**Lycée Raymond Queneau**

18

**Projet BTS Systèmes Numériques : Option EC**

**La maison du futur, connectée...**

**Dufresne Nicolas**

Sommaire

[Mon travail à réaliser dans ce projet 3](#_Toc514858182)

[Matériels mis à disposition 4](#_Toc514858183)

[Détails sur les matériels utilisés 4](#_Toc514858184)

[Les modules Bluetooth 4](#_Toc514858185)

[Première solution 5](#_Toc514858186)

[Deuxième solution 5](#_Toc514858187)

[Les commandes AT 6](#_Toc514858188)

[Mise en place des modules Bluetooth 8](#_Toc514858189)

[Tablette Samsung GALAXY Tab Active 10](#_Toc514858190)

[Caractéristiques : 10](#_Toc514858191)

[Mit app Inventor 11](#_Toc514858192)

[Schéma du montage sous Fritzing 11](#_Toc514858193)

[Développement de l'application pour Smartphone / tablette 12](#_Toc514858194)

[Croquis de l'application 12](#_Toc514858195)

[Prise en main de l'application Web Mit App inventor 2 14](#_Toc514858196)

[Partie design 14](#_Toc514858197)

[Partie blocks 21](#_Toc514858198)

[Les blocs associés 24](#_Toc514858199)

[Bouton 24](#_Toc514858200)

[ListPicker Bluetooth 25](#_Toc514858201)

[Notifier 27](#_Toc514858202)

[Slider 30](#_Toc514858203)

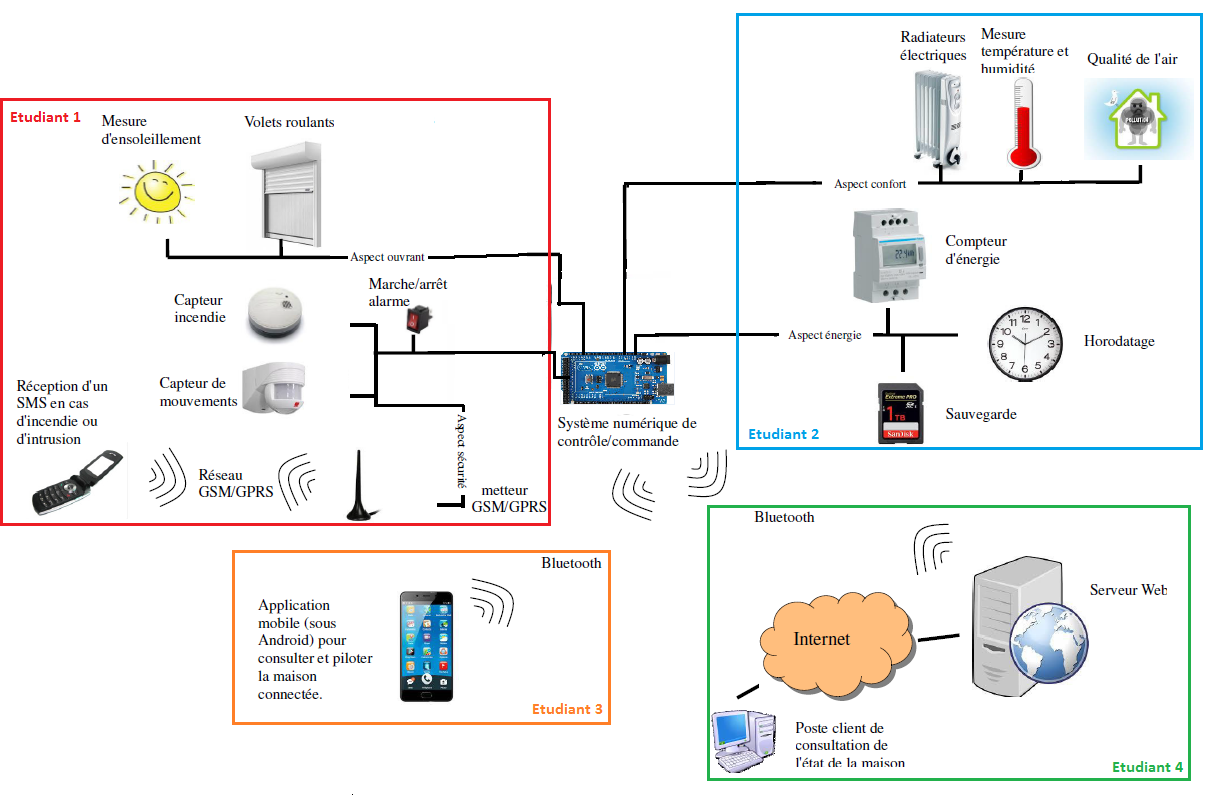
[Récupération d'une trame 32](#_Toc514858204)

[Conclusion 35](#_Toc514858205)

[Annexe 36](#_Toc514858206)

# Mon travail à réaliser dans ce projet

Comme annoncé dans la première partie du rapport sur le projet BTS Systèmes Numériques, le travail est réparti sur 4 étudiants. L'étudiant 1 travaillera avec l’étudiant 2 sur la récupération des données des différents capteurs. L'horodatage et la réception d'un SMS en cas d'incident font également partie de leurs travails. L'étudiant 3 et l'étudiant 4 doivent établir la communication Bluetooth entre la centrale, la tablette ainsi que le serveur Web. Les données de tous les capteurs envoyées par l'étudiant 1 et l'étudiant 2 doivent être récupérées par l'étudiant 3 et l'étudiant 4 par cette communication. Une application permettant l'affichage des données ainsi que la possibilité de gérer les différents appareils connectés de la maison doit être créée par l'étudiant 3. Un site Web où toutes les mesures seront également affichées doit être créé par l'étudiant 4.

Je suis dans ce projet l'étudiant 3, je dois donc concevoir l’application sous Android sur laquelle on pourra gérer et lire des données.

# Matériels mis à disposition

Pour la réalisation de la partie application, nous avons :

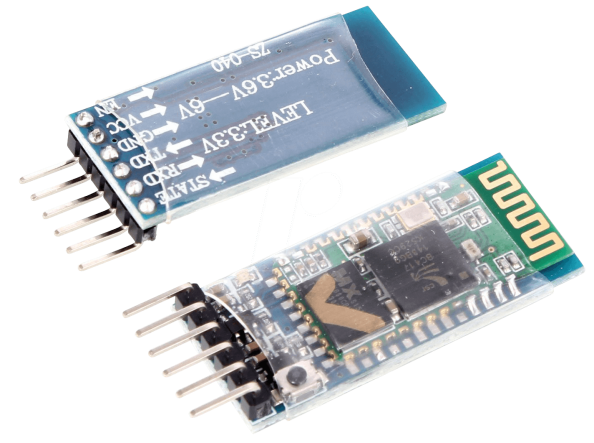
- Un module Bluetooth HC-05.

- Une tablette Samsung GALAXY Tab Active.

- Mit App Inventor 2.

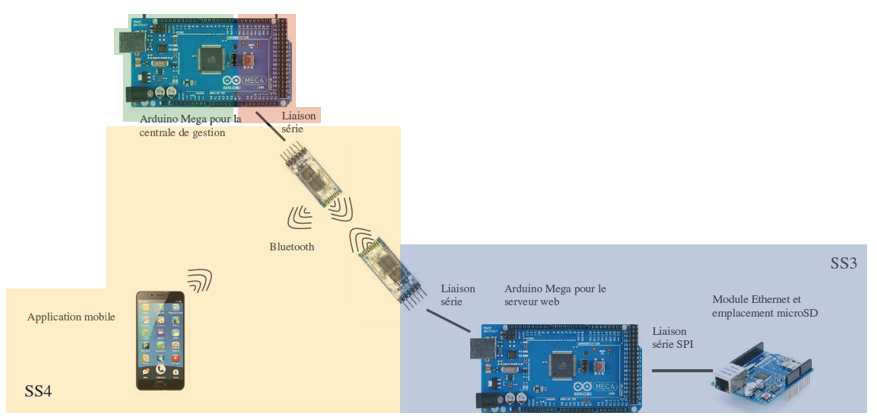
# Détails sur les matériels utilisés

## Les modules Bluetooth

Bluetooth est une norme de communications permettant l'échange bidirectionnel de données à très courte distance en utilisant des ondes radio UHF sur une bande de fréquence de 2,4 GHz. Son objectif est de simplifier les connexions entre les appareils électroniques en supprimant des liaisons filaires.

Nous avons fait l’acquisition de deux modules HC-05 pour notre projet. Nous devions donc configurer les modules pour que la trame envoyée par l’étudiant 1 et l’étudiant 2 (récupération des données des capteurs) soit envoyée d’un module connecté à la centrale de gestion (Arduino Mega 2560), à un autre module sur la partie serveur web de l’étudiant 4 mais également à l’application sur la tablette de l’étudiant 3. Nous appellerons le module connecté à la centrale de gestion « Gestion » et le module connecté au serveur web « Web ». Pour ce faire nous avons réfléchi à plusieurs solutions possibles.

