

TP d'introduction à l'ingénierie

Présentation du projet

Dans ce projet, vous allez concevoir un dé électronique de A à Z, en suivant les étapes clés de la réalisation d'un objet électronique réel. À partir d'un microcontrôleur `ATTiny85`, vous devrez imaginer, tester et fabriquer un circuit capable de simuler le lancer d'un dé à 6 faces.

Vous découvrirez ainsi les principales étapes de la conception électronique : étude des fonctions à implémenter, réalisation du schéma électrique, choix des composants, tests sur breadboard, puis conception du circuit imprimé (PCB) à l'aide de l'outil EasyEDA. Enfin, vous réaliserez l'assemblage final en soudant les composants avec un fer à air chaud.

Ce projet vous permettra de découvrir concrètement le processus de conception et de fabrication d'un circuit électronique.

Table des matières

Présentation du projet.....	1
Étapes de ce projet.....	2
Étude préliminaire du projet.....	2
Présentation de l'AtTiny85.....	2
Étude de l'utilisation de notre AtTiny85.....	3
Conception du schéma électrique.....	3
Présentation de l'étape.....	3
Réalisation de l'étape : utilisation de EasyEDA.....	4
Test sur breadboard (sans microcontrôleur).....	5
Présentation de l'étape.....	5
Réalisation de l'étape.....	5
Programmation du microcontrôleur.....	6
Éléments de base du code Arduino.....	6
Dessin du PCB.....	8

Étapes de ce projet

En pratique, voici la logique suivie par les ingénieurs électroniques :

1. **Schéma électrique** (*logique fonctionnelle*): on commence par dessiner le schéma électrique du circuit (avec un logiciel comme EasyEDA). Ce schéma représente les connexions logiques entre les composants, pas leur disposition physique. Cette étape garantit que le fonctionnement est correct, indépendamment de l'implantation.
2. **Prototype sur breadboard** (*vérification rapide*) : faire un test sur breadboard pour valider certains points : comportement des LED, réponses aux signaux, gestion d'alimentation... Cela permet de vérifier que la logique est bonne sans fabriquer un circuit imprimé potentiellement erroné.
3. **Conception du PCB** : Une fois que le schéma est validé, nous passons au design du circuit imprimé (PCB). À cette étape, il faut tenir compte de l'organisation physique des composants, du routage des pistes, de l'alimentation, etc... Nous partons toujours d'un schéma électrique déjà fonctionnel (*testé et validé*).

Étude préliminaire du projet

Présentation de l'AtTiny85

Un **microcontrôleur** est un **circuit intégré** qui regroupe un **processeur**, de la **mémoire** et des **entrées/sorties** (I/O) sur une seule puce. Il est conçu pour exécuter un programme embarqué afin de contrôler un système électronique.

Dans un circuit imprimé, le microcontrôleur reçoit des signaux d'entrée (*capteurs, interrupteurs...*), effectue un traitement selon le code enregistré en mémoire, puis agit sur des sorties (*LED, moteurs, afficheurs...*).

Il constitue le cœur du projet électronique : à la différence d'un circuit statique composé uniquement de composants passifs (*résistances, condensateurs, moteurs...*), le microcontrôleur donne « une intelligence » au circuit en lui permettant de réagir, de calculer et de s'adapter à son environnement.

Dans ce projet le microcontrôleur choisi est l'ATtiny85 ([Datasheet](#)). Il détectera les mouvements via un *tilt switch*, générera un nombre aléatoire, puis allumera les LED correspondant à la face du dé. Les caractéristiques principales de cette puce sont :

- 8 broches (6 broches d'E/S utilisables)
- Mémoire flash de 2 Ko
- Horloge interne à 8 MHz
- Fonctionne sous 2,7 à 5,5 V

