


Mini écran connecté

Afficher des messages où que vous soyez avec cet écran connecté à internet.

 Difficulté **Moyen**

 Durée **1 heure(s)**

 Catégories **Électronique, Maison**

 Coût **7 EUR (€)**

Sommaire

Introduction

Video d'introduction

Étape 1 - Fabriquer l'écran connecté

Étape 2 - Paramétrer Adafruit IO

Étape 3 - Paramétrer les identifiants

Étape 4 - Test depuis Adafruit IO

Étape 5 - IFTTT - Afficher les notifications d'un téléphone

Étape 6 - Cacher vos identifiants dans le logiciel Arduino

Étape 7 - Changer les images

Étape 8 - Changer la police d'écriture

Étape 9 - Utiliser son propre serveur MQTT

Notes et références

Commentaires


Introduction

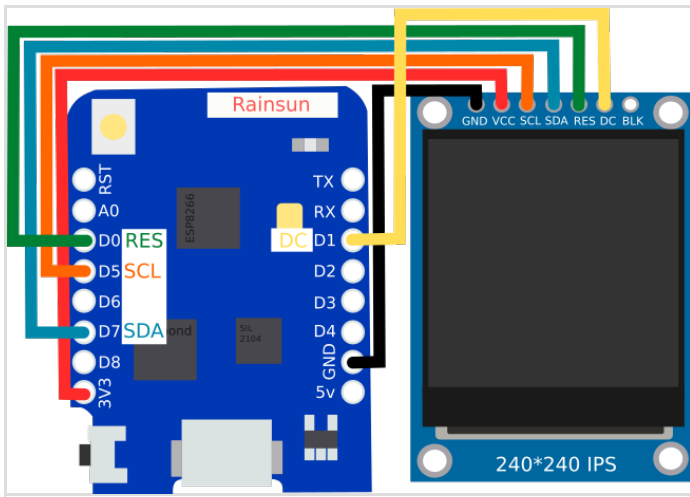
Précédemment, nous avons vu comment afficher une image sur un ST7789, un écran TFT à 3€. (Afficher une image sur un écran ST7789)
Nous allons exploiter cette connaissance pour faire un mini écran connecté.

- Cet écran va afficher les messages reçus depuis un serveur MQTT
- Les messages peuvent contenir des caractères accentués
- La transmission sera chiffrée
- Afin d'éviter les attaques MITM (man in the middle), nous allons vérifier l'identité du serveur MQTT à l'aide de son certificat.

À partir de là nous pouvons même utiliser IFTT (ou **tasker**) afin de transmettre des informations du **web** / de notre **installation domotique** ou de notre **téléphone** sur le serveur MQTT.

Afin de simplifier la partie MQTT, nous allons utiliser **Adafruit IO** qui permet d'avoir un **serveur MQTT gratuitement**.

 Vous pouvez bien évidemment utiliser votre propre serveur MQTT (la partie IFTT par contre ne marchera plus)



Matériaux

- ESP8266
- ST7789
- Câble Wrapping 30 AWG

Outils

- Fer à souder
- Imprimante 3D

Afficher une image sur un écran ST7789

Étape 1 - Fabriquer l'écran connecté

Suivez le tutoriel sur l'écran ST7789, avant toute chose, En résumé,

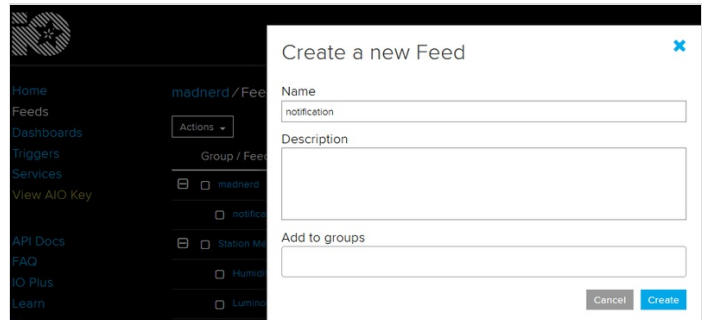
- il faut installer la **bibliothèque tft_espi**
- Changer le fichier **User_Setup.h** directement dans la bibliothèque
- Reliez l'écran à l'esp8266

Afficher une image sur un écran ST7789



Étape 2 - Paramétrer Adafruit IO

- Créer un compte sur Adafruit IO: <https://io.adafruit.com/>
- Cliquer sur **Feeds**
- Cliquer sur **Actions / Create a New Feed**
- Dans **Name** mettez **notifications**



Étape 3 - Paramétrer les identifiants

Tout d'abord il nous faut récupérer le programme

- Télécharger le croquis ici : https://github.com/maditnerd/st7789_mqtt
- Il nous faut aussi la bibliothèque : **Adafruit_MQTT**

Cette bibliothèque va nous permettre de communiquer avec notre serveur MQTT.