# Première solution

Notre première idée est d’effectué une connexion des modules Bluetooth en nuage de point. Soit d’alterner la connexion du module Gestion, avec le module Web et la tablette. La communication de base entre les modules Bluetooth est une connexion en point par point, ce qui veut dire que qu’un module Bluetooth à possibilité de se connecter seulement à un seul autre module Bluetooth. On parle ici de « maître » et « esclave ». Un maître peut se connecter seulement à un seul esclave.

### 

# Deuxième solution

La deuxième idée est de simplement rajouter un troisième module Bluetooth HC-05 que l’on nommera « Application ». Il nous suffit alors de connecter les modules Gestion et Application à la centrale de gestion. Gestion sera connecté à Web et Application sera connecté à la tablette. Chacun des modules connectés à la centrale de gestion aura donc une connexion point par point à un autre module.  
Pour des raisons de simplicité nous avons décidé avec l’étudiant 4 d’opter pour cette seconde idée. En contrepartie, le coût de notre projet et le nombre d’équipements utilisé augmente. Nous avons donc configuré les modules Gestion et Application en tant que maîtres et le module Web et la tablette en tant qu'esclaves.

# Les commandes AT

Les **Commandes AT**, aussi appelées **Commandes Hayes**, constituent un langage de commandes développé à l'origine pour le modem Hayes Smartmodem 300. Il existe une multitude de commande AT.

Ces commandes permettent par exemple :

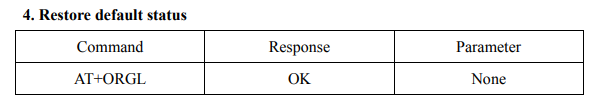
* de composer un numéro de téléphone
* de commander le raccordement du modem à la ligne (l'équivalent de décrocher le téléphone)
* de connaître l'état de la ligne : tonalité d'invitation à transmettre, ligne occupée...
* de spécifier le type de transmission et le protocole de liaison à utiliser
* de régler le volume sonore du haut-parleur interne du modem
* d'envoyer les caractères transmis simultanément vers l'écran
* d'afficher certains renseignements concernant le modem
* de manipuler les registres internes du modem

Source : https://fr.wikipedia.org/wiki/Commandes\_Hayes

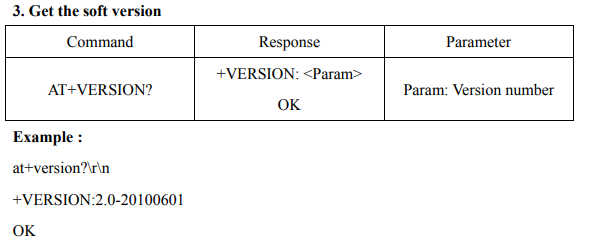
Il y a plusieurs types de commande AT mais chacune comporte le préfixe « AT ».

On peut retrouver :

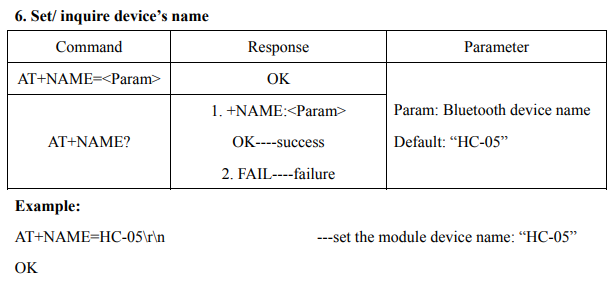
Les commandes qui effectuent une action. Comme dans l'exemple ci-dessous une réinitialisation par défaut de chacune des informations du module.



Les commandes qui nous permettent de recevoir des informations de la part du module. Comme dans l'exemple ci-dessous pour connaître la version du module utilisé.

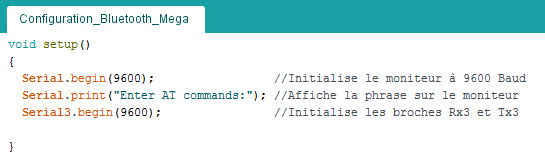


Les commandes qui permettent d’envoyer et de changer les informations concernant le module. Comme dans l'exemple ci-dessous changer le nom du module.



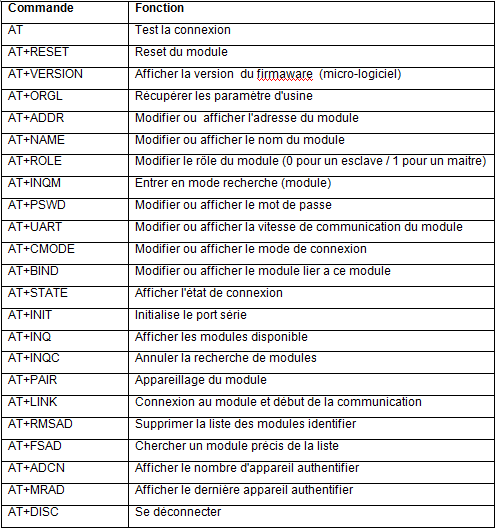
http://www.linotux.ch/arduino/HC-0305\_serial\_module\_AT\_commamd\_set\_201104\_revised.pdf

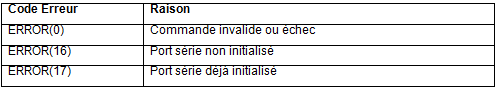
L’utilisation des commandes AT sous Arduino se présentent comme ceux-ci :



### RÃ©sultat de recherche d'images pour "commande at arduino"

# Mise en place des modules Bluetooth

Les commandes que nous avons utilisées pour configurer nos modules sont les suivantes :



### RÃ©sultat de recherche d'images pour "Samsung GALAXY Tab Active"

## Tablette Samsung GALAXY Tab Active

Caractéristiques :

* **Processeur :** **Qualcomm Snapdragon 400 Quad-Core cadencé à 1.2 GHz**
* **Système :** Android 4.4 KitKat
* **Ecran :** 8", résolution 800 x 1280 pixels
* **RAM :** 1.5 Go
* **APN :** **3.15 MP**, flash LED, mise au point automatique, **caméra frontale 1.2 MP**
* **Stockage:** **16 Go**(extensible jusqu'à 64 Go via microSDXC)
* **Connectivité :**BT 4.0, GPS, Wi-Fi 802.11 a/b/g/n, micro-USB, jack 3.5mm
* **Batterie** de 4450 mAh
* **Dimensions :**126.2 x 213.1 x 9.7 mm pour 393 gr

Source : <https://www.ldlc.com/fiche/PB00177233.html>

## Mit app Inventor

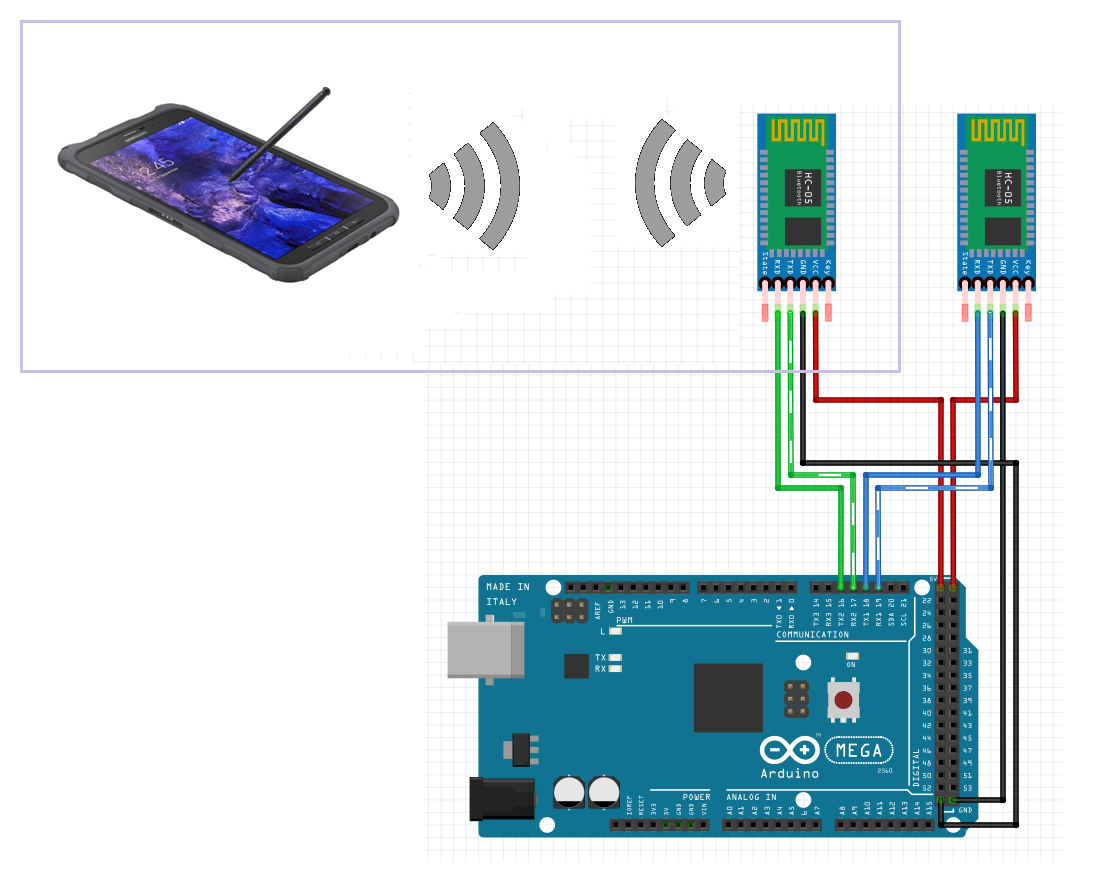
App Inventor pour Android est une application développée par Google. Elle est actuellement entretenue par le MIT.