Étude de l'utilisation de notre AtTiny85

1. L'AtTiny85 possède 8 broches dont une d'alimentation (+) et une de terre (-).
 - a) Observez la disposition classique des points sur un dé. Combien de LED sont nécessaires pour pouvoir représenter toutes les faces d'un dé à 6 côtés ?
 - b) Avons-nous suffisamment de broches pour associer chaque LED à une broche ?
 - c) Est-il raisonnablement possible d'associer plusieurs LED à une même broche ?
 - d) Est-il raisonnablement possible d'associer une LED et notre *tilt switch* à une même broche ?
2. Identifiez les LED ou couple de LED qui s'allument pour plusieurs valeurs du dé. Par exemple, les LED `milieu-gauche` et `milieu-droite` sont allumées seulement pour le nombre 6. À vous de repérer les LED qui sont partagées entre plusieurs chiffres.
3. Associez chaque LED ou couple de LED à une broche du microcontrôleur. Puis, pour chaque valeur possible du dé, indiquez quelles broches devront être activées pour allumer les LED correspondantes.

Exemple :

- La LED centrale est reliée à la broche `PB0`, et doit s'allumer pour les valeurs 1, 3, 5.
 - Pour le chiffre 3, on activera les broches `PB0` et `PB1`, correspondant aux LED nécessaires.
4. Associez le *tilt switch* à une broche restante.

Conception du schéma électrique

Présentation de l'étape

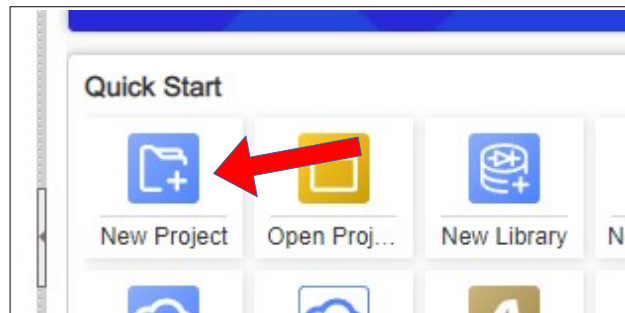
Avant tout test physique, nous devons concevoir le circuit théorique en identifiant les composants nécessaires et leurs connexions. Cette étape consiste à construire le `schéma électrique` du dé électronique. Il représente les liaisons logiques entre les composants.

5. Listez les composants nécessaires : LED, AtTiny85, et les autres.
6. Identifiez les groupes de LED qui seront allumés ensemble selon les valeurs du dé (*par exemple : le nombre 6 allumerait les groupes 1, 2 et 4*).
7. Répartissez dans un tableau récapitulatif chaque groupe de LED sur une broche de l'AtTiny85. Une broche commandera un groupe.

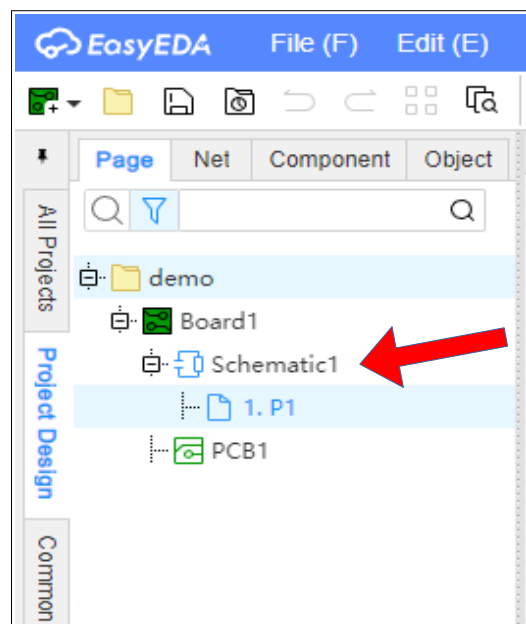
Réalisation de l'étape : utilisation de EasyEDA

Le schéma sera réalisé sur EasyEDA. Cette représentation ne dépend pas encore de la disposition réelle des composants, elle sert à valider le fonctionnement du circuit.

