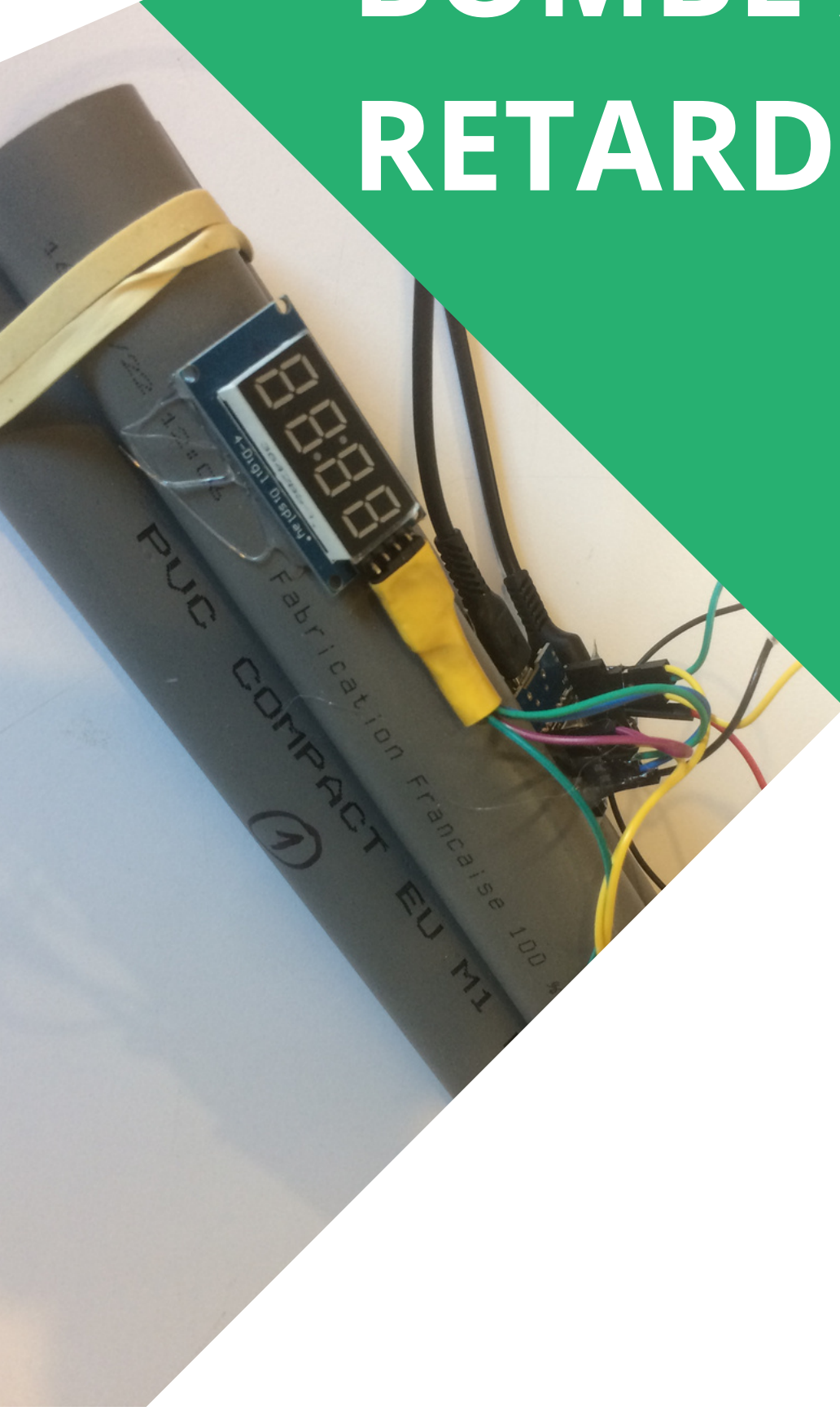




# BOMBE À RETARDEMENT



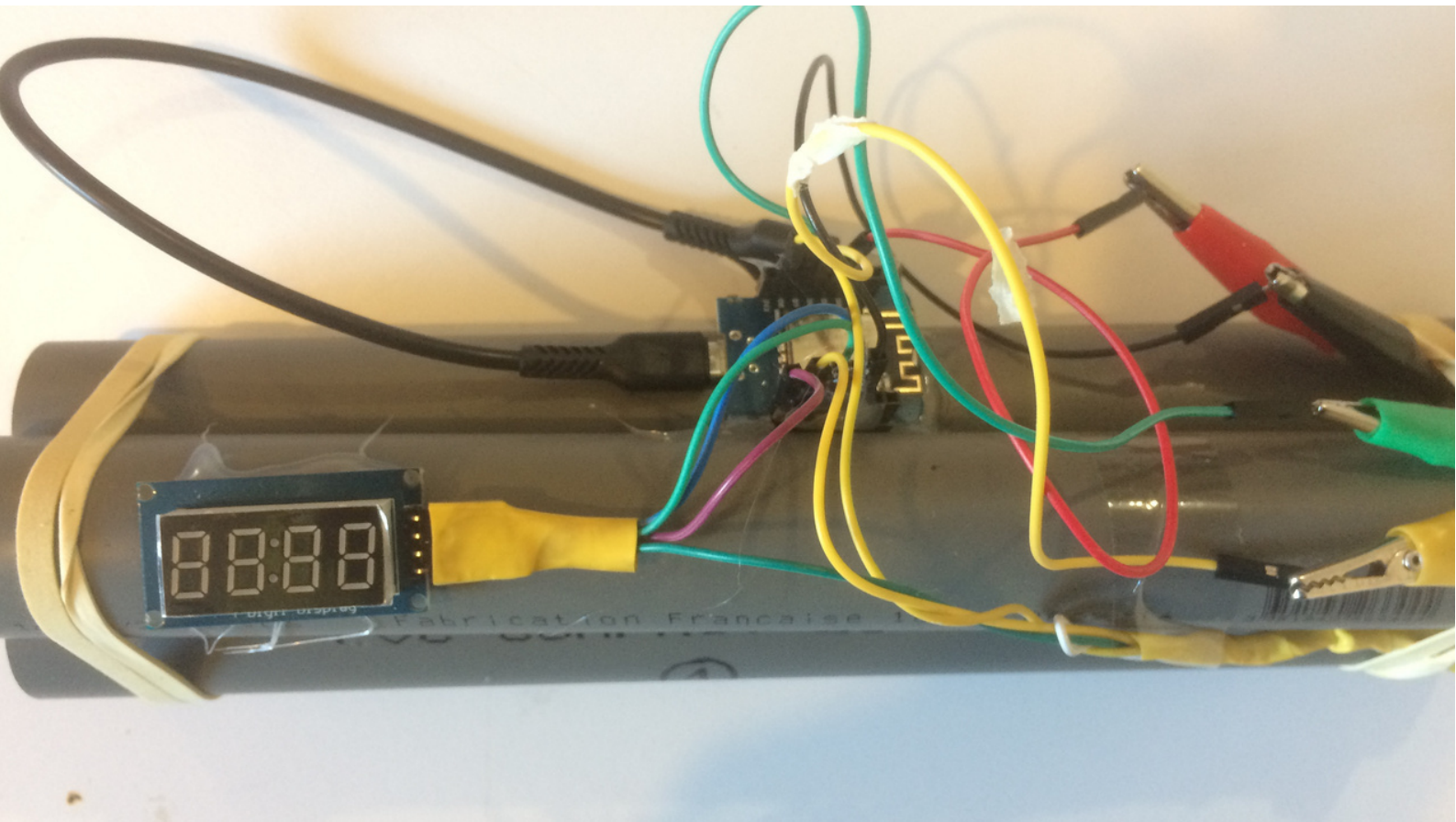
# 01 BOMBE À RETARDEMENT MATÉRIEL NÉCESSAIRE



- Wemos D1 mini
- Pincès Crocodile
- Écran LCD 4 chiffres
- Des câbles de liaison
- Un buzzer
