using CMake 9
9.2 Method 2: Installation Via the vcpkg Package Manager 10
9.3 Summary of Advantages and Disadvantages 10
9.4 Recommendation 10

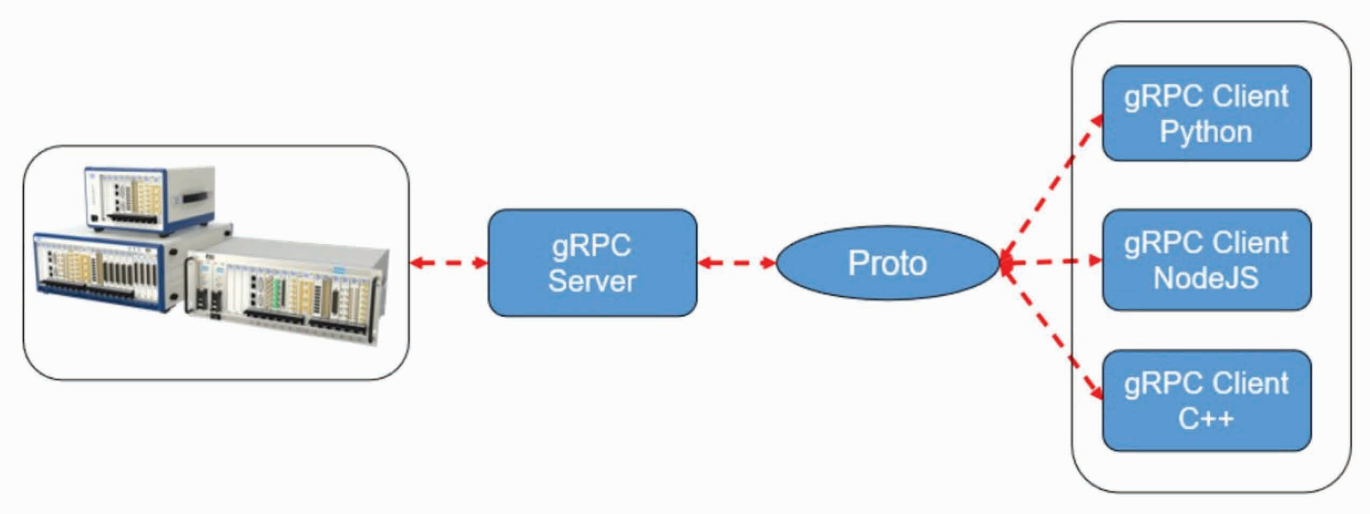
10. Installing gRPC on Linux 11
10.1 Method 1: Manual Compilation Using CMake 11
10.2 Method 2: Installation Via the vcpkg Package Manager 12
10.3 Summary of Advantages and Disadvantages 13
10.4 Recommendation 13

10. Installing gRPC on Linux 13

11. References 13

1. What is gRPC?

gRPC is a modern open-source high performance Remote Procedure Call (RPC) framework that can run in any environment. It can efficiently connect services in and across data centers with pluggable support for load balancing, tracing, health checking and authentication. It is also applicable in last mile of distributed computing to connect devices, mobile applications and browsers to backend services.



Supported languages: C# / .NET, C++, Dart, Go, Java, Kotlin, Node, Objective-C, PHP, Python, Ruby, Swift

2. What is proto File?

A .proto file in gRPC serves as the interface definition between the client and the server. The .proto file specifies: which services the server provides, which functions (RPC methods) the client can call, which parameters are sent, and what data the server returns.

2.1. Example

```
syntax = “proto3”;
package pickering_driver;

service PickeringService {
   rpc CardId (CardIdRequest) returns (CardIdResponse);
}
message CardIdRequest {
   uint32 card_num = 1;
}
message CardIdResponse {
   uint32 error_code = 1;
   string card_id = 2;
}
```

3. Benefits of gRPC for Pickering Software

gRPC offers three key benefits for Pickering software:

- It allows a PXI chassis to be exposed as a network-accessible device, in a similar way to an LXI chassis.
- It enables development of client applications in virtually any programming language.
- It allows multiple clients to connect to a single server, which communicates with the traditional driver without relying on shared memory.

