CHABANIER Aurélia

DALMASSO Dylan

Groupe 4

GameTable

Bibliographie

**Sommaire :**

# Description de notre projet

# Etat de l’art

# La Table de Jeux

## Réalisation de la table

## Réalisation du jeu

### La carte Arduino adéquate

### Comment reconnaître où le joueur appuie ?

### La communication avec le(s) joueur(s)

#### Les modules Bluetooth

#### L’application mobile

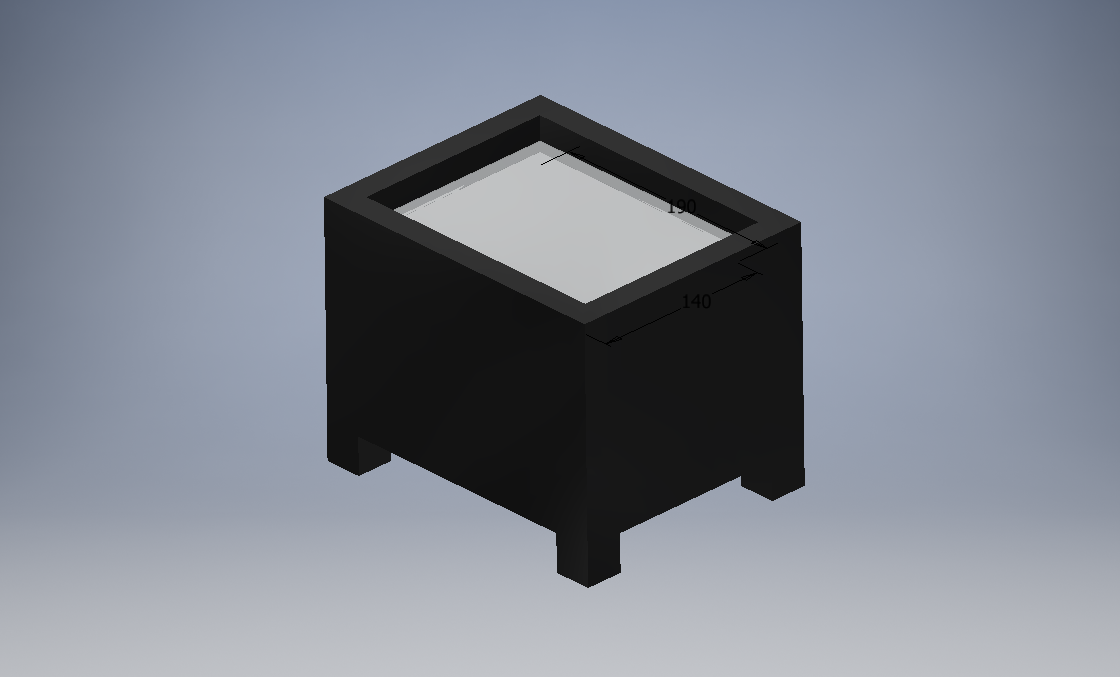
### Quelle LED utiliser ?

**I. Description de notre projet :**

Notre projet est une table de jeux qui sera créée de toutes pièces.

Les jeux seront affichés par un réseau de LED qui s’allumera en fonction du jeux choisi (ces derniers seront des jeux simples et connus, comme le *Morpion* et le *Puissance 4*). Sur ces LED, un écran tactile sera clipsé (celui-ci nous est fourni par M. MASSON comme beaucoup de composants). Cet écran nous permettra de savoir où le joueur appuiera.

De plus, le dispositif de LED et d’écran sont reliés à une carte Arduino équipée d’un dispositif Bluetooth communiquant avec un téléphone. Cette liaison Bluetooth permettra, à travers le téléphone, de choisir le jeu auquel on souhaite jouer, le nom des joueurs, la couleur du joueur si le jeu s’y prête, ainsi que de recevoir les scores. La communication Bluetooth entre la carte Arduino et le téléphone sera facilité par une application que l’on aura créée avec *AppInventor* du MIT.

****

Maquette de notre projet

**II. Etat de l’art :**

En faisant nos recherches pour savoir par où commencer dans notre projet, nous avons trouvé sur Internet quelques projets assez similaires au notre (voire même très similaire au notre), certains beaucoup plus aboutis que d’autres sans que nous soyons pour autant sûrs que tout soit fonctionnel (ce dont nous doutons).

Les deux tables que nous avons trouvées sont différentes de la nôtre dans la forme mais reste assez similaire dans le fond (c’est-à-dire faire une table de jeux piloté par un Arduino). En effet, celles-ci sont des modèles préfabriqués sur lesquels les créateurs viennent greffer un système de LED pour faire un *Tétris* commandé par une carte Arduino et un téléphone portable relié par Bluetooth.

Rentrons plus en détail dans chacun de ces projets.