Le Massachusetts Institute of Technology (MIT) est un institut de recherche et une université américaine, spécialisée

dans les domaines de la science et de la technologie.

Elle simplifie le développement des applications sous **Android** et le rend accessible même pour les novices et ceux qui ne sont pas familiers avec les langages de programmation. Elle est basée sur une interface graphique en bloc.

## Schéma du montage sous Fritzing

Nous nous occuperons uniquement de la partie encadrée.

Nom du module : Web

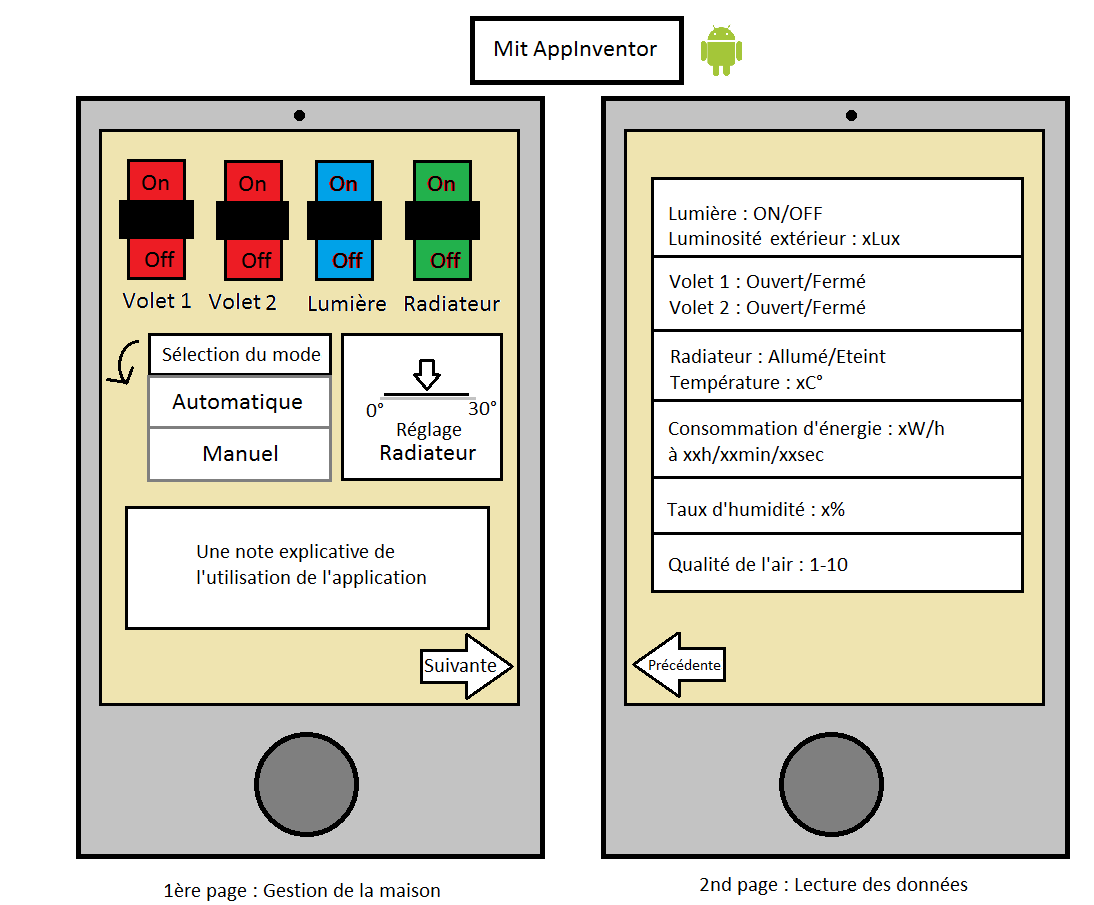
Nom du module : Application

Le module Bluetooth Web est connecté au serveur web géré parl’étudiant 4.

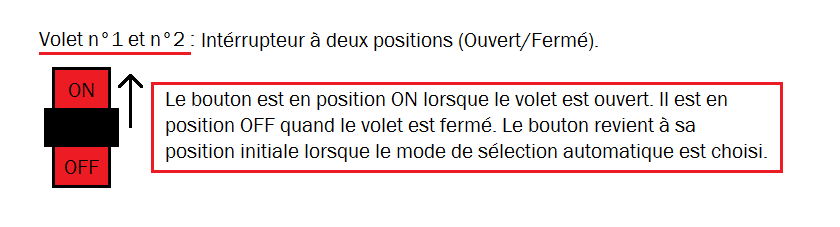
# Développement de l'application pour Smartphone / tablette

## Croquis de l'application

Avant de commencer l'application Android, j'ai dans un premier tant réalisé un croquis rapide sous le logiciel Paint avec la participation des autres étudiants pour répondre à une de nos questions qui était "à quoi notre application doit-elle ressembler ?".

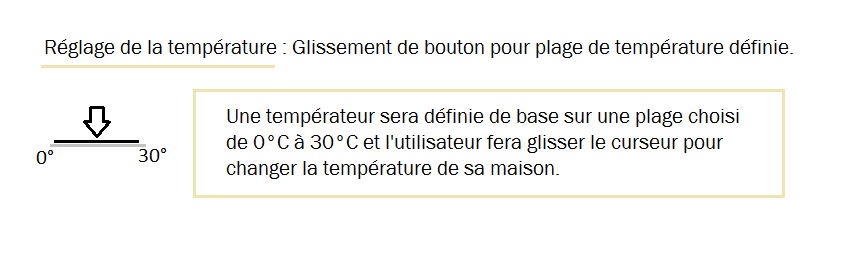
Nous avions donc prévu, en respectant le cahier des charges, de diviser l'application en deux parties. Une partie gestion de la maison où nous pouvons manipuler les différents aspects de notre maison connectée et une partie lectures des données où on regroupe toutes les informations récupérées des différents capteurs présent à l'intérieur et à l'extérieur de notre bâtiment.

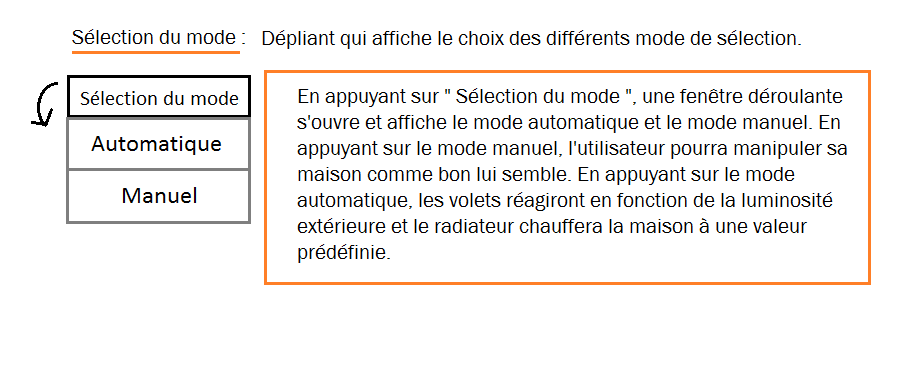
On peut retrouver sur notre première page la configuration qui suit :