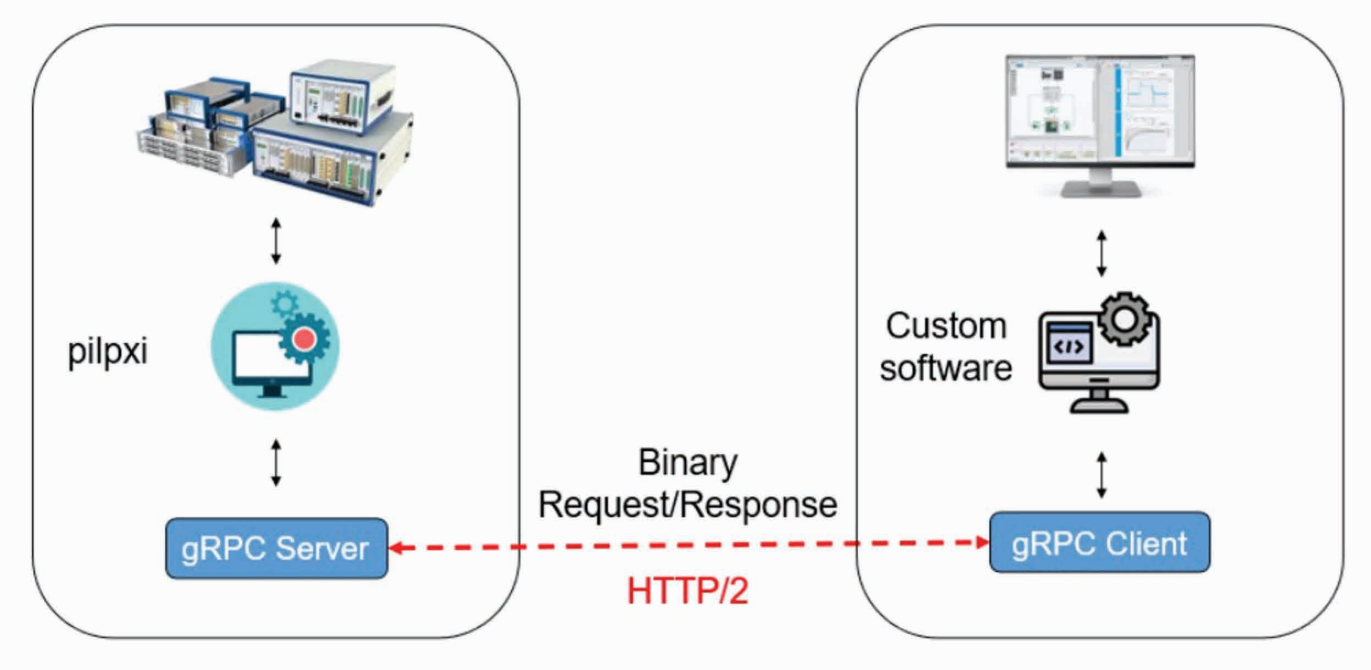
4. What You Need to Install Before Using gRPC

Server Side:

- Install the appropriate PXI or LXI driver:
  - PXI: Download PXI driver
  - LXI: Download LXI driver
- Install the PIRG server:
  - Download gRPC server for Windows
  - Download gRPC server for Linux

Client Side:

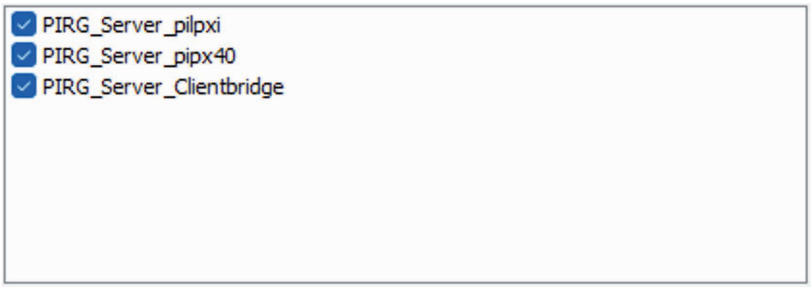
- Install gRPC. Look into sections: Win Section 9 or Linux Section 10
- Download example code: Download gRPC examples