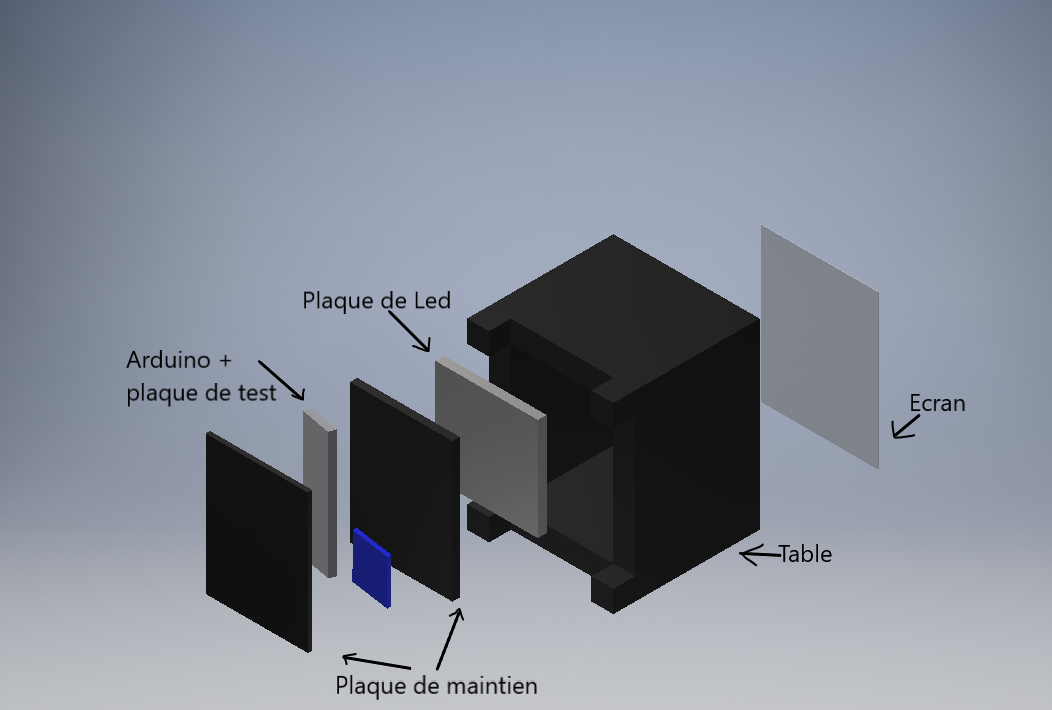
Le premier est un post disponible sur le site Internet create.arduino.cc. En plus de servir de table pour jouer à *Tétris* via son téléphone, il permet aussi de créer une sorte d’ambiance avec les LED et possède une reconnaissance sonore. Pour faire son jeu de *Tétris* le créateur passe par une application Java pour prévisualiser l’interface du menu. Nous ne passerons pas par ce moyen dans notre cas et nous créerons nous-même les jeux dans l’IDE d’Arduino. L’état d’avancement de ce projet semble très correct, même si toutefois, il reste des bugs à corriger et des fonctionnalités manquantes. De plus, les LED utilisées ne sont pas les mêmes que les nôtres.

Le deuxième projet est un article de instructables.com. Sur ce dernier, nous pouvons voir une table servant uniquement de support à un *Tétris*. Ce dernier fonctionne grâce à un Arduino et un module Bluetooth. Ici aussi, les LED ne sont pas les mêmes que les nôtres. Cependant, à la vue des informations donnés par le constructeur, le projet semble fini et fonctionnel.

Nous sommes donc partis de ces deux projets et nous sommes dit : « Que pouvons-nous ajouter de plus à ce type de projet ? Comment réaliser quelque chose de la sorte ? Comment nous démarquer ? ». Nous avons donc choisi de reprendre le concept de table de jeux, en y incorporant des jeux plus simples, multijoueur, et en gardant le contact avec les utilisateurs au travers d’une liaison Bluetooth avec un téléphone portable. Pour le reste des réponses, elle se trouveront au cours de cette bibliographie.

**III. La table de jeux :**

A) Réalisation de la table :



Vue éclatée de notre maquette

Nous souhaitons réaliser notre table en bois peint.

Les plaques de maintien seront fines, et tenues à la table grâce à des fixations en métal qu’on pourra dévisser facilement en cas de besoin.

De plus, afin de faire passer les câbles entre l’Arduino et les composants du dessus, il y aura un trou dans les plaques (non visibles sur la représentation 3D).

Ces trous permettront aussi d’alimenter l’Arduino par-dessous.

B) Réalisation du jeu :

1) La carte Arduino adéquate

Nous avons à notre disposition trois grands modèles de cartes Arduino qui se détachent.