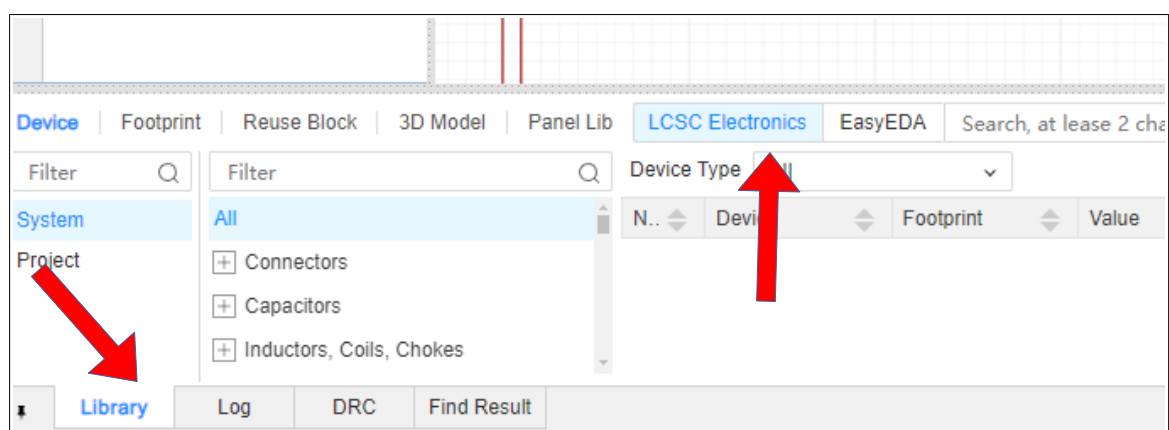
- Créer un projet : depuis l'accueil, cliquez sur **New Project** et nommez le **projet_dé**. Ouvrez le projet.



- Étendez la liste des schémas en ouvrant **Schematic1** dans le volet latéral gauche et ouvrez le schéma **P1**.

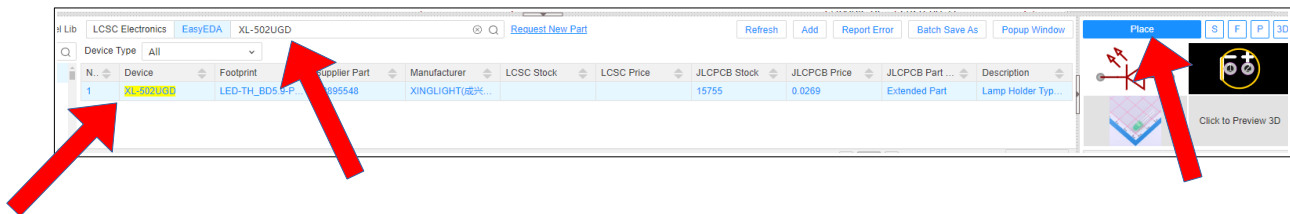


- Ajouter des composants : cliquez sur en bas de l'écran sur **Libraries**, puis **LCSC Electronics**.



11. Grâce à la barre de recherche, trouver les composants suivants dans la librairie et placez-les sur le schéma autant de fois que nécessaire :

- a) ATTINY85-20PU (microcontrôleur) (1)
- b) BS-2-1 (support de pile) (1)
- c) XL-502UGD (LED) (7)
- d) SW-200M-20T10G2 (tilt switch) (1)



12. Positionnez-les de manière claire. Les broches doivent être visibles et accessibles.

13. Précédemment, vous avez identifié les groupes de LED qui seront alimentés ensemble (*car allumés pour les mêmes valeurs d'un dé*). Choisissez 4 lignes de la breadboard pour représenter ces 4 groupes. Branchez les anodes (+) des LED à l'aide de l'outil Wire (raccourci : **W**) aux broches correspondantes de l'ATtiny85, les cathodes (-) à la masse (GND).

14. Connectez le *tilt switch* à une broche d'entrée, et ajoutez la pile bouton pour l'alimentation du circuit. Une fois ce schéma terminé, nous pourrons passer à l'étape de test sur breadboard.

