Utilizing Docker, Visual studio code, PlatformIO and “Dev container “extension on Ubuntu 23.04 for embedded system development (STM32, blue pill)

Nikul Padhya
-

16 de junio de 2023

> DISCLAIMER: I am a software developer, but programming embedded devices is a hobby of mine. Please keep in mind that there may be room for improvement in this article. Take the information presented here with a pinch of salt/coffee/cardamom tea and consider it as a personal perspective.

Alright, listen up! As I mentioned in the disclaimer, I’m a self-proclaimed embedded system programming enthusiast. My personal favorites in the board department include the

- STM32 (commonly known as the blue pill)
- ESP8266
- Arduino (though I am using less and less of them)
- NodeMCU

...but hey, I’m not just your average embedded programmer! I like to spice things up and dabble in other languages, like playing with fire in the Android and Linux OS realms. Recently, in a fit of audacity, I decided to embark on a daring adventure using my primary laptop for a build process. And boy, oh boy, did I end up polluting my poor primary OS with a heap of libraries just for that one task! That’s when I thought, “Hey, there’s gotta be a way to keep my development environment separate, right?”

So, in this guide, I proudly present to you the majestic duo of Ubuntu 23.04 as my primary (Host) OS and ***Docker’s Ubuntu*** image as my guest OS. Together, we shall unveil the secrets of using Visual Studio (VSCode) and Docker for all your embedded system development needs. Prepare yourself for a wild ride!

Things we will need

We will need the following app/extensions to properly setup our environment

- Docker
- – Ubuntu image (You can use any other image as well)
- Visual Studio code (VScode)
- – Dev container

- – PlatformIO

Installing docker to your system

Alright, let’s talk about Docker, shall we? (I know, I probably don’t need to explain this part. We’re all familiar with its fantastic benefits, but hey, it’s my article, so bear with me!).

Now, why is Docker the bee’s knees when it comes to development? Well, my friend, it offers an unparalleled advantage:

The separation of development tool chains

the ability to try out multiple development environments on a single machine. Imagine the power of separating those development tool chains and effortlessly experimenting with different language versions or applications. Need to compare **“Go”** version 1.16 with 1.17 or weigh the charms of **“Python”** 2.x against 3.x? With just a measly 100 MB, you can have separate dev environments for each version, and the best part? It won’t break a sweat!

And that’s where our hero, Visual Studio, swoops in to save the day. It seamlessly integrates with Docker, making it a breeze to switch between these splendidly isolated environments. Gone are the days of fumbling around with setups and configurations. Visual Studio has your back!

OK so as I mention in the disclaimer, I am a hobbyist when it comes to STM32 development, I do it every once in a while. Some time its every day for a month to changing 1 line every 6 month. In this scenario many time I loose the development environment, or sometime I loose my primary OS all together.

Now, as I mentioned earlier, I’m a hobbyist when it comes to STM32 development. Sometimes, I dive into it headfirst, coding away day after day for a month. Other times, I may only tweak a single line every six months. In this unpredictable roller-coaster, I’ve often found myself losing my development environment or, on occasion, bidding farewell to my primary OS altogether. It’s a struggle, my friend.

But wait, there’s more! Imagine the joy of

- Sharing an entire development environment with your team or friends. They can effortlessly dive into writing code without worrying about the arduous setup process. That’s the beauty of Docker. It’s like virtualizing your development environment, complete with all its tools and dependencies.
- And here’s the icing on the cake: you can save this magical environment as an image. So, if the unthinkable happens and you lose your setup, fear not! With a simple jump start, you can resurrect your beloved environment and continue where you left off.

By the way, Docker has its own documentation, and it’s quite handy if you’re eager to install it on Ubuntu. Check it out here and embrace the wonders of containerization ***here***.

