HACKABLE
Électronique numérique & embarqué

336 article(s)

(/Hackable/HK-015/Connectez-un-module-LCD-a-votre-Raspberry-Pi-pour-afficher-la-configuration-reseau)

(/Hackable/HK-015/Connectez-un-module-LCD-a-votre-Raspberry-Pi-pour-afficher-la-configuration-reseau)

(/readlist/add/article/a)

(/readlist/add/article/a)

Bluetooth : pilotez votre Arduino avec votre smartphone

Hackable n° 15 | (/Hackable/HK-015) novembre 2016 | Par Bodor Denis (/auteur/Bodor-Denis)

Électronique (/search/node?domains%5B0%5D=72452) **Mobilité (/search/node?domains%5B0%5D=72462)**

Radio et wireless (/search/node?domains%5B0%5D=72465)

> *À présent que nous avons fait connaissance avec les modules Bluetooth, il est temps de passer aux choses sérieuses ou du moins, aux choses plus utiles. La connectivité Bluetooth est disponible sur bien des systèmes, des PC aux Raspberry Pi, mais la plateforme la plus intéressante est sans l'ombre d'un doute le smartphone. Voyons donc ensemble comment commander un projet Arduino depuis une application Android.*

Précisons de suite ce que cet article n'expliquera pas : le développement d'une application Android. En effet, même s'il est techniquement possible à tout un chacun de créer une application pour son smartphone, ce n'est pas quelque chose qui peut être fait aussi rapidement et aussi facilement qu'un croquis Arduino. Android Studio, l'environnement de développement proposé gratuitement par Google pour développer des applications Android, n'est pas conçu dans un but pédagogique, comme c'est le cas d'Arduino, mais dans le but de... développer des applications.

Bien sûr, la création d'un premier programme, le fameux « Hello World », est quelque chose de relativement rapide et vous aurez, en quelques minutes votre propre application vous faisant « coucou » sur votre smartphone, après avoir suivi un petit tutoriel en ligne. Le problème n'est pas là, mais dans l'étape suivante : sauter le pas et concevoir, avec un langage que vous ne connaissez peut-être pas encore, une application complète avec plusieurs écrans (ou « vues » dans le monde Android), des interactions avec l'utilisateur et le système, ou encore disposant d'une ergonomie digne de ce nom.

La seule création d'une telle application, même simple, détaillée de manière intelligible, nécessiterait un magazine complet (chose de nous avons d'ailleurs fait aux Éditions Diamond, sous la forme d'un hors-série de *GNU/Linux Magazine* (n°82)), car il faudrait aborder beaucoup de concepts et de principes de développement qui n'ont rien de didactique.

Fort heureusement, il ne nous sera pas nécessaire de créer une application pour envoyer des ordres en Bluetooth à notre projet Arduino. En effet, un certain nombre de programmeurs, coutumiers de ce genre de pratiques, proposent leurs applications sur Google Play et il nous suffit de les utiliser en programmant uniquement la partie Arduino.

Bien entendu, si vous désirez une interface spécifique ou souhaitez explorer des domaines ou des fonctionnalités qui n'ont pas été déjà couverts, vous n'aurez d'autres choix que de vous initier à la programmation Android. Il en va de même si vous souhaitez développer une application iOS, à condition bien entendu, d'écarter l'option des modules que nous avons précédemment configurés et de vous lancer dans le monde du Bluetooth Low Energy (puisque les iPhone et iPad ne sont pas compatibles avec les modules Bluetooth utilisant un profil SPP).

1. Contrôlez la couleur de leds

La première application que nous allons utiliser est *LED Control* de Michael Fennel. Celle-ci est disponible gratuitement sur Google Play, mais ne propose pas d'accès à son code source (si vous aviez dans l'idée de l'améliorer). Cette application permet de contrôler la couleur d'une ou plusieurs leds connectées à un montage équipé d'un module Bluetooth SPP comme ceux que j'ai décrits précédemment.


(https://connect.ed-diamond.com/sites/default/files/articles/hackable/hk-015/82836/QR_LedControl1.png)

Pour obtenir LED Control, flashez ce code.

Bien que le développeur ayant créé l'application ne semble pas vouloir exposer son fonctionnement interne, il met à disposition un croquis d'exemple sur GitHub ainsi qu'une documentation succincte détaillant le protocole utilisé (https://github.com/fennel-labs/LED-control/wiki/protocol (https://github.com/fennel-labs/LED-control/wiki/protocol)). En effet, l'application, après connexion en Bluetooth, envoie des données dans un format précis définissant la couleur que doivent prendre la ou les leds à contrôler.