5. PIRG Server on Windows

5.1. Install PIRG Server on Windows

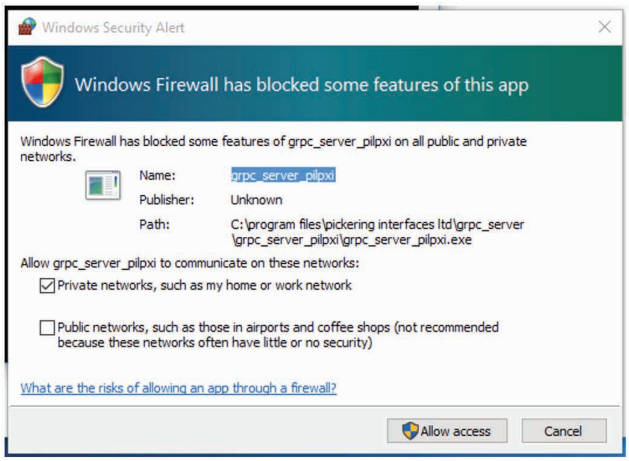
- If you haven't installed the Pickering drivers, install them from: Download driver



- Download the gRPC Server from the Pickering download site: Download server
- Run installer.
- Select server or multiple servers for what you want to install. And click '*Next*'.
- Click '*Install*' (automatically installs selected servers).
- If you want to install windows service for grpc, check checkbox '*Install GPRC as Service*'.
- Click '*Finish*'.

5.2. Run PIRG Server on Windows

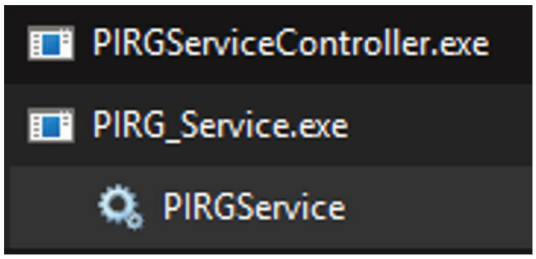
- Install PIRG server.
- Set up the .ini file in the installation folder.
- Start PIRG Server.
  Server can be started from installed folder as "`.\PIRG_Server_pilpxi –ip localhost –port 50051`"
  Or set it up in ini file and next start the exe file."`.\PIRG_Server_pilpxi`"
  Or user use Desktop icon.
  Or if you have installed PIRG service, in taskbar click on icon '*PIRGServiceController*' and select '*Start Server*'.



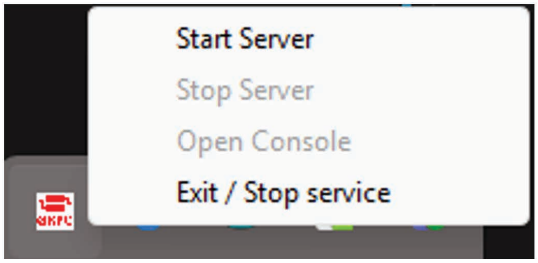
If you see this window, please click on '*Allow Access'*, otherwise, the server will not work.

5.3. How to use PIRG Windows Service

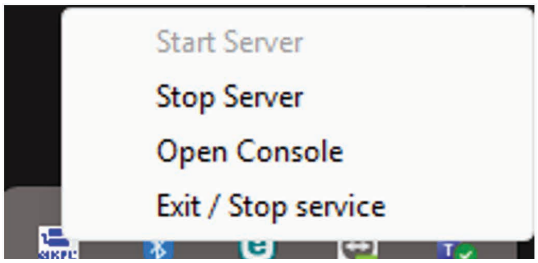
For Windows clients we create windows service.



Service is named *PickeringRemoteGatewayService*, and GUI app in taskbar is named *PickeringRemoteGatewayServiceController*. Both are started after installing the service, and both start after Windows startup. Managing *PickeringRemoteGatewayService* is possible in *Services*, and *PickeringRemoteGatewayServiceController* in *Task Manager* / *Startup apps*. After startup, both apps and all last running servers are started.