Pulling the Linux image

Assuming you now have Docker up and running on your system, it’s time to pull the latest Ubuntu image from the Docker repository. Brace yourself, for it only takes a simple command to make the magic happen:

```
docker pull ubuntu:latest
```

But wait, there’s more! It’s not just Ubuntu that awaits you in this virtual wonderland. Oh no, my friend, the world of OS images is your oyster! Feel free to explore and savor the flavors of other operating systems. You’re probably already familiar with this concept, but hey, I had to mention it. The possibilities are endless!

Running Docker image

Now, let’s embark on the exciting journey of running the newly pulled ***Ubuntu*** image.

But hold your horses, my friend, as we’re not just running any ordinary image here. Since we’re operating in a Docker virtual environment, we must grant our beloved “*ubuntu*” image all the necessary privileges to communicate with USB devices, including the STLINK USB. Fear not, we have the sacred command that will accomplish this feat.

```
docker run -it --name ubuntu -p 4444:4444 -v
"$(pwd)/app":/usr/src/app --privileged -v
/dev/bus/usb:/dev/bus/usb ubuntu /bin/bash
```

By executing this command, we create a Docker container named “*ubuntu*” that assumes its role with divine privileges

Installing VSCode

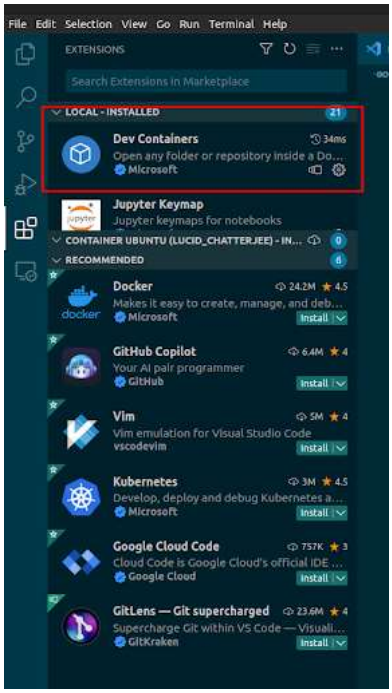
> VSCode is hands down the best IDE (period !!).

On your host system you can install VScode using many tools like “APT, SNAP or manually downloading the package”, we will use the APT way to install it, by running the following commands.

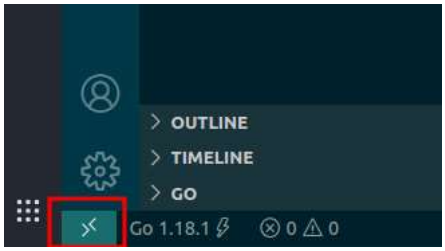
```
sudo apt update && sudo apt upgrade -y && sudo apt
install code -y
```

Installing Dev container extension

Once we have Visual studio code installed, Lets had over to the “**extension**” tab to install the one extension which makes all this possible. (Now that I think about it, I should buy a coffee for the developers @microsoft)

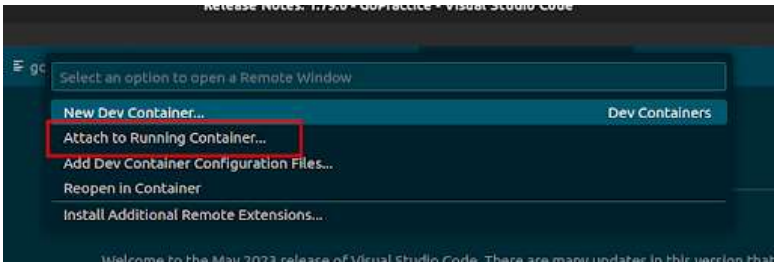


Once installed we will see the *“**Dev environment**”* option at the bottom left of our VScode IDE.