Test sur breadboard (sans microcontrôleur)

Présentation de l'étape

Avant de souder les composants sur le circuit imprimé, vous testerez le fonctionnement de votre schéma sur une **breadboard**. Une breadboard est une plaque de prototypage qui permet de connecter des composants sans soudure, en utilisant des rangées de contacts internes.

Dans cette phase, vous ne brancherez pas encore l'ATtiny85. À la place, vous simulerez son comportement en plaçant manuellement un fil relié au +3V aux points de branchement prévus des broches du microcontrôleur (*comme si ce dernier émettait un signal sur cette broche*). Cela vous permettra de vérifier que chaque point alimente bien les bonnes LED. C'est une étape essentielle pour valider votre logique de câblage, avant de passer à la programmation de la puce.

Réalisation de l'étape

Une breadboard est organisée en lignes et colonnes de trous reliés entre eux électriquement :

- Les lignes verticales (*souvent marquées en rouge et bleu*) servent à distribuer l'alimentation (+ et -). Chaque trou de ces deux lignes est relié aux autres.

- Les colonnes verticales (au centre) sont reliées par groupe de 5 trous. Chacun de ces 5 trous est reliés au 4 autres, et isolés de tous les autres trous de la breadboard.
15. Placez les 7 LED sur la breadboard en reproduisant la forme d'un dé (*matrice 3x3*). Reliez les anodes (+) des LED aux broches correspondantes de l'ATtiny85 et les cathodes (-) à la masse (*GND*).
 16. Testez chaque groupe : alimentez chaque ligne manuellement en y connectant un fil relié au +3V. Vérifiez que seules les LED attendues s'allument pour chaque ligne.

Programmation du microcontrôleur

Le code Arduino programmera le comportement de l'ATtiny85. Il devra :

1. Détecter un mouvement grâce au `tilt switch` (*fermeture du contact*).
2. Générer un nombre aléatoire entre 1 et 6.
3. Allumer les LED correspondantes à la valeur tirée, en activant les bonnes broches de sortie.
4. Maintenir l'affichage quelques secondes, puis éteindre les LED pour attendre le prochain mouvement.

Ce programme sera écrit dans l'Arduino IDE, puis téléversé sur l'ATtiny85 via un programmeur AVR.

Éléments de base du code Arduino

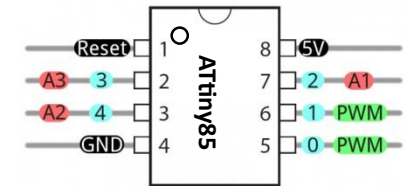
- `setup()` : Fonction exécutée une seule fois au démarrage. On y initialise les broches, par exemple en les configurant en entrée ou sortie.
- `loop()` : Fonction principale exécutée en boucle en continu. Elle contient le comportement du programme.
- `pinMode(pin, mode)` : Permet de définir une broche en entrée (`INPUT`) ou en sortie (`OUTPUT`). Exemple : `pinMode(0, OUTPUT)`; Cela se fait généralement dans la fonction `setup()`.
- `digitalWrite(pin, value)` : Permet de mettre une broche à l'état haut (`HIGH`) ou bas (`LOW`), donc d'allumer ou d'éteindre une LED. Exemple : `digitalWrite(0, HIGH)`;

Ces fonctions sont indispensables pour faire interagir l'ATtiny85 avec les composants du circuit. Voici ci-dessous un résumé des fonctions Arduino associées. Observez le schéma en haut à droite correspondant à la puce. Les pins numérotées de 0 à 5 correspondent aux sorties utilisables de l'AtTiny85 via les fonctions `pinMode` et `digitalWrite`.

17. Écrivez le programme du dé. Vous pouvez commencer par afficher les 6 nombres en boucles sans tenir compte du tilt switch pour tester votre schéma et la puce.