Dans le programme, il y a un fichier **arduino_secrets.h**, c'est ici que nous allons stocker nos identifiants pour le **Wi-Fi** et le **serveur MQTT**

Les identifiants sur Adafruit IO se trouvent en cliquant sur **View AIO Key**

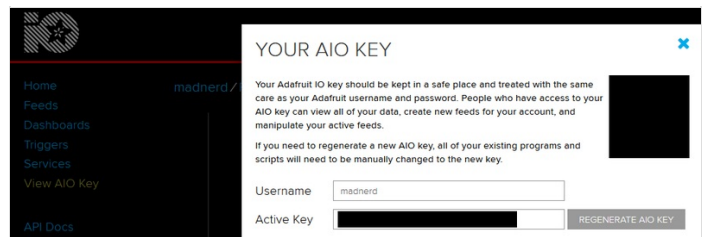
- Dans **HOME_SSID** mettez le nom de votre réseau WI-FI
- Dans **HOME_PASS** mettez le mot de passe de votre réseau WI-FI
- Dans **ADAFRUIT_MQTT_USERNAME** mettez le **Username** d'adafruit IO
- Dans **ADAFRUIT_MQTT_PASS** mettez l'**Active Key** d'adafruit IO

Téléverser le croquis pour tester si tout marche bien.

Normalement vous devriez arriver jusqu'à l'écran **Notifications**

 Vous pouvez vérifier sur le moniteur série (Baudrate : 115200) s'il y a un problème.

```
st7789_mqtt  arduino_secrets.h  arial12pt8b.h  bitmaps.h  connection.h
1 //Wifi Settings
2 #define HOME_SSID      ""
3 #define HOME_PASS     ""
4
5 //MQTT Settings
6 #define ADAFRUIT_MQTT_SERVER "io.adafruit.com"
7 #define ADAFRUIT_MQTT_PORT 8883
8 #define ADAFRUIT_MQTT_USERNAME ""
9 #define ADAFRUIT_MQTT_PASS ""
```



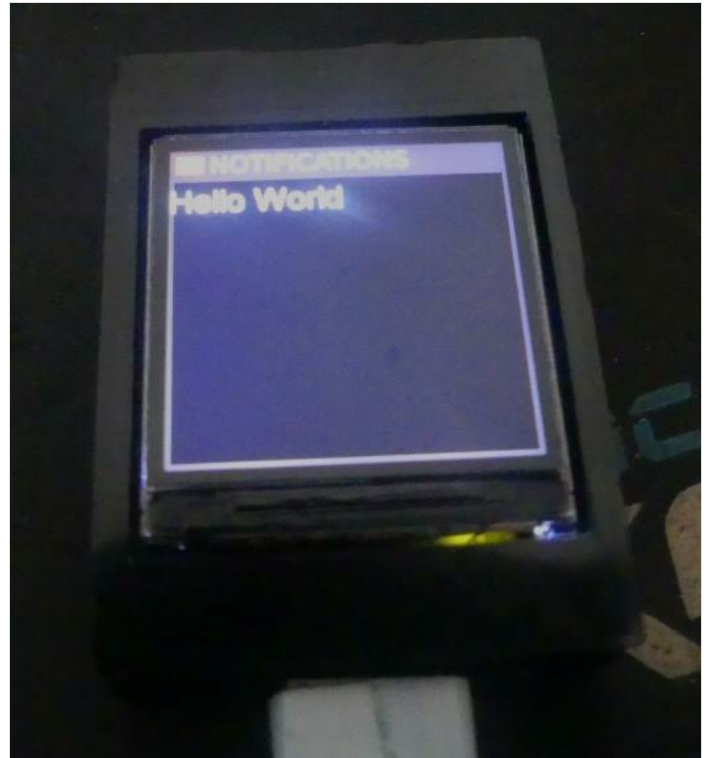
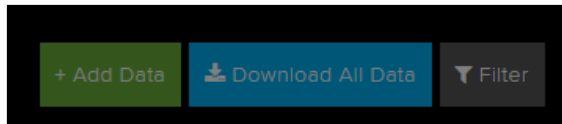


Étape 4 - Test depuis Adafruit IO

Adafruit IO nous permet de créer une donnée qui sera affichée sur l'écran.