Comparons les dans un tableau :

|  |  |  |  |
| --- | --- | --- | --- |
| Modèle de cartes\Caractéristiques | Taille de la carte | Mémoire flash | Nombre de ports |
| Nano | 43mm x 18mm | 32 kB | 12 Entrées-Sorties Numériques  6 d’entre elles sont des PWM  8 Analogiques |
| Uno | 68.6mm x 53.3mm | 32 kB | 14 Entrées-Sorties Numériques  6 d’entre elles sont des PWM  6 Analogiques |
| Méga | 101.6mm x 53.3mm | 256 kB | 54 Entrées-Sorties Numériques  15 d’entre elles sont des PWM  16 Analogiques |

Dans notre projet, nous ne nous occupons pas de la taille de la carte choisie, en effet, notre future table de jeux dépassera les 20cm de longueur donc ce facteur n’a pas d’importance vis-à-vis du choix du modèle. Cependant, les deux autres caractéristiques (c’est-à-dire mémoire vive et Entrées-Sorties disponibles) sont nettement plus importantes.

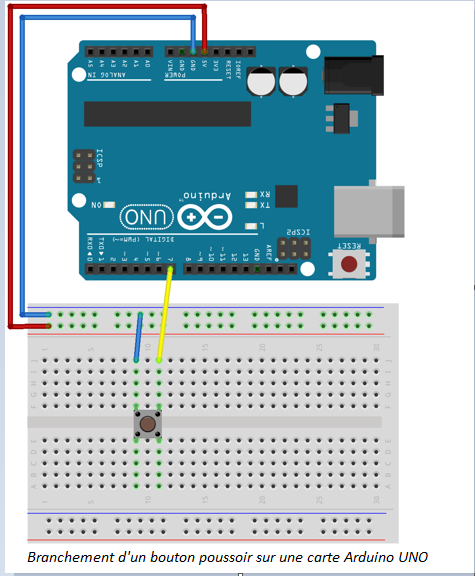
Nous bannissons directement la carte Nano car celle-ci possède trop peu de mémoire vive.

Le choix s’effectuera donc entre le modèle Uno et le modèle Méga. Après consultations des forums sur Internet, nous avons décidés de partir sur le modèle Méga. Celui-ci, avec ses 256kB de mémoire vive, permettra de garder toutes les fonctions d’allumage de LED qu’on détaillera plus tard, ainsi que le reste de notre programme (c’est-à-dire les jeux).

Une fois ce choix fait, il restait pour nous la question de l’alimentation à traiter. Nous avons trouvé un adaptateur se branchant à une prise secteur murale fournissant du 5V à la carte Méga. La plupart voire la totalité de nos composants nécessitant du 5V, cette option semblait correspondre parfaitement à nos attentes. De plus, M. MASSON ayant en stock de tel adaptateurs nous a conforté dans cette idée. Nous sommes donc restés là-dessus !

2) Comment reconnaître où le joueur appuie ? :

Aux prémices de notre projet nous avions pensés à reconnaître où le joueur appuie par un système de bouton poussoir vu en cours.