Icon shows that no servers are currently running. You can choose to start all installed servers or exit the app and terminate both applications.



Icon show if servers running. If servers are running you can show servers log.



Windows with logs on all running servers.

6. PIRG Server on Linux

- If you haven't installed the Pickering drivers, download and install from: Download driver
- Download the PIRG server from the Pickering download site: Download
- Go to folder where the server is located.
- Write command:
  `chmod a+x ./PIRG_Server_pilpxi`
- If you want, set up the .ini file.
- Start Pickering Remote Gateway Server:
  Server can be started as "`.\PIRG_Server_pilpxi –ip localhost –port 50051`"
  Or set it up in the ini file and next start the exe file."`.\PIRG_Server_pilpxi`"

7. How to Startup Client on Windows

7.1. Python

- This procedure is for Pilpxi, but it is similar for other servers.
  You only need to change the proto file name and the server name.
- Install all dependencies for PIRG in python.
  `pip install grpcio grpcio-tools protobuf`
- Download example from: Examples
- Go into the downloaded example folder and generate the necessary files from the proto files.
  `python -m grpc_tools.protoc -I. –python_out=. –grpc_python_out=. pilpxi.proto`
- Start your installed Pilpxi server
  `.\PIRG_Server_pilpxi –ip localhost –port 50051`
- Start client code, connect to server and controlling cards.
  `python.exe .\client.py`

7.2. C++

- Install grpc from section 9.
- Download example from: Examples
- Go into the downloaded folder where the proto files are and generate the necessary files.
  For example:

```
C:\Users\username\Downloads\C-C++_example\client
path/to/protoc.exe --proto_path=. --grpc_out=. --cpp_out=.
--plugin=protoc-gen-grpc="/path/to/grpc_cpp_plugin.exe"  pilpxi.proto
pilfg.proto
```

- Open project and build it.
- Start your installed Pilpxi server `.\PIRG_Server_pilpxi –ip localhost –port 50051`
- Start client application in visual studio solution.

7.3. C#

How to create C client in Visual Studio:

- Create C# project.
- Click on "*Tools->NuGet Package Manager->Package Nuget Packages for Solution..* "
- In Browse find and install this packages: *Grpc.Net.Client*, *Google.Protobuf*, *Grpc.Tools*
- Add to project existing file, proto file.
- Open client.csproj and add there:

```
<ItemGroup>
      <Protobuf Include="pilpxi.proto" GrpcServices="Client" />
</ItemGroup>
```

- Copy to project code from C# example from: Examples
- Build project.

8. How to Startup Client on Linux

8.1. Python

- **Update and install python**

```
sudo apt update
sudo apt install -y python3 python3-pip python3-venv
```

- **Create virtuál PC use venv**

  Go to example folder where the example client is located.

```
python3 -m venv venv
source venv/bin/activate
```

  At the beginning of the line that displays (venv)

```
pip install —upgrade pip
```

- **Install grpc and protbuffer**

```
pip install grpcio grpcio-tools
```

- **Generate the necessary files from the proto file.**

```
python -m grpc_tools.protoc -I. —python_out=. —grpc_python_out=. pilpxi.proto
```

- **Client example:**

```
import grpc
import pilpxi_pb2
import pilpxi_pb2_grpc

def run():
  channel = grpc.insecure_channel('localhost:50051')
  stub = pilpxi_pb2_grpc.PilpxiServiceStub(channel)
  open_response = stub.OpenCards(pilpxi_pb2.OpenCardsRequest())
  print(f"OpenCards: {open_response.num_cards}")

if  name == '_main_':
  run()
```

- Open new terminal and start sterver

```
./PIRG_Server_pilpxi —ip localhost —port 50051
```

- In terminal with virtual PC venv start client

```
python.exe ./client.py
```

8.2. C#

- Installation:

```
wget https://packages.microsoft.com/config/debian/11/packages-microsoft
      -prod.deb -O packages-microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb
rm packages-microsoft -prod.deb
sudo apt update
sudo apt install dotnet-sdk -8.0
```

