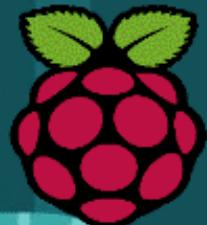


Qu'est ce qu'on Fabrique ?



ATELIER DE CREATION NUMERIQUE

CAHIER DE FICHES

LES FICHES SONT CLASSEES PAR NIVEAU:

'NIVEAU 0 : ZOMBIE

'NIVEAU 1 : INITIE

'NIVEAU 2 : EXPERT

'NIVEAU 3 : MAITRE

LES FICHES REGROUENT LES THEMES SUIVANTS :

HARDWARE

'RASPBERRY PI

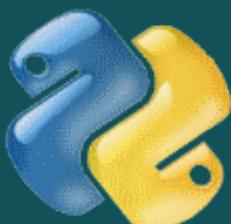
'ARDUINO

'ELECTRONIQUE

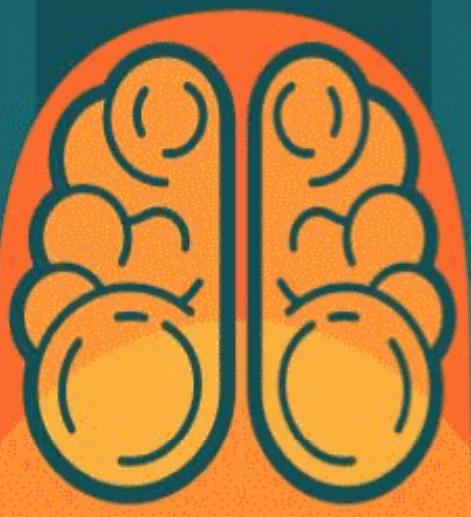
PROGRAMMATION

'PYTHON

'C (ARDUINO)



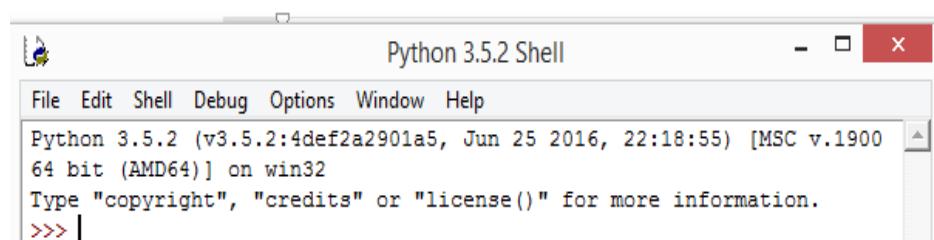
NOM :





Si vous lancez directement Python vous trouverez un écran noir pas très attrayant mais ce n'est pas cela qui nous intéressera. Cherchez IDLE dans votre menu démarrer et vous verrez apparaître le Shell blanc de python (ci-dessous).

Cette interface nous permettra de lancer notre programme et de débugger en cas d'erreur. Pour commencer votre premier programme, cliquez sur **File** puis **File New**.



Un genre de traitement de texte vierge s'ouvre et c'est là que nous pourrons taper notre code.

Premier code :

```
nom=input('Quel est votre nom? ')
age=input('Quel est votre age? ')
naissance=2017-int(age)
print('Bienvenue {0}, vous êtes donc né en {1}'.format(nom,naissance))
```

Quelques explications avant de le lancer notre programme. Input permet de demander une information à l'utilisateur.

La première ligne demande donc le nom et python l'attribue à la variable nom.

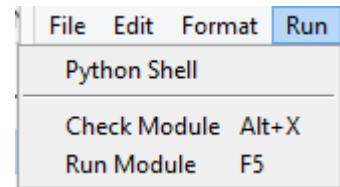
La deuxième ligne demande l'âge et l'attribue à la variable âge.

La troisième crée une variable nommée naissance qui soustrait l'âge à l'année en cours.

Remarque : int permet de transformer l'information renseignée par l'utilisateur en entier (integer)

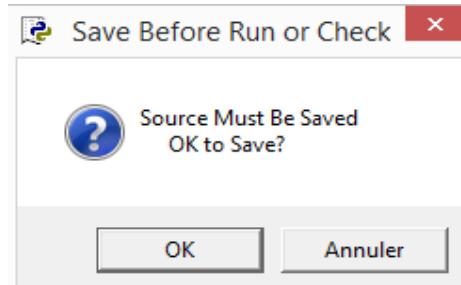
La dernière affiche alors le message où {0} fait référence à la première variable indiquée dans le format en fin de phrase, c'est-à-dire nom et {1} fait référence à la seconde donc naissance.

Lançons à présent le programme pour cela deux possibilités, soit via la touche **F5** de votre clavier, soit en cliquant sur **Run** dans la barre supérieure.



Python vous demande alors de sauvegarder votre programme avant de pouvoir le tester.

Sélectionnez le dossier de votre choix, donnez un nom au fichier sans oublier de finir par **.py**



Maintenant que nous savons interagir avec l'utilisateur, nous allons voir comment faire réaliser à python des tâches qui peuvent paraître répétitives, c'est ce que l'on appelle les boucles.

LES BOUCLES :

```
for i in range(10):
    print('Ca fait {} tour que je fais'.format(i))
```

Nous choisissons un nom pour une variable muette qui représentera le nombre d'itérations à faire réaliser à notre boucle, ici *i*.

Nous souhaitons faire écrire 10 lignes à python chacune indiquant la valeur de *i*. Lançons le programme :

```
Ca fait 0 tour que je fais
Ca fait 1 tour que je fais
Ca fait 2 tour que je fais
Ca fait 3 tour que je fais
Ca fait 4 tour que je fais
Ca fait 5 tour que je fais
Ca fait 6 tour que je fais
Ca fait 7 tour que je fais
Ca fait 8 tour que je fais
Ca fait 9 tour que je fais
```

Python écrit bien 10 lignes mais il s'arrête à *i* = 9 car sans indication contraire python commence toujours à compter à 0.

Nous avons vu la boucle **FOR** (POUR) mais il ne s'agit pas là de la seule boucle possible, nous avons également accès aux boucles **WHILE** (TANT QUE).

Tapez le texte ci-dessous et exécutez-le.

```
i=1
while i<10 :
    print('Ca fait {} tour que je fais'.format(i))
```

Pour l'arrêter appuyez simultanément sur les touches **CTRL** et **C**. Vous êtes tombés dans le piège de la boucle infinie, c'est-à-dire que si on ne l'arrête pas manuellement elle ne s'arrêtera jamais. En effet, ici *i* vaut 1 dès le début mais on ne le fait jamais évoluer. Si on veut faire évoluer *i*, il faut alors le faire augmenter de 1 à chaque tour, pour cela on peut rajouter :

```
i=i+1
```

Si vous êtes habitué d'un autre langage, on peut aussi utiliser la syntaxe suivante pour obtenir la même chose :

```
i+=1
```

On relance alors le programme et on remarque que ce coup-ci la valeur évolue bien en commençant à 1 mais s'arrête à nouveau à 9. En effet, la condition de boucle est que i doit rester inférieur à 10 (strictement) donc 10 ne satisfait pas la condition de boucle. Pour s'arrêter à 10 il faudra alors modifier la condition de la boucle par :

```
while i<11 :
```

On se rend compte que les conditions sont primordiales, nous allons donc les étudier plus précisément.

LES CONDITIONS

Classiquement on parle des conditions IF - ELSE (Si - Sinon). Prenons l'exemple suivant :

```
valeur=input('Entrez un nombre ')
if int(valeur)<10 :
    print('il est plus petit que 10')
else :
    print('il est supérieur à 10')
```

Comme précédemment on demande à l'utilisateur d'entrer une valeur et on va « analyser » cette valeur, vérifier si elle est inférieure à 10 ou non.

Les tests sont « classiques », on peut vérifier si deux valeurs sont strictement supérieures, supérieures, strictement inférieures, inférieures, égales, différentes... Mais le souci habituel est la syntaxe, voici un petit tableau récapitulatif :

<	Strictement Inférieur
>	Strictement Supérieur
<=	Inférieur
>=	Supérieur
==	Egal
!=	Different

Les conditions ne peuvent être réalisées sans comparaisons et sans variable, mais il faut être vigilant car il existe plusieurs sortes de variables.

LES VARIABLES SIMPLES

A la différence d'autres langages, python ne demande pas une déclaration du type de variable, c'est-à-dire que python « comprendra » par lui-même si la variable est un nombre, un mot ou une liste. Voyons quelques exemples :

```
mot='physique'  
print('La seconde lettre est : {}'.format(mot[1]))
```

Ici python reconnaît grâce aux '' que la variable mot est une suite de caractères, on peut alors faire afficher le caractère en seconde position.

Ici on définit deux variables, chiffre et nombre, le premier est entier (int) et le second un décimal (float). On additionne les deux et naturellement python va transformer le résultat de l'addition en un décimal. On peut le transformer en entier en plaçant int avant le résultat.

```
chiffre=1  
nombre=16.0  
print(chiffre+nombre)  
print(int(chiffre+nombre))  
print(float(chiffre+nombre))
```

```
liste=['maison','champignon','tortue']  
print(len(liste))  
print(liste[0])  
print(liste[1][4])
```

A présent nous créons une liste composées de trois mots mais on pourrait y mélanger les types de variables. La commande len permet de connaître le nombre variables dans la liste.

Encore une fois la première variable est la numéro 0. Si on veut afficher la cinquième lettre du second mot de la liste, on pourra alors utiliser la dernière ligne du code.

Reprendons la liste précédente et comme une liste n'est pas une variable figée, on peut y rajouter des termes en fin de liste grâce à la fonction append.

```
liste=['maison','champignon','tortue']  
print(len(liste))  
liste.append('grue')  
print(len(liste))  
print(liste)
```

DÉFIS

Réaliser un jeu du pendu.

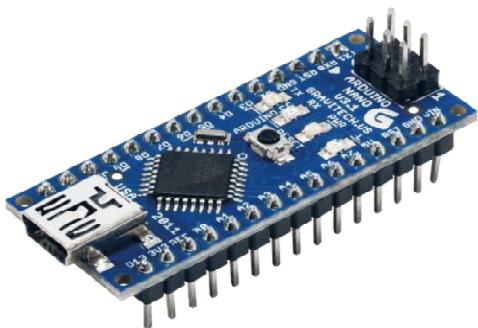
Réaliser un jeu de mastermind avec 4 cases et 5 couleurs (Rouge-Vert-Bleu-Jaune-Orange).



Arduino est une carte de prototypage open-source, c'est-à-dire qu'elle permet de réaliser des prototypes, d'essayer des programmes, de les modifier, de les améliorer, de les re-tester,... Avant de passer à la manipulation, un peu de présentation pour apprendre à bien se connaître.

Le fait qu'elle soit open-source permet à la communauté d'améliorer notamment les logiciels qui nous permettront d'interagir avec ces cartes.

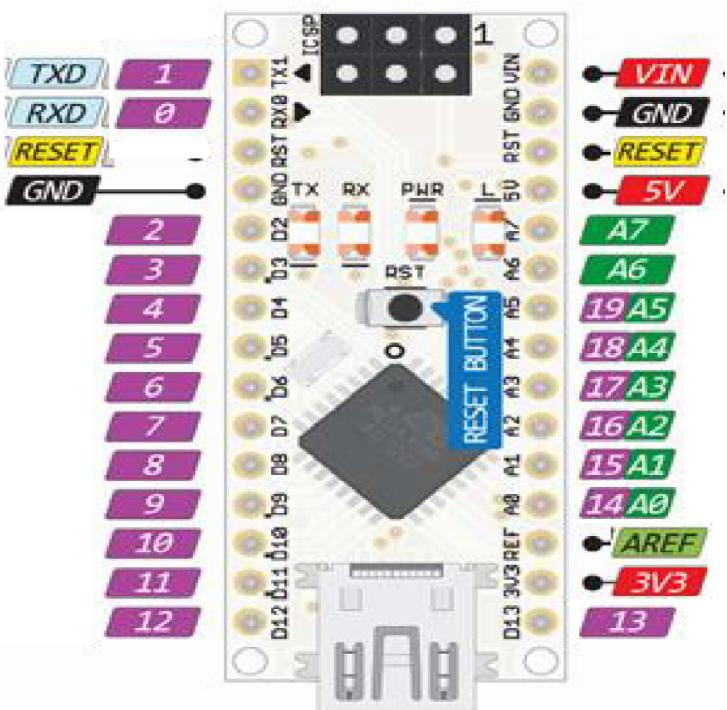
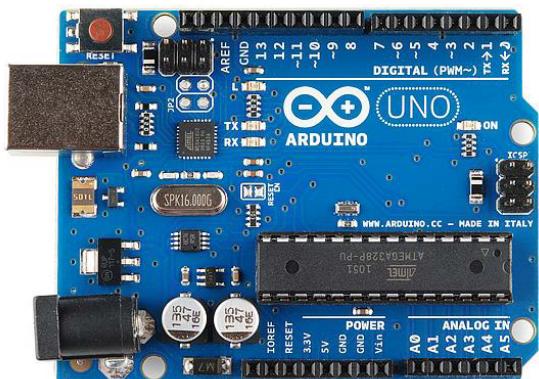
Il existe différentes sortes de cartes que l'on choisira en fonction du projet à réaliser (Consommation énergétique, encombrement, nombre de connexions, de capteurs...). Les plus courantes sont les Arduino UNO qui sont un bon compromis entre les Nano et les Mega en passant par la mini.