- Cliquer sur **Feeds / Notifications**
- Puis cliquer sur **Add Data**
- Écrivez un message

i Si un message est envoyé alors que l'écran n'est pas connecté, celui-ci sera ignoré. MQTT permet toutefois de garder en mémoire les messages avec la fonction Retain




Étape 5 - IFTTT - Afficher les notifications d'un téléphone

IFTTT est un service qui permet d'automatiser des tâches, Adafruit IO est compatible avec celui-ci.

 Si vous voulez utiliser IFTTT avec votre propre serveur, il y a des webhooks qui permettent de faire cela.

Nous allons voir comment afficher les notifications d'un smartphone sur notre écran.

 Bien que ce soit amusant comme projet, n'oubliez pas que vous allez donner accès à vos notifications à deux services sur internet. Même si les communications sont en théorie sécurisée, niveau vie privée c'est une très mauvaise idée.


- Créer un compte sur IFTTT
- Installer l'application android

if

- Sur l'interface web d'IFTTT, cliquer sur **My Applets**
- Cliquer sur **New Applet**
- Choisissez le service **Android Device**
- Choisissez **Notification Received**

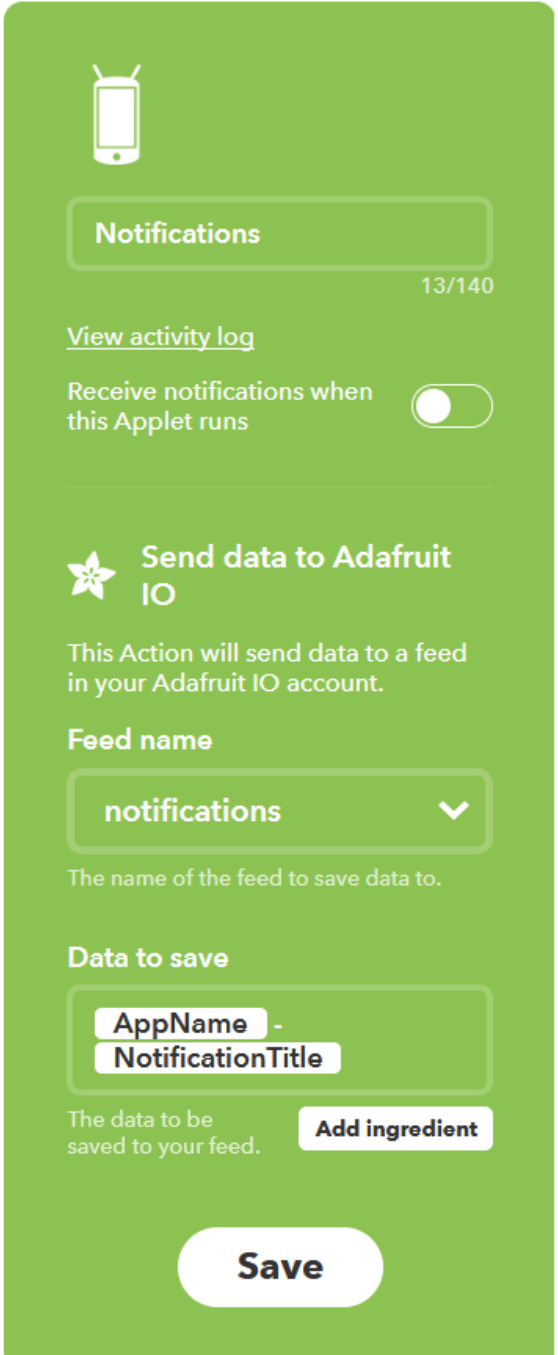
then

- Choisissez **Adafruit**
- Choisissez **Send data to Adafruit IO**
- Dans **Feed Name** mettez **notifications**
- Dans **Data to save** choisissez **AppName** et **Notification Title**

 L'ESP8266 va se déconnecter (puis se reconnecter) du serveur MQTT, si le message est trop long.

