 **Warning:** Trying to access array offset on value of type null in **/home/clients/ee3f943e731fc0a12d3400116692186f/html/wp-content/plugins/unyson/framework/includes/option-types/typography-v2/class-fw-option-type-typography-v2.php** on line **148**

**Warning:** foreach() argument must be of type array|object, null given in **/home/clients/ee3f943e731fc0a12d3400116692186f/html/wp-content/plugins/unyson/framework/includes/option-types/typography-v2/class-fw-option-type-typography-v2.php** on line **148**

**Warning:** Trying to access array offset on value of type null in **/home/clients/ee3f943e731fc0a12d3400116692186f/html/wp-content/plugins/unyson/framework/helpers/general.php** on line **1275**

**Warning:** foreach() argument must be of type array|object, null given in **/home/clients/ee3f943e731fc0a12d3400116692186f/html/wp-content/plugins/unyson/framework/helpers/general.php** on line **1275**



SMART BUD

**Demo** (<https://bud.eco-sensors.ch>) | Tutoriels (<https://eco-sensors.ch/tutoriels/>)



NOUS SOUTENIR

**Shop** (<https://eco-sensors.ch/shop/>) & Sponsoring (<https://eco-sensors.ch/product-category/dons/>)



SOCIAL

**Facebook** (<https://www.facebook.com/ecosensors>) - **Contact** (<https://eco-sensors.ch/contact/>)

# Un raspberry Zero, un LCD et LoRa pour recevoir des données rfm95

**HOME ([HTTPS://ECO-SENSORS.CH/](https://eco-sensors.ch/))**  
**/ LORAWAN / LORA ([HTTPS://ECO-SENSORS.CH/CATEGORY/LORAWAN/](https://eco-sensors.ch/category/lorawan/))**  
**/ UN RASPBERRY ZERO, UN LCD ET LORA POUR RECEVOIR DES DONNÉES RFM95**

**BY ECOSENSORS ([HTTPS://ECO-SENSORS.CH/AUTHOR/ECOSENSORS/](https://eco-sensors.ch/author/ecosensors/))**

/

**25 DÉCEMBRE 2017 ([HTTPS://ECO-SENSORS.CH/UN-RASPBERRY-ZERO-UN-LCD-ET-LORA-POUR-RECEVOIR-DES-DONNEES-RFM95/](https://eco-sensors.ch/un-raspberry-zero-un-lcd-et-lora-pour-recevoir-des-donnees-rfm95/))**

 0  5,709  2  

**LORAWAN / LORA ([HTTPS://ECO-SENSORS.CH/CATEGORY/LORAWAN/](https://eco-sensors.ch/category/lorawan/))**  French





Dans cet article, vous allez apprendre à configurer un Raspberry Zero W avec un carte « chapeau » Radio RF95. Vous allez aussi le faire fonctionner en mode de réception pour qu'il reçoive les paquets envoyés par le Joystick (ou par un autre module LoRa) que j'ai réalisé et configuré dans cet article : un joystick pour orienter un petit robot avec lora (<http://smart-idea.io/un-joystick-pour-orienter-un-petit-robot-avec-lora/>). Vous allez encore configurer un écran écran Oled I2C 128×64 pour afficher les paquets reçus et des messages.

Pour l'installation du Raspberry, veuillez suivre cet article: <https://eco-sensors.ch/2-faire-ses-sauvegardes-sans-connexion/> ([https://eco-sensors.ch/2-faire-ses-sauvegardes-sans-connexion/](https://eco-sensors.ch/2-faire-ses-sauvegardes-sans-connexion/#preparation) #preparation)

## Matériel

- Raspberry Zero W (<https://www.raspberrypi.org/products/raspberry-pi-zero-w/>)
- RFM 95 Lorawan/TTN hat (<https://www.tindie.com/products/>)