- Un tube pvc que vous pouvez couper en trois parties égales
- Un pistolet à colle

Le fichier Arduino est disponible ici : [Télécharger](#)

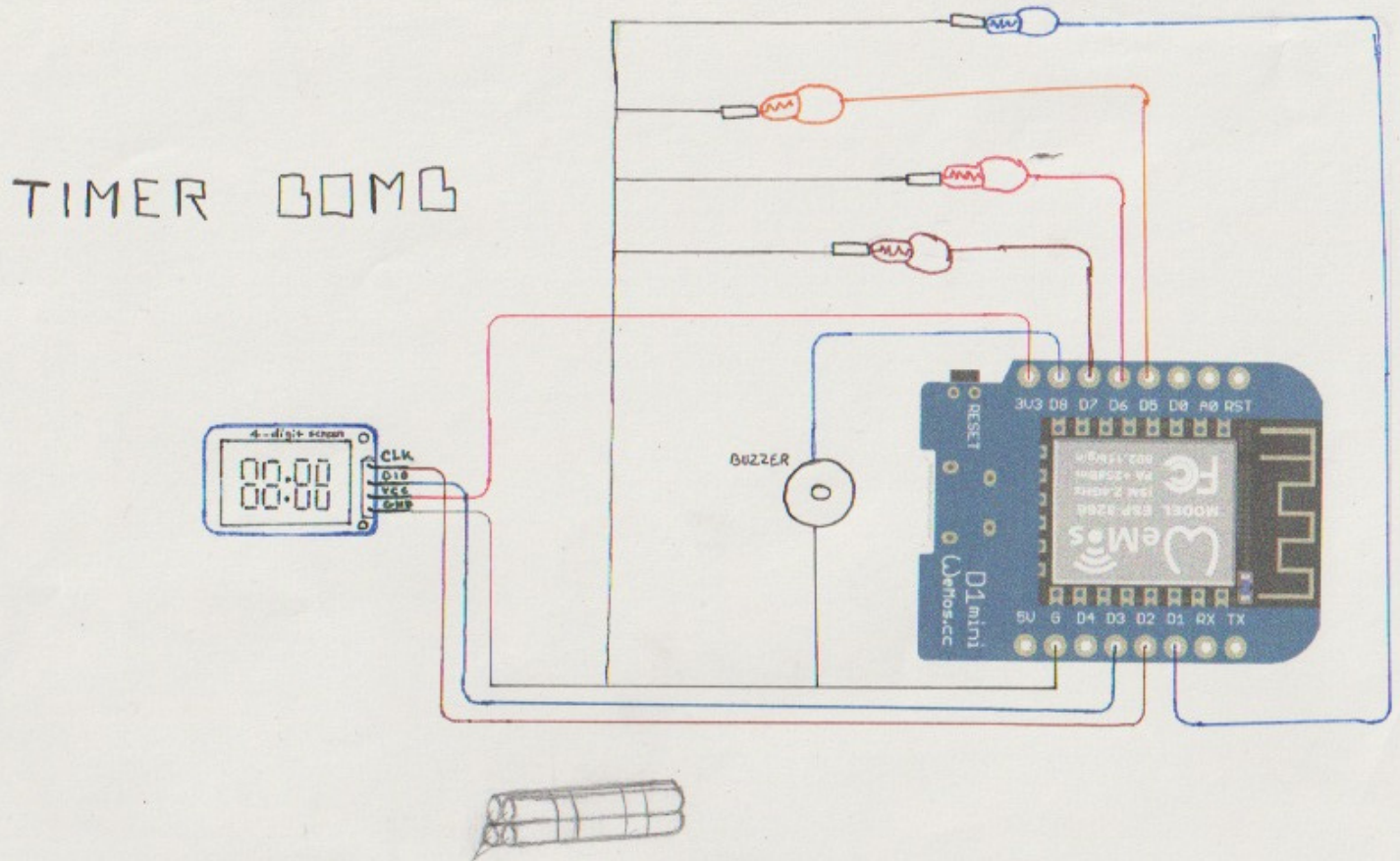
Si vous avez tout rassemblé, vous pouvez passer à l'étape 2.