Le format des messages toujours sur 6 octets est le suivant :

- Octet 0 : numéro ou adresse de la led à contrôler (pour une utilisation future, à l'heure actuelle c'est toujours 0) ;

- Octet 1 : mode de fonctionnement, 0 pour le mode « monochrome » et 1 pour le mode « fondu » (animation) ;

- Octet 2 à 4 : dépend du mode. En mode « monochrome », ces trois octets définissent respectivement les valeurs de rouge, de vert et de bleu entre 0 et 255. En mode « fondu », l'octet 2 précise la vitesse et l'octet 3 l'état de l'animation (0 = oui, 1 = non). L'octet 4 est alors inutilisé ;

- Octet 5 : le caractère ; utilisé pour marquer la fin du message.

L'application présente deux écrans différents : un sélecteur de couleur (*Picker*) et un contrôleur d'animation (*Fader*). Le sélecteur se présente comme un anneau de couleur où l'on peut déplacer un curseur. Au centre, une zone circulaire permet d'alterner la couleur entre celle sélectionnée, le blanc et le noir (led éteinte). Deux curseurs supplémentaires, placés en dessous de l'anneau, permettent de régler la saturation et la valeur de la couleur actuelle (du blanc à la couleur et de la couleur au noir).

Le second écran permet de régler une vitesse entre 0 et 100% (correspondant aux 0 à 255 dans les messages) à l'aide d'un curseur horizontal et l'état de l'animation (on/off) à l'aide un bouton.

Notez que tout ceci forme les paramètres non variables avec lesquels nous allons travailler. La logique première veut que le réglage de la couleur corresponde effectivement à celle de la ou des leds connectées à notre carte Arduino et que le contrôle d'animation fasse varier continuellement l'intensité de chaque composante de couleur (RVB). Cependant, dans l'absolu, vous n'êtes pas du tout obligé de respecter ces principes. L'écran de contrôle de l'animation peut parfaitement servir pour un choix d'effet (de la bibliothèque WS2812FX, par exemple), en décodant arbitrairement le fait que 10% de la vitesse correspond à un certain effet, 15% à un autre, 20% un troisième, etc. Vous ne contrôlez pas la relation application/message, mais vous pouvez faire ce que vous voulez avec la relation message/action côté Arduino.

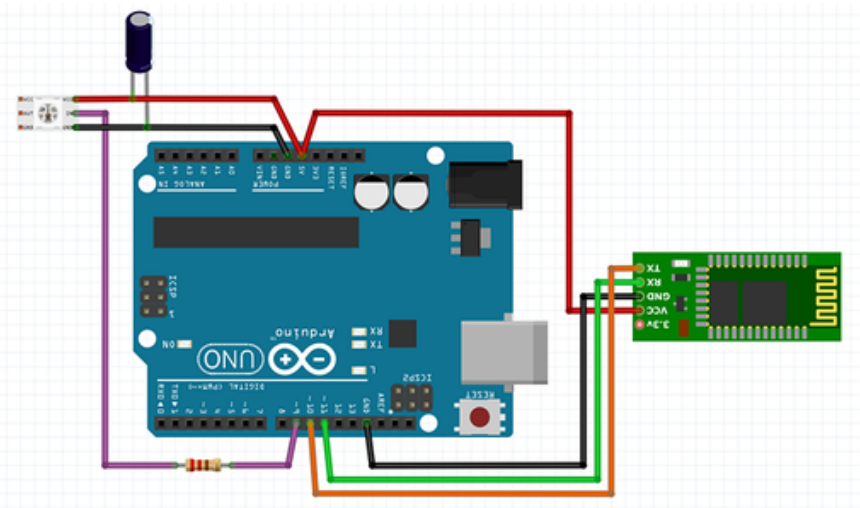
(https://connect.ed-diamond.com/sites/default/files/articles/hackable/hk-015/82836/BT_ledcontrol1.png)

L'application LED Control de Michael Fennel vous permettra de contrôler la couleur d'une ou plusieurs leds, voire d'un éclairage complet, depuis votre smartphone Android.

Michael Fennel fournit un croquis Arduino d'exemple matérialisant sa vision de cette relation. Nous allons ici procéder de même, mais avec une approche plus simple et plus épurée afin de démontrer la facilité qu'offre l'utilisation d'un module Bluetooth pour vos projets.