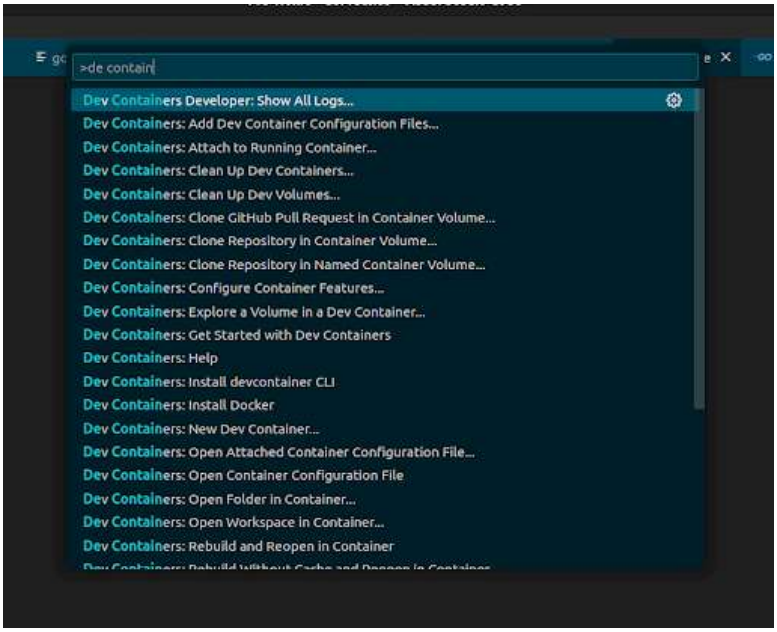
Connecting to running docker image from VScode IDE using Dev container extension

Once you have the “**Dev Container”** installed and ready, you can click on that icon (see above image) to get the **Dev container** Menu



or by opening command palette and typing

```
"ctrl + shift +p

Dev container"
```



The option we are looking for is

```
Dev Container : Attach to Running container
```

Since our docker image is running in the background, the extension will automatically list it and we can connect to it.

Once connected, VScode will spin-up another instance attached to the docker image.

This VScode instance, just like the virtual docker is a separate than your primary instance, so it will be like a newly installed VS code and it wont have any extension you might have installed in your main instance.

The bottom left if this instance will show the indicator that it is attached to which docker image.

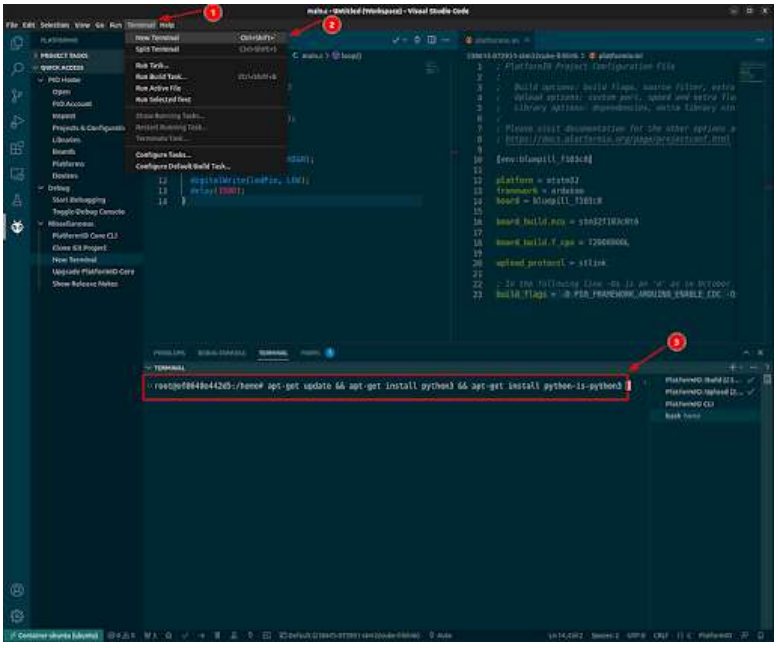
Installing python in the docker image

In the next step we will be installing *PlatformIO* in our VScode IDE, but in order to install ***PlatformIO*** we need to install ***“Python3”.***