Nous travaillerons dans cet exemple et très souvent au labo avec des cartes Arduino Nano. Elles possèdent le même nombre de connexions (Pins) que le UNO mais sont plus facilement connectables sur une plaque de prototypage (aussi appelée BreadBoard).

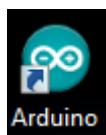
Chacune des pins (D2 à D13 et A0 à A7) peut être utilisée comme une entrée (mesure d'un capteur) ou comme sortie (allumage d'une lampe). Les pins commençant par D (digital) sont pins numériques, c'est-à-dire des pins « tout ou rien » alors que les pins commençant par A sont des pins dites analogiques, autrement dit, elles peuvent dire si le signal d'entrée est plus ou moins « fort ».

Pour ce qui est des autres pins et du fonctionnement particulier de chacune d'entre-elles nous le verrons au fur et à mesure des fiches d'initiation.



Maintenant que nous avons parlé du côté matériel (HardWare) il est nécessaire de parler du côté logiciel (SoftWare) puisque c'est lui qui va nous permettre de programmer et d'envoyer nos programmes dans l'arduino via son câble USB.

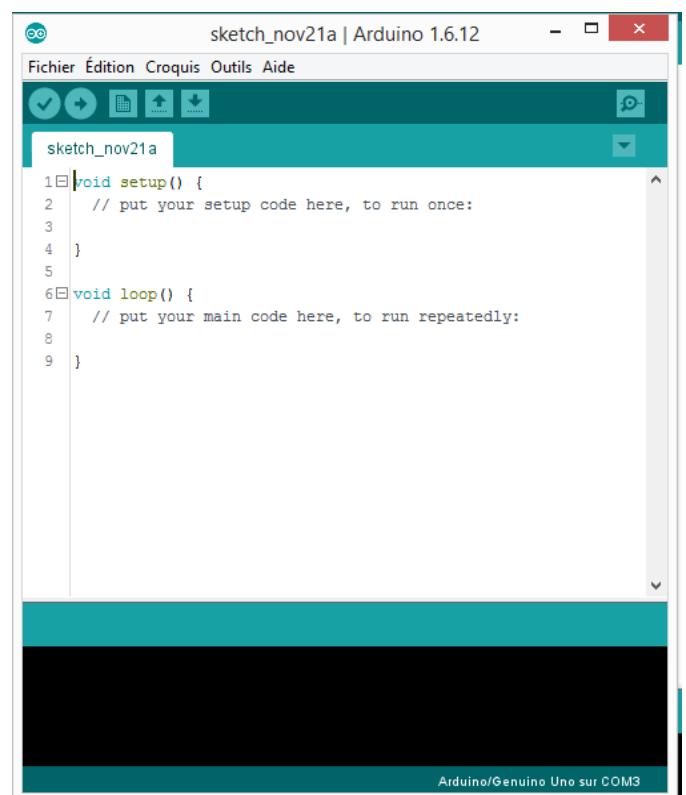
Lancez le programme situé sur le bureau et commençons à nous familiarisé au logiciel. Le software se met alors à se charger :



Vous arrivez à présent sur l'interface de programmation de l'arduino. Elle se décompose en plusieurs parties mais dans un premier temps c'est la structure du programme qui retiendra notre attention.

```
1 void setup() {  
2     // put your setup code here, to run once:  
3 }
```

Cette fonction du code correspond à l'initialisation, elle ne sera réalisée qu'une seule fois au démarrage de votre microcontrôleur. Nous pourrons par exemple y déclarer l'utilisation des pins de l'Arduino. Elle commence à l'ouverture de l'accolade et de terminera par une fermeture d'accolade.



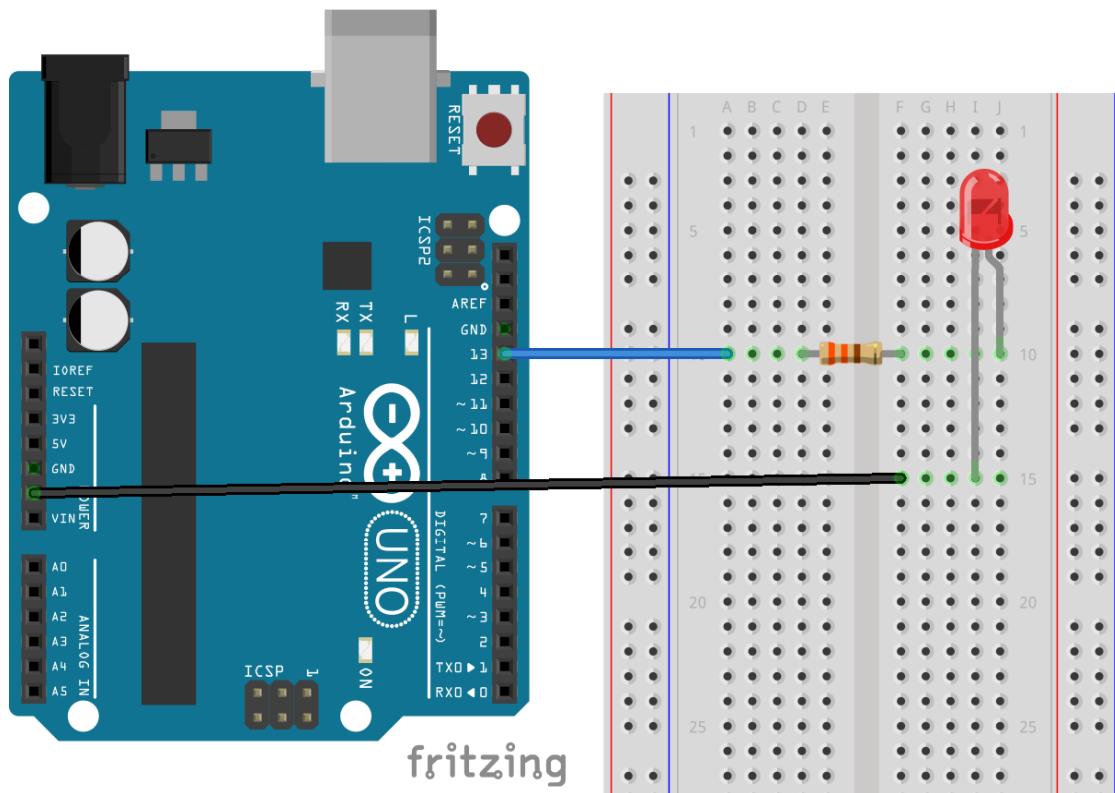
Pour ce qui est de l'utilisation de bibliothèques particulières comme l'utilisation d'un afficheur, d'un thermomètre(...) mais aussi pour les variables ou les constantes, il est de coutume de les déclarer avant cette fonction.

En dessous de cette fonction, nous pouvons voir la fonction loop qui elle est une boucle infinie :

```
6 void loop() {  
7     // put your main code here, to run repeatedly:
```

Cette fonction est limitée également par des accolades et ne s'arrêtera jamais ! Le microcontrôleur la parcourt, réalise ce que lui est demandé puis recommence.

Nous allons à présent commencer par un exemple simple, le clignotement d'une LED (ou diode électroluminescente). La LED est polarisée, c'est-à-dire qu'elle possède une patte à brancher du côté positif (+ cathode) et une autre côté négatif (- anode). Sauf cas particulier, la patte la plus longue est la cathode. Pour éviter tout problème, il est préférable de ne pas brancher directement une LED à l'Arduino mais de rajouter une petite résistance pour limiter la quantité de courant qui pourra traverser. La théorie étant faite, passons à la pratique. Commençons par relier la cathode de la LED à la pin 13 de l'Arduino et l'anode à une patte de la résistance (non polarisée). La patte de la résistance sera elle reliée à la pin GND de l'Arduino qui signifie Ground (terre) et qui n'est rien d'autre que le pôle négatif du microcontrôleur. Vous obtenez le montage ci-dessous :



Avant de brancher l'Arduino à l'ordinateur, il faut à présent préparer le programme.

```
1 int led = 13; //Pin de la borne positive de la LED
```

Pour débuter, une bonne habitude est définir les constantes. Ici nous allons attribuer un nom original de variable à notre diode : led. Cette variable contient la pin sur laquelle est branchée la Led, ici la borne 13.

Remarque : Vous devez finir chacune des lignes par un point virgule sans quoi vous rencontrerez des problèmes lors de l'envoi du programme sur le microcontrôleur.

```
3 void setup() {  
4     pinMode(led, OUTPUT);  
5 }
```

Ensuite nous allons initialiser le mode de cette pin. Comme la pin commandera la sortie, il faut la définir comme étant une sortie (OUTPUT).

```
7 void loop() {  
8     digitalWrite(led, HIGH);  
9     delay(2000);  
10    digitalWrite(led, LOW);  
11    delay(2000);  
12 }
```

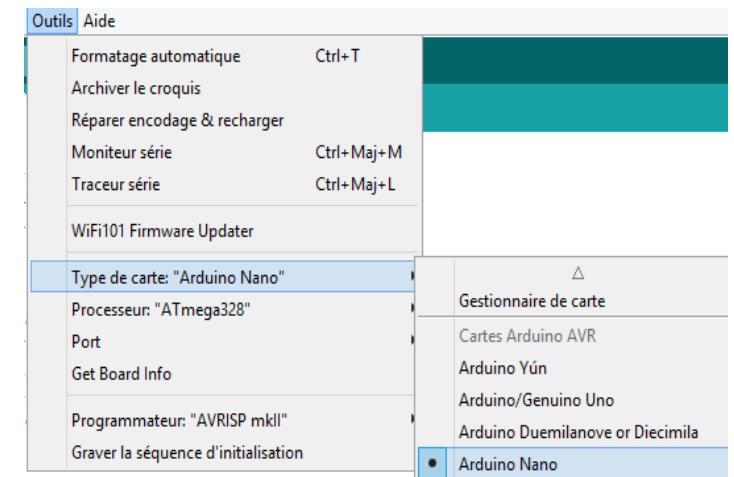
Passons au corps du programme, à la fameuse boucle infinie. Pour mettre notre pin au niveau haut, c'est-à-dire allumée, il faut dire à l'Arduino « d'écrire » sur la pin digital correspondant à notre Led la valeur haute (HIGH). Si on alterne niveau haut et niveau bas, la

fréquence sera telle qu'on ne la verra pas clignoter, pour cela nous rajoutons un « delay » de 2000 millisecondes (2 secondes) durant lequel elle restera allumer avant de s'éteindre à nouveau deux secondes avant que la boucle ne recommence et que la led s'allume à nouveau.

Envoyer le programme vers l'Arduino :

Votre programme fini, vérifiez que le programme reconnaît bien votre carte Arduino, pour cela rendez-vous dans Outils puis choisissez votre carte. Vérifiez également que le Port est bien différent du COM1.

Pour envoyer votre programme vous n'avez plus qu'à le téléverser, pour cela repérez les deux ronds en haut à gauche et appuyez sur :



Erreur à ne pas commettre : n'oubliez pas le second délai sans quoi votre Led s'éteindra mais la boucle recommencera instantanément elle se rallumera sans que vous ne l'ayez vu s'éteindre.

```
7 void loop() {  
8     digitalWrite(led, HIGH);  
9     delay(2000);  
10    digitalWrite(led, LOW);  
11 }
```

Remarque : La pin 13 n'a pas été choisie au hasard, elle possède déjà une mini-led qui sert de témoin. Vous pouvez voir directement sur l'Arduino une diode s'allumer et s'éteindre au même rythme que votre diode.

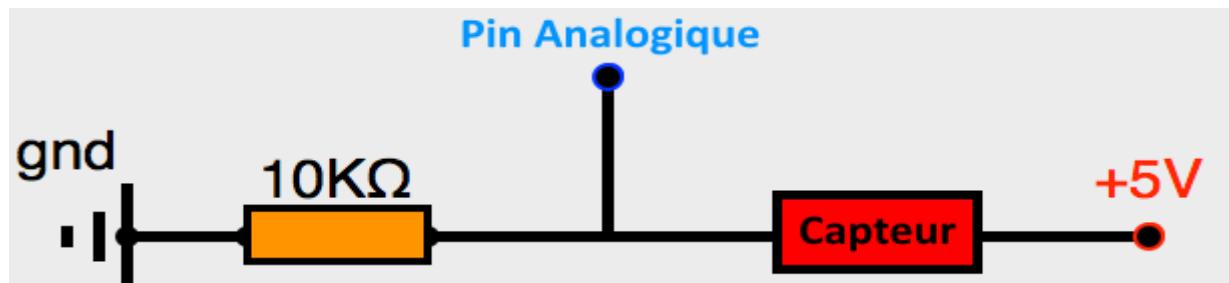
DÉFI: Réaliser un feu rouge de signalisation routière où le rouge restera allumé 10 secondes avant de passer au vert pendant également 10 secondes puis au orange pendant seulement 3 secondes.

Il est possible avec un Arduino de mesurer

- des grandeurs analogiques : la tension lue est proportionnelle à la valeur mesurée
- des grandeurs numériques : valeurs précises codées en 0 et 1 ou bien le 0 correspond à absence et 1 à présence de gaz, de lumière...