Sparkfun Electronics ATtiny85 Arduino Quick Reference Sheet



Structure

/* Each Arduino sketch must contain the following two functions. */

```
void setup()
{ // this code runs once at the
  // beginning of the code execution.
}
```

```
void loop()
{ // this code runs repeatedly over
  // and over as long as the board is
  // powered.
}
```

Comments

```
// this is a single line comment
/* this is
   a multiline
   comment */
```

Setup

```
pinMode(pinNum, INPUT/OUTPUT/INPUT_PULLUP);
/* Sets the mode of the digital I/O pin.
All pins are general I/O on the board. You
must define what the pin will be used for at
the beginning of your code in setup() */
```

Control Structures

```
if(condition)
{ // if condition is true, do something here
}
else
{ // otherwise, do this
}
```

```
for(init; condition; increment)
{
  // do this, increment, and
  // repeat while condition is true.
}
```

Digital I/O

```
digitalWrite(pin, val);
```

/* val = HIGH or LOW write a HIGH or a LOW value to a digital pin. */

```
buttonVal = digitalRead(pin);
```

/* Reads the value from a specified digital pin, either HIGH or LOW. */

Analog I/O

```
analogWrite(pin, val);
```

/* Writes an analog voltage (using PWM) to a pin. val = integer value from 0 to 255 */

```
sensorVal = analogRead(pin);
```

/* Reads the voltage from the specified analog pin. 0V returns 0; Vcc returns 1023*/

Time

```
delay(time_ms);
```

/* Pauses the program for the amount of time (in milliseconds). */

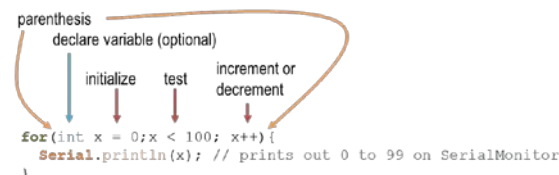
```
millis();
```

/* Returns the number of milliseconds since the board began running the current program. max: 4,294,967,295 */

Serial Communication

A separate USB to serial adapter like FTDI is needed for Serial communication with the ATtiny. And. the ATtiny must be flashed to run at 8 MHz instead of 1 MHz.

The ATtiny does not support Serial natively. You need to use the **SoftwareSerial** library to enable this function.



ATtiny85 Pins

Pins 0 - 4 : general purpose I/O pins (GPIO).

Both **digitalWrite()** and **digitalRead()** can be used with any of these pins.

Pins 0 & 1 : setup for PWM output using **analogWrite()**.

Pins A1, A2, A3 : setup for reading sensor input with **analogRead()**.

Data Types

void // nothing is returned

boolean // 0, 1, false, true

char // 8 bits: -128 to 127

byte // 8 bits: 0 to 255

int // 16 bits: -32,768 to 32,767

unsigned int // 16 bits (unsigned)

long /* 32 bits: -2,147,483,648 to 2,147,483,647 */

unsigned long // 32 bits (unsigned)

float // 32 bits, signed decimal

```
#include <SoftwareSerial.h> // include library
```

```
SoftwareSerial tinySerial(3, 4);
```

/* Put above setup() and loop() - declares tinySerial using 3 & 4 for Transmit (tx) and Receive (rx) */

```
tinySerial.begin(9600); /* begin Serial at 9600 baud. Put this line in setup() */
```

