

Git:

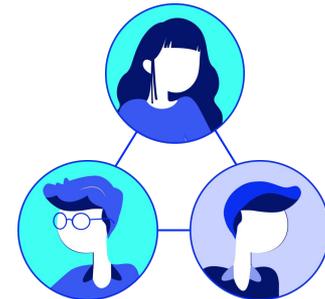
Desarrollo colaborativo

Módulo 4

Desarrollo colaborativo

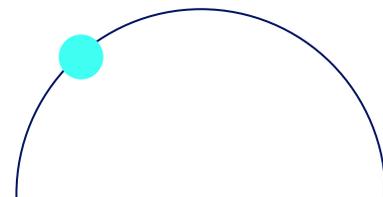
Desarrollo colaborativo

GitHub está diseñado alrededor de un flujo de **trabajo focalizado en Pull Requests**. Esto permite que los **desarrolladores colaboren** tanto en repositorios unificados por grupo, como en aquellos que son distribuidos en forma global a través de compañías o incluso personas desconocidas.



Por lo general, la acción de colaborar funciona de la siguiente manera:

1. Hacemos un **Fork** de un proyecto.
2. Creamos un **branch** nuevo desde el **branch master**.
3. Realizamos algunos **commits** para mejorar el proyecto.
4. Hacemos un **push** del **branch** al proyecto de GitHub.
5. Abrimos un **Pull Request** en GitHub.
6. Generamos una discusión al respecto de nuestro trabajo realizado y, en forma alternativa, seguimos haciendo más **commits**.
7. Sincronizamos el **branch master** nuevamente a nuestro **Fork**.



Git fetch

Como ya hemos mencionado, el comando **git clone** agrega implícitamente un **origin** (origen) remoto en el repositorio local.

Veremos el caso en el que se trabaje con otros desarrolladores.

Si en el repositorio local, no tenemos el historial de cambios realizados por otros usuarios, podemos obtener esa información con:

```
> git fetch <alias_remoto>
```

git fetch traerá todas las referencias a objetos **commit** que estén presentes en el repositorio remoto - que no tengamos copiado en el local -.

Esto brinda la posibilidad de inspeccionar el repositorio con posterioridad o bien, si estamos seguros de que es un trabajo testeado, unirlo con nuestro trabajo actual.

En caso de estar trabajando **con un repositorio clonado, simplemente se puede correr el comando `git fetch origin`** (ya que, como vimos, clonar un repositorio remoto crea automáticamente un remoto local con alias “**origin**”).

Es importante notar que el comando **`git fetch`** solo descarga el historial de cambios al repositorio local, sin embargo **no realiza ningún merge con el repositorio actual**. Para ello, tenemos que integrar ambas ramas de trabajo en forma manual.



Git pull

Si la rama actual está configurada para apuntar a una rama remota, podemos alternativamente usar el siguiente comando:

```
> git pull
```

A diferencia del comando **git fetch**, **git pull** **descargará todos los cambios nuevos** que no tengamos en nuestro repositorio local desde el repositorio remoto **y va a intentar integrar automáticamente nuestro trabajo local actual con el que acaba de traer.**

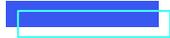
Esto puede ser de gran ventaja para el flujo de trabajo ya que **no dependemos de una integración manual** en caso de estar seguros de lo que estemos descargando desde el proyecto remoto.



Pull Request

Pull Request

Imaginemos que tenemos un **Fork** de algún proyecto en nuestra cuenta de usuario. Queremos realizar modificaciones en él y sugerirlos como cambios potenciales dentro del proyecto original.



Pasos a seguir:

1. Generar el **Fork** del proyecto.
2. **Clonarlo** en forma local.
3. Crear una **rama de trabajo nueva**.
4. **Realizar cambios** de código.
5. **Verificar** que los cambios estén bien y hayan sido **testeados**.
6. Confirmar nuestros cambios con un **commit**.
7. Realizar un **push** de esa rama al **Fork** que tenemos en GitHub.