I. L'analogique :

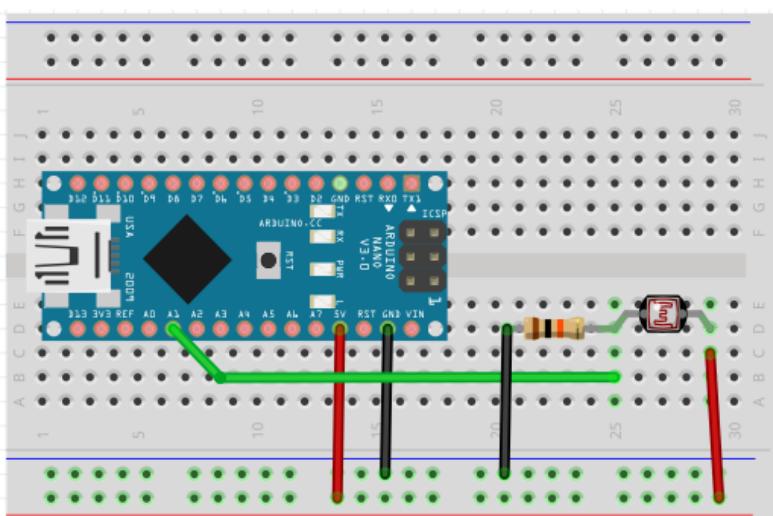
L'Arduino est capable de mesurer des tensions entre 0 et 5V sur les huit entrées commençant par la lettre A. Ces tensions sont divisées en 1024 niveaux (entre 0 et 1023). Pour lire la valeur du capteur, on utilisera un montage électronique appelé pont diviseur (sur lequel on ne s'attardera pas) :



• La photo-résistance :

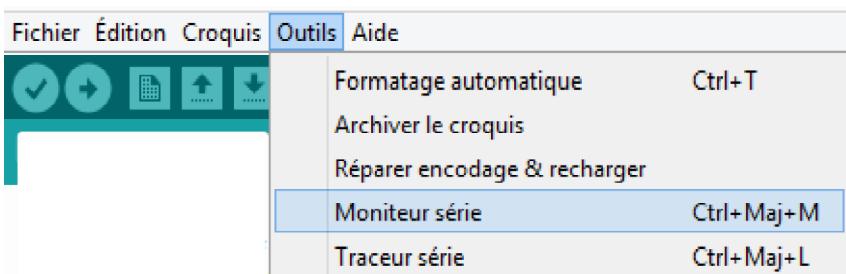
Elle permet de mesurer la luminosité ambiante, plus celle-ci est faible et plus la résistance est importante. Notre montage nous permettra alors d'obtenir une valeur proche de 1023 quand la luminosité est forte et proche de 0 dans l'obscurité.

Passons à la pratique :



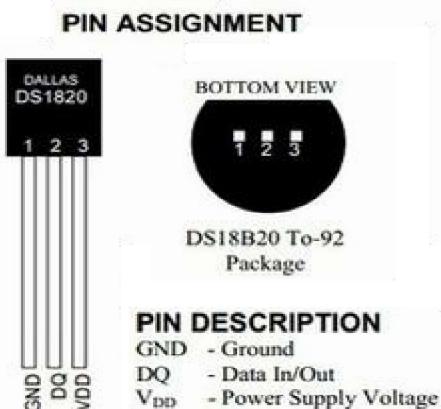
```

1 void setup() {
2     //Initialise l'affichage des
3     //valeurs sur le moniteur:
4     Serial.begin(9600);
5 }
6
7 void loop() {
8     // On lit l'entrée analogique A1
9     //et on l'affiche sur le moniteur:
10    Serial.println(analogRead(A1));
11    //On rajoute un délai d'attente
12    //pour plus de stabilité
13    delay(100);
14 }
```



Pour voir la valeur mesurée il suffit de rester dans le logiciel Arduino et d'ouvrir le moniteur série ou le traceur série.

II. Le numérique :



Comme dit en introduction, il existe deux sortes de capteurs numériques ceux donnant une réponse binaire, présence ou absence, appelés capteurs tout-ou-rien et ceux donnant une valeur précise mesurée par le capteur.

Ici nous étudierons un capteur de température renvoyant une valeur exacte de la température, le capteur Dallas 18B20. Ce capteur possède une plage de température allant de -55°C à 125°C avec une tolérance comprise entre -0.5°C/+0.5°.

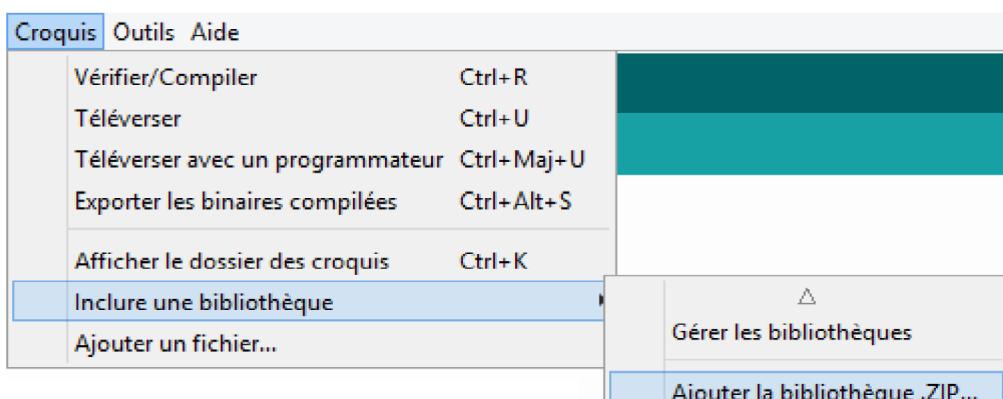
L'avantage d'être un capteur numérique est que l'on peut le brancher sur n'importe quelle borne de l'Arduino (et non pas uniquement sur l'une des 6 entrées analogiques), et qu'il est moins sujet aux parasites.

Pour lire ce capteur il faudra travailler avec **deux librairies** toutes prêtes :

La première librairie s'appelle **One-Wire** (Un Fil) car on verra plus tard que l'on peut brancher dix capteurs sur une seule broche de l'Arduino.

La seconde s'appelle **DallasTemperature**. Elle permet de directement lire les capteurs de chez Dallas.

Arduino n'a pas ces librairies pré-installées, il va donc nous falloir les importer, pour cela rendez-vous dans **Croquis** puis **Inclure une bibliothèque** et enfin **Ajouter la bibliothèque .ZIP** :

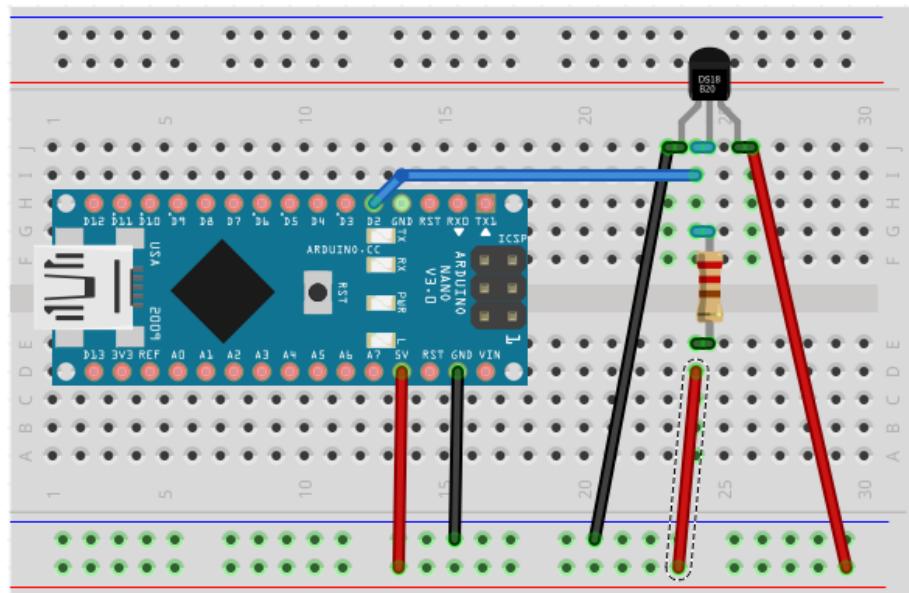


Allez chercher les deux fichiers zip : *OneWire.zip* et *DallasTemperature.zip*

Passons à la pratique :

Pour réaliser le montage il faudra une résistance de 4,7kOhms placée entre les broches VDD et DQ.

La borne GND sera reliée à la masse, la broche DQ (Data) sera reliée à une pin de l'Arduino (ici D2) et la broche VDD au 5V du circuit.

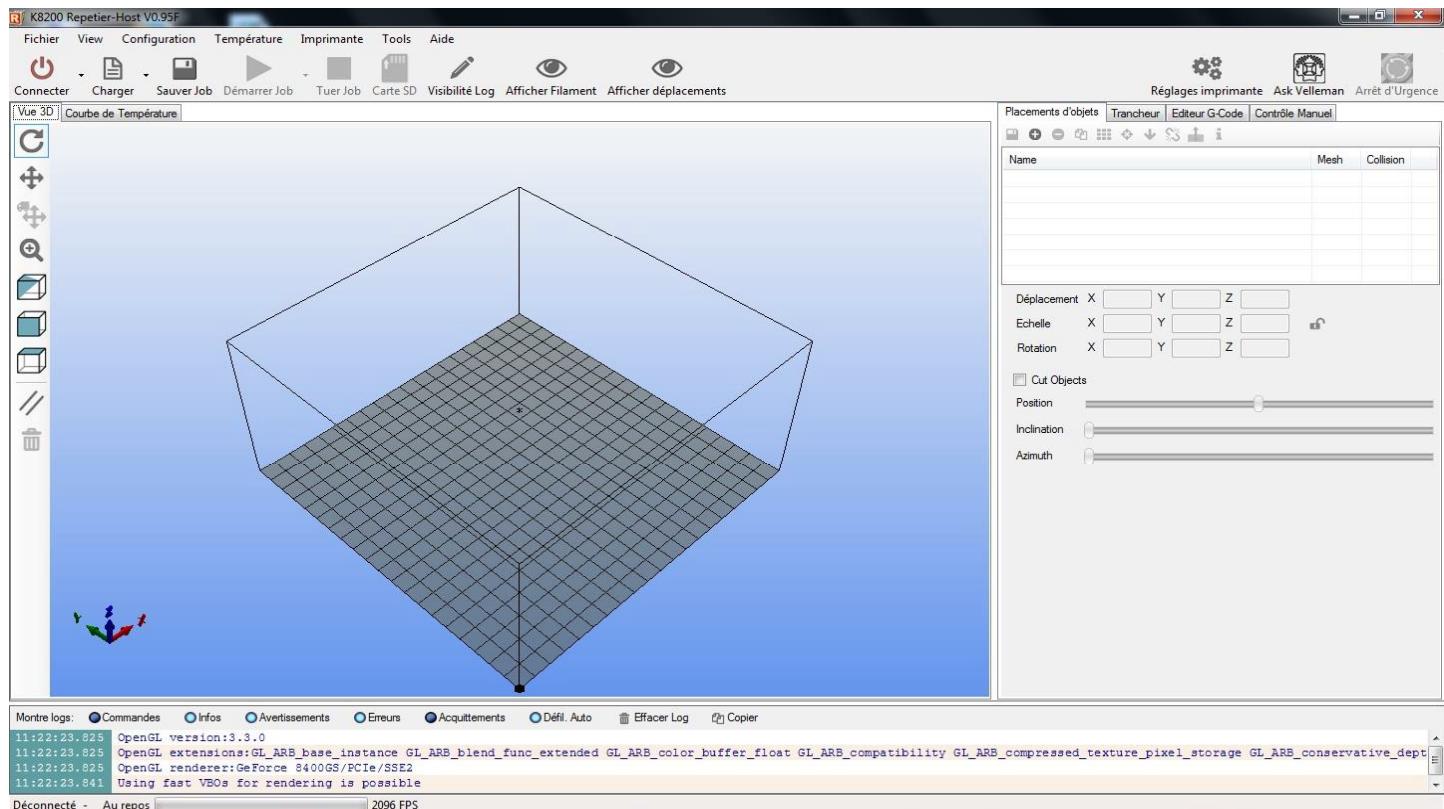


Du côté du code, l'appel aux deux librairies se fait dès le début et ainsi nous pourrons appeler les fonctions au cours du sketch :

```
1 //On inclue les librairies
2 #include <OneWire.h>
3 #include <DallasTemperature.h>
4
5 //On connecte la broche de lecture à la pin 2 de l'Arduino
6 #define ONE_WIRE_BUS 2
7 //On cherche les thermomètres connectés à cette pin:
8 OneWire oneWire(ONE_WIRE_BUS);
9 //On envoie les références trouvées à la librairie Dallas
10 DallasTemperature sensors(&oneWire);
11
12 void setup(void)
13 {
14     //On initialise la communication avec le moniteur série
15     Serial.begin(9600);
16     //On initialise les capteurs dallas
17     sensors.begin();
18 }
19
20 void loop(void)
21 {
22     //On envoie la commande de récupération de température
23     sensors.requestTemperatures();
24     Serial.print("La température est: ");
25     //On lui demande d'afficher la valeur du premier capteur (le 0)
26     Serial.println(sensors.getTempCByIndex(0));
27     //On laisse souffler une seconde avant de relancer.
28     delay(1000);
29 }
```

D E F I S

Réaliser un montage où une led s'allumera en vert quand la température est inférieure à 20°C, en orange entre 20 et 30° et enfin en rouge si elle supérieure à 30°C. .