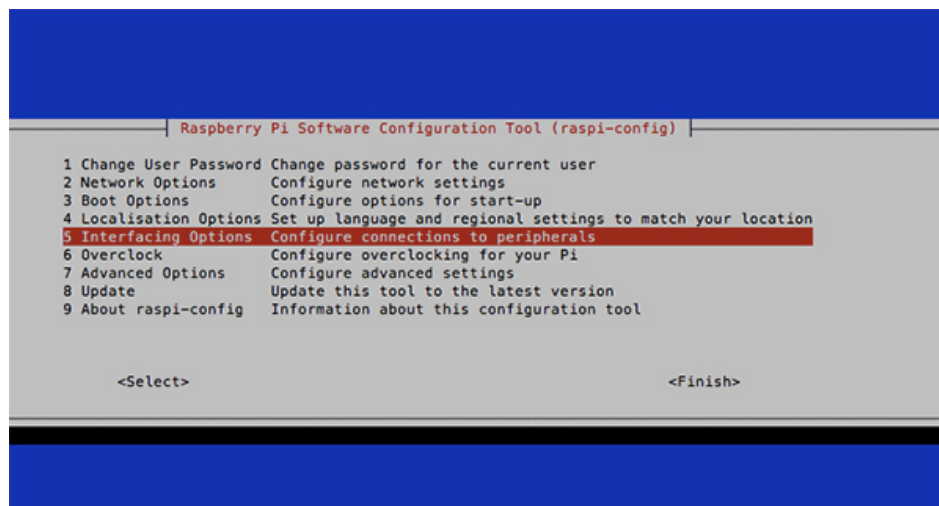
# LCD OLED 128×64 I2C

Référence: <https://hallard.me/adafruit-oled-display-driver-for-pi/>  
(<https://hallard.me/adafruit-oled-display-driver-for-pi/>)

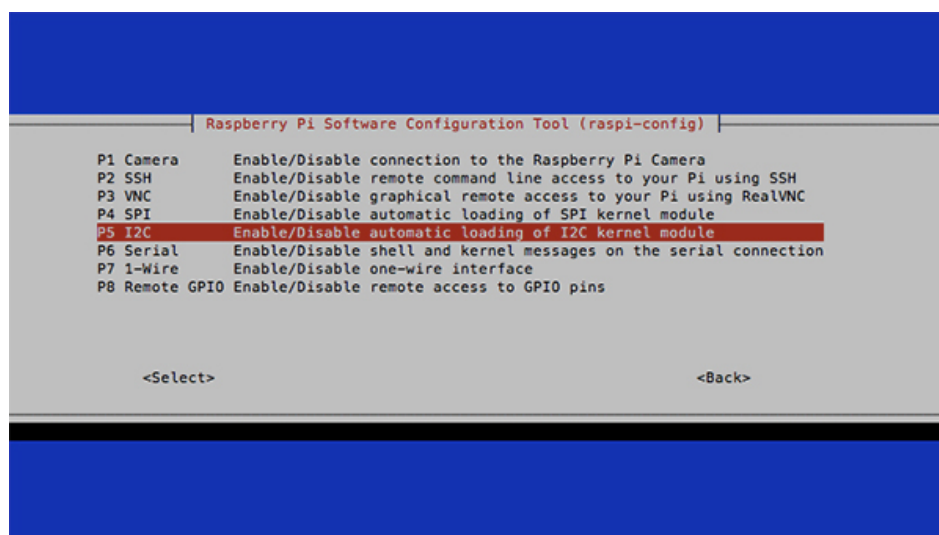
Nous allons d'abord vérifier que I2C est activé sur votre Raspberry. Ouvrez votre terminal et tapez la commande suivante

```
1 sudo raspi-config
```

Sélectionner **5 Interfacing Option**



Puis **P5 I2C**

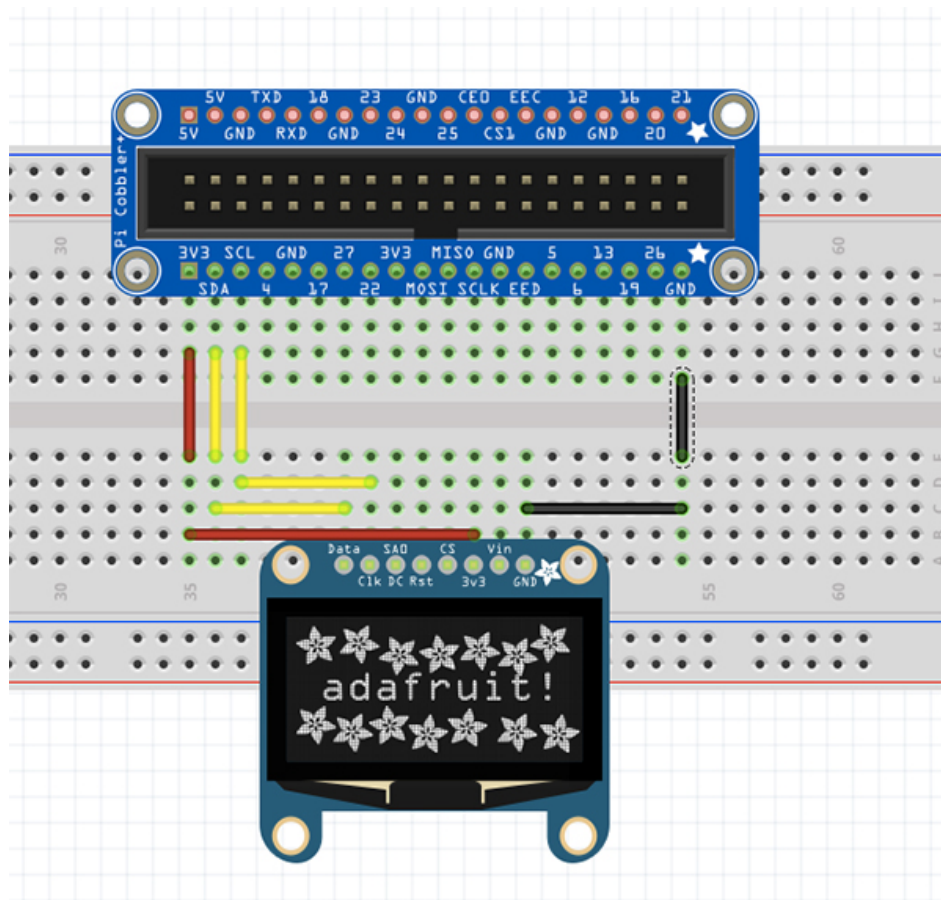


Puis, répondez YES pour l'activer. Puis cliquez sur <Finish>

 (/)   (/)