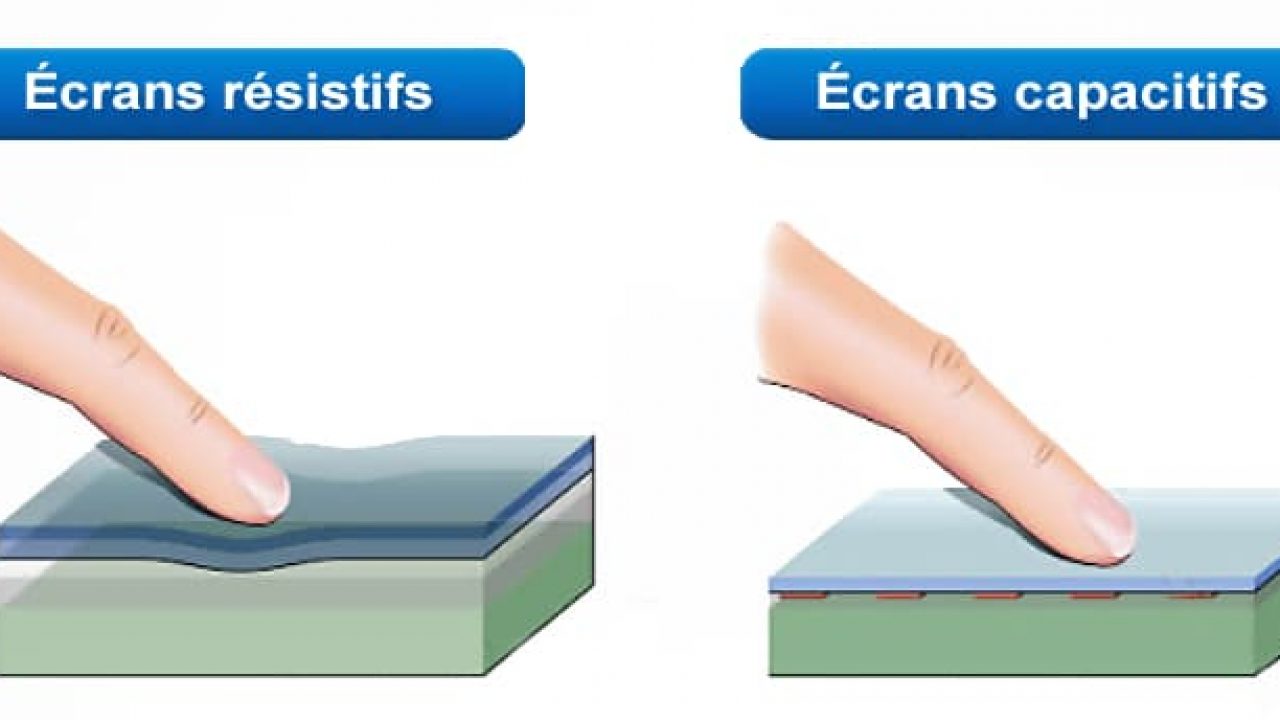


Cependant, après avoir consulté notre professeur référent, nous nous sommes aperçus que cela serait vraiment trop compliqué à mettre en place. Outre le fait qu’il faudrait des dizaines de boutons poussoirs, c’est surtout la partie câblage qui aurait eu un rendu vraiment brouillon. De plus, il aurait fallu souder tous les boutons poussoir sur la table, autant le dire d’emblée cela aurait été un vrai casse-tête.

Néanmoins, après nous avoir dit cela, M. MASSON nous a orienté vers un écran tactile déjà utilisé dans les années précédentes qu’il avait en stock. Avant de parler de son fonctionnement parlons des deux types d’écrans tactiles utilisés dans l’industrie.

Premièrement les écrans résistifs (c’est un écran de ce type que nous aurons) sont des écrans possédant trois sous-couches. La première est une couche de polyester souple sur laquelle on vient appuyer. L’appui sur cette couche va pousser des matériaux conducteurs sur une plaque en verre (deuxième et troisième couche). Ce rapprochement entre les deux couches va créer un contact électrique et en mesurant la tension au point d’impact, on est capable de trouver l’endroit où la personne a appuyé. Cependant ce genre d’écrans peuvent perdre en précision à force de les utiliser.

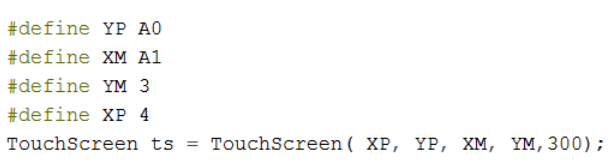
Deuxièmement les écrans capacitifs fonctionnent non pas par appui mais par effleurement. Cet effleurement se base sur la capacité conductrice du corps humain, et par cette conductivité de notre corps nous allons venir altérer le champ électrique de la plaque. L’appareil se charge alors de calculer la position.



Dans notre projet nous aurons donc un écran résistif à quatre fils. Pour le brancher nous devront brancher deux entrées (les entrées + en X et Y) sur des ports analogiques de la carte Arduino. Les deux autres entrées pourront être branchées à notre guise sur un port analogique ou numérique (et donc sur du 5V/GND normalement).

Pour détecter l’endroit où le joueur appuie nous avons plusieurs solutions, deux pour être précis. La première est un diviseur de tension sur les points d’entrée. Pour faire ce diviseur pour par exemple lire la coordonnée en Y il faut placer Y1 à 5V et Y2 en GND et venir lire la tension sur X1. En faisant la même chose pour lire X et donc en remplaçant Y par X et vice-versa on peut écrire une fonction qui nous renvoie exactement les bonnes valeurs. Cependant, nous ne savons pas si l’exécution de ce programme serait assez rapide pour satisfaire nos besoins.

La deuxième méthode est l’importation de la bibliothèque <TouchScreen.h>. Cette bibliothèque utilise un objet TouchScreen (donc notre écran) au travers de cette commande :



Ceci vient d’un projet d’une année précédente. Le 300 représente la valeur rx c’est-à-dire la résistance dans l’écran.

Grâce à cette bibliothèque, nous avons accès aux fonctions suivantes :

- isTouching(); qui renvoie true ou false ce qui permet de vérifier que l’on appuie bien sur l’écran ;

- readTouchY() ; qui renvoie la position en Y sous forme d’entier de là où l’utilisateur appuie ;

- readTouchX(); qui renvoie la position en X de là où l’utilisateur appuie ;

3) La communication avec le(s) joueur(s) :

a) Les modules Bluetooth :

Deux modules Bluetooth sont à notre disposition : le module HC-05 et le HC-06.

Le module HC-05 est à la fois maître et esclave : il peut à la fois initier et recevoir la communication alors que le module HC-06 est uniquement esclave : il ne peut que recevoir la communication Bluetooth.