Le montage de démonstration sera relativement simple puisque nous allons tout bonnement connecter le module Bluetooth à deux broches arbitrairement choisies de la carte Arduino, en plus de l'alimentation 5V et de la masse. Nous n'allons, en effet, pas utiliser le port série de la carte, mais la bibliothèque *SoftwareSerial* nous permettant de mettre en place ce type de liaison de façon logicielle. Ceci fonctionnera exactement comme `Serial()` tout en conservant la communication avec l'IDE Arduino et son moniteur série pour peaufiner nos réalisations.

En ce qui concerne la led que nous allons contrôler, le choix le plus simple consiste à utiliser une WS2812b (alias NeoPixel). Le croquis de démonstration de Michael Fennel utilise des sorties analogiques (PWM) et une led RVB standard, mais ceci n'a pas vraiment d'importance. Le croquis que nous allons utiliser pourra être facilement adapté à presque n'importe quel usage et n'importe quel composant. Notre unique WS2812b (qui est en réalité un clone au format 8 mm translucide ici) sera connecté à la broche 9 en plus des 5V et de la masse. Là encore, c'est un choix arbitraire sans importance. Pour correctement faire les choses, il est recommandé d'ajouter un condensateur de quelques 100µF entre l'alimentation et la masse, ainsi d'une résistance entre la broche et l'entrée de données de la led, en particulier si votre environnement de travail est riche en perturbations électrostatiques.



(https://connect.ed-diamond.com/sites/default/files/articles/hackable/hk-015/82836/BT_un_led_montage1.png)

Le montage de démonstration est simple, mais pourra être facilement étendu ou modifié. Ici nous contrôlons une seule led WS2812b, mais rien ne vous empêche d'en prendre une importante quantité en charge ou encore de détourner le sens des messages reçus pour en contrôler plusieurs indépendamment.

Il ne nous reste plus, à présent, qu'à rédiger le croquis qui sera extrêmement court. Nous utilisons les bibliothèques *SoftwareSerial* pour la création d'un port série supplémentaire et *Adafruit_NeoPixel* pour la prise en charge de la led WS2812b.

La création des objets représentants les deux éléments se fera respectivement avec :

- **`SoftwareSerial btmodule(10, 11)`** qui nous permet d'obtenir un **`btmodule`** utilisable exactement comme **`Serial`**. Nous passons en argument les broches correspondants aux signaux RX et au TX.

- **`Adafruit_NeoPixel ws = Adafruit_NeoPixel(1, 9, NEO_RGB + NEO_KHZ800)`** qui rendra **`ws`** utilisable dans le reste du croquis. Là, les arguments sont un peu plus nombreux avec, dans l'ordre, le nombre de leds (**`1`**), la broche utilisée (**`9`**) et une combinaison de macros précisant le type d'encodage des valeurs RVB (**`NEO_RGB`** pour notre led 8mm, mais **`NEO_GRB`** pour une vraie WS2812b) et la vitesse de communication (**`NEO_KHZ800`** pour 800 khz).

Nous pouvons ensuite initialiser l'ensemble (port série, port série logiciel et led) dans la fonction **`setup()`** avant de spécifier une couleur noire pour la led. En effet, le modèle utilisé s'illumine en bleu dès sa mise sous tension, ce qui n'est pas très agréable.

```
#include <SoftwareSerial.h>
#include <Adafruit_NeoPixel.h>
// Déclaration NeoPixel
// NEO_RGB pour la led 8mm
// NEO_GRB pour une vraie WS2812b
Adafruit_NeoPixel ws = Adafruit_NeoPixel(1, 9, NEO_RGB + NEO_KHZ80
0);
// Déclaration port série logiciel
SoftwareSerial btmodule(10, 11); // RX, TX
void setup() {
  // moniteur série en 115200 bps
  // (utile pour débuger)
  Serial.begin(115200);
  while (!Serial);
  // port logiciel en 19200
  // vitesse configurée avec AT+UART=19200
  btmodule.begin(19200);
  // initialisation NeoPixel
  ws.begin();
  // noir
  ws.setPixelColor(0, ws.Color(0,0,0));
  // affichage
  ws.show();
}
void loop() {
  // Des données à lire ?
  if (btmodule.available()) {
    // Oui, on récupère la commande
    String commande = btmodule.readStringUntil(';');
    // on a 5 octets (sans le ";") ?
    if(commande.length() == 5) {
      // oui, c'est une commande valide
      // le second octet est une commande de couleur ?
      if(commande.charAt(1) == 0x00) {
```

```
          // oui, on donne au NeoPixel la couleur demandée
          ws.setPixelColor(0, ws.Color(commande.charAt(2),commande.c
harAt(3),commande.charAt(4)));
          ws.show();
        }
      }
    }
}
```