# Câblage

Référence: <https://learn.adafruit.com/monochrome-oled-breakouts/wiring-1-dot-3-128x64#using-with-i2c> (<https://learn.adafruit.com/monochrome-oled-breakouts/wiring-1-dot-3-128x64#using-with-i2c>)






## Installation des librairies (C) et de git

```
1 sudo apt-get install build-essential git-core
```

# Installation d'ArduiPi\_OLED

Note: ArduiPi\_OLED ([https://github.com/hallard/ArduiPi\\_OLED](https://github.com/hallard/ArduiPi_OLED))

 ([/](#))   ([/](#))  
replace ArduiPi\_SSD1306 ([https://github.com/hallard/ArduiPi\\_SSD1306](https://github.com/hallard/ArduiPi_SSD1306))

```
1 mkdir ~/Soft
2 cd ~/Soft
3 git clone https://github.com/hallard/ArduiPi_()
4 cd ArduiPi_OLED/
5 sudo make
```



Le nouveau script compilera la librairie (libArduPi\_OLED \*) et l'installera pour vous (c'est pourquoi nous devons utiliser sudo) dans

```
/usr/local/lib/
```

et en-têtes de bibliothèque dans

```
/usr/local/include/
```

## Testons!

Allez dans

```
1 cd ~/Soft/ArduPi_OLED/examples
```

et compilez

```
1 sudo make
```

puis lancez le fichier de démo pour un écran 128×64  
fonctionnant sur I2C

```
1 sudo ./oled_demo --verbose --oled 3
```

De mon côté, ça marche très bien

# Radio RFM95

Installation des librairies C pour BCM 2835

```
1 mkdir ~/Soft
2 wget http://www.airspayce.com/mikem/bcm2835/bc
3 tar zxvf bcm2835-1.52.tar.gz
4 cd bcm2835-1.52
5 sudo ./configure
6 sudo make
7 sudo make check
```

```
8 sudo make install
```



(/)



(/)

## RadioHead

Référence : <https://github.com/hallard/RadioHead> (<https://github.com/hallard/RadioHead>)

```
1 cd ~/Soft/
```

```
2 git clone https://github.com/hallard/RadioHead
```

 French



```
3 cd RadioHead/  
4 cd ~/Soft/RadioHead/examples/raspi  
5 sudo nano RasPiBoards.h
```

Il faut maintenant corriger la définition des pins pour qu'elle correspond à mon PCB Radio RF95

Commentez et ajoutez ces lignes

```
// Module 1, 2 and 3 are example of module type soldered on  
// change to fit your needs  
  
// Module 1 on board RFM95 868 MHz (example)  
// #define MOD1_LED_PIN RPI_V2_GPIO_P1_07 // Led on GPIO4 soldered on pin 07  
// #define MOD1_CS_PIN RPI_V2_GPIO_P1_24 // Slave Select on pin 24  
// #define MOD1_IRQ_PIN RPI_V2_GPIO_P1_22 // IRQ on GPIO25 soldered on pin 22  
// #define MOD1_RST_PIN RPI_V2_GPIO_P1_29 // Reset on GPIO5 soldered on pin 29  
  
#define MOD1_LED_PIN RPI_V2_GPIO_P1_18  
#define MOD1_CS_PIN RPI_V2_GPIO_P1_08  
#define MOD1_IRQ_PIN RPI_V2_GPIO_P1_25  
#define MOD1_RST_PIN RPI_V2_GPIO_P1_05
```

Pour quitter et sauver : **<ctrl+X>**, puis **<maj+Y>**, puis **<enter>**

Ensuite nous allons scanner votre carte

```
1 cd /Soft/RadioHead/examples/raspi/spi_scan  
2 sudo make  
3 sudo ./spi_scan
```

Si tout c'est bien passé, vous devriez voir ceci:

```
Checking register(0x42) with CS=GPI006 => Nothing!  
Checking register(0x10) with CS=GPI006 => Nothing!  
Checking register(0x42) with CS=GPI007 => Nothing!  
Checking register(0x10) with CS=GPI007 => Nothing!  
Checking register(0x42) with CS=GPI008 => SX1276 RF95/96 (V  
Checking register(0x10) with CS=GPI008 => Nothing!  
Checking register(0x42) with CS=GPI026 => Nothing!  
Checking register(0x10) with CS=GPI026 => Nothing!
```