Placez-vous sur l'ordinateur relié à l'imprimante 3D.

Lancer le logiciel Repetier-Host placé sur le bureau. Il permet de faire chauffer le lit et la tête de l'imprimante mais aussi de la réinitialiser pour qu'elle revienne à son point d'origine.

Une fois le logiciel lancé il faut le connecter à la machine, en cliquant sur le bouton Connecter situé en haut à gauche de votre écran.



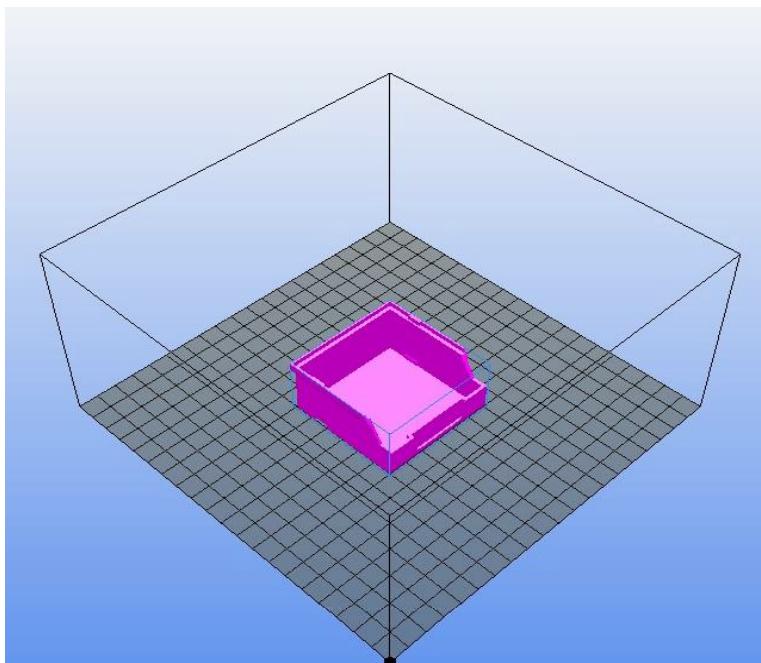
Si tout se passe bien, le bouton se transforme et devient vert avec « Déconnecté » écrit en dessous. Votre imprimante est alors prête à recevoir des instructions.

Nous allons à présent choisir la pièce à imprimer, pour cela dans les onglets situés en haut à droite de votre écran choisissez « Placements d'objets »

[Placements d'objets](#) [Trancheur](#) [Print Preview](#) [Contrôle Manuel](#) [Carte SD](#)



Pour cela cliquez sur le bouton « Charger » placé à côté du bouton « Déconnecter ». Choisissez à présent le fichier « **Box.stl** ».

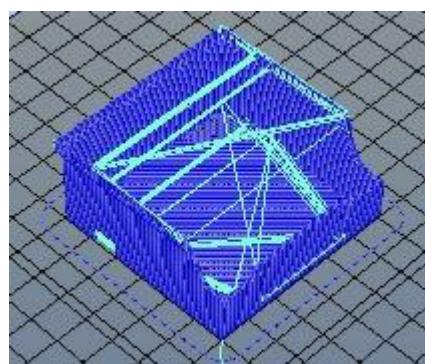


La pièce apparaît alors au milieu du plateau, on vérifie qu'elle ne dépasse pas de la zone de travail, qu'elle soit bien dans le bon sens... Si tout va bien, on va à présent passer dans l'onglet « Trancheur »

Placements d'objets Trancheur

Il est nécessaire de demander à l'ordinateur de « trancher » l'objet en fines couches pour pouvoir l'imprimer, c'est le travail du trancheur, nommé ici Slic3r. Le trancheur est déjà configuré pour fonctionner avec cette imprimante, il suffit donc de cliquer sur « trancher » pour laisser calculer l'ordinateur.

(Attention suivant la complexité de la pièce et la puissance de votre ordinateur, cela peut prendre plusieurs minutes voir heures, soyez juste patients.)



Le tranchage terminé, l'objet n'est plus rose mais bleu et on peut voir les trajectoires prévues pour la buse d'impression. On peut également regarder le temps estimé d'impression qui se trouve en bas du GCode qui vient d'apparaître sur votre droite :

R1 C1 Insère Couche 0 Extrudeur 0 Temps d'impression:53m:48s

Cette information vous permet de préparer votre patience et il est enfin temps de démarrer le job.

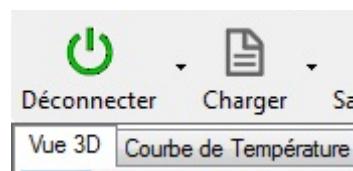
Pour lancer le travail , il faut appuyer sur le bouton correspondant surmonté du logo « Play ».



Extrudeur: 24,0/200°C Plateau: 24,0/80°C

Le plateau se met alors à chauffer tout comme la buse d'impression. Rien ne se passera tant que ces deux éléments ne seront pas arrivés à température.

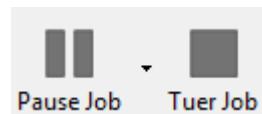
Vous pouvez suivre l'évolution de ces températures dans l'onglet « Température » situé dans le coin supérieur gauche,



Impression...Fin: 39m:11s Couche 3/62

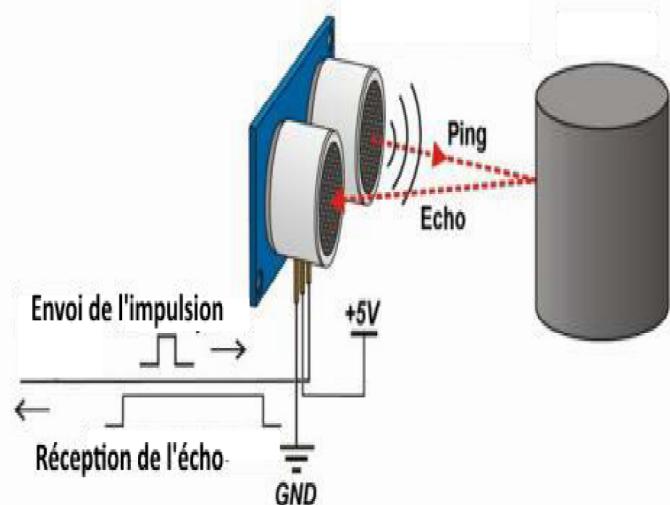
Une fois les températures contraintes atteintes, le plateau se met à bouger et le filament commence à couler de la buse. Votre impression est partie, surveillez-la au moins les cinq premières minutes ou du moins les deux premières couches car il s'agit là du moment stratégique qui permet une bonne accroche de la pièce au plateau pour la suite de l'impression.

En cas de problème, n'hésitez pas à mettre le job en pause pour éviter d'endommager l'imprimante. En cas de souci d'accroche au plateau, d'alimentation en filament ou si le processus se passe mal, n'hésitez pas à « Tuer le Job » et à recommencer l'impression après avoir nettoyé le plateau.



Le capteur ultra-son :

On a vu qu'avec un Arduino, il est possible de mesurer des grandeurs telles que la température, la luminosité mais il est également possible de mesurer des distances, notamment grâce à un émetteur récepteur à ultrason, le HC-SR04.



Ce capteur utilise le principe du sonar, le haut-parleur T émet un ultra-son qui va se réfléchir contre l'obstacle et revenir vers le micro R.

Le temps mesuré entre l'envoi de l'impulsion et de sa réception correspond au temps nécessaire pour effectuer un aller-retour jusqu'à l'obstacle. On sait que dans l'air, le son parcourt environ 340 mètres en une seconde, on peut alors déterminer la distance parcourue par le son et en déduire la distance nous séparant de l'objet en divisant par deux cette dernière.

Il faut dans un premier temps se focaliser sur les caractéristiques du capteur :

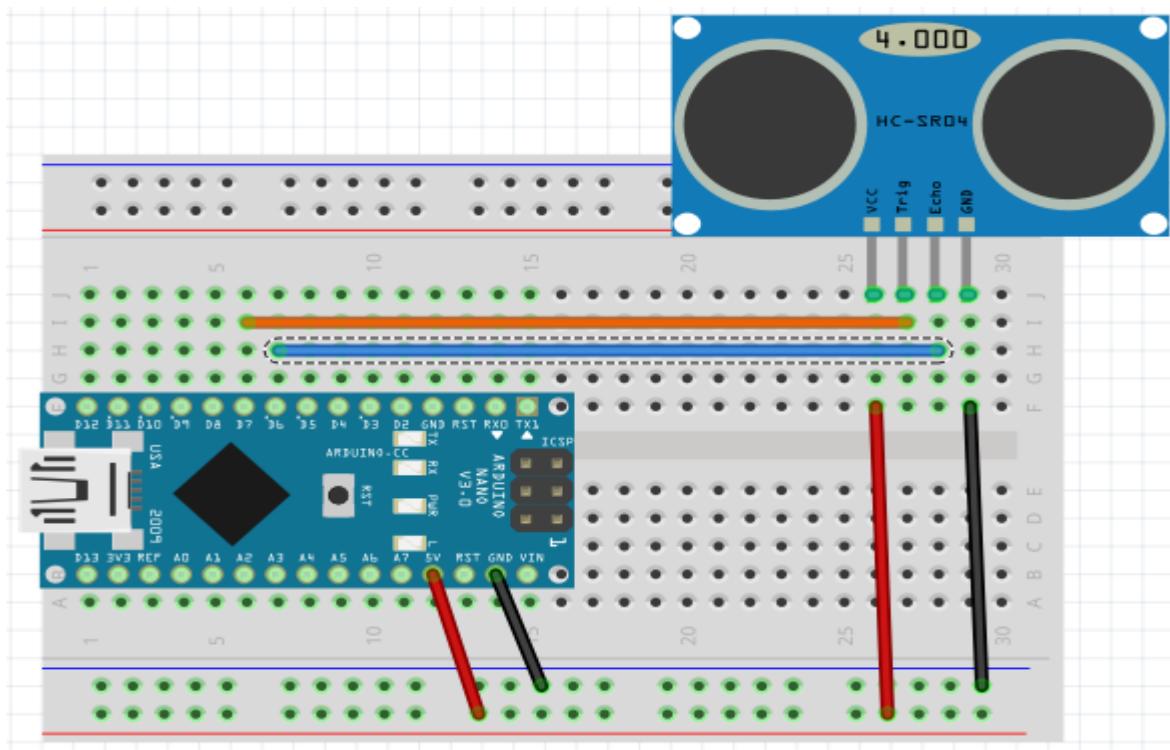
Distance de lecture de 2 à 450cm
 Tension d'entrée : 5V
 Angle de lecture : 15° maximum
 Signal d'initialisation : 10 µS

Le montage :

Cette étape est la plus facile, le capteur présente quatre broches :

- VCC est l'alimentation électrique du capteur et que nous prendrons directement à la pin 5V de l'Arduino.
- GND est la masse (ground) et qui se prendra sur la pin homonyme de l'Arduino
- TRIG est la pin envoyant le signal (pulsation) ultrason, dans notre exemple on la mettra sur la pin D7.
- ECHO est la pin recevant le signal (pulsation) ultrason, dans notre exemple on la mettra sur la pin D6.

On obtient alors le montage suivant :



Le code :

Le code n'est pas évident mais il sera commenter directement au-dessus de chaque ligne pouvant poser problème au cours de la programmation.

```
1  /* Utilisation du capteur de distance à Ultrason SR04 */
2
3 //Assignment des pins de l'Arduino
4
5 #define echoPin 6
6 #define trigPin 7
7
8 //On initialise les valeurs utiles par la suite
9 int mesure = 0;
10 int total = 0;
11 //La vitesse du son variant en fonction de la température
12 //et de l'atmosphère, elle pourra être réglée par
13 //étalonnage
14 int vitesse=340;
15
16 void setup()
17 {
18     Serial.begin(9600);
19     //On définit les modes des pins
20     pinMode(echoPin, INPUT);
21     pinMode(trigPin, OUTPUT);
22     //On s'assure que l'émetteur est éteint
23     digitalWrite(trigPin, LOW);
24 }
```

```

26 void loop()
27 {
28     //On définit le temps et la distance pour le SR04
29     long temps, distance;
30     //variable servant à mesurer plusieurs fois la distance
31     //avant d'en afficher la moyenne
32     mesure=mesure+1;
33     //On allume l'émetteur ultrason
34     digitalWrite(trigPin, HIGH);
35     //Durée d'initialisation du capteur
36     delayMicroseconds(10);
37     //Pulsion ultrasonore émise, on éteint l'émetteur
38     digitalWrite(trigPin, LOW);
39     //On définit la pin Echo comme en attente de message
40     pinMode(echoPin, INPUT);
41     //On recoit l'impulsion sur le récepteur
42     temps = pulseIn(echoPin, HIGH);
43     //On définit la distance comme étant le produit de la
44     //vitesse par le temps divisé par deux pour l'A-R.
45     //Le temps étant en microsec(10^-6) et pour avoir d
46     //en cm (10^-2), on divise par 10 000 (10^-4)
47     distance = temps * vitesse/(2*10000);
48     //On crée une variable total qui sera la distance totale
49     // pour 10 mesures.
50     total=total+distance;
51     //Si les 10 mesures sont faites :
52 if (mesure==10){
53     //On fait la moyenne des 10 mesures
54     distance=total/10;
55     // on affiche l'information sur le moniteur série:
56     Serial.print("Distance: ");
57     Serial.print(distance);
58     Serial.println(" cm");
59     //On remet mesure à 0 pour recommencer une série
60     mesure=0;
61 }
62 }

```

Le capteur à ultrason SR04 ne peut être qualifié de capteur numérique mais il pourra être utilisé aussi bien sur une pin numérique qu'analogique.