Il ne reste ensuite plus qu'à gérer les messages arrivant dans la fonction **`loop()`** en vérifiant tout d'abord la présence de données reçues, avec **`btmodule.available()`**. Si tel est le cas, nous utilisons la méthode **`readStringUntil()`** en précisant comme argument le caractère marquant la fin d'un message (*`;`*). Si l'opération réussie, **`commande`** contiendra le message en question, débarrassé de son point-virgule.

Celui-ci, pour être valide, devra avoir une taille de 5 caractères/octets. Si ce n'est pas le cas, le message est corrompu et inutilisable et nous arrêtons là. Dans le cas contraire, nous avons un message à traiter. **`charAt()`** peut alors être utilisé pour vérifier s'il s'agit d'une commande de changement de couleur tel que décrit dans le protocole de Michael Fennel, avec la valeur comme second octet.

À ce stade, nous sommes donc relativement confiants quant à l'état et la nature du message et nous utilisons encore une fois **`charAt()`** pour extraire les octets correspondants à chaque valeur de couleur et les utiliser directement pour changer les valeurs utilisées par la led. Enfin, nous « poussons » ces changements avec **`show()`**.

Le test sera fort simple après enregistrement du croquis dans la carte Arduino : nous utilisons l'application de Michael Fennel, via le menu ***Connection Settings*** pour choisir notre périphérique Bluetooth que nous avons précédemment appairé et jouons avec les contrôles et curseurs à l'écran. En temps réel, la couleur de la led devrait changer selon notre bon plaisir.

Je n'ai pas ici pris en charge la fonction d'animation (message avec le second octet à 1), mais on peut facilement imaginer ajouter une simple condition et, par exemple, remplacer tous les appels aux fonctions et méthodes de *Adafruit_NeoPixel* par celles de *WS2812FX*, qui elles-mêmes utilisent *Adafruit_NeoPixel*, et définir un ou des effets en conséquence.

2. Contrôler des relais ou d'autres périphériques

Une autre application que je trouve très intéressante (gratuite et sans pub), car très polyvalente est *BlueTooth Serial Controller* de NEXT PROTOTYPES. Celle-ci peut sembler complexe au premier regard, mais elle permet de faire des merveilles. Là, nous sommes dans une situation différente de celle de l'application précédente qui était très spécialisée. *BlueTooth Serial Controller* vous propose d'associer les boutons d'un pavé de 9 à 25 boutons sur l'écran avec l'envoi de messages de votre choix.

(https://connect.ed-diamond.com/sites/default/files/articles/hackable/hk-015/82836/QR_BTSerialController1.png)

Pour obtenir BlueTooth Serial Controller, flashez ce code.

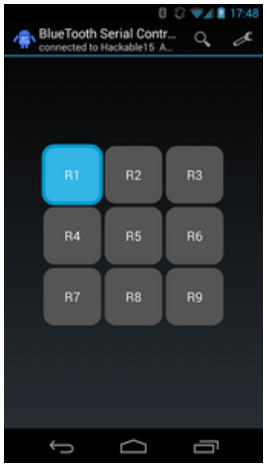
Après installation de l'application, on fera un tour dans les préférences et commencera par activer le mode « Orientation/Paysage » (pour un smartphone) ainsi que le mode « 9 BUTTON MODE » afin de masquer le second pavé de 16 boutons.

Pour une utilisation en bascule, comme pour le contrôle de relais, la logique sera la suivante : l'application propose 5 écrans de contrôle (*controllers*) différents nommés de A à E. Par défaut, on n'utilise qu'un seul contrôleur en envoyant des messages lors du contact avec un bouton, mais ici, nous voulons une bascule. Nous allons donc configurer le contrôleur A avec les messages à envoyer pour activer le ou les relais, et le contrôleur B pour les messages désactivant ces relais.

Nous allons donc commencer par faire un tour, toujours dans les préférences, dans le menu ***BUTTON/Name***, ce qui nous conduit à un autre écran de configuration. Là se trouve une longue liste de 26 boutons que nous pouvons nommer comme bon nous semble. Très originalement, nous allons libeller nos 9 boutons en R1 à R9 (en mode 9 boutons, les boutons 10 à 25 n'ont pas d'importance).