Votre carte radio a bien été vue à la GPIO08

Maintenant, nous allons le faire fonctionner en mode récepteur.  
C'est à dire qu'il va attendre qu'un émetteur lui envoie des paquets.

```
1 cd ~/Soft/RadioHead/RadioHead/examples/raspi/  
2 sudo make  
3 sudo ./rf95_server
```



Je vais utiliser le Joystick que j'ai fait dans cet article : un joystick pour orienter un petit robot avec lora (<https://eco-sensors.ch/un-joystick-pour-orienter-un-petit-robot-avec-lora/>) pour que ce Joystick lui indique sa position (haut, bas, gauche, droite).

J'ai le grand bonheur de constater que lorsque le Jostick est bougé de haut en bas et de gauche à droite, mon petit Raspberry Zero W affiche ces informations

```
pierrot@smartidea2:~/Soft/RadioHead/RadioHead/examples/rasp
rf95_server
RF95 CS=GPI08, IRQ=GPI025, RST=GPI022, LED=GPI023 OK NodeID
Listening packet...
Packet[20] #255 => #255 -34dB: 75 70 00 00 00 00 00 00 00 00 00
Packet[20] #255 => #255 -34dB: 72 75 70 00 00 00 00 00 00 00 00
Packet[20] #255 => #255 -35dB: 64 6F 77 6E 00 00 00 00 00 00 00
Packet[20] #255 => #255 -35dB: 72 64 6F 77 6E 00 00 00 00 00 00
Packet[20] #255 => #255 -36dB: 75 70 00 00 00 00 00 00 00 00 00
Packet[20] #255 => #255 -35dB: 72 75 70 00 00 00 00 00 00 00 00
Packet[20] #255 => #255 -24dB: 75 70 00 00 00 00 00 00 00 00 00
Packet[20] #255 => #255 -24dB: 72 75 70 00 00 00 00 00 00 00 00
Packet[20] #255 => #255 -29dB: 72 69 67 68 74 00 00 00 00 00 00
Packet[20] #255 => #255 -28dB: 72 72 69 67 68 74 00 00 00 00 00
```

Ceci indique bien qu'il reçoit bien les paquets du Joystick via la technologie LoRa TF95 :o)

Nous allons voir plus bas, comment démarrer cette fonctionnalité automatiquement au démarrage du Raspberry, sans que vous aillez à tapez la commande

```
1 sudo ./rf95_server
```

## Modification du Makefile RadioHead pour afficher des textes sur l'écran OLED

Nous allons modifier maintenant un fichier. Avant, nous allons faire une copie afin de garder l'original. Ouvrez votre terminal et taper la commande suivante

```
1 cd ~/Soft/RadioHead/Soft/RadioHead/examples/rf95_server
2 cp rf95_server.cpp rf95_server.ccpcopy
3 cd ~/Soft/RadioHead/Soft/RadioHead/examples/rf95_server
4 cp Makefile Makefilecopy
```




Nous allons ensuite éditer le fichier Makefile

```
1 cd ~/Soft/RadioHead/examples/raspi/rf95
2 sudo nano Makefile
```

et nous allons remplacer la ligne

```
LIBS = -lbcm2835
```

par

 LIBS = -lbcm2835 -lArduiPi\_OLED  
 

Pour rappel, pour quitter et sauver le fichier après vos modifications, il faut appuyer sur les touche **<ctrl>+x** et ensuite **<ctrl>+O** ou **<ctrl>+Y**

Nous allons encore éditer le fichier rf95\_server.cpp

```
1 cd ~/Soft/RadioHead/examples/raspi/rf95
2 sudo nano rf95_sever.ccp
```

et nous allons ajouter ces lignes juste dessous les autres lignes qui commencent par #include ...

```
1  /*****
2  * For OLED LCD *
3  *****/
4  #include "ArduPi_OLED_lib.h"
5  #include "Adafruit_GFX.h"
6  #include "ArduPi_OLED.h"
7  #include <getopt.h>
8  // Instantiate the display
9  ArduPi_OLED display;
10 // Config Option
11 struct s_opts
12 {
13     int oled;
14     int verbose;
15 } ;
16 int sleep_divisor = 1 ;
17 // default options values
18 s_opts opts = {
19     OLED_ADAFRUIT_I2C_128x64, // Default oled (
20     false // Not verbose
21 };
```

C'est pas fini. Recherchez la fonction `main()` en faisant une recherche avec les touches **<ctrl>+W** (le critère de recherche est **'main ('**)

et ajoutez ceci:

**⚠ ATTENTION N'AJOUTEZ QUE CE QU'IL SE TROUVE ENTRE  
/\* FOR OLED \*/ ET /\* END OLED \*/**

```
1 //Main Function
2 int main (int argc, const char* argv[] )
3 {
4
5     /* FOR OLED */
6
7     // I2C change parameters to fit to your LCD
8     if ( !display.init(OLED_I2C_RESET,opts.oled)
9     {
10     exit(EXIT_FAILURE);
11     };
12     display.begin();
13
14     display.clearDisplay(); // clears the screen
15
16     // text display tests
17     display.setTextSize(1);
18     display.setTextColor(WHITE);
```



```

19 display.setCursor(0,0);
20 display.print("Welcome aboard\n");
21 display.print("EcoSensors\n");
22 display.print("\n");
23 display.print("Starting\n");
24 display.print("rf95_server\n");
25 display.display();
26 sleep(3);
27
28 /* END OLED */
29
30 unsigned long led_blink = 0;
31

```

compilez

```

1 cd ~/Soft/RadioHead/examples/raspi/rf95
2 sudo make

```

et exécutez rf95\_server

```
1 $ sudo ./rf95_server
```

 (/)   (/)

et vous devriez voir le message s'afficher

```
Welcome aboard  
EcoSensors  
  
Starting  
rf95_server
```

De mon côté, ça marche!

Voici quelques fonctions

```
1 display.setTextSize(1); // Taille du texte  
2 display.setTextColor(WHITE); // Couleur du te
```



```
3 display.setTextColor(WHITE, BLACK); // Inverse
4 display.clearDisplay(); // Efface l'écran
5 display.setCursor(0,0); // Positionne le curseur
6 display.print("message"); // Message à afficher
7 display.display(); // Affiche les print() précédents
```

1 if (rf95.available()) {}  
🗑️ (/) 🔍 👤 (/)

plus récemment ici: [https://github.com/pierrot10/RadioHead/blob/master/examples/raspi/rf95/rf95\\_server.cpp#L179](https://github.com/pierrot10/RadioHead/blob/master/examples/raspi/rf95/rf95_server.cpp#L179) ([https://github.com/pierrot10/RadioHead/blob/master/examples/raspi/rf95/rf95\\_server.cpp#L179](https://github.com/pierrot10/RadioHead/blob/master/examples/raspi/rf95/rf95_server.cpp#L179)).

Sous la ligne 179, ajoutez ceci:

```
1 /* OLED */
2 display.clearDisplay();
3 display.setTextColor(BLACK, WHITE); // 'inverted'
4 display.setCursor(0,0);
5 display.print(" Listing ");
6 display.setTextColor(WHITE, BLACK); // 'inverted'
7 display.print("Packet[");
8
9 snprintf(buf_print,bufprintsize,"%d",len);
10 display.print(buf_print);
11 display.print("]\n");
12 display.print("#");
13
14 snprintf(buf_print,bufprintsize,"%d",from);
15 display.print(buf_print);
16 display.print(" => ");
17
18 snprintf(buf_print,bufprintsize,"%d",to);
19 display.print(buf_print);
20 display.print("\n");
21 display.print("rssi:");
22
23 snprintf(buf_print,bufprintsize,"%d",rssi);
24 display.print(buf_print);
25 display.print("\n\n");
26
27 snprintf(buf_print,bufprintsize,"%s",buf);
28 display.print(buf_print);
29 display.print("\n");
30 display.display();
```

Je vous laisse voir vous, comment faire mieux et où vous jugez  
utilise d'afficher d'autres messages sur votre écran.

## rf95\_server en tant que service

Pour que le Raspberry fonction automatiquement en mode de  
réception des paquets, dès qu'il a démarré, il faut créer un  
nouveau service.

Créez le fichier suivant

```
1 sudo nano /lib/systemd/system/rf95_server.serv
```

Coller ce texte (modifiez le chemin *ExecStart*, si nécessaire)

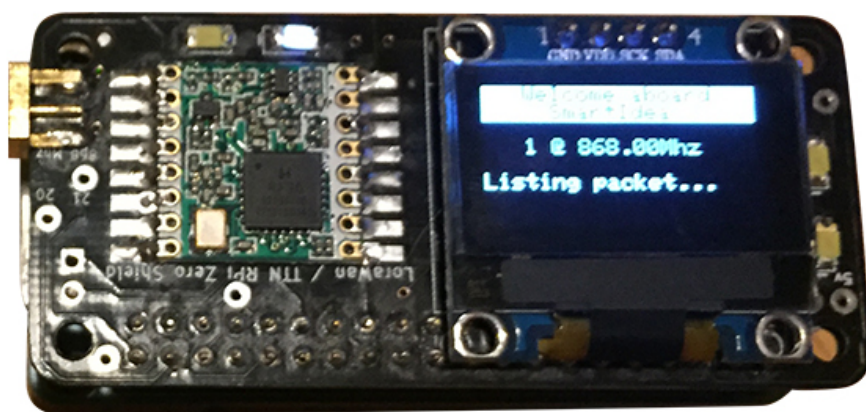
```
1 [Unit]
2 Description=Launch welcome RF95 server to lis
3 After=multi-user.target
4
5 [Service]
6 Type=idle
7 ExecStart=/home/pierrot/Soft/RadioHead/example
8 [Install]
9 WantedBy=multi-user.target
```