Tu primer Pull Request

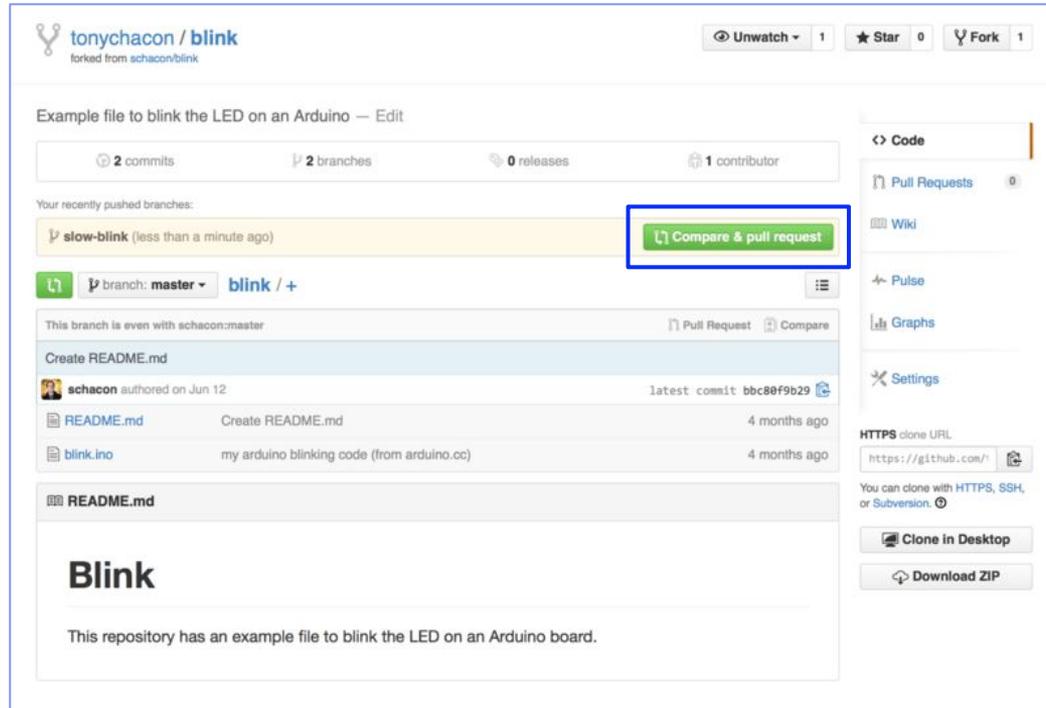
Una vez dentro de GitHub podemos observar cómo la página reconoce que fue creado y subido un nuevo **branch**. Además, va a aparecer con un gran botón verde con la leyenda **Compare & pull request**, que nos permitirá generar un primer **Pull Request**.



Veamos un ejemplo en el siguiente slide.



Verás en pantalla:



The screenshot shows the GitHub interface for the repository 'tonychacon / blink', which is a fork of 'schacon/blink'. The repository has 2 commits, 2 branches, 0 releases, and 1 contributor. A recently pushed branch 'slow-blink' is highlighted in yellow, with a green 'Compare & pull request' button next to it, which is circled in blue. The repository contains a README.md file and a blink.ino file. The README.md file is displayed below the file list, showing the title 'Blink' and the text 'This repository has an example file to blink the LED on an Arduino board.'

tonychacon / blink
forked from schacon/blink

Unwatch 1 Star 0 Fork 1

Example file to blink the LED on an Arduino — Edit

2 commits 2 branches 0 releases 1 contributor

Your recently pushed branches:

slow-blink (less than a minute ago) Compare & pull request

branch: master blink / +

This branch is even with schacon:master Pull Request Compare

Create README.md

schacon authored on Jun 12 latest commit bbc80f9b29

File	Commit	Time
README.md	Create README.md	4 months ago
blink.ino	my arduino blinking code (from arduino.cc)	4 months ago

README.md

Blink

This repository has an example file to blink the LED on an Arduino board.