Nous revenons ensuite en arrière pour tapoter l'entrée ***BUTTON/Commande***. Une liste très similaire s'affiche nous proposant de saisir, pour chaque bouton, un message à envoyer. Pour la démonstration, nous nous contenterons de configurer le premier bouton en lui associant le message « r1on ».

Nous revenons ensuite à l'écran précédent et défilons jusqu'à l'entrée ***ON/OFF Mode*** qui nous conduit à un autre écran de configuration. Là, activez ***Enable ON/OFF Mode*** et juste dessus tapez sur ***Another Command's Source*** et choisissez ***Controller B***. Ceci paramètre l'application pour le mode bascule : en activant un bouton, nous envoyons le message définit précédemment, mais en désactivant ce même bouton, nous comptons envoyer le message du contrôleur B et non A.



(https://connect.ed-diamond.com/sites/default/files/articles/hackable/hk-015/82836/BT_SerialContr1.png)

BlueTooth Serial Controller est une application Android offrant la possibilité de configurer l'effet d'une série de boutons présentés à l'écran. La configuration est un peu touffue, mais l'étendue des possibilités est très importante puisque c'est vous qui décidez de la nature et la forme des messages envoyés en Bluetooth.

De ce fait, revenez à l'écran précédent et, tout en haut des options, passez sur le contrôleur B, puis retournez sur ***BUTTON/Command***. Vous constaterez que le message associé au bouton 1 est la valeur par défaut (« def ») et non plus « r1on ». C'est normal, nous configurons le contrôleur B et non le A. Changez la configuration pour envoyer le message « r1off » cette fois puis, enfin, revenez à l'écran précédent pour rebasculer sur la contrôleur A.

Pour terminer, quittez la configuration pour afficher l'écran principal avec les 9 boutons. Un « clic » sur le bouton 1, nommé R1 enverra « r1on » tout en l'activant (il devient bleu) et un second « clic » désactivera le bouton (redevenant gris) tout en envoyant « r1off » (du contrôleur B). Il ne nous reste alors plus qu'à créer le croquis Arduino correspondant.

Celui-ci ne sera que très légèrement différent du précédent, car il nous suffit d'éliminer toutes traces du support pour la led et d'utiliser la broche 9 pour contrôler un relais (ou autre chose comme une led pour test) :

```
#include <SoftwareSerial.h>
// Déclaration port série logiciel
SoftwareSerial btmodule(10, 11); // RX, TX
void setup() {
  // moniteur série en 115200 bps
  // (utile pour débuger)
  Serial.begin(115200);
  while (!Serial);
  // port logiciel en 19200
  // vitesse configurée avec AT+UART=19200
  btmodule.begin(19200);
  pinMode(9, OUTPUT);
  digitalWrite(9, HIGH);
}
void loop() {
  // Des données à lire ?
  if (btmodule.available()) {
    // Oui, on récupère la commande jusqu'à LF
    String commande = btmodule.readStringUntil('\n');
    if(commande.length() > 0) {
      // on a une commande
      if(commande == "r1on")
        digitalWrite(9, LOW);
      if(commande == "r1off")
        digitalWrite(9, HIGH);
    }
  }
}
```

On retrouve ici le **`readStringUntil()`**, mais cette fois avec **`\n`** en argument, qui est le caractère « nouvelle ligne », alias LF. Il est possible de choisir, dans la configuration de l'application, ce marqueur de fin de ligne (par défaut) ou CR (**`\r`**) ou CR et LF. Nous testons également si la chaîne obtenue est vide (en cas de problème) et finalement, on active ou désactive la sortie en fonction du message « r1on » ou « r1off ». Notez au passage que le module relais utilisé ici étant contrôlé par un transistor PNP SS8550, la sortie est inversée.

Nous n'avons ici configuré qu'un bouton, une paire de messages et un relais mais, vous l'aurez compris, une simple carte Arduino, avec cette application, pourra très facilement contrôler 9 relais de la même manière, avec une interface relativement sympathique.