Le module Bluetooth HC-05 possède 6 broches :

- VCC

- GND : connecté à la masse

- RX (réception) : connectée à la broche de transmission de l'Arduino

- TX (transmission) : connectée à la broche de réception de l'Arduino

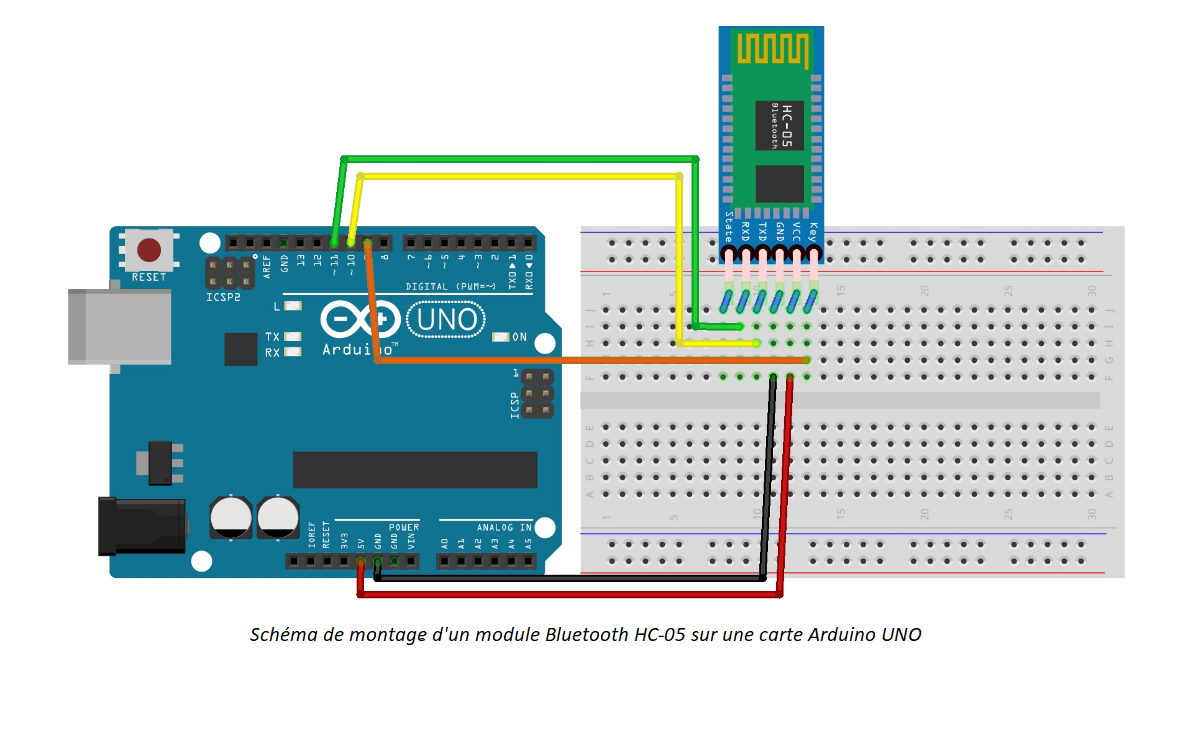
- State : retourne 1 quand le module est connecté

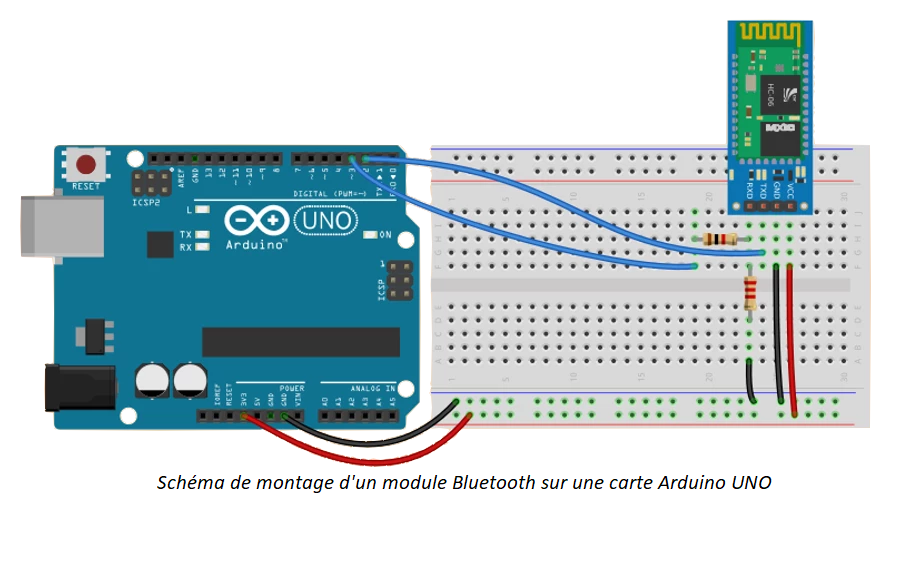
- Key/EN : alimenté en mode configuration, non alimenté en mode communication

Le module HC-06 quant à lui n’en a que 4 : la VCC, la GND, la RX et la TX.

La broche RX du module HC-06 doit être alimentée au maximum par une tension de 3,3V, il faut donc créer un diviseur de tension pour éviter d’abimer la broche.

Voici, ci-dessous, les schémas de câblages des deux modèles Bluetooth venant d’être présentés.





Ces deux modules sont donc semblables, seulement le module HC-06 nécessite de créer un diviseur de tension, et est branché sur du 3V3. Cependant, le reste de nos composants ont besoin d’être branché sur du 5V. Le module HC-05 est donc idéal pour notre projet, nous avons donc choisi ce module.