Code

Pull Requests 0

Wiki

Pulse

Graphs

Settings

HTTPS clone URL

https://github.com/tonychacon/blink

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop

Download ZIP

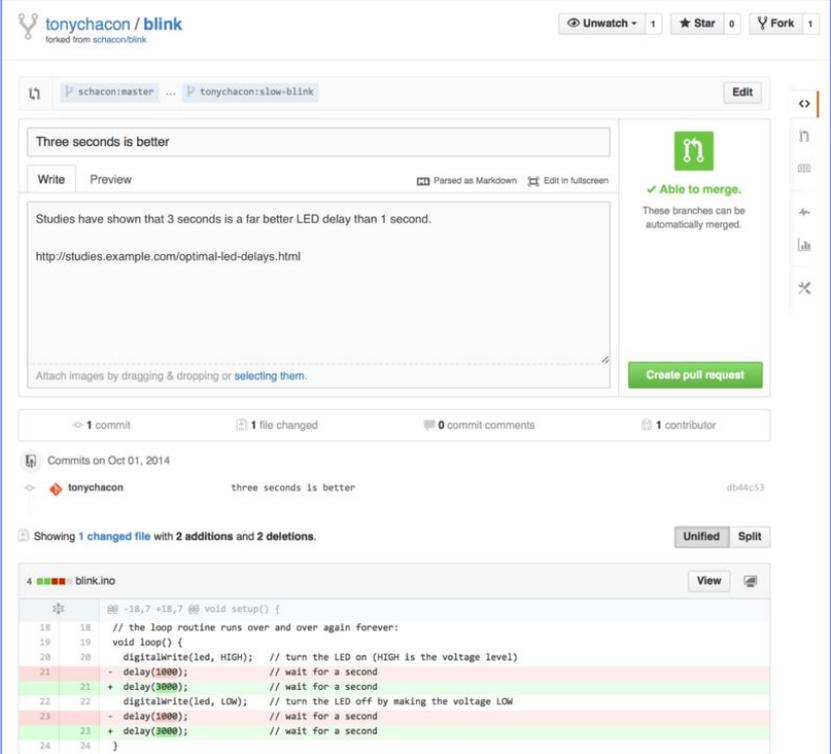
Si hacemos clic en el botón, accedemos a una ventana que **nos pedirá un título y una descripción** de nuestro **Pull Request**.

Usualmente, vale la pena esforzarnos en pensar ambos datos, ya que va a ser de mucha ayuda para el dueño del proyecto original a determinar qué cambios estuvimos haciendo y si los tiene que aceptar o no.

También podemos ver una lista de uno o más **commits** en nuestro **branch** que están por sobre (**"ahead"**) el **branch master** y un visor de diferencias unificadas que van a tomar efectos en caso de que nuestro trabajo sea aceptado.



Verás la lista así:



tonychacon / blink
forked from sachaon/blink

Unwatch 1 Star 0 Fork 1

schacon:master ... tonychacon:slow-blink Edit

Three seconds is better

Write Preview Parsed as Markdown Edit in fullscreen

Studies have shown that 3 seconds is a far better LED delay than 1 second.
http://studies.example.com/optimal-led-delays.html

Attach images by dragging & dropping or selecting them.

✓ Able to merge.
These branches can be automatically merged.

Create pull request

1 commit 1 file changed 0 commit comments 1 contributor

Commits on Oct 01, 2014

tonychacon three seconds is better db44c53

Showing 1 changed file with 2 additions and 2 deletions. Unified Split

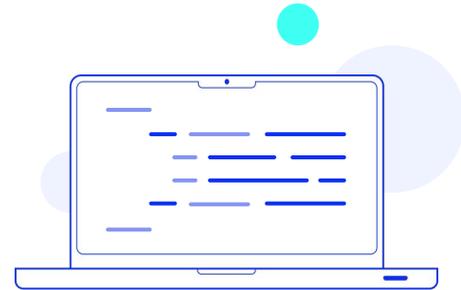
blink.ino View

```
@@ -18,7 +18,7 @@ void setup() {  
18 18 // the loop routine runs over and over again forever:  
19 19 void loop() {  
20 20   digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)  
21 21   - delay(1000); // wait for a second  
22 22   + delay(3000); // wait for a second  
23 23   digitalWrite(led, LOW); // turn the LED off by making the voltage LOW  
24 24   - delay(1000); // wait for a second  
25 25   + delay(3000); // wait for a second  
26 26 }
```