- Verify .NET SDK version:

```
dotnet --version
```

- Creating test client

```
dotnet new console -o GrpcClient
cd GrpcClient
dotnet add package Grpc.Net.Client
dotnet add package Google.Protobuf
dotnet add package Grpc.Tools
```

- Copy the pilpxi.proto to current directory

```
cp ~/pilpxi.proto ./
```

- Open GrpcClient.csproj and add following line within the element

```
<Protobuf Include="pilpxi.proto" GrpcServices="Client" />
```

- Build Client

```
dotnet build
dotnet run
```

9. Installing gRPC on Windows

On Windows, there are two main installation methods for gRPC:

- Manual compilation using CMake.
- Installation via the vcpkg package manager.

Below are the details of both methods, including advantages, disadvantages, and step-by-step instructions.

9.1. Method 1: Manual Compilation Using CMake

Procedure:

- **Set up the environment:** Download and install CMake (version 3.16 or higher) and Visual Studio with the C++ workload.

- **Download the source code:**
```
git clone --recurse-submodules -b v1.74.0 --depth 1 https://github.com/grpc/grpc
cd grpc
```

- **Compile and install:**
```
mkdir cmake\build
cd cmake\build
cmake -DgRPC_INSTALL=ON -DgRPC_BUILD_TESTS=OFF -DCMAKE_CXX_STANDARD=17 -
        DCMAKE_INSTALL_PREFIX=%USERPROFILE%\g rpc . .
cmake --build . --config Release --target install
```

- **Set environment variables:**
```
set  PATH=%PATH%;%USERPROFILE%\grpc\bin
```

- **Test the installation:**
```
cd examples\cpp\helloworld\cmake\build
.\greeter_server.exe
.\greeter_client.exe
```

  Expected output:
```
Greeter received:   Hello world
```

Detailed instructions can be found in the official guide: Quick start | C++.

9.2. Method 2: Installation via the vcpkg Package Manager

Procedure:

- **Install vcpkg:**
  ```
  git clone https://github.com/microsoft/vcpkg.git
  cd vcpkg
  .\bootstrap-vcpkg.bat
  ```

- **Install gRPC:**

Method	Description
`.\vcpkg install grpc`	Install all packages, it takes a long time.
`.\vcpkg install grpc:x64-windows`	(Recommended) Install packages for x64 windows, dynamicly linked lib.
`.\vcpkg install grpc[core]:x64-windows`	This will install just the base without tests and other optional stuff.

- **Integrate with Visual Studio:**
  `.\vcpkg integrate install`

- **But now are not protoc and grpc in PATH:**
- **Paths needed to generate the necessary files from the proto file.**
  `protoc = "path/to/vcpkg/installed/x64-windows/tools/protobuf/protoc.exe"`
  `grpc_cpp_plugin = "path/to/vcpkg/installed/x64-windows/tools/grpc/grpc_cpp_plugin.exe"`
  all you need is in this folder: `"path/to/vcpkg/installed"`

- **Test the installation:** Create a new project in Visual Studio, add a dependency on gRPC, and verify functionality.

9.3. Summary of Advantages and Disadvantages

Method	Advantages	Disadvantages
Manual compilation (CMake)	Full control over configuration, ability to customize the build process.	Complex setup, requires build tools and time.
vcpkg	Easy installation, automatic dependency management.	Less flexible configuration, dependency on an external tool.

9.4. Recommendation

- **Beginners:** We recommend starting with vcpkg installation, which is simpler and faster.
- **Advanced users:** If you need specific configuration or optimizations, manual compilation with CMake is preferable.

10. Installing gRPC on Linux

On Linux, there are two main ways to install gRPC:

- Manual compilation using CMake.
- Installation via the vcpkg package manager (standard, classic, or minimalistic variant).

Below are the detailed instructions, advantages, and disadvantages of each approach.