Modifiez les droits, recharger le daemon, activez le nouveau service et redémarrez votre Raspberry

```
1 sudo chmod 644 /lib/systemd/system/rf95_server.service
2 sudo systemctl daemon-reload
3 sudo systemctl enable rf95_server.service
4 sudo reboot
```

Vous devriez voir ceci s'afficher



*reapsberry zeo rfm95*

# Autres tests

J'ai également alimenté mon Joystick (<https://eco-sensors.ch/un-joystick-pour-orienter-un-petit-robot-avec-lora/>) et le module (récepteur) reçoit et affiche bien « up » quand le joystick (émetteur) poussé vers le haut, ainsi que down, left, right, etc...

Finalement, j'ai préparé un Feather MO Radio avec un module LoRa (<https://learn.adafruit.com/adafruit-feather-m0-radio-with-lora-radio-module/pinouts?view=all>) (émetteur) que j'ai, en plus, glissé dans un tube en aluminium :o). Le module Raspberry (récepteur) reçoit aussi les paquets avec un RSSI de -22 à -35, quand il est dans le tube, malgré le fait que le tube soit un obstacle aux ondes radio. Je n'ai malheureusement pas pu éloigner l'émetteur de plus de 4m.

## Bravo!!!

Vous venez de configurer votre Raspberry pour qu'il affiche du texte. Mais vous venez surtout de configurer votre Raspberry pour qu'il puisse recevoir des paquets Radio avec LoRa. Vous avez aussi modifié la librairie RadioHead pour que le contenu des paquets reçus s'affichent sur l'écran OLED. Vous avez aussi fait en sorte que votre module écoute les paquets radio envoyés dès sont démarrage!

## Soutenez-nous

Si vous avez aimé cet article, faites un don (<https://eco-sensors.ch/product/don/>) de quelques Euro pour le financement de mes projets et du matériel.

*Ceci est conservé comme historique des opérations qui ont été effectuées mais qui ne sont plus d'actualité dans le cadre de cet article*

## Installation des librairies (Python):

Référence : <https://learn.adafruit.com/ssd1306-oled-displays-with-raspberry-pi-and-beaglebone-black/usage> (<https://learn.adafruit.com/ssd1306-oled-displays-with-raspberry-pi-and-beaglebone-black/usage>)

Comme nous utilisons un Raspberry, ouvrez votre terminal et installez les librairies suivantes:

```
1 sudo apt-get update
2 sudo apt-get install build-essential python-d
3 sudo pip install RPi.GPIO
```

Puis installez encore ces deux librairies dont python-imaging  
(<http://effbot.org/imagingbook/pil-index.htm>)

```
1 sudo apt-get install python-imaging
2 sudo apt-get install python-smbus
```

Si ce n'est pas encore fait, installez git

```
1 sudo apt-get install git
```

Puis télécharger **Adafruit\_Python\_SSD1306**

```
git clone https://github.com/adafruit/Adafruit_Python_SSD1306
cd Adafruit_Python_SSD1306
```

et installez-le

```
1 cd ~/Soft/SSD1306/Adafruit_Python_SSD1306
2 sudo python setup.py install
```

## Usage

Référence : <https://learn.adafruit.com/ssd1306-oled-displays-with-raspberry-pi-and-beaglebone-black/usage> ([https://](https://learn.adafruit.com/ssd1306-oled-displays-with-raspberry-pi-and-beaglebone-black/usage)  
[learn.adafruit.com/ssd1306-oled-displays-with-raspberry-pi-and-beaglebone-black/usage](https://learn.adafruit.com/ssd1306-oled-displays-with-raspberry-pi-and-beaglebone-black/usage))





beaglebone-black/usage)

 (/)   (/)

Cet exemple est basé sur le lien ci-dessus. Je vais faire un exemple simplifié qui ne vas qu'afficher un message de bienvenue, quand vous allumez votre Raspberry.

Créez un fichier welcome.py et éditez-le

```
1 sudo mkdir ~/Python/welcome.py
2 sudo vi ~/Python/welcome.py
```

Vous trouverez sur Github cet exemple (<https://github.com/pierrot10/oled/blob/master/welcome.py>) qui est suffisamment commenté pour compléter le fichier `welcome.py`

Vous pourrez l'exécuter avec cette commande

```
1 sudo python ~/Python/welcome.py
```

## Affichage du texte au démarrage de votre Raspberry

Référence: <https://www.raspberrypi-spy.co.uk/2015/10/how-to-autorun-a-python-script-on-boot-using-systemd/> (<https://www.raspberrypi-spy.co.uk/2015/10/how-to-autorun-a-python-script-on-boot-using-systemd/>)

Je vais finalement déplacer le fichier `welcome.py` dans `/opt`

```
1 sudo mv ~/Python/welcome.py /opt/
```

Puis, je vais créer un lien symbolique dans ~/Python

```
1 cd ~/Python
2 ln -s /opt/welcome.py welcome.py
```

Ce qui revient presque au même sauf que maintenant nous avons un lien dans ~/Python qui pointe sur le fichier qu'on a créé qui se trouve maintenant dans /opt.