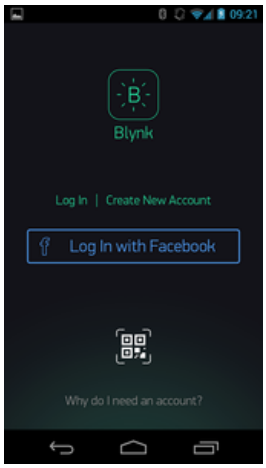
Si vous procédez à une recherche sur Google Play, avec « Bluetooth relay », vous trouverez une quantité impressionnante d'applications. La plupart sont trop restrictives, peu configurables ou, disons-le franchement, carrément moches et boguées. Après en avoir testé une bonne douzaine, j'ai retenu celle-ci pour sa simplicité (relative) et sa souplesse. Mais il est possible qu'il en existe une plus personnalisable encore.

3. Plus loin : un mot à propos d'ArduDroid, de Blynk et MIT App Inventor 2

Dans ma tentative de sélections d'applications démonstratives, je suis tombé sur ce qu'on pourrait appeler des classiques qui semblent plébiscitées par bon nombre d'utilisateurs, mais que j'ai écartées pour des raisons personnelles et/ou des préférences politico-philosophiques.

ArduDroid de Hazim Bitar, par exemple, est très sympathique, car cette application permet, à l'aide d'un croquis dédié, de prendre le contrôle sur toutes les broches d'une carte Arduino. En entrée comme en sortie, y compris en analogique. Le problème cependant, est que cela ne reste intéressant que pour des tests et non pour un usage courant ou une démonstration grandiloquente de vos capacités techniques (vous savez le « *non, mais moi je contrôle tout mon appart avec mon téléphone* », « *oui, c'est moi qui ai tout fait* »). Contrôler les broches n'est pas la même chose que de choisir une couleur sur un écran...

Une autre application très très à la mode est Blynk. Le concept est en effet extrêmement séduisant puisque l'interface de l'application est totalement modulaire et graphiquement très agréable. C'est un système complet fonctionnant aussi bien en Bluetooth qu'en Wifi et ce avec des cartes Arduino ou des choses plus musclées comme une Raspberry Pi. Jauges, graphiques, curseurs, boutons, afficheurs, voyants... tout y est ! Mais voilà, une application qui me demande de me connecter ou créer un compte pour utiliser mon propre matériel a une fâcheuse tendance à me mettre les nerfs à vif, tout autant qu'une liste longue comme le bras de permissions demandées par l'application (incluant un accès à la caméra, aux données GPS, et aux achats « Achats via l'application »).



(https://connect.ed-diamond.com/sites/default/files/articles/hackable/hk-015/82836/BT_Blynk1.png)

Blynk est une application non seulement très populaire, mais de très bonne qualité. Cependant, lorsque je suis accueilli par un

écran me demandant de me connecter à une infrastructure et donc de créer un compte pour quelque chose d'aussi simple que de contrôler une carte Arduino en Bluetooth, j'ai le réflexe d'immédiatement fermer l'application et la désinstaller...

Certes, il est possible d'installer son propre serveur (en Java) pour se passer de celui de Blynk, mais c'est une approche que je n'apprécie pas même si, oui, en effet, ceci rend votre projet accessible de n'importe où, et qu'on se trouve aujourd'hui dans un monde de cloud, d'objets connectés (IoT) et aussi de fuites de données privées... Peut-être reviendrai-je sur le sujet d'ici quelque temps, mais pour l'heure le « *Connectez-vous avez Facebook* » en écran d'accueil de l'application m'a clairement effarouché.

Enfin, nous avons, non pas une application, mais une solution pour créer facilement des applications Android : MIT App Inventor 2. Je dois avouer que là, ma réticence est plus technique et fortement en lien avec ma vision de ce qu'est la programmation. MIT App Inventor 2 se présente sous la forme d'une application à installer sur son smartphone et d'un environnement de développement en ligne (sur le Web) permettant de créer des applications puis de les tester et les déployer sur son smartphone. Je suis déjà réticent quant à une telle architecture, puisque forcément il faut créer un compte, mais cela se cumule avec la façon que propose cet environnement de créer une application : non pas un langage, mais une représentation visuelle du programme façon Scratch (créé également au MIT Media Lab). Trouvez-moi vieux jeu si vous voulez, mais j'ai cette fâcheuse tendance à penser que le fait d'agencer des pièces de puzzle n'est pas « programmer ». Une imbrication de pièces multicolores peut effectivement être très attrayante, voire inspiratrice, pour une jeune audience mais, selon moi, ce n'est pas une solution viable à long terme pour créer un programme lisible, facile à modifier et à maintenir.