Pour gérer la communication avec les modules, on utilise la librairie SoftwareSerial, avec notamment les fonctions suivantes :

- SoftwareSerial MODULE(broche\_RX, broche\_TX) : où MODULE est le nom que l’on donne à la série de communication (dans les autres fonctions on l’appellera Bluetooth, mais on aurait pu l’appeler autrement).

- Bluetooth.begin(X) : où X représente la vitesse de transmission.

- Bluetooth.available() : renvoi le nombre de bit disponibles dans le buffer.

- Bluetooth.read() : retourne un caractère lu sur l’entrée RX.

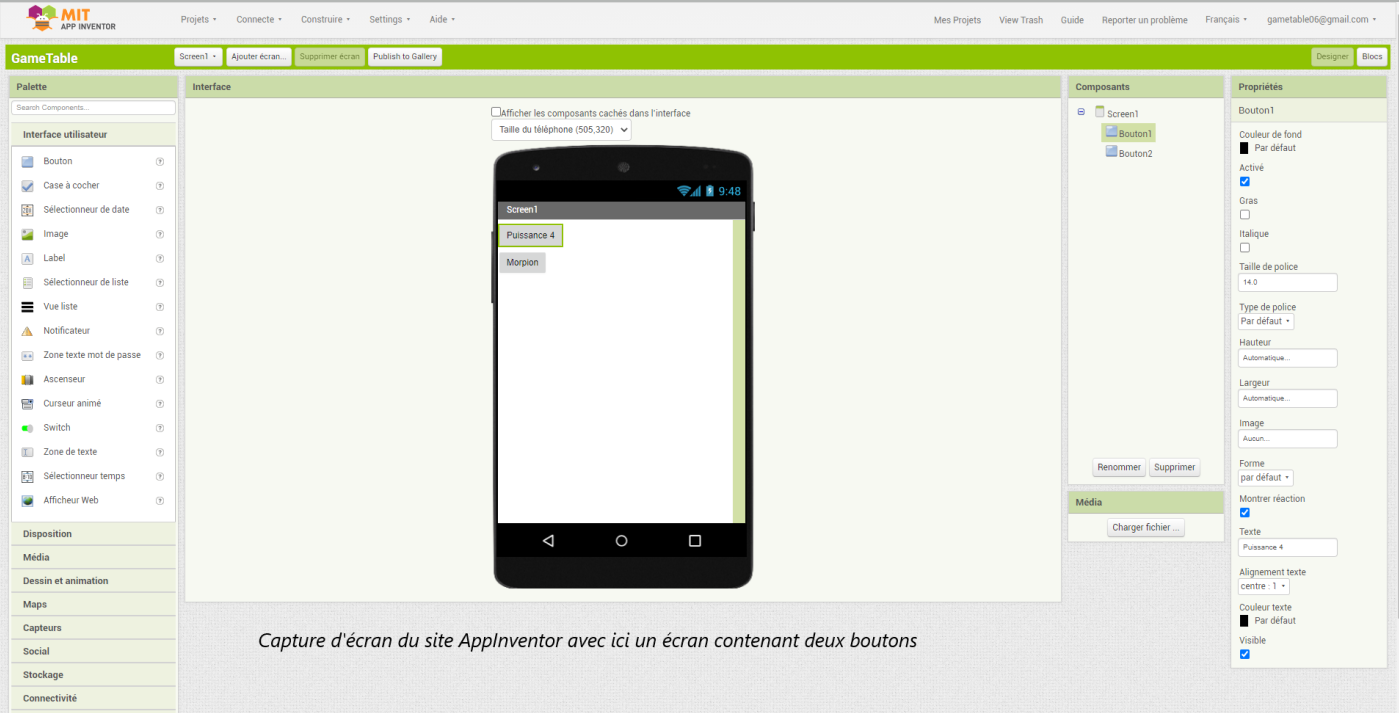
- Bluetooth.print(X) : envoie X sur la sortie TX.

b) L’application mobile

Comme dit dans la description de notre projet, pour établir la communication via Bluetooth entre notre carte Arduino et notre téléphone nous allons utiliser une application mobile créée avec *AppInventor*, une application en ligne du MIT.

Cette création se décompose en trois phases :