Le capteur de gaz :

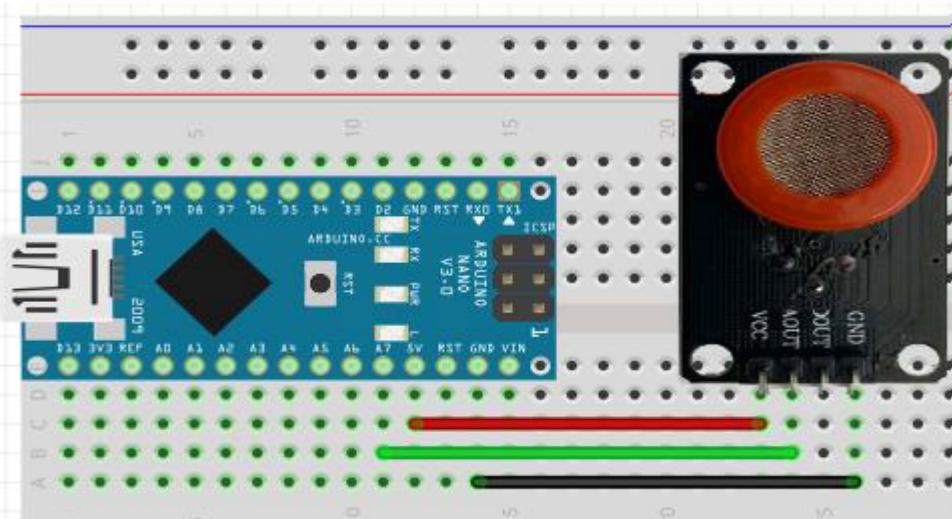


Le capteur a besoin d'une résistance de charge à placer entre la sortie et la masse. Sa valeur peut être comprise entre deux et quarante-sept kilo Ohms. Plus sa valeur sera faible et moins le capteur sera sensible mais moins il sera possible de mesurer de fortes concentrations en gaz (saturation). Si seulement un gaz spécifique est mesuré alors la résistance de charge peut-être calibrée en

utilisant d'une concentration connue de ce gaz. Si ce n'est pas le cas, alors la résistance de charge doit être choisie de sorte à ce que la valeur de sortie soit de l'ordre de 1V dans une atmosphère "normale".

Capteur	Sensibilité	Préchauffage		Autre
MQ-2	Méthane, Butane, GPL et fumées	48h	5V	
MQ-3	Alcool, Ethanol et fumées	24h	5V	
MQ-4	Méthane (CH ₄).	48h	5V	De 300 à 10000 ppm
MQ-5	Gaz naturel, GPL	24h	5V	De 300 à 50000 ppm
MQ-6	GPL, butane	48h	5V	De 200 à 10000 ppm
MQ-7	Monoxyde de carbone (CO).	48h		De 20 à 2000 ppm. La tension alterne entre 5 et 1.4V(librairie spec)
MQ-8	Hydrogène	24h	5V	De 100 à 10000 ppm
MQ-9	Monoxyde de carbone (CO), méthane (CH ₄)			La tension alterne entre 5 et 1.5V. Si seulement CO testé : 1.5V
MQ131	Ozone		6V	La résistance de charge doit être entre 100 et 200 kOhms car les mesures ne sont pas en ppm mais en ppb (partie par milliard).
MQ135	Qualité de l'air	24h	5V	
MQ136	Sulfure d'hydrogène gazeux (H ₂ S).	24h	5V	De 1 à 1000 ppm
MQ137	Ammoniac.			De 5 à 500ppm
MQ138	Benzène, Toluène, Alcool, Acétone, Propane, Formaldéhyde, Hydrogène	24h		De 10 à 1000ppm sauf pour NH ₃ , de 10 à 3000 ppm
MQ214	Méthane propane butane GPL	24h		3000ppm à 20000ppm 500ppm à 10000ppm 500ppm à 10000ppm
MQ216	Gaz naturel, gaz de houille, Propane, CH ₄	24h		
MQ303A	Alcool, Ethanol, fumées	+ 48h		
MQ306A	GPL, butane	+ 48h		Identique au MQ-6 mais avec une tension de chauffage plus basse (0.9V)
MQ307A	Monoxyde de carbone (CO)	+ 48h		Identique au MQ-7 mais avec une tension variant de 0.2 à 0.9V
MQ309A	Monoxyde de carbone, gaz inflammables	+ 48h		Identique au MQ-9 mais avec une tension variant de 0.2 à 0.9V

Une fois le capteur choisi, si celui-ci ne possède pas de potentiomètre au dos, prenons une résistance de 2k0ms et passons au montage :



Le capteur choisi ici possède quatre pins et nous n'en utilisons que trois. La pin inutilisée DOUT est la sortie digitale, c'est-à-dire numérique qui ne donne donc comme unique information : Présence (1) ou Absence (0) de gaz. Nous privilégierons la sortie analogique qui elle permet une précision sur 1024 niveaux de concentration.

Le Code :

Pour le code, il s'agit d'un simple code de lecture d'une pin analogique, ici la A7. Pour ce qui est de la valeur affichée, on pourra choisir d'afficher directement la valeur lue, comprise entre 0 et 1023 ou si on connaît la gamme de mesure du capteur, on pourra faire un petit calcul d'après la concentration maximale mesurable. Dans notre exemple du MQ5, nous connaissons la gamme : De 300 à 50000 ppm.

```
1 /*Lecture entrée analogique*/
2 int concentration_max=50000;
3 void setup() {
4     // initialize serial communication at 9600 bits per second:
5     Serial.begin(9600);
6 }
7
8 void loop() {
9     // Lecture de la valeur de la pin A7:
10    int sensorValue = analogRead(A7);
11    // Conversion de la valeur (0 - 1023) en concentration:
12    float concentration = sensorValue * (concentration_max / 1023.0);
13    // print out the value you read:
14    Serial.print(concentration);
15    Serial.println(" ppm");
16 }
```

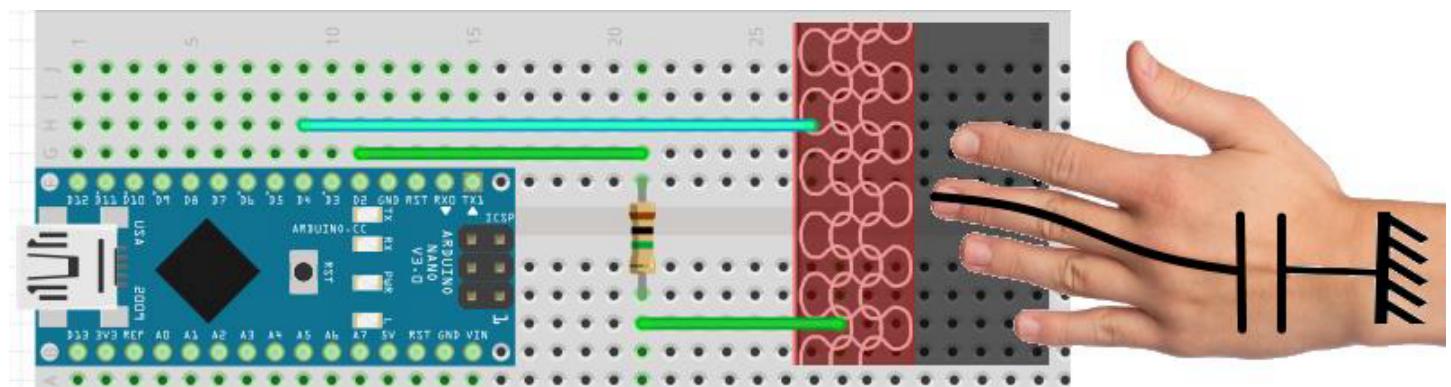
Le bouton inductif :

Le principe du bouton inductif ou bouton tactile est simple notre corps est chargé d'électricité statique, et considéré comme une masse électrique. On place une surface conductrice reliée à l'Arduino et lorsque notre peau entre en contact avec cette surface conductrice, cette électricité statique est captée et le potentiel électrique entre les deux pins change d'un coup.

Dans notre exemple nous pourrons prendre du papier aluminium de cuisine ou une plaque métallique sur laquelle il sera plus facile de souder des fils. Il nous faudra également une très grande résistance, de l'ordre de 100 kilo à 50 Méga Ohms. Cette résistance sera à ajuster lors des tests, plus elle sera élevée et plus le capteur sera sensible mais moins ce dernier répondra vite.

Remarque : Pour plus d'informations vous pouvez effectuer une recherche sur l'oscillateur électrique RC.

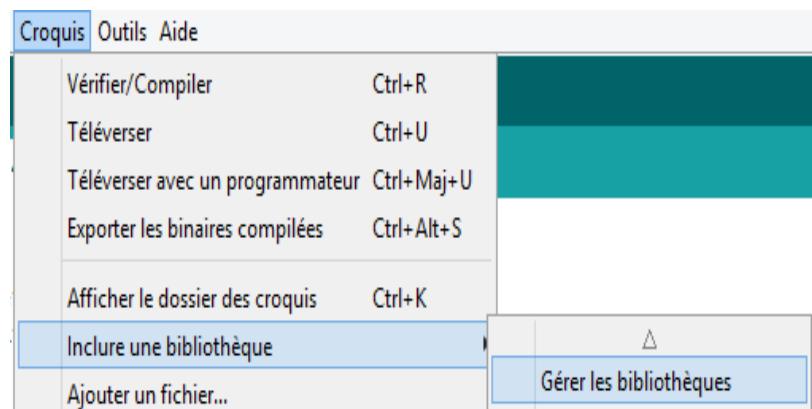
On se place entre les pins D2 et D4 afin d'obtenir le montage suivant :



Le Code :

Avant de passer à la programmation à proprement parler et surtout pour se simplifier la vie, nous allons installer une bibliothèque :

CapacitiveSensor. Pour cela dans le logiciel Arduino, rendez-vous dans *Croquis* puis *Inclure une bibliothèque* et enfin *Gérer les bibliothèques*.



Dans la ligne de recherche à droite, saisissez le nom de la bibliothèque **CapacitiveSensor** puis appuyez sur Entrée. Deux bibliothèques vous sont proposées, choisissez celle des deux Paul, Bagder et Stoffregen :

Type Tout Sujet Tout CapacitiveSensor

CapacitiveSensor by Paul Bagder, Paul Stoffregen
Create capacitive sensors that can detect touch or proximity. The capacitiveSensor library turns two or more Arduino pins into a capacitive sensor, which can sense the electrical capacitance of the human body. All the sensor setup requires is a medium to high value resistor and a piece of wire and a small (to large) piece of aluminum foil on the end. At its most sensitive, the sensor will start to sense a hand or body inches away from the sensor.
[More info](#)

Version 0.... Installer

Vous pouvez alors cliquer sur *Installer*.

Grâce à cette librairie nous n'aurons que très peu de code à saisir pour mesurer la valeur de la capacité électrique. Vous pourrez ensuite l'adapter pour allumer des diodes ou déclencher des actions par exemple. Passons donc au sketch Arduino relatif à la bibliothèque :

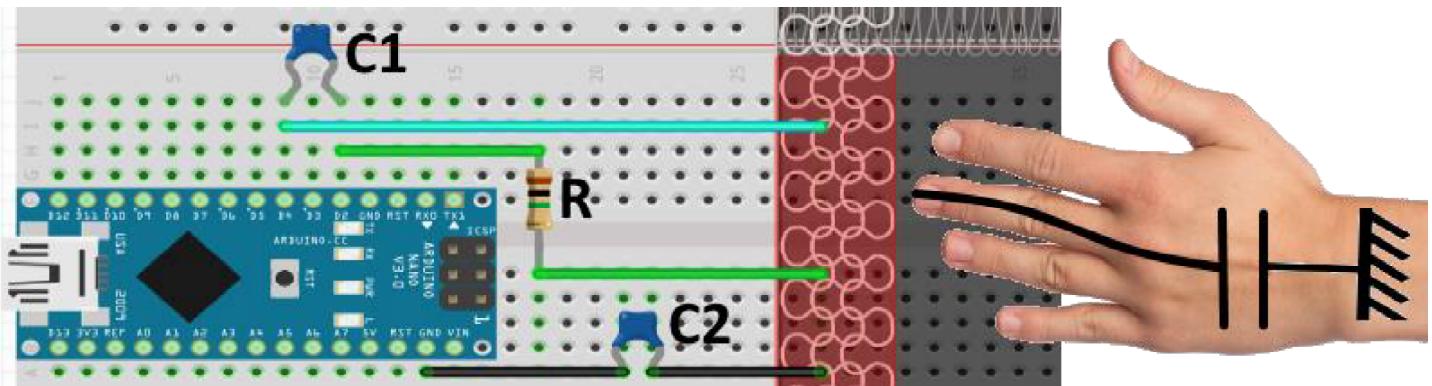
```

1 #include <CapacitiveSensor.h>
2 //La pin qui envoie le signal est la D4 (sendPin) et
3 //la variation est lue sur la pin 2 (ReceivePin) :
4 CapacitiveSensor surface = CapacitiveSensor(4,2);
5
6 void setup() {
7   Serial.begin(9600);
8 }
9
10 void loop() {
11   long start = millis();
12   //On règle la sensibilité du capteur à 30
13   //l'unité est arbitraire
14   long valeur = surface.capacitiveSensor(30);
15   //Quelques lignes pour connaître le temps de réponse.
16   // en millisecondes
17   Serial.print(millis() - start);
18   //Caractère de tabulation du port série
19   Serial.print("\t");
20   Serial.println(valeur);
21   if (valeur>6000){
22     Serial.println("On a touché!");
23   }
24   delay(200);
25 }

```

Améliorations :

Les expériences ont montré que rajouter des condensateurs permet d'améliorer la stabilité et répétabilité de la mesure.