Apprendre à utiliser Java et l'environnement de développement Android demande du temps et des efforts, certes, mais cet apprentissage sera de toute façon nécessaire lorsqu'il s'agira de passer d'une solution « puzzle » à la création d'une vraie application.

Article rédigé par

(/auteur/Bodor-Denis)

Bodor Denis (/auteur/Bodor-Denis)

Chef des rédactions, Rédacteur en chef du magazine Hackable - Éditions Diamond (/auteur/Bodor-Denis)

619 articles

Par le(s) même(s) auteur(s)

Édito (/Hackable/HK-035/Edito)

Hackable n° 35 (/Hackable/HK-035) | octobre 2020

Par Bodor Denis (/auteur/Bodor-Denis)

« *Alerte de sécurité critique* »

Voilà qui a de quoi effrayer et c'est clairement l'objectif. Je dis bien « effrayer » et non « sensibiliser », car, « *Quelqu'un vient d'utiliser votre mot de passe pour essayer de se connecter à votre compte à partir d'une application n'appartenant pas à Google* » (notez avec quelle condescendance il est précisé qu'il ...

Motoriser une antenne directionnelle avec un ESP8266 (/Hackable/HK-035/Motoriser-une-antenne-directionnelle-avec-un-ESP8266)

Hackable n° 35 (/Hackable/HK-035) | octobre 2020

Par Bodor Denis (/auteur/Bodor-Denis)

Électronique (/search/node?domains%5B0%5D=72452)

Radio et wireless (/search/node?domains%5B0%5D=72465)

Cet article aurait pu s'intituler « pointez les choses dans le ciel avec un ESP8266 », car en réalité, l'application de ce qui va suivre à la réception de signaux venus de l'espace n'est qu'une utilisation parmi tant d'autres. Notre objectif ici sera de motoriser une antenne de façon à la pointer automatiquement en direction d'un émetteur mobile, et plus exactement, un satellite en orbite basse (< 2000 km). Le tout, bien entendu, en le suivant alors qu'il se déplace. ...

Le module du moment : Décodeur DTMF (/Hackable/HK-035/Le-module-du-moment-Decodeur-DTMF)

Hackable n° 35 (/Hackable/HK-035) octobre 2020

Par Bodor Denis (/auteur/Bodor-Denis)

Électronique (/search/node?domains%5B0%5D=72452)

DTMF, pour *dual-tone multi-frequency* est un encodage où chaque symbole d'un alphabet de 16 correspond à un couple de deux fréquences audibles, parmi une collection de 8, utilisées de concert. Initialement créé pour la téléphonie fixe, ce système permet historiquement d'encoder les pressions sur les touches d'un téléphone (« 0 » à « 9 », « A » à « D », plus « * » et « # ») et de transmettre le signal correspondant sous forme de sons. ...

Pilotez de manière optimale vos afficheurs LED (/Hackable/HK-034/Pilotez-de-maniere-optimale-vos-afficheurs-LED)

Hackable n° 34 (/Hackable/HK-034) juillet 2020

Par Bodor Denis (/auteur/Bodor-Denis)

Code (/search/node?domains%5B0%5D=72464)

Électronique (/search/node?domains%5B0%5D=72452)

Trop souvent, dans les forums et/ou sur les sites web, certains ont tendance à conseiller l'approche « facile » plutôt que l'approche « efficace ». Qui n'a jamais vu un jour quelqu'un répondre « mais utilises donc xxx(), ça marche et c'est plus simple » en réponse à une problématique précise ? C'est là, généralement, le fait de personnes qui n'ont que peu

...

Le module du moment : afficheur matrice led 8x32 (/Hackable/HK-034/Le-module-du-moment-afficheur-matrice-led-8x32)

Hackable n° 34 (/Hackable/HK-034) juillet 2020

Par Bodor Denis (/auteur/Bodor-Denis)

Électronique (/search/node?domains%5B0%5D=72452)

La bibliothèque MD_MAX72XX de Marco Colli (alias MajicDesigns) permet de piloter un ou plusieurs modules en configuration linéaire (les uns après les autres, sur une ligne), mais elle forme également la base de la bibliothèque MD_MAXPanel supportant des agencements en panneaux et fournissant des primitives graphiques intéressantes (points, lignes, rectangles, cercles, etc.). ...