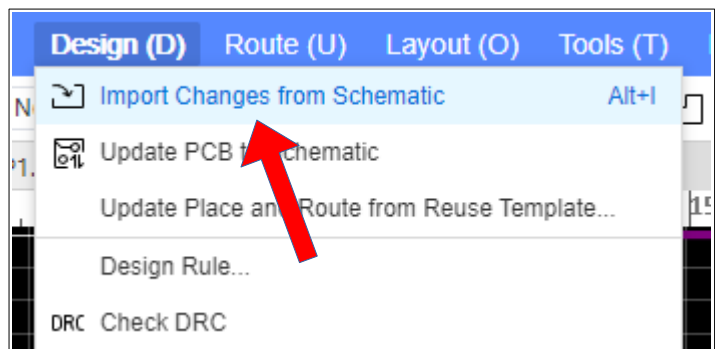
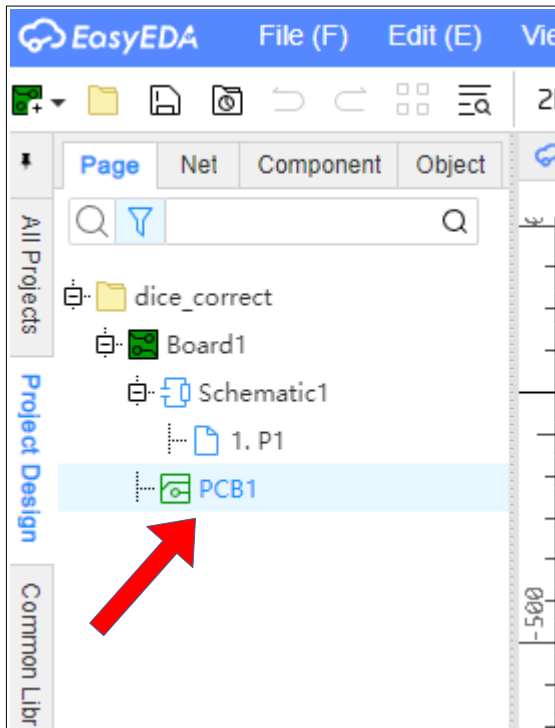
```
tinySerial.print(""); /* sends data on TX line - to your receiving computer. */
```

```
tinySerial.println(""); /* sends data to Serial Monitor with CRLF. */
```

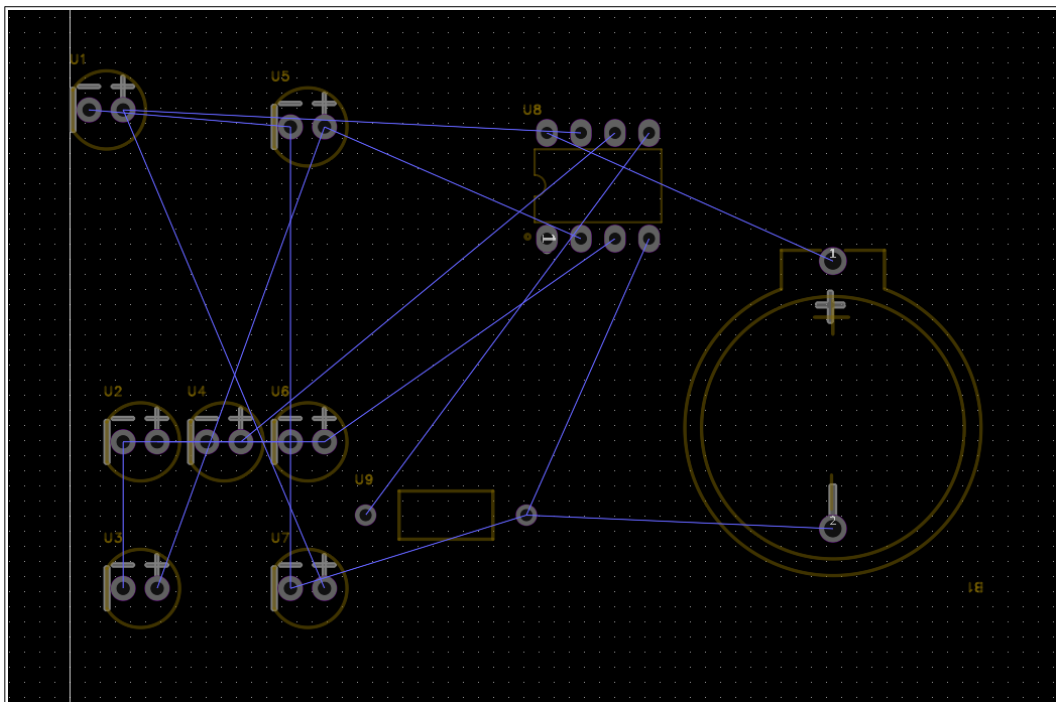
```
inChar = tinySerial.read();
```

Dessin du PCB

Maintenant que le code fonctionne et que le schéma électrique a été validé, nous allons passer à la conception du circuit imprimé. Cette étape consiste à placer physiquement les composants sur une carte et à tracer les pistes de connexion entre eux. Le dessin du PCB se fera dans EasyEDA, à partir du schéma déjà réalisé.