1. So lets had over to the “***terminal***”
2. Select a “***New Terminal***”

3. use the following command to install the python3 *(with this command we are installing one more extiension, so linux recognize python3 as python)*

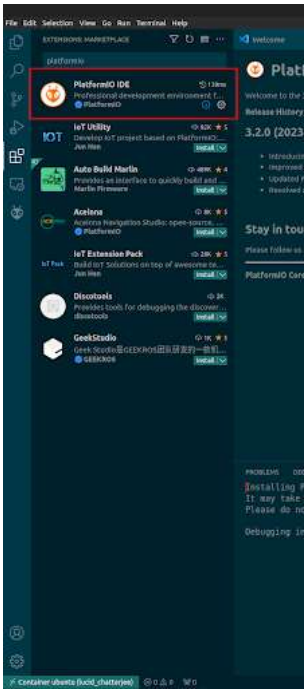
```
apt-get update && apt-get install python3 -y && apt-get install python3-venv -y && apt-get install python-is-python3
```



Installing PlatformIO in VScode IDE

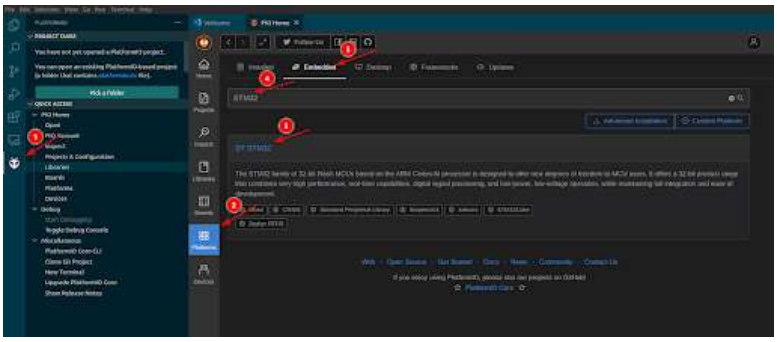
Like I mentioned the new instance is like a newly installed VS code so we will have to install all the needed/favorite extensions again, but this is a 1 time activity.

So now lets had over to “extensions”tab again, keeping in mind that we are in the docker connected VS code and not the original one (quick tip. You can close the original VScode if not needed so we don’t get confused). In the extensions tab, lets install “**PlatformIO IDE**” extension as well. This is the extension that helps VScode to communicate to the *blue pill.* (TODO to self, I should buy a coffee for the developers @microsoft)

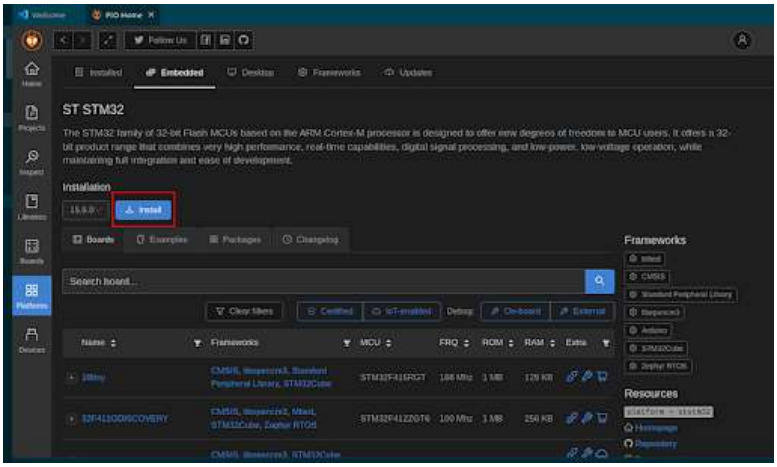


Installing required platform for STM32 development in the docker

Once the PlatformIO extension is installed, you will have a new section in the “docker connected” VScode instance named “PIO Home”