## 

## 





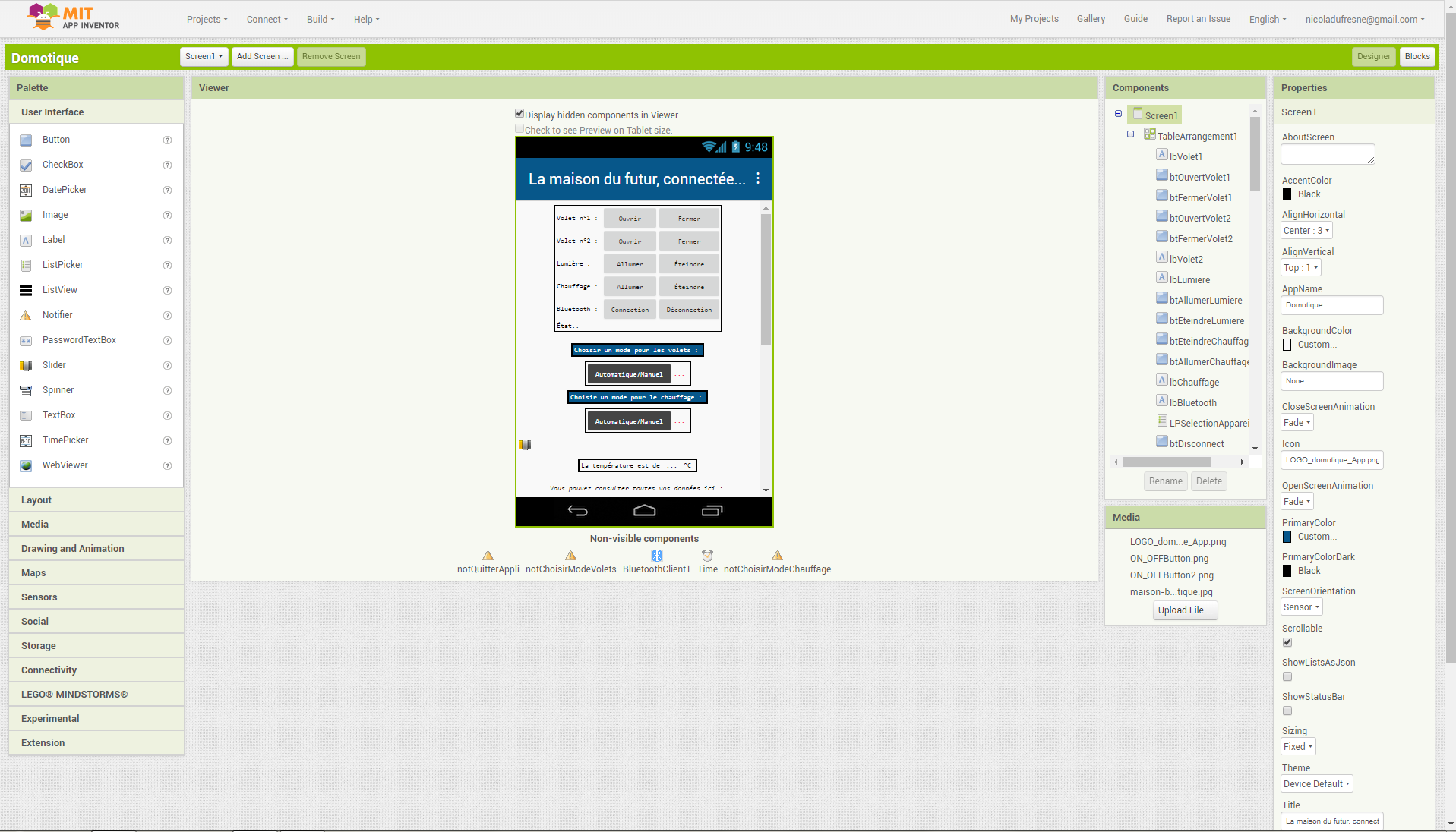
## Prise en main de l'application Web Mit App inventor 2

Mit App Inventor 2 possède deux interfaces uniques différentes. D'un coté une partie **design** sur laquelle on s'occupera du visuelle de l'application. De l'autre, une partie **block** qui complètera les actionneurs de la partie design.

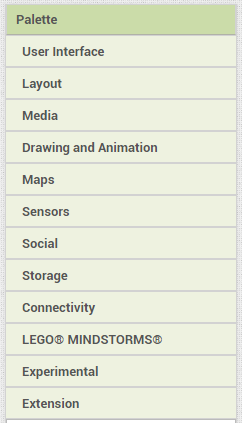
### Partie design

La partie design est accessible en cliquant sur le bouton en haut à gauche de la page :

Mit App Inventor 2 se présente de la manière suivante :



L'application se présente de façon simple et ordonnée. Dans un premier temps il nous faut associer un compte Google à Mit App Inventor qui permettra la récupération de ses données sur n'importe quelle autre support informatique. Par la suite, il est demandé de créer un projet que l'on nommera " Domotique ". Une page apparaît avec plusieurs encadrés disponible.

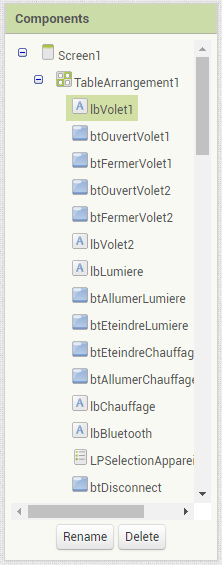
La **palette** situé sur le coté gauche regroupe l’ensemble des composants supportés par App Inventor 2.

Ils se divisent en plusieurs catégories regroupant les composants relatifs :

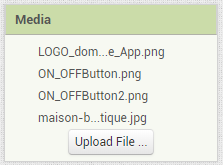
* à l’interface utilisateur (bouton, case à cocher, image),
* à la disposition des blocs (c’est à dire l’affichage, le design, de votre application),
* aux médias (tout ce qui est relatif aux médias traditionnels : sons, vidéos, appareil photo),
* au dessin et à l’animation (tout ce qui va afficher des animations à l’écran tel que des GIF animés, des mouvements, d’images),
* aux capteurs (les composants techniques propres aux smartphones tels que l’orientation du téléphone, le temps qui passe...),
* aux interactions sociales (tout ce qui concerne la communication, comme les réseaux sociaux, les appels téléphoniques...),
* au stockage (tout ce qui est lié à la base de données),
* à la connectivité (communication via les réseaux tels que le Bluetooth, le Web),
* à LEGO MINDSTORMS (robotique liée à des produits spécifiques de la marque LEGO),
* aux fonctionnalités expérimentales (nouvelles fonctionnalités testées par AI 2, il s’agit de Firebase),
* aux extensions (cela concerne tous les développements spécifiques à AI 2).

La partie graphique de l'application s’effectue dans la zone **Viewer**.

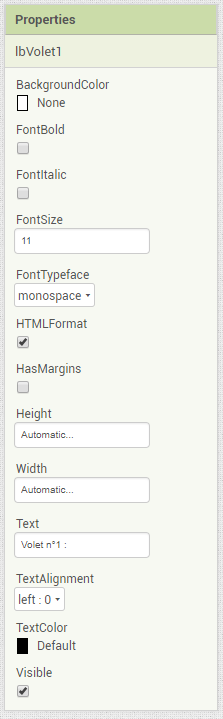
Afin d'appliquer un composant de la tablette à l'interface graphique, il suffit simplement de glisser l'objet en question, sur la partie **Viewer**.

La partie **Components** affiche tous les composants glissé dans la partie **Viewer** ce qui permet de manipuler, de renommer ou de rechercher rapidement un objet.

La partie **Media** permet de télécharger des images et de les incruster dans la partie **Viewer**. Les dernières images y sont également stocké.