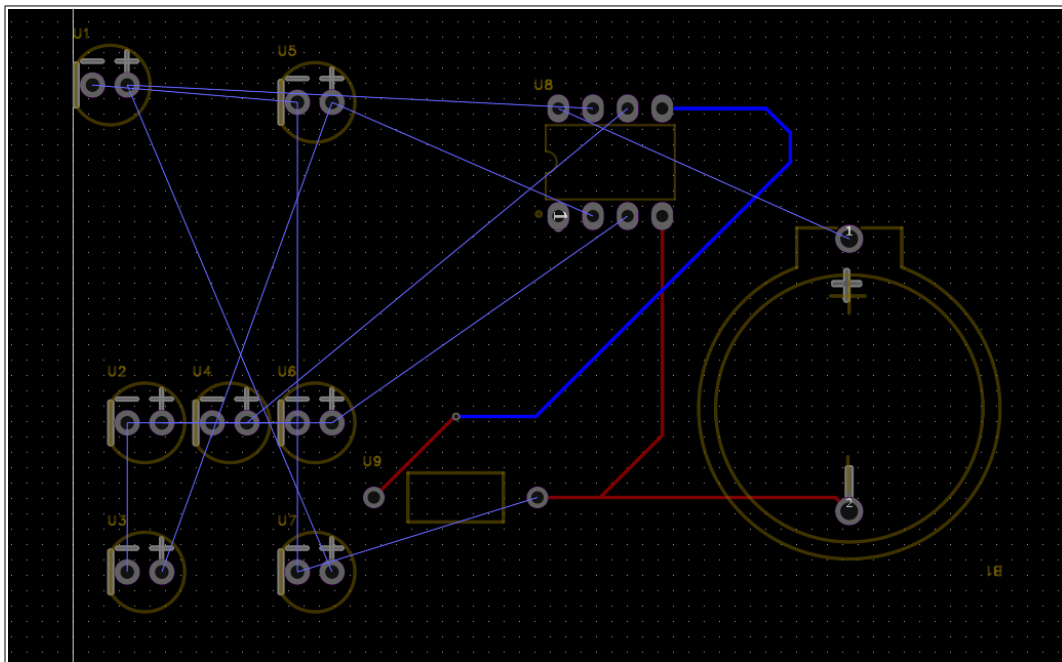


En suivant ces deux opérations, cela vous ouvrira une zone de travail comme ci-dessous :



Dans la vue PCB d'EasyEDA, nous passons du schéma logique au circuit physique. Voici les éléments essentiels :

- *Traits bleus* : ce sont les **ratlines**, des liaisons automatiques générées à partir du schéma électrique. Elles indiquent quels composants doivent être connectés. Ce ne sont pas encore des pistes de cuivre.
- *Création des connexions* : pour transformer les traits bleus en vraies pistes, il faut utiliser l'outil **Route** (ou **Ctrl+W**). On clique sur une broche, puis on trace une piste manuelle en suivant le chemin souhaité jusqu'à l'autre composant. Il est possible de double-cliquer pendant le dessin du trait pour passer d'une surface à l'autre du PCB (création d'un **pipe**).
- *Placement physique des composants* : Il faut placer chaque composant de manière logique et compacte, sans que les pistes ne se croisent inutilement.
- *Contours du PCB* : ils sont visibles en violet. C'est le bord réel de la carte. Il peut être dessiné ou modifié avec l'outil **Board Outline**.



Voici un exemple avec quelques pistes déjà tracées. Les traits rouges correspondent aux connexions réalisées sur une face du PCB, tandis que les traits bleus sont sur la face opposée. Cette méthode permet de faire passer une piste sous une autre sans croisement direct, car deux pistes ne peuvent pas se croiser physiquement sur une même couche.

18. Dessinez le PCB.

Une fois les cartes et les composants reçus, à vos fers à souder !

