

BATS: Bricolabs Arduino Training Shield (con kicad)

Bricolabs

20 de xunio de 2015

NOTA IMPORTANTE Lee a sección **META** para clonar este proxecto.

Qué imos facer? Qué se describe en este documento?

Imos deseñar un shield para Arduino. Usando [KiCad](#)

Daremos unha visión xeral da funcionalidade da suite KiCad pero non vamos a redactar un tutorial detallado nin de deseño de circuitos nin do mesmo KiCad. Daremos as pistas xustiñas para empezar a traballar con KiCad, se queredes un tutorial detallado, en youtube tedes un super recomendable, elaborado por [TutoElectro](#)

Tamén comentaremos a nivel xeral os pasos para desenvolver este proxecto.

Qué é KiCad?

KiCad é unha suite de deseño electrónico automatizado ¹. KiCad permite o deseño tanto de esquemas de circuitos como das placas de circuito impreso a nivel profesional. Hai versións de KiCad dispoñibles para Windows, Linux, Apple OS X. A suite está dispoñible para Windows, Linux e Apple OS X. É un programa gratuito e **libre** distribuído baixo licencia **GNU GPL v2**.

Mellor aínda, a suite KiCad é a elegida polo CERN para o desenvolvemento e deseño de electrónica. É de esperar que con este respaldo a suite mellore aínda máis.

¹EDA suite en inglés

Requisitos do shield

Queremos facer un shield que poida servir para iniciarse no mundo da programación con Arduino, especialmente centrado no ensino (tecnoloxía 3º - 4º ESO). Neste eido, traballar sobre breadboard pode ser complexo, moitas veces o material é compartido e isto implica que, cada clase, hai que montar e desmontar o circuito que logo se programará. Isto leva o seu tempo e ocasiona erros de montaxe que son difíciles de detectar, polo que o uso dun shield parece que pode ser unha solución.

Malia isto, empregando un shield xa ensamblada se perde a oportunidade de profundizar os coñecementos de electrónica e circuitos que se adquiren ao facer un mesmo a montaxe, así que un shield na que o alumno/a teña que colocar e soldar el mesmo os compoñentes, segundo avanza na súa aprendizaxe, parécenos a solución de compromiso idónea.

Queremos que o shield sexa escalable, que sirva para traballar cuns poucos compoñentes se non se quere profundizar demasiado, pero que permita tamén chegar a niveis máis avanzados e con máis compoñentes sen cambiar de shield.

Estes son os compoñentes que se pensan incluír:

- 3-4 x Botóns
- 1 x RGB
- 1 x LDR
- 1 x Pines servo
- 1 x sensor temp (dudoso)
- 2 x potenciómetros
- 1 x LED vermello
- 1 x LED amarelo
- 1 x LED verde
- 1 x Zoador
- 1 x Botón de reset

Instalación de KiCad (en Ubuntu)

Para instalar o KiCad en Ubuntu basta con facer o típico ciclo de instalación:

```
sudo apt-get install kicad
```

Se queremos estar á última temos o ppa de Monsieur Reynaud dispoñible:

```
sudo apt-add-repository ppa:js-reynaud/ppa-KiCad
sudo apt-get update
sudo apt-get install kicad
```

Nos escollimos esta opción.

Se non usades un linux baseado en Debian, teredes que consultar na rede como facer a instalación para o voso sistema operativo. De todos os xeitos a instalación é moi doada, donde podemos atopar algún problemíña é na instalación das bibliotecas de compoñentes que vos contaremos cos mais detalle mais adiante.

Configuración de directorios para este proxecto

Además de desenvolver o proxecto con KiCad queremos ter o proxecto dispoñible en github.

Agora que temos KiCad instalado imos preparar un directorio de traballo ao que chamamos **bats**.

O directorio **bats** será o “repositorio” ou depósito do noso proxecto para git. Contén os seguintes subdirectorios:

doc Contén a documentación do proxecto (o que estás a leer agora mesmo) redactada en [Pandoc](#)

kicad Contén o proxecto KiCad

Unha vez que temos preparado o directorio do proxecto activamos git para iniciar o control de versións.

Describir a configuración de git??

Biblioteca de compoñentes incluíndo un shield para Arduino

As bibliotecas de KiCad están organizadas en dúas partes:

- Un ficheiro que contén os símbolos dos compoñentes para usarse no editor de esquemas electrónicos **Eescheme**
- As pegadas dos compoñentes electrónicos, é dicir, a forma que ten que ter a pista da placa de circuito impreso (*PCB*) para poder soldar o compoñente.

O KiCad non trae por defecto unha biblioteca de compoñentes que inclúa shields de Arduino. Pero non hai problema hai bibliotecas que podemos descargar da rede.