On rajoute **C1** de l'ordre de 20 - 400 pF en parallèle de la mesure de la capacité de l'individu.

On rajoute **C2** de l'ordre de 100 pF placé entre la masse et la touche tactile.

On peut également remplacer la résistance **R** par un potentiomètre pour effectuer un réglage plus fin de la résistance.

Apparu le 29 février 2012, ce micro-ordinateur de la taille d'une carte de crédit était destiné à l'apprentissage de l'informatique à moindre coût. Cet ordinateur a su évoluer et se diversifier pour afficher à ce jour deux modèles phares :



- Le Pi 3, plus performant (processeur Broadcom BCM2837 64 bit à quatre coeurs ARM Cortex-A53 à 1,2 GHz, accompagné de 1Go de RAM et d'une puce Wifi 802.11n et Bluetooth 4.1 intégrée) présenté entre 35 et 40€. Il sera intéressant comme support de programmation, média center, serveur, console rétro ...

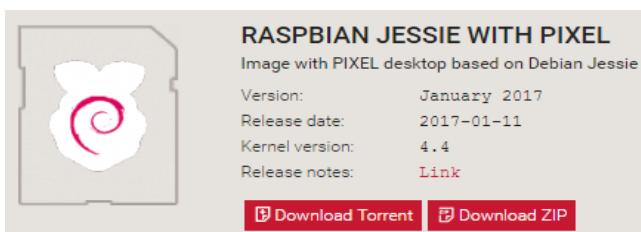


- Le Pi Zéro qui lui présente de moins bonnes caractéristiques (processeur 1 GHz simple core ARM11 accompagné de 512Mo de RAM pas de Wifi ni de Bluetooth) mais ses atouts sont ailleurs. Ses dimensions sont très inférieures et surtout son prix : 5\$, lui permettant de correspondre à pas mal de projets embarqués.

Le prix étant son principal avantage, il faut penser qu'il vous faudra rajouter tout ce qui se place autour et surtout une carte micro-SD qui embarquera le système d'exploitation (un windows pour Raspberry). L'avantage de toutes ces cartes est également la présence de ports GPIO (pins) nous permettant d'interagir avec des capteurs, des relais, ... Seul point noir, aucun de ces GPIO n'est analogique. Passons donc à la préparation de ce fameux système d'exploitation, noté OS, le plus courant étant appelé Raspbian doté du bureau Pixel.

Téléchargement de l'image :

1. Le fichier image ISO:



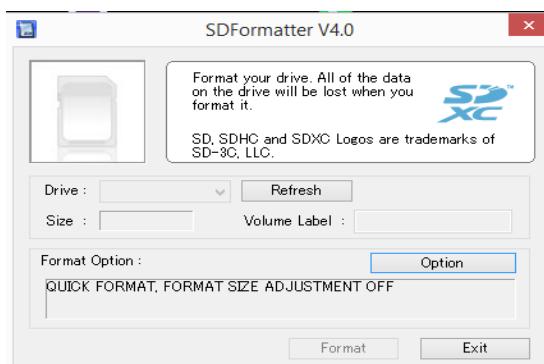
Il vous faut tout d'abord télécharger l'image de l'OS qui ira sur la carte SD à l'adresse :

<https://www.raspberrypi.org/downloads/raspbian/>

Pour se simplifier la vie, cliquez sur *Download ZIP*, attendez que l'image soit rapatriée sur votre PC.

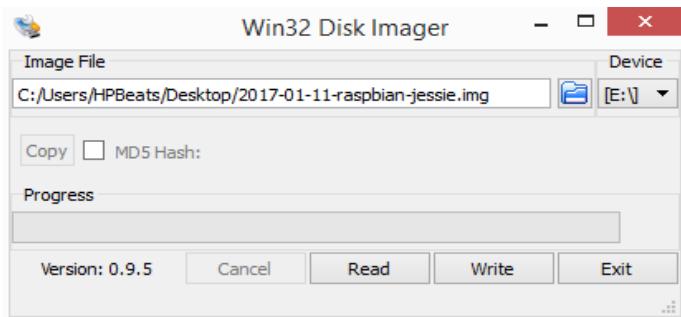
2. Les logiciels sur Windows :

Pendant que votre image se télécharge, vous avez le temps d'aller télécharger, à l'aide d'une recherche google, les deux logiciels utiles pour la suite de la création : *SDFormatter 4* et *Win32DiskImager*.

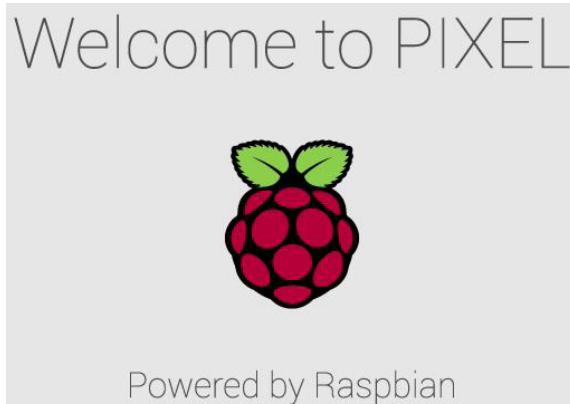


Insérer la carte SD dans votre ordinateur et lancer SDFormatter pour l'effacer correctement. Choisissez la carte dans *Drive*, vérifiez les options de formatage puis cliquez sur *Format*.

Une fois l'image téléchargée, décompressée la et rendez-vous dans Win32DiskImager, sélectionnez l'image puis la lettre correspondant à la carte SD. Il ne vous reste plus qu'à cliquer sur Write pour que l'image soit clonée sur votre carte. Soyez patient et sans 5 min tout sera prêt.

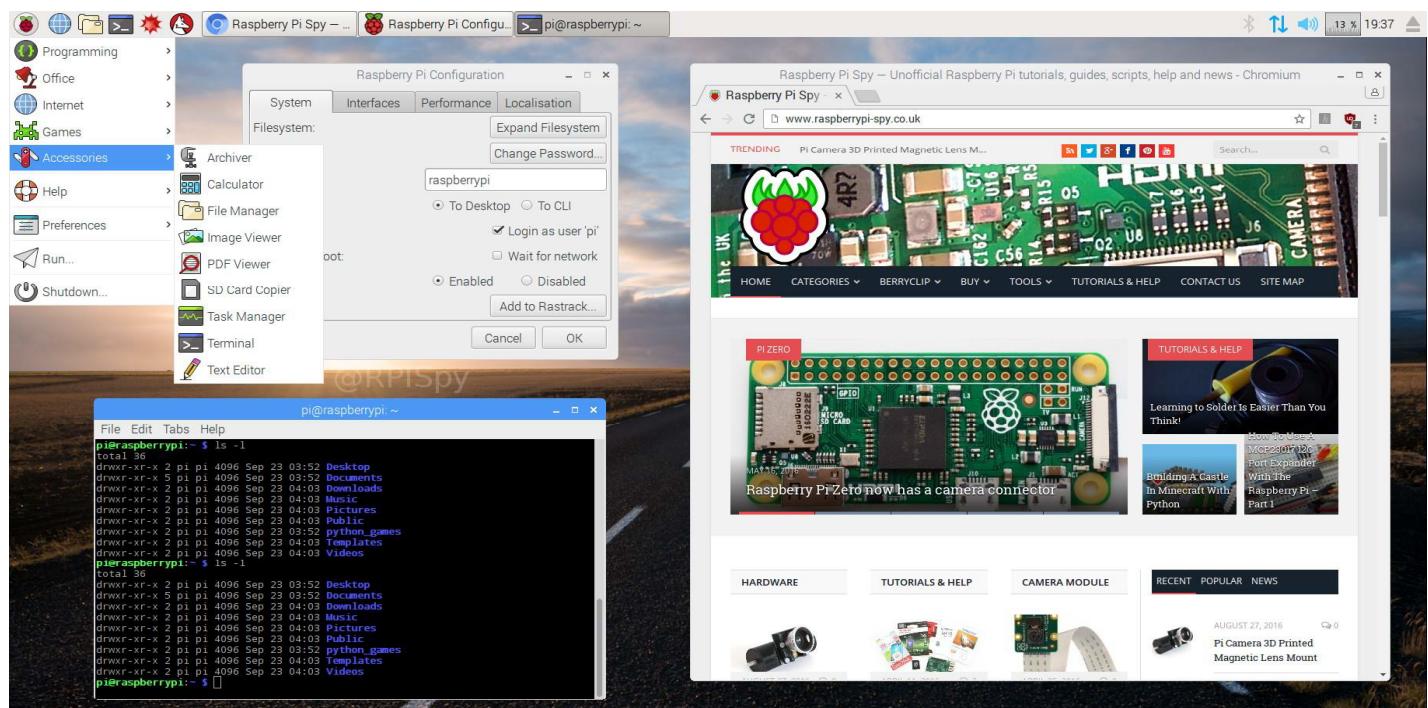


Votre carte SD est à présent prête, vous pouvez l'insérer dans votre Raspberry, le brancher à son alimentation électrique et laissez le système s'amorcer.



Présentation de Raspbian sous pixel :

Raspbian est un dérivé de la distribution Linux Debian, même si cette distribution ne vous est pas familière, vous allez voir par la suite qu'elle est très facile à prendre en main et que vous ne serez que très peu chamboulé par rapport à l'utilisation d'un ordinateur sous Windows.

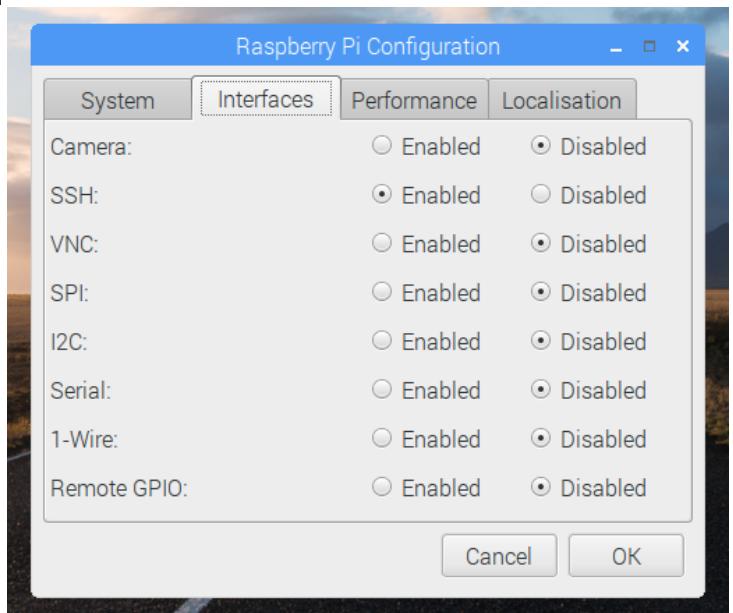


Le bureau est très classique, vous trouverez le menu principal en haut à gauche avec tous les programmes nécessaires à la vie de tous les jours : LibreOffice, Navigateur internet, Python, visualiseur de PDF, ... Mais ce qui nous intéressera le plus lorsque l'on voudra pousser l'utilisation de Raspbian est le Terminal.



Avant de se lancer dans la découverte de la distribution, il est nécessaire de la paramétrer pour l'utilisation que nous en aurons par la suite, pour cela rendez-vous dans le menu Framboise puis dans Préférences et enfin Raspberry Pi Configuration.

Rq : Vous trouverez beaucoup de tutos vous demandant de faire cela en ligne de commande mais ceux-ci sont anciens.



Les deux onglets qui nous concerteront principalement sont *Interfaces* et *Localisation*.

Les interfaces nous permettrons de communiquer avec ou depuis le Raspberry.

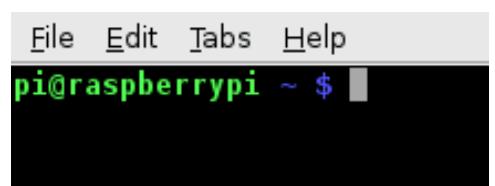
- Le SSH nous permettra de nous connecter au RPi et de lui envoyer des commandes.
- VNC nous permettra de « déporter » l'écran du RPi sur un autre ordinateur.
- 1-Wire nous permettra de communiquer avec des capteurs numériques connectés aux pins du RPi.