1. Lets head over to ***PIO Home***, and we need to get all the dependencies for our board, in our case its STM32.
2. Click on the “***Platforms***” tab
3. Select “***Enbedded***” from the top options
4. Search for “***STM32***”
5. The result should get up an “***ST STM32***” result
6. Select the “***ST STM32”,*** and click ***install***

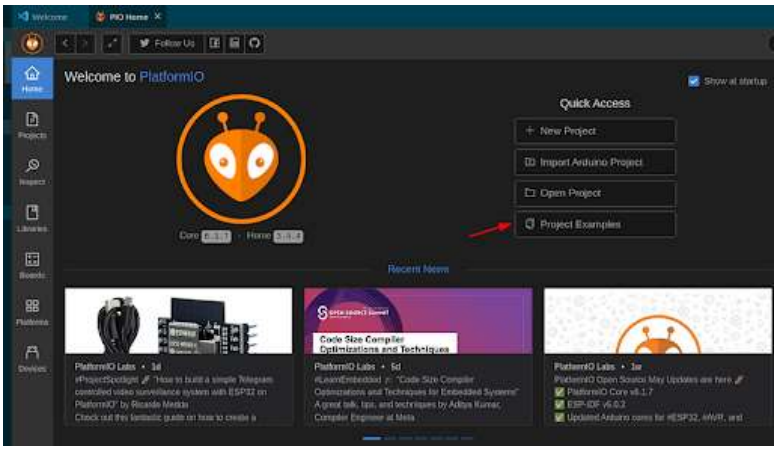


This should install all the required libraries for us to get started with the development

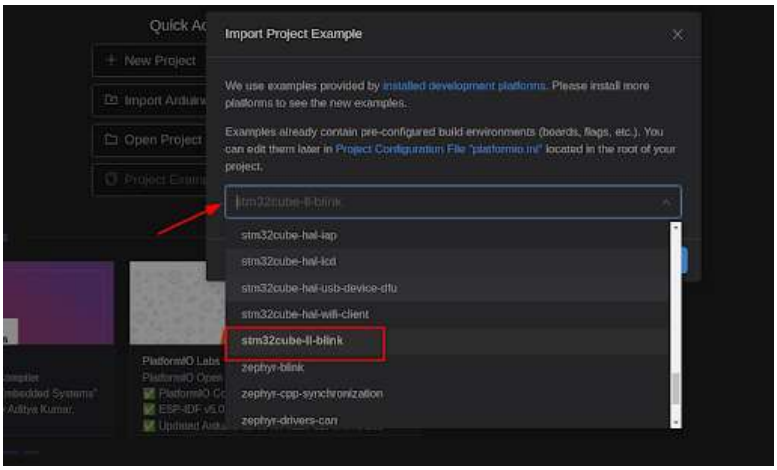
Getting sample project and building it

To get started with the development, “**PlatformIO**” gives many example project. And just like “Hello World !” of programming we will start with “blink” example for STM32

Click on “***Project Examples***”



Select “***stm32cube-ll-blink***” and select import.



This should take a minute of two, but at the end of the process, you will have the “***blink***” example setup in your own virtual docker container

Making changes to the sample project and building and deploying it to the hardware.

We need to make changes to 2 files in order to build and deploy the “blink” sample to our STM32.

1. ***platform.ini***

Platform.ini file gives the required matadata to ***PlatformIO*** on how to deploy the build. So had over to platform.ini and change the content to the following (here is the explaination for this settings)

```
[env:bluepill_f103c8]

platform = ststm32
framework = arduino
board = bluepill_f103c8

board_build.mcu = stm32f103c8t6

board_build.f_cpu = 72000000L

upload_protocol = stlink

; In the following line -Os is an 'o' as in October

build_flags = -D PIO_FRAMEWORK_ARDUINO_ENABLE_CDC -
Os
```

here, I am using STLINK device to connect to my STM32.

```
upload_protocol = stlink
```

For more detail on this options, you can refer to PlatformIO’s documentation. Which has a really well documented configurations for other boards here