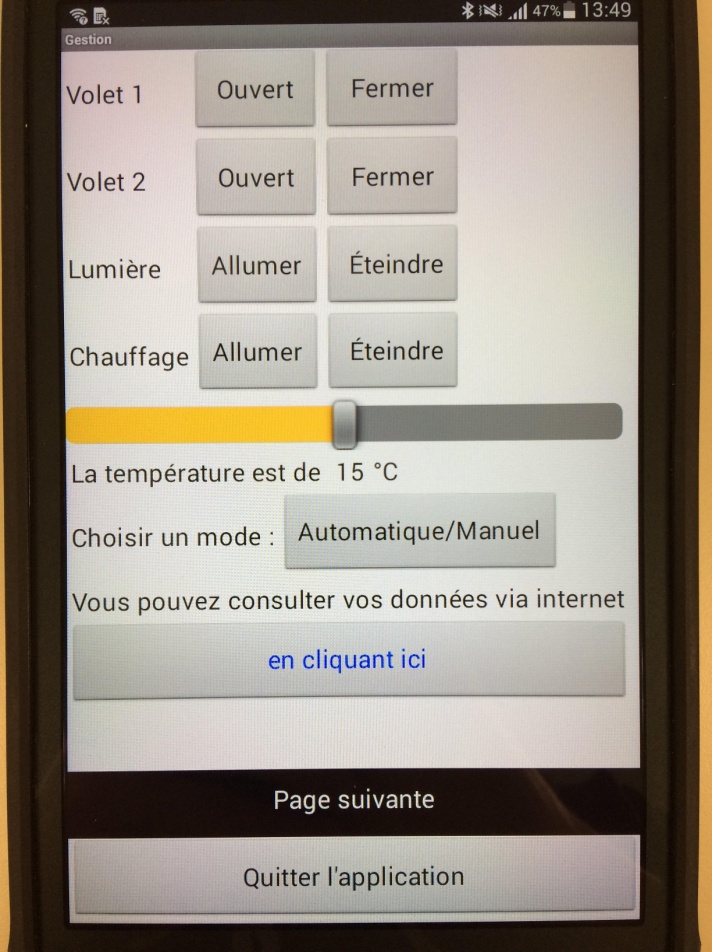
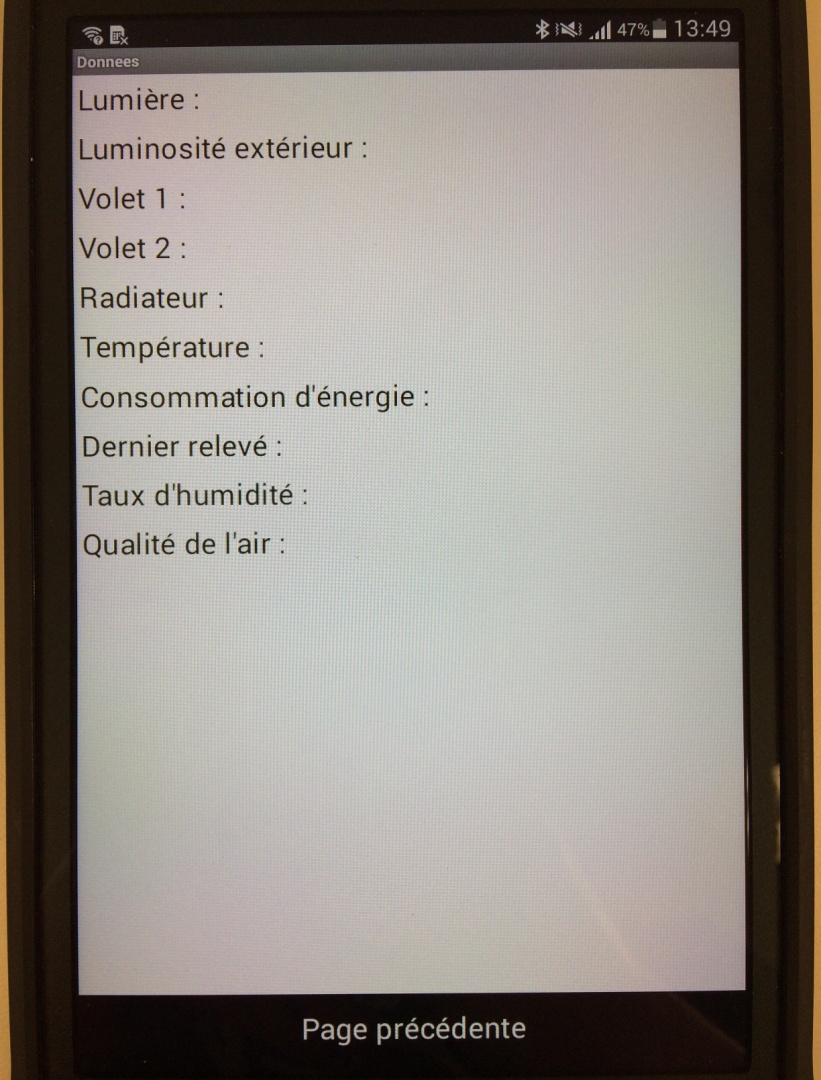


Lorsque l'on sélectionne dans la partie **Components** un élément au choix, ses propriétés apparaissent dans la colonne **Properties** visible sur la droite. On peut ensuite modifier comme bon nous semble un composant.



Maintenant que nous avons fait le tour de **Mit App Inventor 2**, nous allons essayer de représenter notre croquis sur la partie **Viewer**.

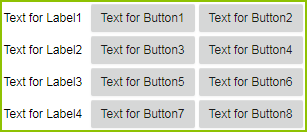
Le résultat est le suivant :

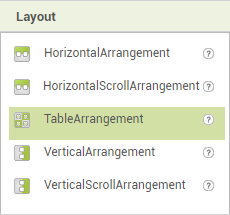


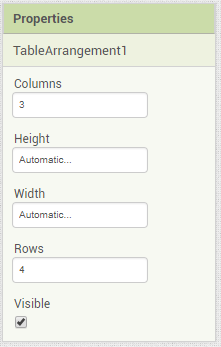
On retrouve par rapport au croquis, la première page **Gestion** qui comporte non pas 4 mais 8 actionneurs regroupant les voles 1 et 2 , la lumière et le chauffage. Il y a aussi un potentiomètre numérique pour la température, un bouton pour le choix du mode et un bouton facultatif qui amène sur le site web de l'étudiant 3. On y retrouve aussi la seconde page **Données** avec chacune des valeurs que l'on doit récupérer.

On peut observer sur ces deux pages :

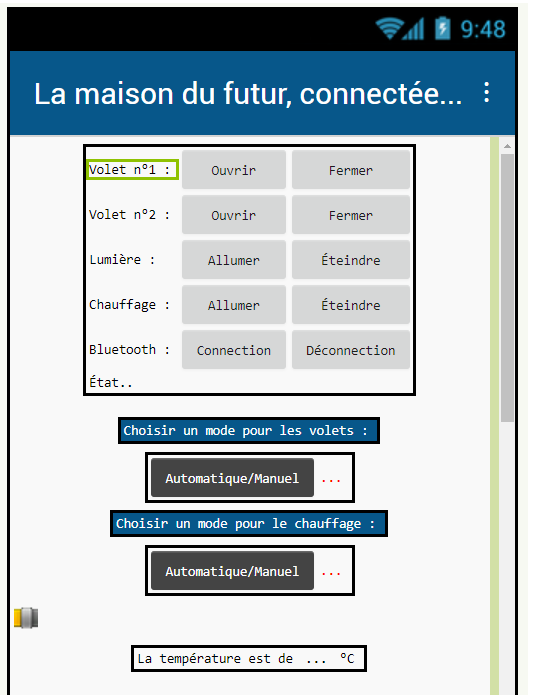
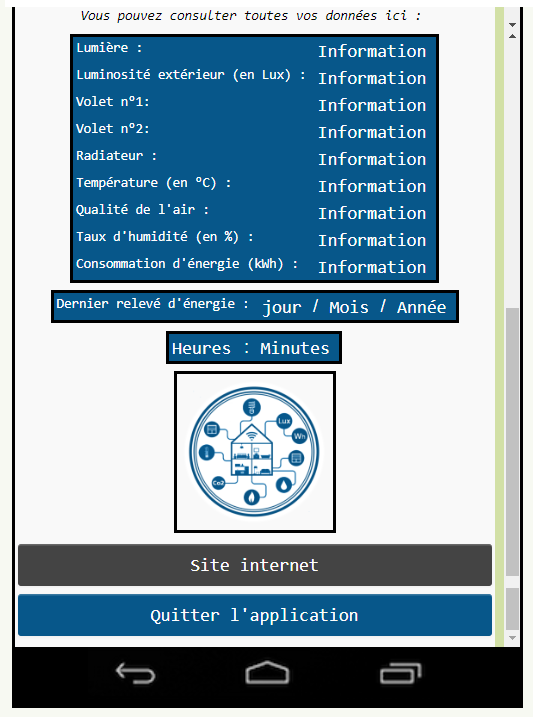
* des boutons (**button**) : 
* un potentiomètre numérique (**slider**) : 
* des zones de textes (**label**) : 
* un page internet (**Webviewer**) : 
* un selectionneur de liste (**ListPicker**) : 
* une notification (**notifier**) : 

Mit App Inventor 2 ne permet pas une disposition libre de ses composants, nous avons donc à disposition des **Layout**. Chacun des composants sont rangé de manière strict dans une table d'arrangement. On peut modifier sur ses tableaux le nombre de colonnes, de ligne mais aussi la taille horizontal et vertical.





À partir de ce prototype nous allons ensuite améliorer la partie graphique de notre application.



Pour des raisons de simplicité nous avons décidé moi et mon équipe de faire de notre application, une page glissante (**scrollable**) :  disponible dans les propriétés de l'écran.