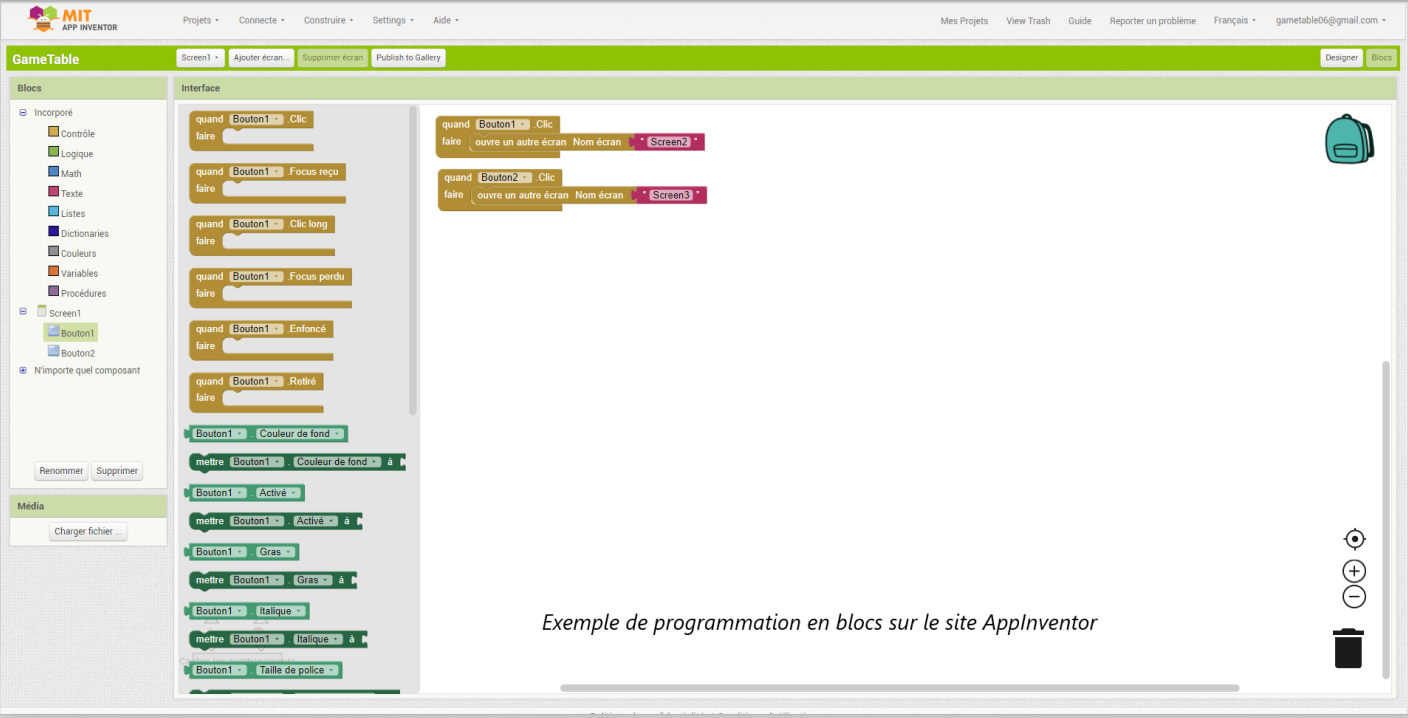
Premièrement, nous allons créer le visuel de notre application. Pour ce faire nous pourrons ajouter des boutons, des images, des listes, des zones de texte et encore beaucoup d’autres choses. Nous voyons en temps réel ce que nous sommes en train de faire, et surtout nous pouvons modifier énormément de choses (taille de la police, couleur, disposition des boutons etc.) pour chaque item que nous ajoutons. De plus, nous pouvons ajouter plusieurs écrans pour faire une application très complète.



Ensuite, vient le temps du codage.

Celui-ci se réalise aussi sur le site et se fait par système de bloc de commande déjà créé et qu’il faut juste emboiter. On y retrouve les traditionnelles commandes « if » ; « else » ; « elif » ; « for » ; « while », ainsi que les booléens « true » et « false ». Et puis, on y retrouve aussi les couleurs, les textes et énormément de personnalisation.

Mais le plus important : des actions propres à chaque item de l’application. Par exemple pour les boutons, on peut décider de faire une action particulière lorsqu’on appuie sur un bouton par un clic long ou une autre par un clic court.



Enfin, nous pouvons uploader l’application sur notre téléphone via un QR-code qui nous renvoie vers un site pour télécharger notre application. Nous pouvons faire tout ce qui a été programmé sur le site et le visuel est bien identique à ce que nous voyons. Pour donner un exemple, nous avons essayé au tout début de la prise en main du logiciel faire en sorte d’avoir deux boutons qui nous renvoyaient sur deux écrans différents. Arrivé sur un de ces deux nouveaux écrans, un nouveau bouton était présent ainsi qu’une zone de texte. Lorsque nous appuyions sur ledit bouton, le texte changeait ainsi que sa couleur de fond.

4) Les LED choisies :

La première solution que nous avons envisagée est d’utiliser des bandes de LED Neopixel.

Cependant, pour brancher toutes les bandes nécessaires pour couvrir toute la surface de l’écran, il aurait fallu un nombre très important de câbles, et la gestion de l’allumage des LED était compliqué puisqu’il aurait fallu s’occuper de chaque bande une par une…

C’est donc pour ça que nous avons opté pour une matrice LED.

Afin que la matrice ne soit pas trop grande par rapport à l’écran, nous avons choisi une matrice de taille 32x32 led.