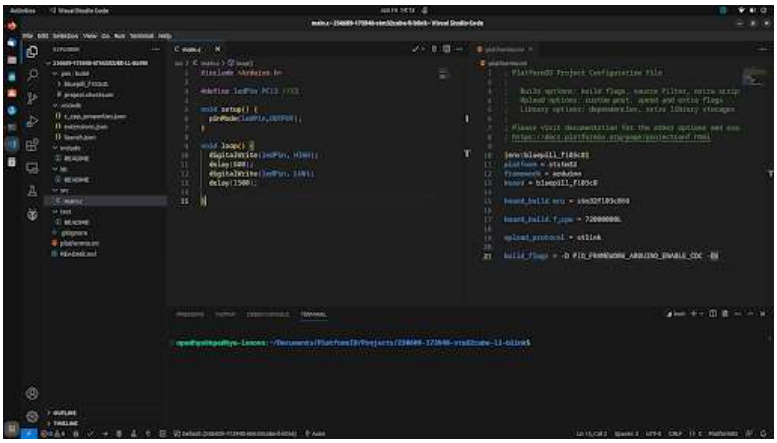
2. ***main.c***

Had over to ***main.c*** and change its content to the following. This wont do any black magic but it will allow us to test our setup. On the **“Blue pill”** board there is a *LED* connected to *Pin-13*.

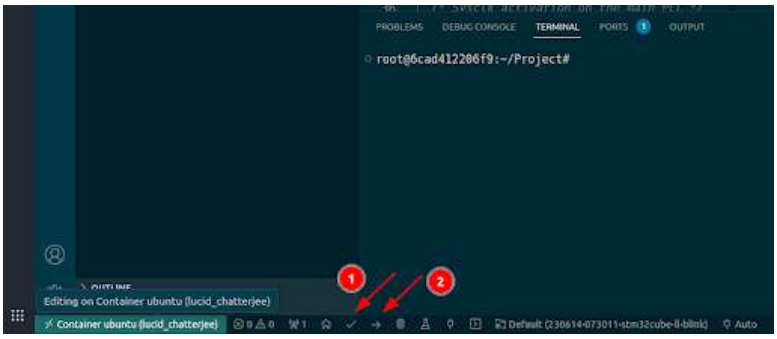
```
# include <arduino.h>
# define ledPin PC13

void setup() {

 pinMode(ledPin,OUTPUT);

}

void loop() {

 digitalWrite(ledPin, HIGH);

 delay(500);

 digitalWrite(ledPin, LOW);
```

```
    delay(1500);

  }
```

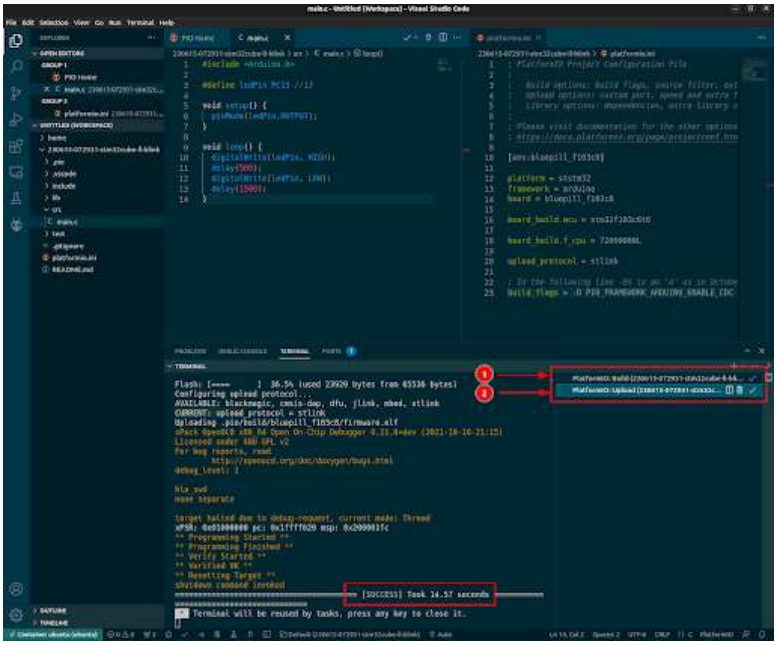
so ultimately the 2 files we changed should look like this