Unha biblioteca moi completa é a de Freetronics que podemos atopar tamén en github en:

https://github.com/freetronics/freetronics_KiCad_library.git

As bibliotecas de KiCad poden estar almacenadas en diferentes directorios do noso ordenador. Poderíamos engadir as bibliotecas que usemos en algún sub-directorio de */usr/share/kicad* ou de */usr/local/share*. Esta podería ser unha boa estratexia nun servidor compartido por varios usuarios. Tamén poderíamos descargar todas as bibliotecas a un directorio común do noso *home*. Pero como estamos facendo un control de versións do noso proxecto con git a propia páxina da biblioteca suxírenos o xeito mais adoitado de facer a instalación: coma un submódulo git do noso proxecto.

Describir as vantaxes de usar un git submodule

Engadir a biblioteca como un submódulo de git

Dende o directorio principal de noso proxecto descarregamos a biblioteca de Freetronics coma un submodule do noso proxecto:

```
git submodule add https://github.com/freetronics/freetronics_kicad_library.git kicad/ftlib
```

Despois de engadir a biblioteca como un submódulo se consultamos o estado git do noso proxecto aparecerán dous novos ficheiros:

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   .gitmodules
    new file:   kicad/ftlibrary
```

Git engadiu automaticamente os dous novos ficheiros, o directorio que contén a nosa biblioteca eo ficheiro **.gitmodule** que levará o control de todos os submódulos que usemos.

En realidade os ficheiros que compoñen a biblioteca non pertencen ao noso depósito de software, git só leva conta da versión da biblioteca de Freetronics que estamos a usar.

Se queres saber mais de esta potente funcionalidade de git podes leer: <http://www.git-scm.com/book/en/v2/Git-Tools-Submodules>

Configurar *Eescheme* para usar a nova biblioteca

No menú Preferences -> Component Library sinalamos na ventá inferior o directorio do noso proxecto. Na ventá superior engadimos o ficheiro da biblioteca.

No github da biblioteca nos aconsellan poñer a biblioteca de primeira na nosa lista por que definen todo tipo de compoñentes. Nos de momento seguimos o consello de Freetronics, e a puxemos de primeira.

Configurar acceso aos datos de pegadas (*footprints*) en pcbnew

Configuramos un ficheiro para o noso proxecto declarando os *footprint* extra que imos a utilizar.

```
(fp_lib_table
  (lib
    (name FT)
    (type KiCad)
    (uri ${KIPRJMOD}/ftlibrary/freetronics_footprints.pretty)
    (options "")
    (descr "Freetronics Kicad Library")
  )
)
```

Engadimos o novo ficheiro ao noso repositorio

```
git add fp-lib-table
```

Abrimos *Pcbnew* e no menú *Preferences->Footprint Libraries Manager* comprobamos que na pestaña *Project Specific Libraries* figura o noso ficheiro.

Tutorial

Checklist: Pasos a seguir para rematar o proxecto

- Abrir o proxecto (e mellor tér un directorio dedicado ao proxecto creado en adianto)
- Crear o esquema do circuito (usando Eescheme)
- Chequeo de erros (opción *Perform Electrical Rules Check*)
- Xerar o ficheiro NET (opción *Generate netlist*)
- Asignar as pegadas (*footprints*) aos compoñentes (opción *run CvPcb* dende o *Eescheme*)

A pantalla xeral

Pantalla xeral de KiCad opcións, citar a lista de hotkeys

Abrindo un proxecto

Abrimos un novo proxecto: File::New Project (Ctrl+N) **bats**

falar das propiedades do documento

Crear e Editar o esquema do circuito

O primeiro que imos facer é o esquema do circuito. Para isto temos que usar a ferramenta *Eeschema* que podemos atopar en tres lugares diferentes ² na barra de iconos de ferramentas, no menú de KiCad no título da fiestra, ou có atallo **Ctrl+E**.

Abrimos eescheme e creamos un novo ficheiro de esquema.

²Isto de ter varios xeitos de facer unha cousa é habitual en KiCad como iremos vendo

ERC: comprobación dos erros no circuito

Tips

- Falar dos flags
 - Power flags
 - Not used flag
- Falar dos ficheiros de pegadas
- Falar da asignación automática de pegadas

Enlaces útiles

[Conceptos de circuitos impresos](#) [Instalación das bibliotecas](#) [Mais bibliotecas para KiCad](#)

Dende o seguinte enlace podese descargar unha biblioteca de compoñentes moi currada:

[git://smisioto.eu/KiCad_libs.git](https://smisioto.eu/KiCad_libs.git)

META

Este repositorio usa submódulos de git para a xestión das bibliotecas de compoñentes. Para clonar o repositorio tes que clonar o repositorio e despois actualizar os submódulos executando:

```
$ git clone https://github.com/brico-labs/BATS
$ git submodule update --init
```

Este documento está escrito en [Markdown-Pandoc](#). Pandoc e un sistema moi sinxelo de documentación que permite xerar múltiples formatos de saída.

As fontes do documento están no directorio **doc/src**. Os formatos de saída son este fichero **README.md** en formato Markdown-github e os documentos que podes atopar no directorio **doc/out** incluíndo un pdf.

Os documentos xeneranse automaticamente a partir do ficheiro fonte sen máis que executar:

```
$ cd doc
$ ./makeDoc
```

É importante cambiar ao directorio doc antes de executar o **makeDoc**.