Nous avons également rajouté le logo de notre projet grâce à **Image** : 

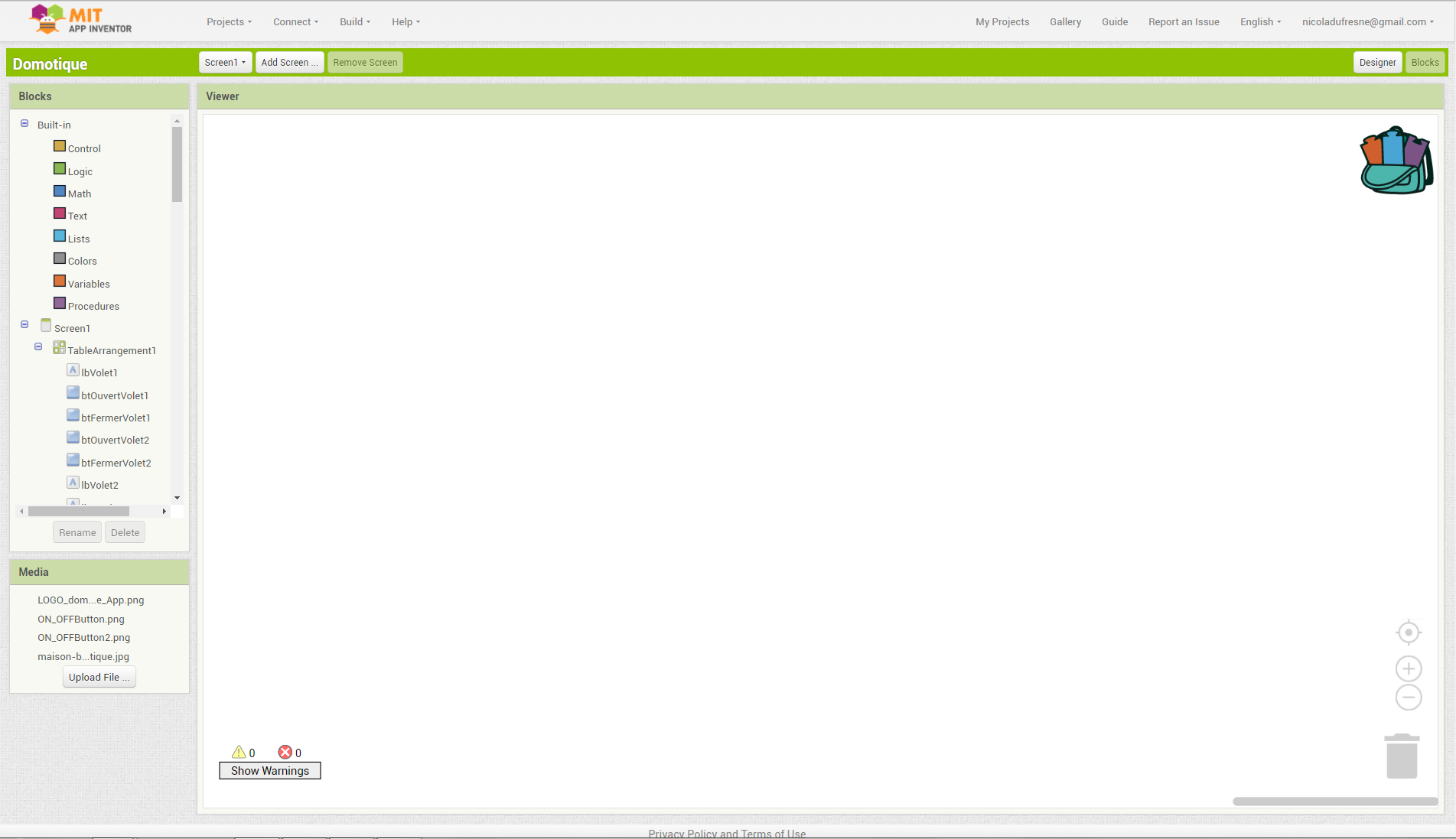
### Partie blocks



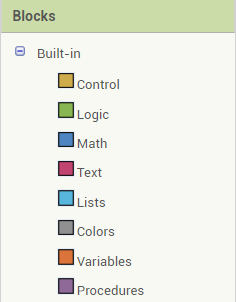
La partie **blocks** est accessible en cliquant sur le bouton en haut à gauche de la page :

Les blocs logiques sont des morceaux qui représentent du code et qui vont nous permettre de déterminer les comportements attendus de la part de notre application.

L’écran des blocs logiques se présente de la manière suivante

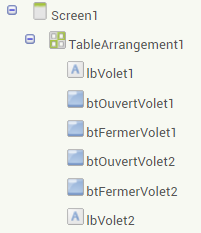


Dans la colonne de gauche se trouve l’ensemble des instructions que nous pouvons utiliser afin de réaliser votre application.

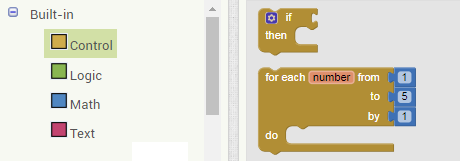


Ces instructions permettent de lier les comportements des composants que nous avons ajouté à l’ensemble de l’application.

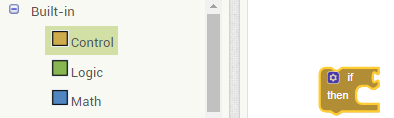
Tous les composants que nous avons ajouté dans la partie **Design** vont apparaître en dessous de ces blocs d’instruction avec le nom donné identique.



Lorsque l'on souhaite utiliser un bloc logique, il nous suffit de le sélectionner en cliquant dessus, un nouveau menu va alors apparaître

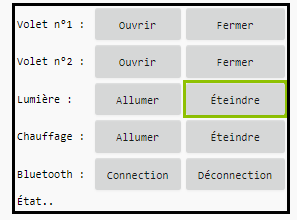
.

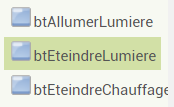
Il nous suffit ensuite de choisir le bloc logique de notre choix en fonction du comportement que l'on souhaite que notre application provoque et de faire glisser le bloc logique choisi dans l’espace vide à droite du menu :

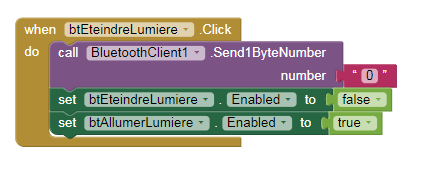


## Les blocs associés

### Bouton

Partie **Design** :



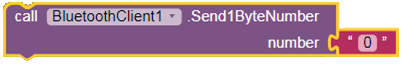
Partie **Blocks** :

L'exemple ci dessus s'applique pour les boutons associé aux volet n°1, volet n°2, lumière et chauffage.

Ce bouton effectue donc l'action suivante :

Quand on **click** sur le bouton **btEteindreLumiere**, faire..



appeler la connexion **BluetoothClient1** et envoyer la valeur " **0** "

.

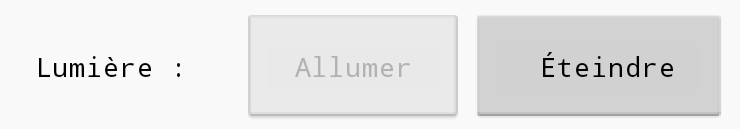
appliquer au bouton **btEleindreLumiere** une activation à **faux**.

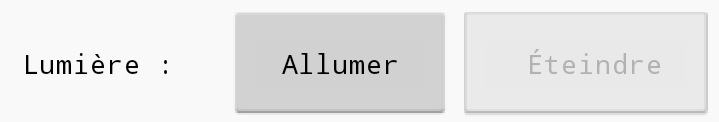


appliquer au bouton **btAllumerLumiere** une activation à **vrai**.



Le résultat est montré ci dessous :

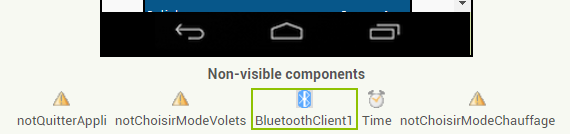
Si on appuie sur **Allumer**, le bouton devient inutilisable. Même chose si on appuie sur le bouton **Éteindre**.



Les valeurs envoyées seront alors manipulé dans un programme créé par les étudiants 1 et 2 pour modifier l'état des actionneurs.

### ListPicker Bluetooth

Tout d'abord, il faut sélectionner une communication Bluetooth dans la **Palette**, puis dans **Connectivity**. Etant donné que la tablette est l'esclave, il faut lui appliqué **BluetoothClient.**

Le Bluetooth apparait ensuite en bas de l'écran.

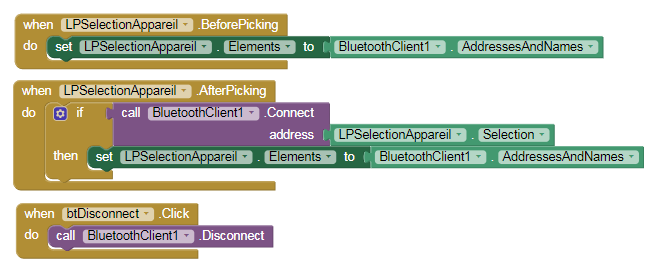
Partie **Design** :

Un **ListPicker** ressemble à un bouton au niveau représentation graphique. Il permet cependant une toute autre application.