Now, you can use the build and deploy action in VScode to deploy this code to your STM32 blue pill, by using the following two commands



If everything is good, We will see “**success**” in the terminal as we click the “**PlatforIO: Upload**” (2nd button in the above screenshot)



KUDOS..... You have successfully learn something new today.

Some points to keep in mind

To ensure smooth operations within a Linux environment, it is essential to grant your OS User the necessary privileges and permissions. These permissions include accessing files, installing software, and utilizing USB devices.

It is important to be mindful of the dual nature of VSCode, with separate instances running in the Guest OS and Host OS. Therefore, exercise caution to execute commands in the appropriate instance.

Before launching the Docker container, ensure that it possesses the required access permissions. Keep the *“docker run”* command readily available for each instance you intend to run. This will ensure a seamless experience whenever you initiate the container, granting it the necessary privileges to function effectively within your environment.

NEXT STEP

As an exercise, you can save this docker environment and publish the image for future use.

Denunciar este artículo

Comentarios

20 • 5 comentarios

Recomendar Comentar Compartir

Añadir un comentario...

Nikul Padhya • +3er — 1 año •••

-

Ryan F., Thanks! Yaa highly technical topics need to be engaging, even I have a habit to wonder off, if the read is not interesting.

Mostrar traducción

Recomendar · 2 | Responder

Varad Joshi • +3er — 1 año •••

Cybersecurity/Observability Presales | Story Teller | Funny Soul | Chess Enthusiast | Wannabe Content Creator

Always knew about your programming skills. Now I know about your writing skills too

Mostrar traducción

Recomendar · 2 | Responder · 2 respuestas

Nikul Padhya • +3er — 1 año •••

-

Thanks dude !!!

Mostrar traducción

Recomendar | Responder

Ryan F. • +3er — 1 año •••

Customer Minded Professional Driven by Results Without Compromising Values

Agreed! Brilliantly witty and engaging for the relatively non-tech audience (me). Nice work.

Mostrar traducción

Recomendar · 1 | Responder

Mostrar más comentarios

¿Te ha gustado este artículo?

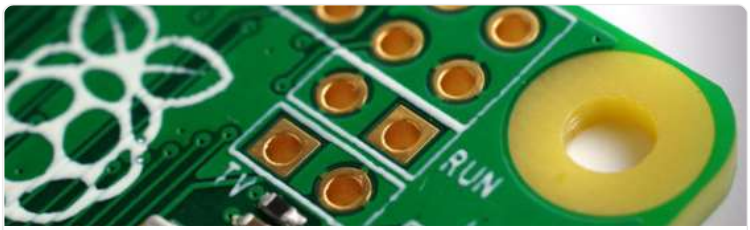
Sigue para no perderte ninguna novedad.

Nikul Padhya

-

Seguir

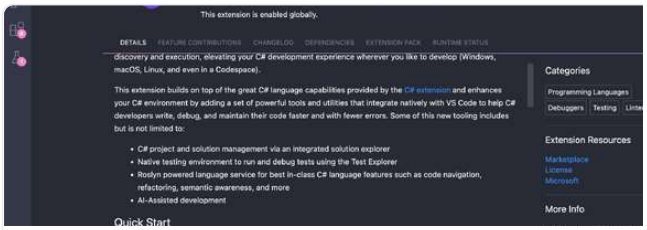
Más artículos para ti



Visual Studio Code comes to Chromebooks, Raspberry Pi

Manish Katoch

1



Master C# Development in VSCode: Writing and Running Tests

Rachel G.

21