10.1. Method 1: Manual Compilation Using CMake

Procedure:

- **Prepare the environment:**
```
sudo apt update
sudo apt upgrade
sudo apt install -y git cmake build-essential pkg-config curl zip unzip tar
```

- **Clone the gRPC source code:**
```
git clone --recurse-submodules -b v1.74.0 --depth 1 https://github.com/grpc/grpc
cd grpc
```

- **Build and install:**
```
mkdir -p cmake/build
cd cmake/build
cmake -DgRPC_INSTALL=ON -DgRPC_BUILD_TESTS=OFF -DCMAKE_CXX_STANDARD=17 \
        -DCMAKE_INSTALL_PREFIX=$HOME/grpc ..
cmake --build . --config Release --target install
```

- **Update environment variables:**
```
export  PATH=$PATH:$HOME/grpc/bin
```

- **Verify installation:**
```
cd examples/cpp/helloworld/cmake/build
./greeter_server
./greeter_client
```

  Expected output:
```
Greeter received:   Hello world
```

Detailed instructions can be found in the official guide: Quick start | C++.

10.2. Method 2: Installation via the vcpkg Package Manager

The vcpkg package manager allows easy installation of gRPC and its dependencies. There are three main variants: **standard**, **classic**, and **minimalistic**.

Preparation (common for all variants):

```
sudo apt update
sudo apt upgrade
sudo apt install -y net-tools curl zip unzip tar git cmake build-essential

git clone https://github.com/microsoft/vcpkg.git
cd vcpkg
./bootstrap-vcpkg.sh
```

Variant A: Standard Installation:

```
./vcpkg install grpc
./vcpkg integrate install
```

This installs gRPC with default settings and automatically integrates it with your compiler or IDE.

Variant B: Classic Installation:

```
sudo apt install -y pkg-config
./vcpkg install grpc OR ./vcpkg install grpc:x64-linux
# or explicitly with protobuf
./vcpkg install protobuf grpc OR ./vcpkg install grpc:x64-linux protobuf:x64-linux

# update system and export environment paths sudo apt update
export PATH=$HOME/vcpkg/installed/x64-linux/tools/grpc:$PATH
export PATH=$PWD/installed/x64-linux/tools/protobuf:$PATH

# verify installation
protoc --version
```

This approach allows full installation with optional protobuf integration.

Variant C: Minimalistic Installation:

```
mkdir -p vcpkg/triplets
nano vcpkg/triplets/x64-linux-release.cmake
```

Inside the file, define:

```
set(VCPKG_TARGET_ARCHITECTURE x64)
set(VCPKG_BUILD_TYPE release)       # Release-only build
set(VCPKG_LIBRARY_LINKAGE static)   # Static libraries minimize runtime dependencies
set(VCPKG_CRT_LINKAGE dynamic)      # Can be static or dynamic based on your needs
```

Then install only the necessary components:

```
./vcpkg install grpc:x64-linux-release protobuf:x64-linux-release
```

This variant minimizes build time and dependencies—ideal for lightweight deployments.

10.3. Summary of Advantages and Disadvantages

Method	Advantages	Disadvantages
Manual compilation (CMake)	Full control over the build process; can enable or disable specific components.	Complex setup, requires build tools and time.
vcpkg - Standard variant	Easiest to install; fully automated dependency management.	Less control over build configuration.
vcpkg - Classic variant	Supports protobuf and full customization; compatible with IDEs.	Slightly more setup steps required.
vcpkg - Minimalistic variant	Lightweight, static linking, fast installation.	Requires manual editing of triplet files, limited flexibility.

10.4. Recommendation

- **Beginners:** Use the vcpkg standard variant for the simplest setup.
- **Advanced users:** The classic variant provides more flexibility for custom environments.
- **Optimized deployments:** Use the minimalistic variant to build lightweight, static binaries for production use.

11. Tips

- Every time you remove or insert a card, you need to restart the Pickering Remote server.
- If a gRPC server is running and the shared memory driver version is not installed, no other program can control the cards.

12. References

[1] gRPC *A high performance, open source universal RPC framework*. From: https://grpc.io/, visited 2025-09-23.

[2] GitHub. *gRPC github repository*. From: https://github.com/ni/grpc-device, visited 2025-09-23.