Maintenant vous pouvez également vous rendre dans l'onglet Localisation afin d'indiquer que vous êtes en France et que vous possédez un clavier AZERTY. Une fois le paramétrage fini, vous cliquez sur Ok, et la carte vous demandera de redémarrer pour appliquer les changements.

Nous voilà prêt et nous allons pouvoir passer au côté « Nerd » de l'utilisation du RPi pour le mettre à jour et installer des logiciels.

Pour cela rendez-vous dans le menu à la framboise puis dans Accessoires et choisissez enfin Terminal

Vous pouvez également cliquer directement sur l'icône présente dans la barre supérieure :



Une fenêtre toute noire (voir ci-contre) apparaît alors sur votre écran.

Nous y taperons directement les commandes Linux mais aussi pas mal de programmes en Python.

Le terminal accepte beaucoup de commandes mais afin d'éviter les mauvaises manipulations, certaines vous seront interdites car réservée à l'administrateur. Pour devenir administrateur, rien de plus simple, il suffit de commencer sa ligne de commande par « sudo » qui est le condensé de « Super Utilisateur Do ».

Commençons par mettre à jour la liste des programmes installables à l'instant t sur notre Raspberry, pour cela tapez : `sudo apt-get update` puis une fois la liste mise à jour, vous pourrez taper `sudo apt-get upgrade` afin que le Rpi mette à jour les logiciels (appelés paquets) déjà présents sur votre carte SD.

Installons maintenant un logiciel. Si par exemple vous souhaitez installer l'excellent logiciel vlc pour lire des vidéos, il faudra l'installer en qualité de super utilisateur. Pour cela tapez : `sudo apt-get install vlc`

Vous verrez alors apparaître :

```
pi@raspberrypi ~ $ sudo apt-get install vlc
[sudo] password for initiald:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libcddeb2 libdvbpsi6 libebml3 libiso9660-7 libmatroska3 libSDL-image1.2 libtar libupnp3 libva-x11-1 libvcdinfo0
  libvlc5 libvlccore4 libx264-106 libxcb-keysyms1 libxcb-randr0 libxcb-xv0 vlc-data vlc-nox vlc-plugin-notify
  vlc-plugin-pulse
Suggested packages:
  mozilla-plugin-vlc videolan-doc
The following NEW packages will be installed:
  libcddeb2 libdvbpsi6 libebml3 libiso9660-7 libmatroska3 libSDL-image1.2 libtar libupnp3 libva-x11-1 libvcdinfo0
  libvlc5 libvlccore4 libx264-106 libxcb-keysyms1 libxcb-randr0 libxcb-xv0 vlc vlc-data vlc-nox vlc-plugin-notify
  vlc-plugin-pulse
0 upgraded, 21 newly installed, 0 to remove and 1 not upgraded.
Need to get 13.3 MB of archives.
After this operation, 42.2 MB of additional disk space will be used.
Do you want to continue [Y/n]? █
```

Vous devrez alors valider l'installation du logiciel et des paquets nécessaires en appuyant sur la touche Y si vous êtes resté en Anglais ou O si vous avez bien fait le changement de localisation. Passons à présent à la partie qui nous intéressera le plus pour la suite est l'utilisation des ports GPIO.

G P I O :

Il faut installer la librairie WiringPi : `sudo apt-get install wiringpi` puis vérifiez en tapant `gpio readall`

BCM	wPi	Name	Mode	V	Pi 3			Mode	Name	wPi	BCM
					Physical	V					
		3.3v			1	2			5v		
2	8	SDA.1	IN	1	3	4			5V		
3	9	SCL.1	IN	1	5	6			0v		
4	7	GPIO. 7	IN	1	7	8	0	IN	TxD	15	14
		0v			9	10	1	IN	RxD	16	15
17	0	GPIO. 0	IN	0	11	12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13	14			0v		
22	3	GPIO. 3	IN	0	15	16	0	IN	GPIO. 4	4	23
		3.3v			17	18	0	IN	GPIO. 5	5	24
10	12	MOSI	IN	0	19	20			0v		
9	13	MISO	IN	0	21	22	0	IN	GPIO. 6	6	25
11	14	SCLK	IN	0	23	24	1	IN	CE0	10	8
		0v			25	26	1	IN	CE1	11	7
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31	1
5	21	GPIO.21	IN	1	29	30			0v		
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26	12
13	23	GPIO.23	IN	0	33	34			0v		
19	24	GPIO.24	IN	0	35	36	0	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37	38	0	IN	GPIO.28	28	20
		0v			39	40	0	IN	GPIO.29	29	21

La colonne qui nous intéressera dans un premier temps est la BCM, elle suit la numérotation en rouge si on regarde directement le Raspberry Pi 3 par-dessus :

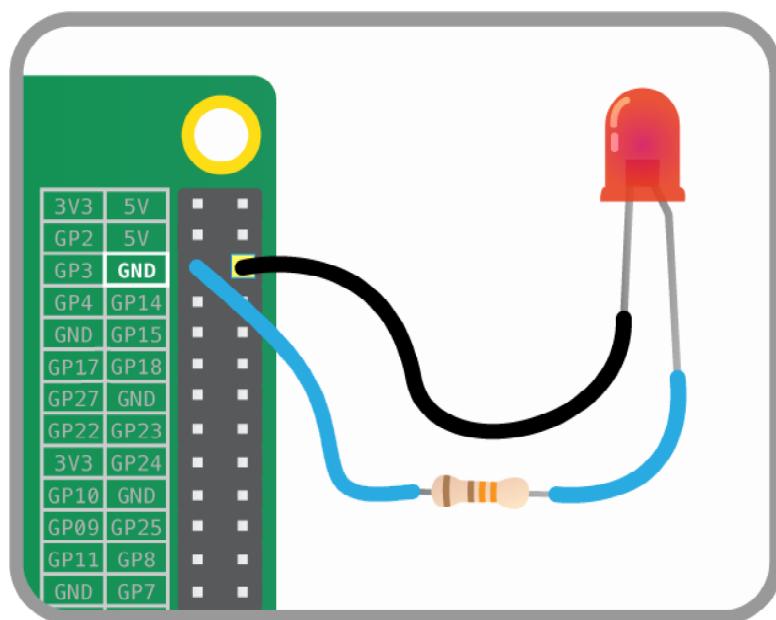


(En noir la numérotation GPIO.BOARD. En rouge, la numérotation GPIO.BCM)

Dans les exemples qui suivent, nous travaillerons avec le port GPIO.BCM numéro 3 qui est noté 5 en numérotation Physique. Si l'on souhaite utiliser ce port GPIO comme sortie, il va falloir le signaler au terminal, pour cela on saisira : `gpio export 3 out`

Maintenant si l'on veut « allumer » cette sortie, il va falloir taper : `gpio -g write 3 1`

En revanche si l'on souhaite « l'éteindre », vous taperez : `gpio -g write 3 0`



Passons à la pratique, réalisez le montage ci-contre, en vous rappelant que sur une LED, la plus longue « patte » correspond au pôle plus, ici le GPIO.BCM 3 alors que nous connecterons le pôle moins à une masse notée GND pour ground.

En exécutant les commandes vues précédemment, vous pourrez allumer et éteindre la Led.

Toutes les pins ne peuvent être utilisées comme entrée ou sortie, il existe des alimentations permanentes en 3,3V ou en 5V, des masses, ... Le tableau de la page suivante récapitule les différents ports utilisables et leur nom. La numérotation BCM est celle indiquée après le mot GPIO. Si celui-ci n'est pas un GPIO, il ne sera alors pas utilisable grâce à la librairie WiringPi.

Nous verrons dans une prochaine fiche comment piloter ces ports grâce au langage Python et donc directement dans nos programmes pour déclencher des évènements ou bien lire des capteurs.

Raspberry Pi 3 GPIO Header

<i>Pin#</i>	<i>NAME</i>		<i>NAME</i>	<i>Pin#</i>
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I ² C)		DC Power 5v	04
05	GPIO03 (SCL1 , I ² C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I ² C ID EEPROM)		(I ² C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

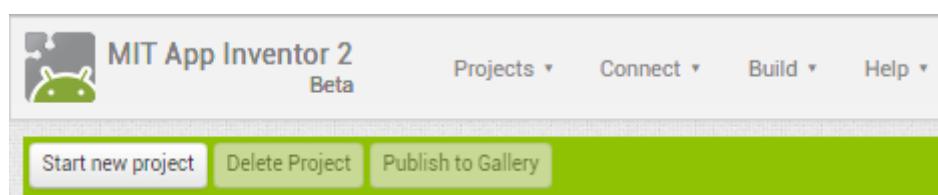


App Inventor 2 est un logiciel en ligne permettant de réaliser des applications Android assez rudimentaires mais souvent très pratiques pour qui ne souhaite pas se plonger dans l'apprentissage complexe du langage Android.

Pour les habitués de la programmation graphique sur le logiciel Scratch, vous ne serez pas déstabilisés. En revanche pour les autres, il faudra un moment d'adaptation pour trouver les outils et les fonctions.

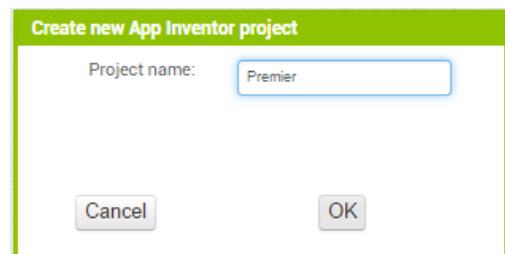
Tout d'abord pour commencer il vous faudra un compte google (gmail) pour pouvoir vous connecter et réaliser votre première application. A présent rendez-vous sur la page d'identification :

ai2.appinventor.mit.edu

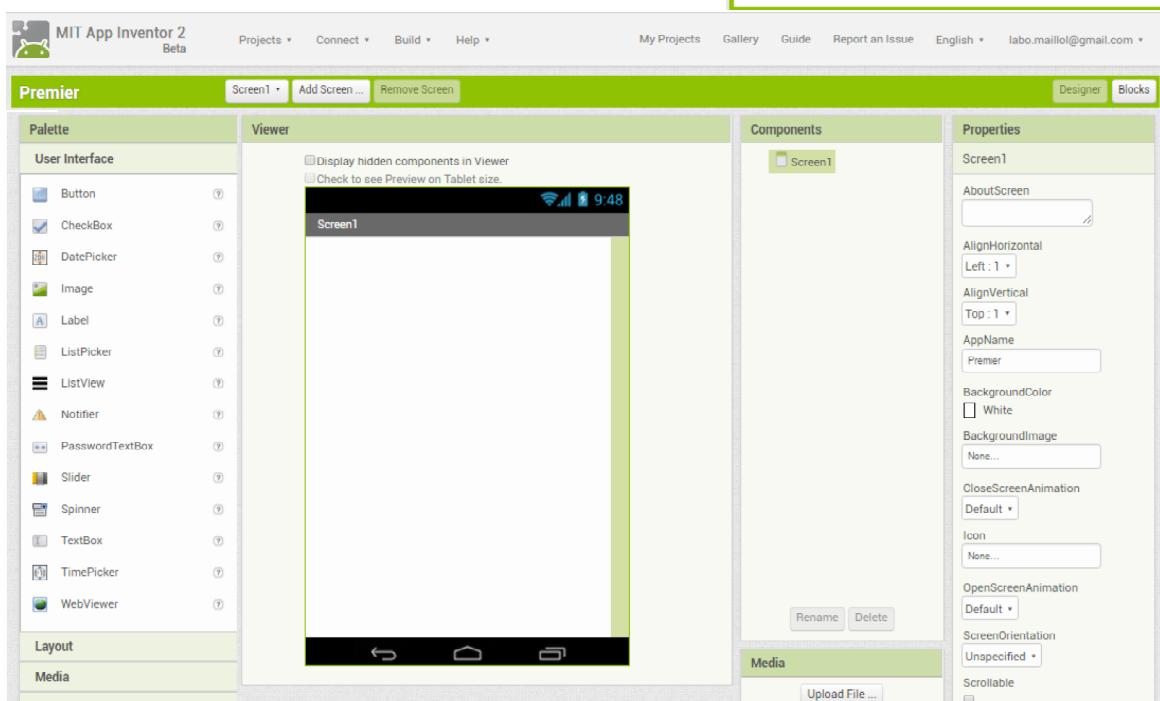


Une fois identifié, vous voilà sur la page principale qui vous permettra de démarrer un nouveau projet :

[Start new project](#)



On nous demande alors de donner un nom à notre projet avant d'arriver à la page principale dite « Designer »



Sur la gauche on trouve l'ensemble des outils qui pourront être utilisés dans notre application. On choisit, un de ces outils, on le fait glisser sur la fenêtre du milieu représentant notre téléphone puis on trouve toutes les options et les paramétrages sur la droite.



