Setup Guide

# Setting up the Graphiti

1. To start, connect your device to the Graphiti using a USB-A to Micro USB cable by connecting the USB-A to your device and the Micro USB cable to the Graphiti located on the left side of the Graphiti closest to Dot 3.
2. Next power the device on and press Space + Dot 8 to put the device into API mode. Alternatively, one can press Space + Dot 1 to put the device into Draw mode to enable the Draw API which includes Draw events.
3. Open the device manager from the Windows search bar in the taskbar by searching device manager
4. Under “Ports (COM & LPT)” identify the COM port the device has connected to which is “COM” followed by a number e.i. COM4 and note this for later

# Setting up your coding environment

1. Download the Graphiti library to anywhere on your computer that is under your user.
2. Now open the library in visual studio code by selecting “File” in Visual Studio Code’s activity bar. Now select “Open Folder”. Then select the place that you installed the library folder
3. Next, install the “C/C++ Extension Pack” by Microsoft  
   A screenshot of a computer

   AI-generated content may be incorrect.
4. Next, install the powershell extension “Powershell” by Microsoft

A screenshot of a computer

AI-generated content may be incorrect.

1. Now, open the powershell extension by selecting “Terminal” in Visual Studio Code’s activity bar. Then select “New Terminal”. This should open a powershell window under the terminal page on the bottom half of your screen in Visual Studio Code.
2. Before we continue, ensure that you have git installed as well as a valid generator and compiler. Make sure that your version of C++ is 23 or greater. I recommend using the Ninja generator and g++ compiler. If you already have a generator and compiler for C/C++ skip to step 7
   1. To install g++ and Ninja download msys2 from <https://www.msys2.org> and run the installer and follow the instructions
   2. Next open MSYS2 MSYS from the Start Menu
   3. Run this command in MSYS2 shell:
      * pacman -Syu
   4. Close the shell when prompted then reopen it and run again:
      * pacman -Su
   5. Next run:
      * pacman -S mingw-w64-x86\_64-gcc mingw-w64-x86\_64-ninja
   6. Then add this to your PATH
      * C:\msys64\mingw64\bin
3. Next let’s ensure you can run a ps1 script file. In the library there is an exe file that will allow the current user to execute ps1 script files. Now run this command in powershell in the to ensure that you have allowed ps1 scripts to execute the exe file on your computer as your user. It will prompt you to give it permission to run, when this happens select yes. Then a message box will appear that says, “Execution Policy set to RemoteSigned”, press okay or close the window to exit. In the powershell window run the following command to execute the exe file:

.\scripts\Set-ExecutionPolicy.exe

If you wish to read the contents of this exe file, it is in the Set-ExecutionPolicy.ps1 file that the exe is made from.

1. With the powershell window open in Visual Studio Code or “VS Code”, install the library and it’s dependencies by running the script file called “install\_library\_vcpkg.ps1”. To do this enter the command in powershell:

.\scripts\install\_library\_vcpkg.ps1

If the command does not execute, ensure you have valid permissions. If the window prompts us to give the file permission to execute, then do so. If you do not have valid permissions, run this command in an administrator terminal by opening a terminal as administrator, then navigating to the directory you installed the library, then running the same command.

1. If you wish to install the library using a specific generator and compiler do these steps. Navigate to the lib folder from the root directory by doing:

cd lib  
 If vcpkg is installed under your user profile you can compile the library using the library.ps1 script file. This file allows you to specify the generator and/or compiler using parameters like so:

.\library.ps1 -Generator "Ninja" -Compiler "g++"

Or you can specify no parameters by doing:

.\library.ps1

This will use your default generator and compiler.

1. Now add the include paths of your installations to the Include path of your C/C++ Configurations tool. You can open this any time with Ctrl+Shift+P

${env:USERPROFILE}\graphiti\include

${env:USERPROFILE}\vcpkg\installed\x64-windows\include

${env:USERPROFILE}\vcpkg\installed\x64-windows\include\hidapi

C:\Program Files\Java\jdk-20\include

C:\Program Files\Java\jdk-20\include\win32

${workspaceFolder}/\*\*

1. Now your environment is all set up. If you wish to test the CPP library the run the tests using:

.\scripts\tools.ps1 standard

OR

.\scripts\tools.ps1 standard\_debug

1. The wrappers can be used as stated in their README.md files

# Writing a CPP project

1. Start with making a CMakeLists.txt file with the contents below:

cmake\_minimum\_required(VERSION 3.15)

set(CMAKE\_CXX\_STANDARD 23)

add\_definitions(-D\_WIN32\_WINNT=0x0601)

# Set vcpkg toolchain (adjust path as needed)

# Make sure vcpkg has asio

set(CMAKE\_TOOLCHAIN\_FILE "$ENV{USERPROFILE}/vcpkg/scripts/buildsystems/vcpkg.cmake")

# Set path for Graphit library

set(CMAKE\_PREFIX\_PATH "$ENV{USERPROFILE}/graphiti")

# Your project

project(RunGraphiti)

# Find asio for Graphiti

find\_package(asio REQUIRED)

# Find Graphiti

find\_package(Graphiti REQUIRED)

# Create Executable

add\_executable(main src/main.cpp)

# Link Executable to Graphiti Library

target\_link\_libraries(main PRIVATE Graphiti::Graphiti)

1. To include the graphiti library do

#include <Graphiti/Extension.hpp>

This includes all the functions of the Graphiti API you’ll need for this project

1. Next call startUpVCP with your COM port identified in step four of “Setting up the Graphiti”, followed by whether you wish to enable or disable key and touch events using bool values. For example, startUpVCP(“COM4”, false, false) will disable both key and touch events and use the COM port COM4
2. Next run any code you wish interacting with the graphiti api by doing

“graphiti->function” where “function” is any function call you wish.

1. Finally call shutDownVCP with the same boolean values followed by delete graphiti to delete the graphiti object.
2. Generally between most calls to the graphiti a graphiti->sleep(2) is called or just a time sleep for two seconds in any way to give time for the Graphiti to send a response then checking the response.
3. Example code is given below:

#include <cstdio>

#include <Graphiti/Extension.hpp>

GraphitiExtension\* graphiti = new GraphitiExtension();

void sleepAndOutput(){

    graphiti->sleep(2);

    auto output = graphiti->getNextOutputEvent();

    if(output.has\_value()) {

        std::cout << output.value() << std::endl;

    } else {

        std::cout << "No value" << std::endl;

    }

}

int main() {

    bool keyEventsBool = false;

    bool touchEventsBool = false;

    std::cout << "Starting" << std::endl;

    graphiti = new GraphitiExtension();

    if(!graphiti->startUpVCP("COM4", keyEventsBool, touchEventsBool)){

        return 1;

    }

    graphiti->getDateAndTime();

    sleepAndOutput();

    graphiti->shutDownVCP(keyEventsBool, touchEventsBool);

    delete graphiti;

    std::cout << "Ending" << std::endl;

    return 0;

}