Partie **Blocks** :

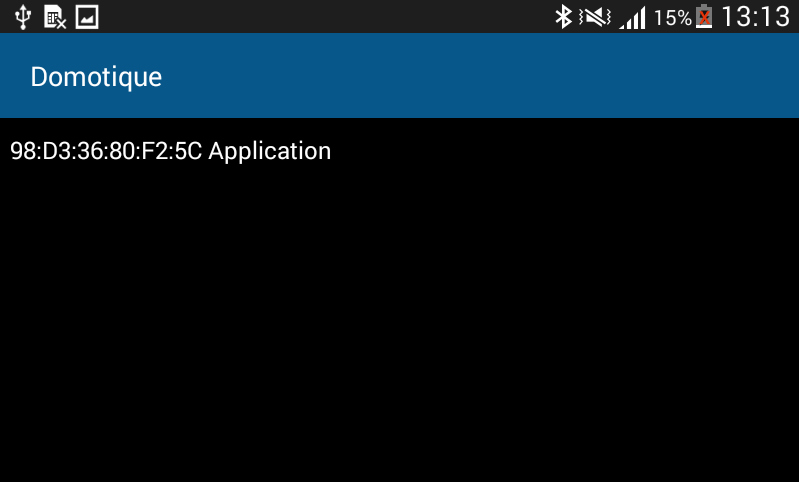
Les blocs suivant permettent d'établir une connexion Bluetooth avec notre module.



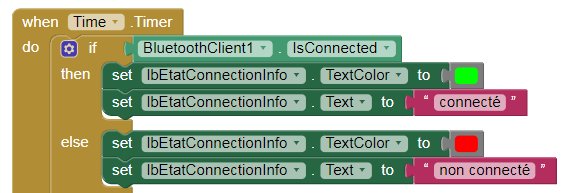
Le bouton **btDisconnect** permet de couper la communication Bluetooth précédemment établi.

L'action provoqué est la suivante :

Après avoir appuyer sur **LPSelectionAppareil**, une page s'ouvre.



Cette page affiche les connexions Bluetooth disponible. On peut observer l'adresse MAC Bluetooth ainsi que le nom du module Bluetooth précédemment configuré via les commandes AT.

Ce bloc permet au **Label** appelé **lbEtatConnectionInfo** d'afficher si la connexion Bluetooth est correctement établie ou non.



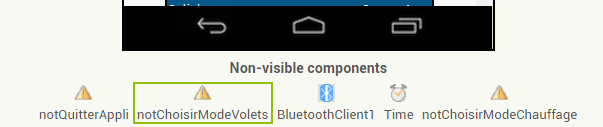
Si la connexion est établie, le **Label** affiche connecté en vert :

Si la connexion n'est pas établie, le **Label** affiche non connecté en rouge :

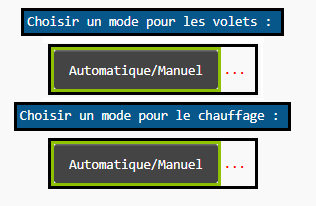
### Notifier

Partie **Design** :

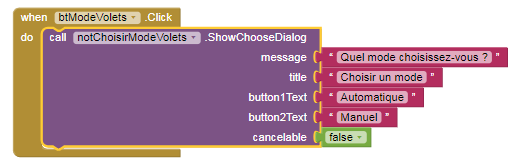
Tous comme le **BluetoothClient1**, le **Notifier** apparait en dessous de l'écran sous la frome d'un triangle jaune.



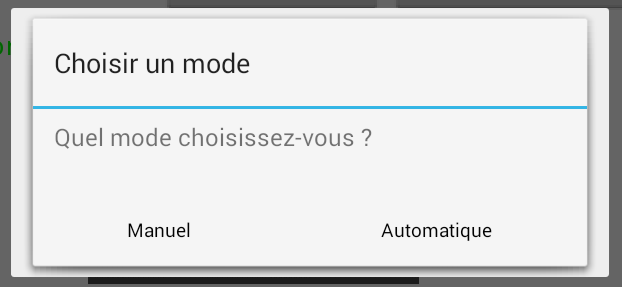
Une **notification** apparait après une action. Ici nous avons utilisé les boutons **btModeVolets** et **btModeChauffage** qui nous permettront de définir un mode **automatique** ou **manuel** répondant à notre cahier des charges.

Les boutons sont les suivants :

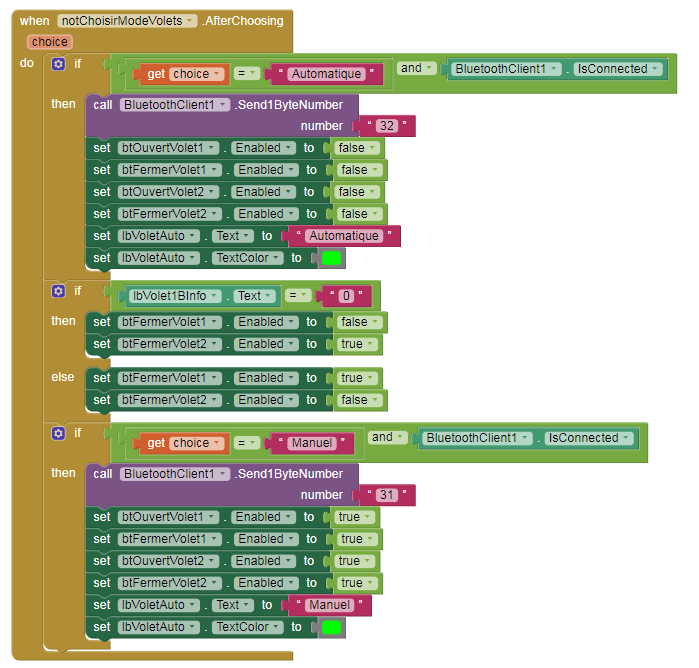
Partie **Blocks** :

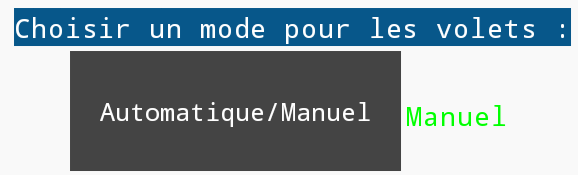
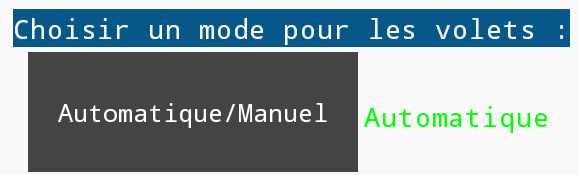
Les blocs suivant permettent d'afficher une notification à l'écran après l'appui sur le bouton **btModeVolets**.

Après avoir appelé la notification **notChoisirModeVolets**, on peut donc choisir le message à afficher, le titre, le texte sur le bouton 1 et le texte sur le bouton 2 ainsi que d'afficher un non un bouton **annuler**.

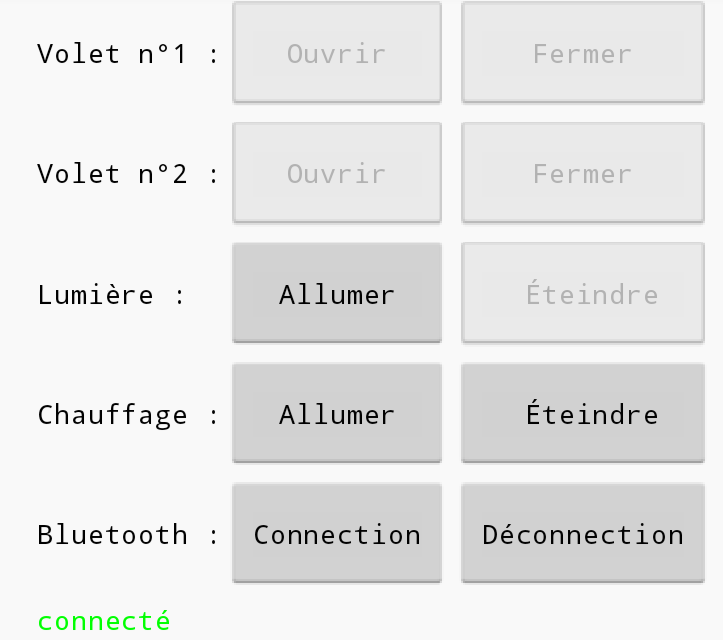
La notification apparu à l'écran est la suivante :

Les boutons pour choisir le mode de chauffage **btModeChauffage** et pour quitter l'application **btQuitterAppli** fonctionnent de la même manière.