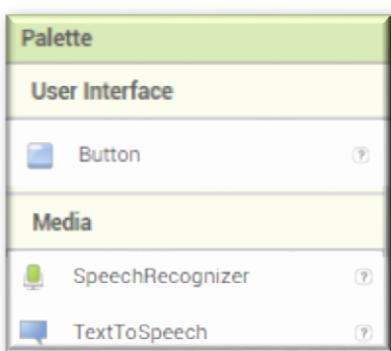
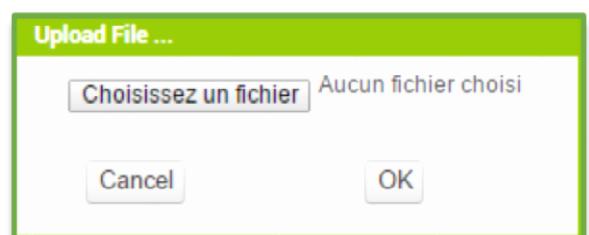
Commençons par un petit exemple : **Le perroquet**

On va commencer par changer le fond d'écran, pour cela on peut regarder dans le menu Propriétés de droite et chercher la zone *BackgroundImage* (si comme moi vous êtes resté en Anglais).

Cliquez alors sur *None...* puis sur *Upload File*. Une nouvelle fenêtre s'ouvre et n'attends qu'une chose, que vous cliquez sur *Choisissez un fichier*. Sélectionnez le dans votre ordinateur puis cliquez sur *Ok* et là quelques secondes de patience avant de voir votre perroquet apparaître au milieu de votre écran de téléphone virtuel.



Upload File ...



Placez-vous à présent dans la partie Palette et insérez les outils ci-contre.

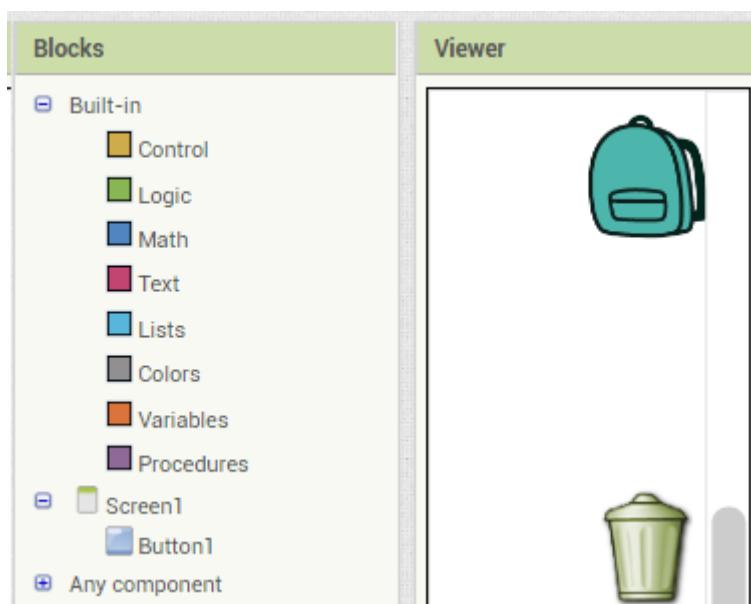
Occupons-nous dans un premier temps des Propriétés du Bouton.

BackgroundColor <input type="color"/>	Change la couleur du fond du bouton.	Image <input type="file"/>	Image de fond du bouton.
Enabled <input checked="" type="checkbox"/>	Le bouton sera actif si cette case est cochée.	Shape <input type="button"/>	Style du bouton.
FontBold <input type="checkbox"/>	La police sera Gras.	ShowFeedback <input checked="" type="checkbox"/>	Renvoie une information si la case est cochée.
FontItalic <input type="checkbox"/>	La police sera Italique.	Text <input type="text"/>	Texte du bouton écrit sur l'écran.
FontSize <input type="text"/>	Taille de la police.	TextAlignment <input type="button"/>	Alignement du texte sur le bouton.
FontTypeface <input type="button"/>	Police de l'écriture.	TextColor <input type="color"/>	Couleur du texte.
Height <input type="text"/>	Hauteur du bouton dans l'écran du téléphone.	Visible <input checked="" type="checkbox"/>	Ce bouton ne se verra que si cette case est cochée.
Width <input type="text"/>	Largeur du bouton dans l'écran du téléphone.		

Ce qu'il ne vous faudra pas changer ici c'est *Enable* pour ne pas désactiver le bouton et surtout *Visible* pour que l'utilisateur puisse le voir.

Dans le coin en haut à droite, on peut voir les deux petits boutons ci-contre. Ils permettent de basculer entre la représentation du téléphone (propriétés...) et la partie « programmation » qui permet de gérer le comportement des outils.

Designer Blocks



En résumé vous trouverez en cliquant sur Blocks, la possibilité de faire agir ou même interagir les différents composants que vous aurez intégré dans le Designer. La colonne Blocks, vous permet de récupérer les blocs classiques de programmation dans la partie Built-in, vous y trouverez les déclarations de variables, les boucles, les calculs... Les blocs spécifiques aux composants ajoutés, se trouvent directement en cliquant sur le composant.

Pour ce qui est de la poubelle, la signification est assez claire même si vous pouvez supprimer un bloc sélectionné en appuyant tout simplement sur la touche *Suppr* de votre clavier.

En revanche, l'utilisation du sac à dos est plus particulière, il s'agit en réalité du système copier-coller. Vous placez le bloc ou l'ensemble de blocs sur le sac à dos et vous pourrez alors y avoir accès dans une autre fenêtre de votre application. Vous pouvez également réaliser un clic droit sur l'objet de votre choix pour le dupliquer ou l'ajouter au *backpack* (sac à dos).

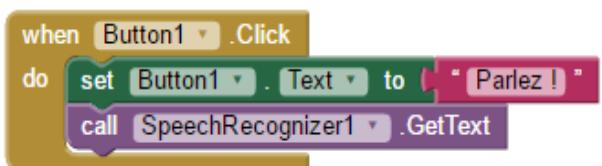
Maintenant que les présentations sont faites, faisons notre première application. Restez dans la partie Blocks et tentez de reproduire les blocs ci-dessous.

Ce bloc est la suite d'instructions qui sera réalisée lorsque

l'on appuiera sur le Bouton1 :

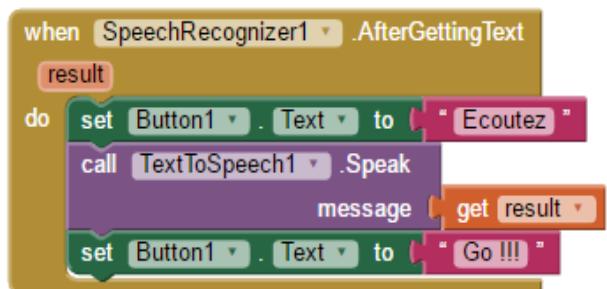
Le texte du bouton changera et deviendra « Parlez ! ».

Ensuite le téléphone appellera la fonction reconnaissance vocale d'Android, qui attendra que vous ayez énoncé l'intégralité de votre phrase.



Ce bloc à présent représente ce que le téléphone réalisera lorsque la reconnaissance vocale a détecté un texte.

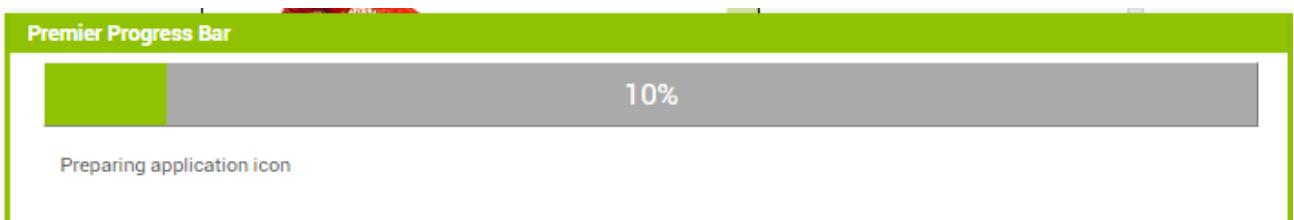
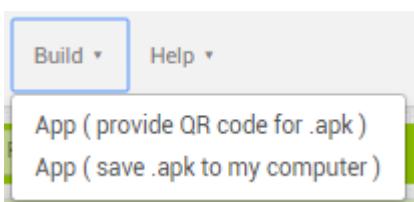
Il changera le texte du bouton par « Ecoutez » puis appellera la synthèse vocale d'Android pour lire le résultat obtenu. Une fois la lecture du texte finie, le texte est à nouveau modifié pour retrouver sa valeur initiale.



Notre première application étant finie il va être nécessaire de la « compiler », c'est-à-dire de la transformer en un langage compréhensible de notre téléphone. Pour cela, il nous suffira de cliquer sur le bouton Build

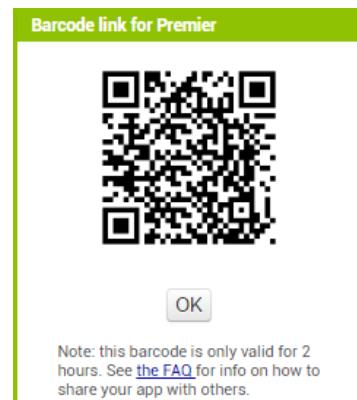
puis de faire le choix entre un QR code que vous lirez directement sur votre écran à l'aide d'une application installée sur votre téléphone ou un téléchargement direct de l'application sur votre ordinateur au format apk (format classique des applications Android).

Une fois votre choix fait, vous verrez pendant quelques secondes une barre de progression avancée sur votre écran.



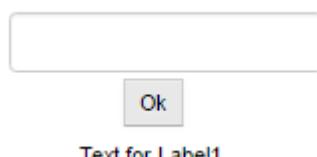
Celle-ci représente l'évolution de la compilation de votre projet et lorsque tout est terminé soit vous verrez votre fichier apk dans le dossier Téléchargements de votre ordinateur, soit une fenêtre comme ci-contre s'affichera et vous n'aurez plus qu'à scanner pour que l'application arrive directement sur votre téléphone.

(L'application MIT AI2 Companion sera peut-être nécessaire pour scanner ce code)

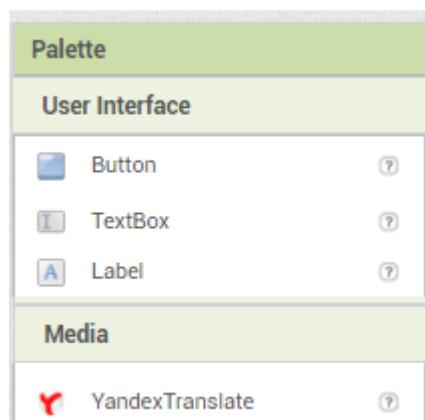


Continuons par un second petit exemple : **Le traducteur**

Comme pour l'application précédente, rendez-vous dans le Designer et insérez les composants énoncés ci-contre.



L'objectif va être de saisir un mot ou une phrase dans la TextBox et lorsque l'on appuiera sur le bouton, cette dernière sera traduite dans la langue choisie.



Lorsque l'on cliquera sur le bouton 1 (Ok), une requête sera envoyée au site Yandex (une connexion est donc nécessaire). On insère alors un texte vierge dans la partie « Langage vers lequel on veut traduire » puis on renseigne le code correspondant à la langue d'après le tableau suivant.

"ar" pour la langue : "Arabe"	"he" pour la langue : "Hébreu"	"no" pour la langue : "Norvégien"
"az" pour la langue : "Azerbaïjan"	"hr" pour la langue : "Croate"	"pl" pour la langue : "Polonais"
"be" pour la langue : "Biélorusse"	"hu" pour la langue : "Hongrois"	"pt" pour la langue : "Portugais"
"bg" pour la langue : "Bulgare"	"hy" pour la langue : "Arménien"	"ro" pour la langue : "Roumain"
"bs" pour la langue : "Bosniaque"	"id" pour la langue : "Indonésien"	"ru" pour la langue : "Russe"
"ca" pour la langue : "Catalan"	"is" pour la langue : "Islandais"	"sk" pour la langue : "Slovaque"
"cs" pour la langue : "Tchèque"	"it" pour la langue : "Italien"	"sl" pour la langue : "Slovénie"
"da" pour la langue : "Danois"	"ka" pour la langue : "Georgien"	"sq" pour la langue : "Albanien"
"de" pour la langue : "Allemand"	"lt" pour la langue : "Lithuanien"	"sr" pour la langue : "Serbe"
"el" pour la langue : "Grec"	"lv" pour la langue : "Letton"	"sv" pour la langue : "Suédois"
"en" pour la langue : "Anglais"	"mk" pour la langue : "Macédonien"	"tr" pour la langue : "Turc"
"es" pour la langue : "Espagnol"	"ms" pour la langue : "Malaisien"	"uk" pour la langue : "Ukrainien"
"et" pour la langue : "Estonien"	"mt" pour la langue : "Maltais"	"vi" pour la langue : "Vietnamien"
"fi" pour la langue : "Finlandais"	"nl" pour la langue : "Néerlandais"	"zh" pour la langue : "Chinois"
"fr" pour la langue : "Français"		



Une fois que le site Yandex a reçu et traité la requête, il renvoie une traduction. Nous remplacerons alors le texte présent dans le Label1 par la traduction reçue en cliquant sur le bouton **traduction**

D E F I S

Rajouter en bas de l'application perroquet un Label qui affiche le texte reconnu par Google.

Améliorer le traducteur en permettant de lui parler et qu'il vous lise la traduction en Anglais

Améliorer encore cette dernière application en permettant de traduire le français dans la langue choisie par l'utilisateur entre l'anglais, l'espagnol et catalan.