+

(/search/node?authors%5B0%5D=70942)

Articles qui pourraient vous intéresser

Fonctionnement d'un téléphone mobile sous l'eau (/Hackable/HK-035/Fonctionnement-d-un-telephone-mobile-sous-l-eau)

Hackable n° 35 (/Hackable/HK-035) octobre 2020

Par Carry Emile (/auteur/Carry-Emile), Friedt Jean-Michel (/auteur/Friedt-Jean-Michel)

Radio et wireless (/search/node?domains%5B0%5D=72465)

Parmi les idioties que nous lisons quotidiennement dans la presse grand public, la technologie, et les liaisons radiofréquences en particulier, est souvent en haut du palmarès, faute de compétence des auteurs et du moindre esprit critique sur leurs affirmations, sans néanmoins atteindre le niveau des statistiques qui sont l'apothéose de l'art de cacher le mensonge [1]. ...

Une carte pilote de LED RGB hackée en kit de développement FPGA à bas coût (/Hackable/HK-035/Une-carte-pilote-de-LED-RGB-hackee-en-kit-de-developpement-FPGA-a-bas-cout)

Hackable n° 35 (/Hackable/HK-035) octobre 2020

Par Marteau Fabien (/auteur/Marteau-Fabien)

Électronique (/search/node?domains%5B0%5D=72452)

Dans cet article, nous allons vous proposer un Hack au sens noble du terme, qui consiste à détourner un produit de son usage prévu initialement pour en faire autre chose. Le produit en question est une carte de contrôle de panneaux de LED disponible pour 15 $ sur le site chinois de vente en ligne AliExpress. Il s'avère que la Colorlight 5A-75B est constituée d'un FPGA ECP5. Voilà qui peut faire un excellent kit de développement ECP5 à très bas coût. ...

Identification automatique de signaux grâce à la fonction d'autocorrélation (/MISC/MISCHS-022/Identification-automatique-de-signaux-grace-a-la-fonction-d-autocorrelation)

MISC HS n° 22 (/MISC/MISCHS-022) octobre 2020

Par Kuchly Denis (/auteur/Kuchly-Denis)

Sécurité (/search/node?domains%5B0%5D=72467)

Radio et wireless (/search/node?domains%5B0%5D=72465)

Vous connaissiez la chasse au trésor… Initiez-vous maintenant à la chasse au signal (signal hunting en anglais). La chasse au signal consiste à rechercher les signaux qui nous entourent dans l'espace invisible du spectre électromagnétique. Mais plus besoin de rester l'oreille collée sur un haut-parleur à tourner le bouton pour régler la fréquence. La SDR (Software Defined Radio) a révolutionné tout cela : une radio numérique et un PC est vous voilà armé pour découvrir ce ...

Le module du moment : Décodeur DTMF (/Hackable/HK-035/Le-module-du-moment-Decodeur-DTMF)

Hackable n° 35 (/Hackable/HK-035) octobre 2020

Par Bodor Denis (/auteur/Bodor-Denis)

Électronique (/search/node?domains%5B0%5D=72452)

DTMF, pour *dual-tone multi-frequency* est un encodage où chaque symbole d'un alphabet de 16 correspond à un couple de deux fréquences audibles, parmi une collection de 8, utilisées de concert. Initialement créé pour la téléphonie fixe, ce système permet historiquement d'encoder les pressions sur les touches d'un téléphone (« 0 » à « 9 », « A » à « D », plus « * » et « # ») et de transmettre le signal correspondant sous forme de sons. ...

Simulation d'un ordinateur mécanique en scriptant sous FreeCAD (/Hackable/HK-035/Simulation-d-un-ordinateur-mecanique-en-scriptant-sous-FreeCAD)

Hackable n° 35 (/Hackable/HK-035) octobre 2020

Par Friedt Jean-Michel (/auteur/Friedt-Jean-Michel), Testault Olivier (/auteur/Testault-Olivier), Carry Emile (/auteur/Carry-Emile)

Radio et wireless (/search/node?domains%5B0%5D=72465)

Code (/search/node?domains%5B0%5D=72464)

L'évolution du traitement du signal est une histoire fascinante largement déroulée par David Mindell dans ses divers ouvrages [1] et citations [2]. Partant de l'ordinateur mécanique avec ses rouages, poulies, bielles et crémaillères, le passage à l'électrique au début du 20ème siècle, puis à l'électronique intégrée avec l'avènement du transistor et des circuits intégrés (VLSI) nous ont fait oublier les stades initiaux qui ont amené à notre statut actuel d'ordinateurs infiniment puissants, ...

ABONNEMENTS

À PROPOS

Tél. : 03.67.10.00.20