Vous devez encore rendre le fichier exécutable en exécutant cette commande

```
1 sudo chmod +x /opt/welcome.py
```

Vous pouvez faire comme vous voulez. Moi j'ai préféré regrouper dans /opt, mes applications pour qu'elles ne dépendent pas des dossiers qui se trouvent dans les dossiers personnels.

## Option 1:

 (/)   (/)

Editez le fichier /etc/rc.local et ajoutez la ligne avant # Print...

```
1 python /opt/welcome.py
2 # Print the IP address
```

Puis la commande

```
1 sudo rapsi-config
```

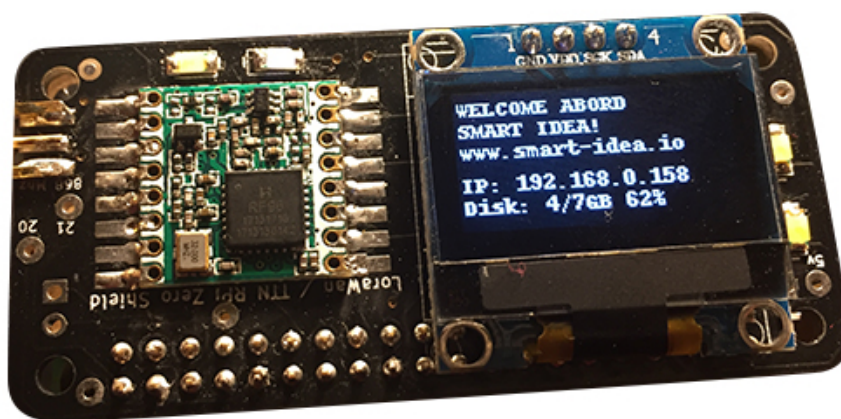


Choisissez **3 Boot options** puis **B2 Wait for Network at Boot** et activez-le.

Redémarrer votre Raspberry

1 `sudo reboot`

Et vous devriez voir ceci:



## Option 2:

Pour que ce fichier soit exécuté quand votre Raspberry démarre, **il faut donc le démarrer en tant que service.**

Pour cela, créez et éditez le fichier suivant

```
1 sudo nano /lib/systemd/system/welcome.service
```

Ajoutez les lignes suivantes

```

[Unit]
Description=Launch welcome script which print IP and disk s
After=multi-user.target

[Service]
Type=idle
ExecStart=/usr/bin/python /opt/welcome.py > /var/log/oled-w
[Install]
WantedBy=multi-user.target

```

Puis tapez les touche **<ctrl>+X**, **<maj>+Y**, **<enter>**, pour quitter en sauvant.

Changez encore les droits de ce fichier

```
1 sudo chmod 644 /lib/systemd/system/welcome.se
```

Pour terminer, vous devez encore taper ces deux commandes dans votre terminal

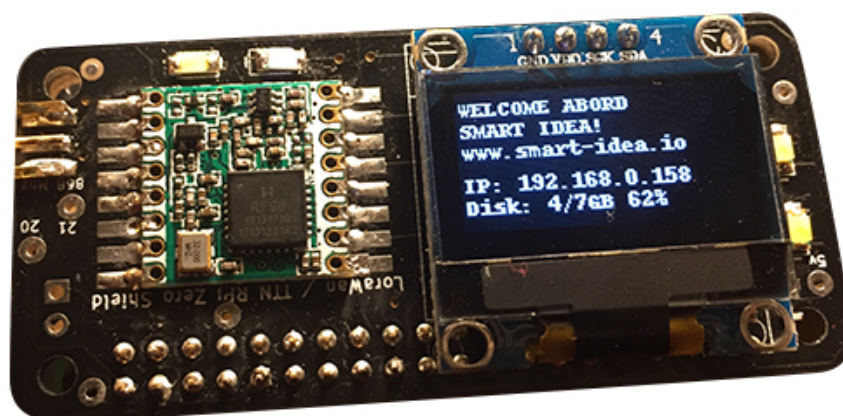
```
1 sudo systemctl daemon-reload
2 sudo systemctl enable welcome.service
```

pour recharger le daemon et activer votre nouveau service.