Pour gérer la communication avec la matrice, on utilise les librairies qui fonctionnent ensemble : Adafruit BusIO, Adafruit GFX, Adafruit NeoMatrix, Adafruit NeoPixel et RGBmatrixPanel.  
On utilisera notamment les fonctions suivantes :

- RGBmatrixPanel matrice = new RGBmatrixPanel(A, B, C, CLK, LAT, OE, true) : sert à créer l’objet matrice 32x32

- matrice.begin() : permet de faire marcher la matrice de led.

- matrice.drawPixel(x, y, c) : permet d’allumer la led de coordonnées (x,y) avec la couleur c.

- matrice.fillscreen(c) : permet de remplir la matrice led de la couleur c.

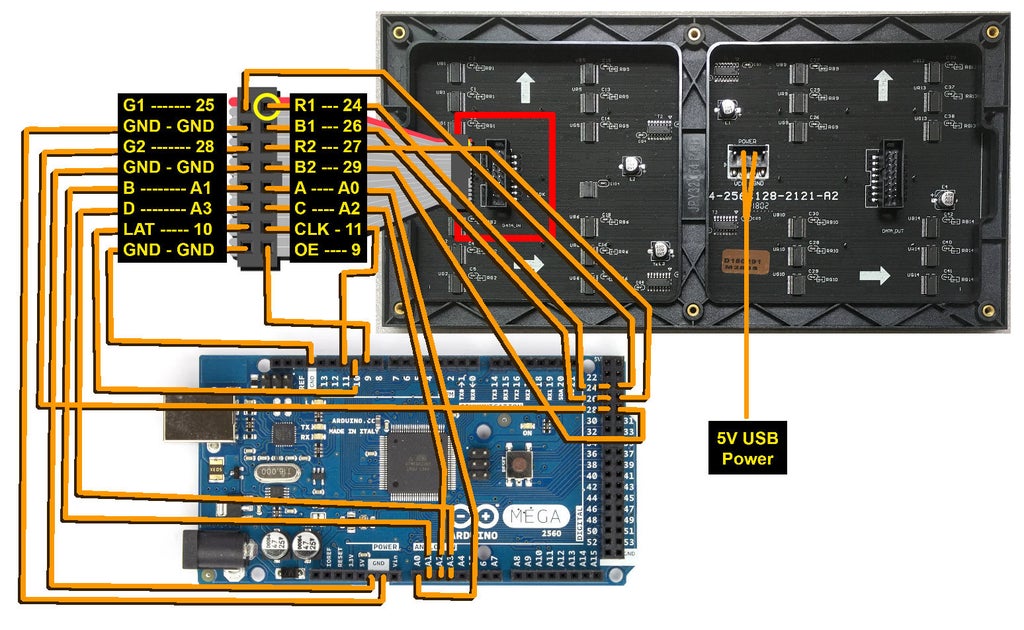
Pour brancher la matrice à l’Arduino, il faut faire les branchements suivants (la broche D n’existe pas dans le cas d’une matrice 32x32) :

Schéma de montage d’une matrice de led à l’Arduino Mega

**Liens :**

Table de jeux :

<https://create.arduino.cc/projecthub/AntoineKia/interactive-led-table-for-50-650b83?ref=tag&ref_id=bluetooth&offset=17>

<https://www.youtube.com/watch?v=9mYyU1978k4>

<https://www.youtube.com/watch?v=X24BZuQ2AWc>

<https://www.instructables.com/Arduino-LED-Coffee-Table/>

LED :

<https://create.arduino.cc/projecthub/glowascii/neopixel-leds-arduino-basics-126d1a>

<http://www.fablabredon.org/wordpress/2017/12/17/lumiere-sur-larduino-avec-de-la-couleur-et-des-led/>

<https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library-use>

<https://www.youtube.com/watch?v=XH4mNRYMCSA>

<https://www.instructables.com/64x32-RGB-LED-Matrix-With-Arduino-Mega/>

<https://www.arduino.cc/reference/en/libraries/esp32-64x32-led-matrix-hub75-dma-display/>

<https://www.youtube.com/watch?v=9plJ-0MhVDI>

Ecran :

<https://blog.integral-system.fr/difference-entre-le-tactile-resistif-et-le-tactile-capacitif/>

Application mobile :

<http://ai2.appinventor.mit.edu/#5518053553340416>

<https://www.youtube.com/watch?v=fr0yT7Ad1AE>

<https://www.youtube.com/watch?v=opKcik96F58>

Connectivité Bluetooth :

<https://pecquery.wixsite.com/arduino-passion/copie-de-le-detecteur-a-ultrasons-h-1>

<https://knowledge.parcours-performance.com/arduino-bluetooth-hc-05-hc-06/>

<https://www.youtube.com/watch?v=jR4wX7GF9gQ>

<https://www.youtube.com/watch?v=SX54SIAUVcM>

<https://www.aranacorp.com/fr/votre-arduino-communique-avec-le-module-hc-05/>

Alimentation :

<https://navlab.fr/projets/outils/arduino/alimenter-votre-carte-arduino/>

<https://www.youtube.com/watch?v=yuSYzujbz-o>