# 02 BOMBE À RETARDEMENT CÂBLAGE



Suivez ce schéma pour réaliser votre prototype sur une breadboard d'abord.





# 03 BOMBE À RETARDEMENT

## TÉLÉCHARGER LE CODE



Chargez le code téléchargé précédemment sur votre carte Arduino et essayez votre prototype. Le code est légèrement compliqué mais vous devez comprendre qu'une partie est pour la musique que le buzzer va jouer en cas d'échec et l'autre est pour le processus de désamorçage.

Dans ce code, vous n'avez que quelques paramètres que vous pouvez modifier.

- Ligne 32 : C'est le temps de pénalité lorsqu'un mauvais câble est déconnecté.
- Ligne 43 : L'ordre de coupe des câbles (0 signifiant ne pas couper et 1 à 3 l'ordre).
- Ligne 53 : La durée du minuteur en minutes

```
cutthewires pitches.h
26
27 // CONSTANTS
28 const byte numWires = 4;
29 // Note that Wemos D1 mini has PULLDOWN resistor on D8, but no PULLUP
30 const int wirePins[numWires] = {D7, D6, D5, D1};
31 // Amount of time (in ms) to be deducted when an incorrect wire is cut
32 const unsigned long timePenalty = 600000;
33
34
35
36 // GLOBALS
37 // Create a display with the specified CLK/DIO lines
38 TM1637Display display(D2, D3);
39 int lastState[numWires];
40
41 // What is the order in which wires need to be cut
42 // 0 indicates the wire should not be cut!
43 int wiresToCut[numWires] = {0, 1, 3, 2}; //0, 1, 2, 3, so you need to cut 1 and then 3 and not 0 and 2
44 byte wiresCutCounter = 1;
45 // Keep track of the current state of the device
46 enum State {Inactive, Active, Defused, Exploded};
47 State state = State::Inactive;
48 // This is the timestamp at which the bomb will detonate
49 // It is calculated by adding on the specified number of minutes in the game time
50 // to the value of millis() when the code initialised.
51 unsigned long detonationTime;
52 // The game length (in minutes)
53 int gameDuration = 45;
54
55 void Activate(){
56     state = State::Active;
57     // Set the detonation time to the appropriate time in the future
58     detonationTime = millis() + (unsigned long)gameDuration*60*1000;
59     Serial.println(F("Bomb activated!"));
60 }
61
62 void Deactivate() {
63     Serial.println(F("Bomb defused"));
64     state = State::Inactive;
65 }
66
67 void Detonate() {
68     state = State::Exploded;
69     Serial.println("BOOM!");
70 }
```

# 04 BOMBE À RETARDEMENT FABRICATION DE LA BOMBE



Vous êtes prêts à assembler ta bombe et à terminer le prototype.

Coupez votre tube PVC en trois et collez ces tubes-là ensemble ou serrez-les avec des élastiques.

Lorsque cela semble stable, vous devez d'abord coller la carte Arduino quelque part au-dessus des tubes. Faites de même pour l'écran.

Connectez toutes les pinces crocodiles et attachez-les également pour les rendre stables.

Si tout est connecté, vous êtes prêt à brancher votre bombe et à voir le compte à rebours commencer !