Redémarrer votre Raspberry

**1** `sudo reboot`

Et vous devriez voir ceci:



128X64 ([HTTPS://ECO-SENSORS.CH/TAG/128X64/](https://eco-sensors.ch/tag/128X64/))  
ARDUINO ([HTTPS://ECO-SENSORS.CH/TAG/ARDUINO/](https://eco-sensors.ch/tag/arduino/))  
ARDUINO\_OLED ([HTTPS://ECO-SENSORS.CH/TAG/ARDUINO\\_OLED/](https://eco-sensors.ch/tag/arduino-oled/))  
ARDUINO\_SSD1306 ([HTTPS://ECO-SENSORS.CH/TAG/ARDUINO\\_SSD1306/](https://eco-sensors.ch/tag/arduino-ssd1306/))  
GPIO ([HTTPS://ECO-SENSORS.CH/TAG/GPIO/](https://eco-sensors.ch/tag/gpio/))  
HAT ([HTTPS://ECO-SENSORS.CH/TAG/HAT/](https://eco-sensors.ch/tag/hat/))  
LCD ([HTTPS://ECO-SENSORS.CH/TAG/LCD/](https://eco-sensors.ch/tag/lcd/))  
LORA ([HTTPS://ECO-SENSORS.CH/TAG/LORA/](https://eco-sensors.ch/tag/lora/))  
MAKEFILE ([HTTPS://ECO-SENSORS.CH/TAG/MAKEFILE/](https://eco-sensors.ch/tag/makefile/))  
OLED ([HTTPS://ECO-SENSORS.CH/TAG/OLED/](https://eco-sensors.ch/tag/oled/))  
PYTHON ([HTTPS://ECO-SENSORS.CH/TAG/PYTHON/](https://eco-sensors.ch/tag/python/))  
RADIO ([HTTPS://ECO-SENSORS.CH/TAG/RADIO/](https://eco-sensors.ch/tag/radio/))  
RADIOHEAD ([HTTPS://ECO-SENSORS.CH/TAG/RADIOHEAD/](https://eco-sensors.ch/tag/radiohead/))  
RASPBERRY ([HTTPS://ECO-SENSORS.CH/TAG/RASPBERRY/](https://eco-sensors.ch/tag/raspberry/))  
RFM95 ([HTTPS://ECO-SENSORS.CH/TAG/RFM95/](https://eco-sensors.ch/tag/rfm95/))  
RFM95\_SERVER ([HTTPS://ECO-SENSORS.CH/TAG/RFM95\\_SERVER/](https://eco-sensors.ch/tag/rfm95-server/))  
SERVICE ([HTTPS://ECO-SENSORS.CH/TAG/SERVICE/](https://eco-sensors.ch/tag/service/))  
ZERO ([HTTPS://ECO-SENSORS.CH/TAG/ZERO/](https://eco-sensors.ch/tag/zero/))

PREV POST

NEXT POST

(<https://eco-sensors.ch/snap-si-tu-bouges-je-timmortalise/>)

(<https://eco-sensors.ch/un-raspberry-pour-lire-vos-cartes-rfid-nfc/>)

Christophe

19 DÉCEMBRE 2019 ([HTTPS://ECO-SENSORS.CH/UN-RASPBERRY-ZERO-UN-LCD-ET-LORA-POUR-RECEVOIR-DES-DONNEES-RFM95/#COMMENT-340](https://eco-sensors.ch/un-raspberry-zero-un-lcd-et-lora-pour-recevoir-des-donnees-rfm95/#COMMENT-340))

Merci pour ton tuto, il m'a été très utile pour réaliser ma Gateway avec la PCB de Philippe Cadic que je connais bien. Pour ma part mon projet consiste à connecter une ruche sur un réseau lora

**REPLY ([HTTPS://ECO-SENSORS.CH/UN-RASPBERRY-ZERO-UN-LCD-ET-LORA-POUR-RECEVOIR-DES-DONNEES-RFM95/?REPLYTOCOM=340#RESPOND](https://eco-sensors.ch/un-raspberry-zero-un-lcd-et-lora-pour-recevoir-des-donnees-rfm95/?REPLYTOCOM=340#RESPOND))**

EcoSensors

5 MARS 2020 ([HTTPS://ECO-SENSORS.CH/UN-RASPBERRY-ZERO-UN-LCD-ET-LORA-POUR-RECEVOIR-DES-DONNEES-RFM95/#COMMENT-341](https://eco-sensors.ch/un-raspberry-zero-un-lcd-et-lora-pour-recevoir-des-donnees-rfm95/#COMMENT-341))

 French





Merci pour votre commentaire. Votre projet est très intéressant. Avez-vous un site internet qui le présente? Bonne soirée

**REPLY (HTTPS://ECO-SENSORS.CH/UN-RASPBERRY-ZERO-UN-LCD-ET-LORA-POUR-RECEVOIR-DES-DONNEES-RFM95/?REPLYTOCOM=341#RESPOND)**

2 comments **In This Topic:**

*FULL NAME*

*EMAIL ADDRESS*

*PHONE NUMBER*

*YOUR COMMENT*

**SUBMIT NOW**