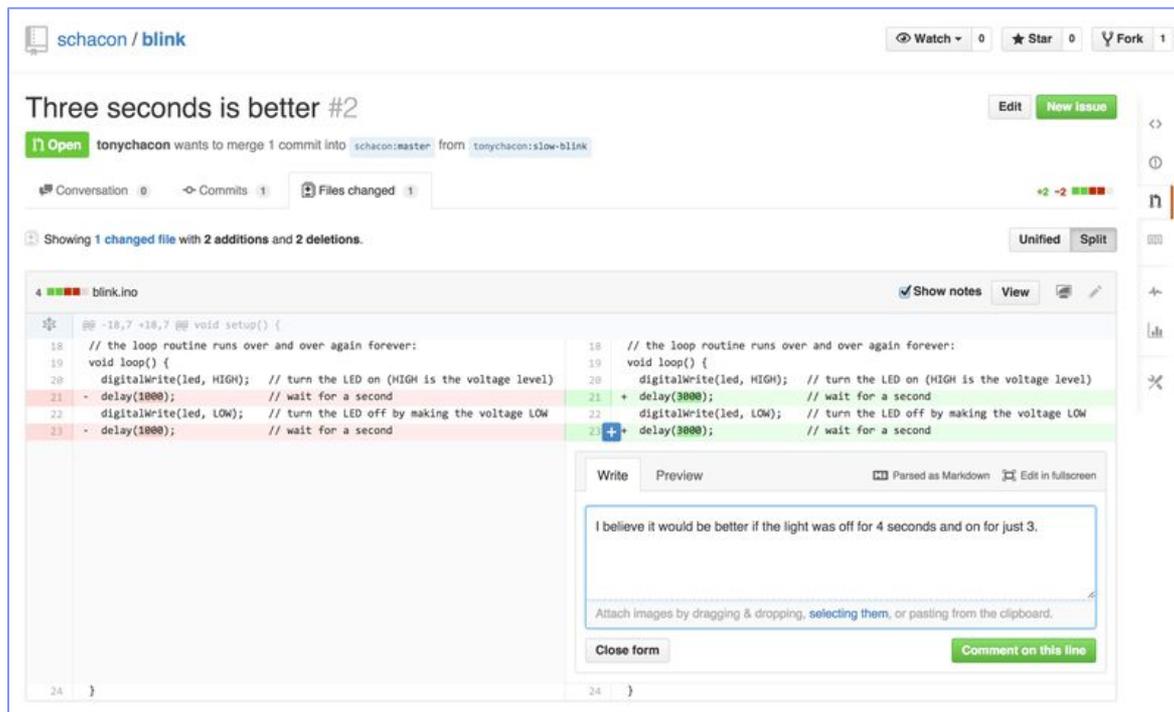
Cuando hagamos clic en el botón **Create pull request**, el dueño del proyecto del que hicimos el **Fork** va a recibir una notificación de que alguien está sugiriendo cambios y va a poder acceder a toda la información de los cambios que propusimos.

En este punto, el dueño del proyecto puede ver los cambios sugeridos, integrarlos con el repositorio original, rechazarlos o realizar comentarios sobre ellos.

En la **vista de diferencias**, el dueño del proyecto puede hacer clic en cualquier línea de cambio propuesta para dejar un comentario sobre alguna de ellas.



Aparece en pantalla:



schacon / blink

Watch 0 Star 0 Fork 1

Three seconds is better #2

Open tonychacon wants to merge 1 commit into schacon:master from tonychacon:slow-blink

Conversation 0 Commits 1 Files changed 1

Showing 1 changed file with 2 additions and 2 deletions.

Unified Split

4 blink.ino

```
@@ -10,7 +10,7 @@ void setup() {
18 // the loop routine runs over and over again forever:
19 void loop() {
20   digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
21   - delay(1000); // wait for a second
22   digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
23   - delay(1000); // wait for a second
18 // the loop routine runs over and over again forever:
19 void loop() {
20   digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
21   + delay(3000); // wait for a second
22   digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
23   + delay(3000); // wait for a second
24 }
```

Write Preview

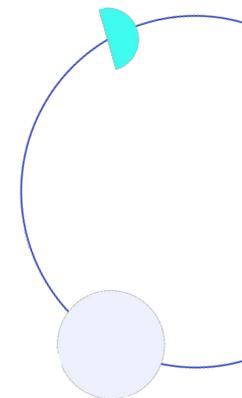
Parsed as Markdown Edit in fullscreen

I believe it would be better if the light was off for 4 seconds and on for just 3.