Le bloc de code qui suit permet selon le choix sélectionné **d'envoyer** **une** **valeur** par la communication Bluetooth, de **désactiver** **les** **boutons** devant être inutilisable selon le mode et d'afficher sur un **Label** le **type** **de** **mode** **choisi**.

On pourra donc apercevoir sur notre application :

ou

Les boutons désactivé sont affiché de la manière suivante :

Selon les valeurs récupérés, un programme permettra aux volets en mode automatique de s'ouvrir et de se fermer en fonction de la luminosité extérieur récupéré par un capteur ( Voir l'étudiant 1 ).

Les modes automatiques et manuel sont géré de la même façon pour le radiateur, à la différence où ce sera la **Slider** qui sera désactivé.

De même pour la bouton quitter application **btQuitterAppli** qui selon le choix quittera ou non l'application.

### Slider

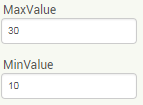
Le Slider est un potentiomètre numérique qui permet de choisir une valeur sur une plage donnée avec sont doigt de manière simple et efficace.

Partie **Design** :

Le **Slider** est représenté sous cette forme sur Mit App Inventor 2 :

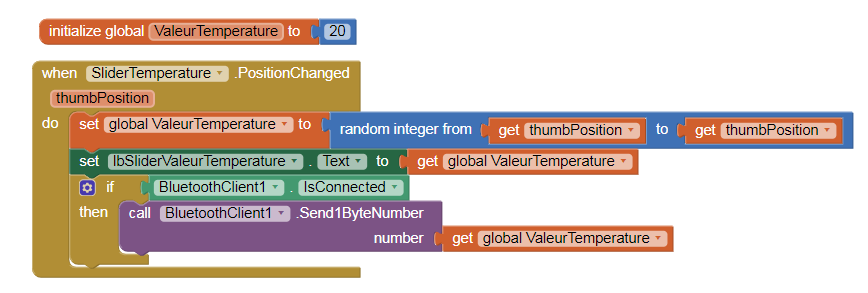
Et sous cette forme sur l'application :

La plage de valeur à choisir se trouve dans le cadre **Properties** :



Pour une plage de température dans une maison nous avons choisi de 10°C à 30°C.

Partie **Blocks** :



Dans les blocs de code précèdent, on initialise une variable global à 20. C'est donc la valeur par défaut de la température.

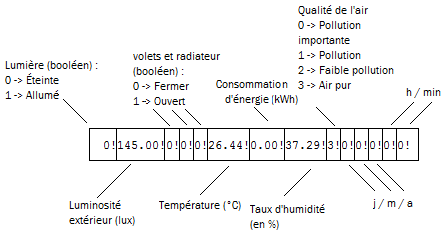
Ensuite, si la position du **Slider** nommé **SliderTemperature** change, on applique la valeur du changement à la variable global que l'on envoie ensuite via la communication Bluetooth. Cette valeur permettra dans un programme de modifier la valeur en °C du radiateur mis à notre disposition.

Un **Label** prendra également la valeur de la position du **Slider**.



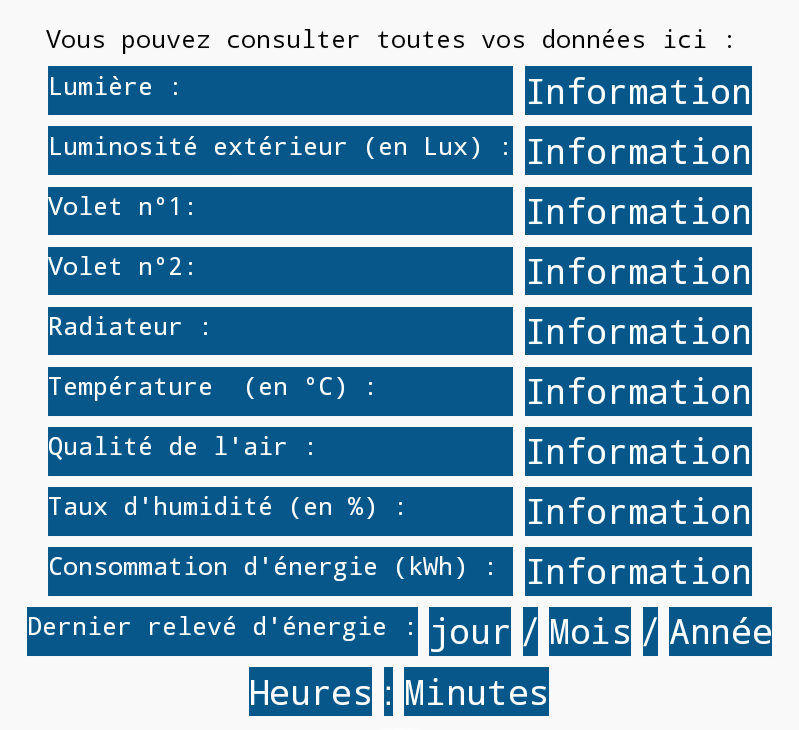
### Récupération d'une trame

La partie la plus complexe est la récupération de la trame envoyé par les étudiants 1 et 2. La trame comprend :



Nous avons donc pour objectif de récupérer chacun des valeurs émises par la trame et de les afficher sur la partie donnée de l'application sous Android.

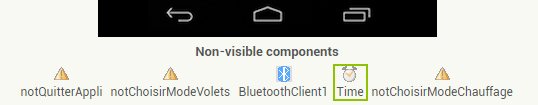
Partie **Design** :

Chacune des données viendront se superposer sur un label qui lui est attribué.

Partie **Blocks :**

Pour commencer la partie **Blocks** de la récupération de notre trame, nous allons parler du **Timer** ou **Clock**.

**Clock** est disponible dans la **palette** de la partie **Designer** dans l'onglet **Sensors**. Il est présenté sous la forme d'une horloge et est visible en bas de l'écran avec le **ClientBluetooth1** et la **notification**.

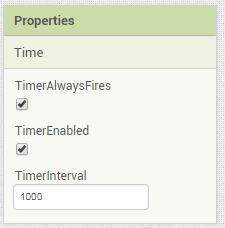


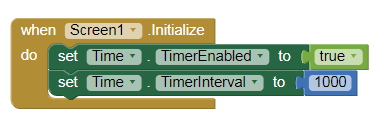
Dans la partie **Blocks** il se présente sous la forme suivante :



Ce composant permet d’utiliser des fonctionnalités de temps. Cela est très utile pour définir des comptes à rebours, afficher l’heure, déclencher des comportements au bout d’un temps précis... On peut retrouver se composant à la page 27 pour afficher si la connexion est toujours présente ou non.

Nous avons défini un intervalle de temps de 1000ms, ce qui veut dire que l'écran de notre application rafraichira les valeurs reçu toute les secondes.

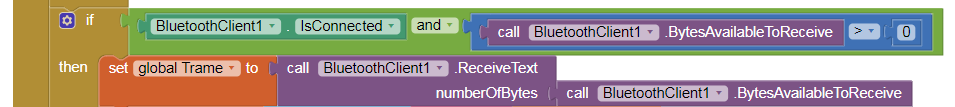




Etant donné que les blocs permettant la récupération de la trame sont identiques, nous allons voir pour récupérer une seul valeur.

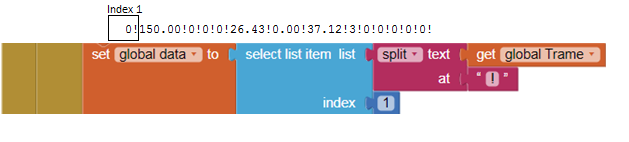
Dans un premier temps il nous faut déclarer une variable global et autant de variable de donnée que de valeur à récupérer.

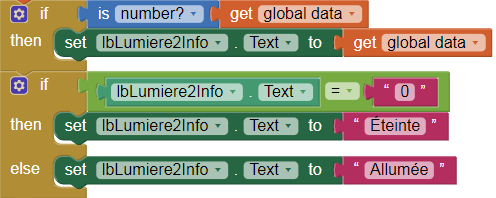


Ensuite il faut vérifier si la communication Bluetooth est connecté et que l'on reçois correctement des valeurs supérieurs à 0. Si c'est le cas, on donne à la variable global Trame les valeurs de la trame complète.

Puis, entre chacune des valeurs se trouve un " ! ". Ce caractère défini la séparation entre chaque valeur. Ici, le premier global data à pris la valeur du premier chiffre de la trame.

Index est le numéro qui correspond à la position de la valeur dans la trame.



Enfin, si la valeur est un numéro, on le récupère et on l'affiche sur le Label lbLumiere2Info. Si le numéro est égal à 0, le Label affiche Éteinte. Si c'est une autre valeur, affiche Allumée.

## Conclusion

# Annexe

Gestion de la maison

Lecture des données