Votre applet devrait ressembler à ceci.

Aller sur votre téléphone, lancer IFTTT et autoriser l'accès aux notifications.



The screenshot shows the configuration page for an IFTTT applet named "Notifications". At the top, there is a mobile phone icon and a "Notifications" header with a "13/140" indicator. Below this is a "View activity log" link and a toggle switch for "Receive notifications when this Applet runs", which is currently turned off. The main section is titled "Send data to Adafruit IO" with a star icon. It explains that the action will send data to a feed in the user's Adafruit IO account. The "Feed name" is set to "notifications" in a dropdown menu. Below this, it says "The name of the feed to save data to." The "Data to save" section shows two selected ingredients: "AppName" and "NotificationTitle". There is an "Add ingredient" button next to them. At the bottom, there is a large "Save" button.

Étape 6 - Cacher vos identifiants dans le logiciel Arduino

Nous avons utilisé le fichier `arduino_secrets.h` qui est dans notre croquis pour **sauvegarder nos identifiants**.

! Il vaut mieux éviter de faire cela, car vous risquez de partager accidentellement vos mot de passe!

Une solution pour éviter ça et de créer une **bibliothèque** pour nos identifiants.

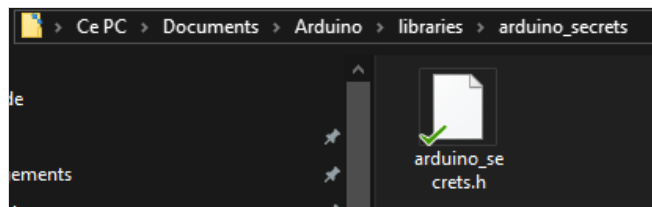
L'autre avantage c'est que nos identifiants seront **accessibles pour tous nos croquis** !

i C'est Andreas Spiess qui a eu cette idée
<https://www.youtube.com/watch?v=CAGQ8h8PKX4&t=347s>

- Créer dans `Documents/Arduino/libraries` un dossier `arduino_secrets`
- Copier `arduino_secrets.h` dans ce dossier

Dans le croquis, changez l'include au début du code

```
///include "arduino_secrets.h" include <arduino_secrets.h>
```



Étape 7 - Changer les images

Si vous voulez changer les images affichées, elles sont sauvegardées dans `bitmaps.h`

Pour convertir vos images en code, suivez ce tutoriel

Afficher une image sur un écran ST7789

Les images en version vectorielle SVG et PNG sont disponibles dans le dossier `images`

Pour afficher une image, il faut juste utiliser cette commande:

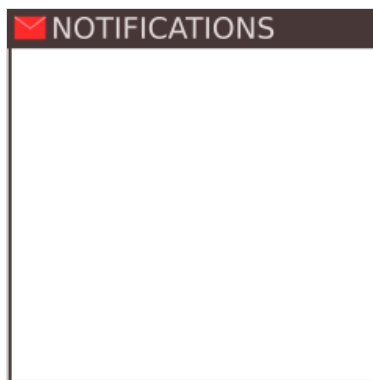
```
tft.pushImage(0,0,240,240,adafruit_io);
```

Pour écrire du texte sur plusieurs lignes, positionnez le texte puis écrivez le

```
tft.setCursor(0,50); tft.println(data); //Print text
```

Vous pouvez aussi écrire une seule ligne avec cette commande

```
tft.drawString(HOME_SSID, 25, 20);
```



Notes et références

Voilà ce tutoriel est fini, j'espère que cela vous donnera des idées pour faire des applications sympas avec.

Évidemment dans l'idéal

- Il faudrait pouvoir **paramétrer notre écran** depuis une **interface web**
- Éviter d'utiliser des **images** de la **taille de l'écran** quand ce n'est pas nécessaire

Mais cela devrait vous donner un bon point de départ.

Suivez-moi sur Twitter si vous voulez être au courant des prochains tutoriels en avance : <https://twitter.com/m4dnerd>