Attach images by dragging & dropping, selecting them, or pasting from the clipboard.

Close form Comment on this line

Una vez que este cambio fue comentado, el usuario que abrió el **Pull Request** (y cualquier otra persona que le esté dando seguimiento al repositorio) va a recibir una notificación. Si tenemos configurada la cuenta para poder recibir notificaciones por e-mail, podríamos ver un correo similar al siguiente:



Además, cualquier usuario puede dejar comentarios en “generales” en el **Pull Request**.

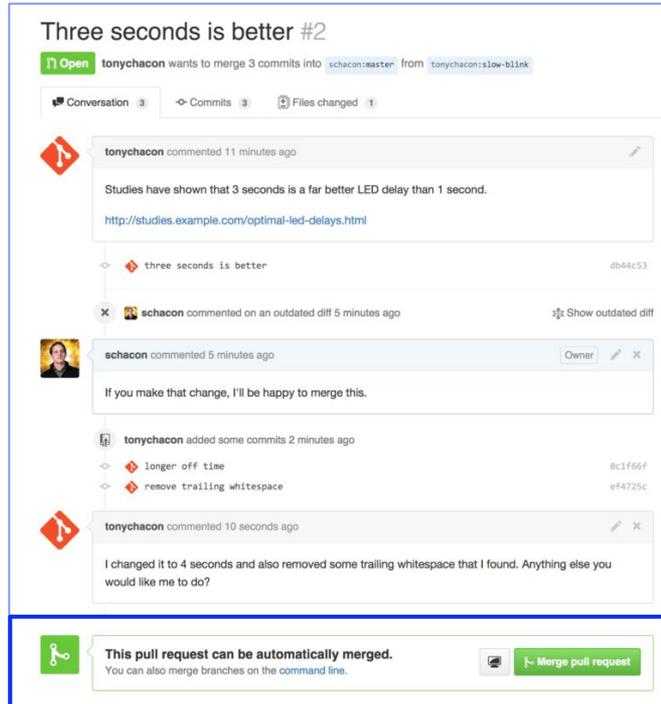
Ahora el usuario que está contribuyendo al proyecto original simplemente puede realizar más **commits** en el branch en cuestión y volver a hacer un **push**.

Esta acción actualizará automáticamente el Pull Request. Si se afectan líneas de código que habían sido previamente comentadas por alguien, se verán colapsadas en la página de la discusión del **Pull Request**.

Agregar **commits** a un **Pull Request** existente no activa una notificación. Una vez que se hayan realizado los cambios o corregido el trabajo, lo ideal es **dejar un nuevo comentario para informar sobre ellos al dueño del proyecto**.



Ejemplo:



The screenshot shows a GitHub pull request titled "Three seconds is better #2". At the top, it indicates that "tonychacon" wants to merge 3 commits into "schacon:master" from "tonychacon:slow-blink". The interface includes tabs for "Conversation", "Commits", and "Files changed".

The conversation history shows:

- A comment from **tonychacon** (11 minutes ago) stating: "Studies have shown that 3 seconds is a far better LED delay than 1 second." with a link to <http://studies.example.com/optimal-led-delays.html>. Below the comment is a commit diff for "three seconds is better" (commit hash: db44c53).
- A comment from **schacon** (5 minutes ago) stating: "If you make that change, I'll be happy to merge this." with a "Show outdated diff" link.
- A comment from **tonychacon** (2 minutes ago) stating: "I changed it to 4 seconds and also removed some trailing whitespace that I found. Anything else you would like me to do?". Below this is a commit diff showing changes: "longer off time" (commit hash: bc1f66f) and "remove trailing whitespace" (commit hash: ef4725c).
- A comment from **tonychacon** (10 seconds ago) stating: "I changed it to 4 seconds and also removed some trailing whitespace that I found. Anything else you would like me to do?".

At the bottom of the interface, a green banner states: "This pull request can be automatically merged. You can also merge branches on the command line." To the right of this banner is a green button labeled "Merge pull request".

Algo interesante es que **GitHub verifica que el Pull Request pueda ser integrado de manera limpia y sin problemas**. En ese caso, se mostrará un botón de atajo para realizar esta operación.



**¡Sigamos
trabajando!**