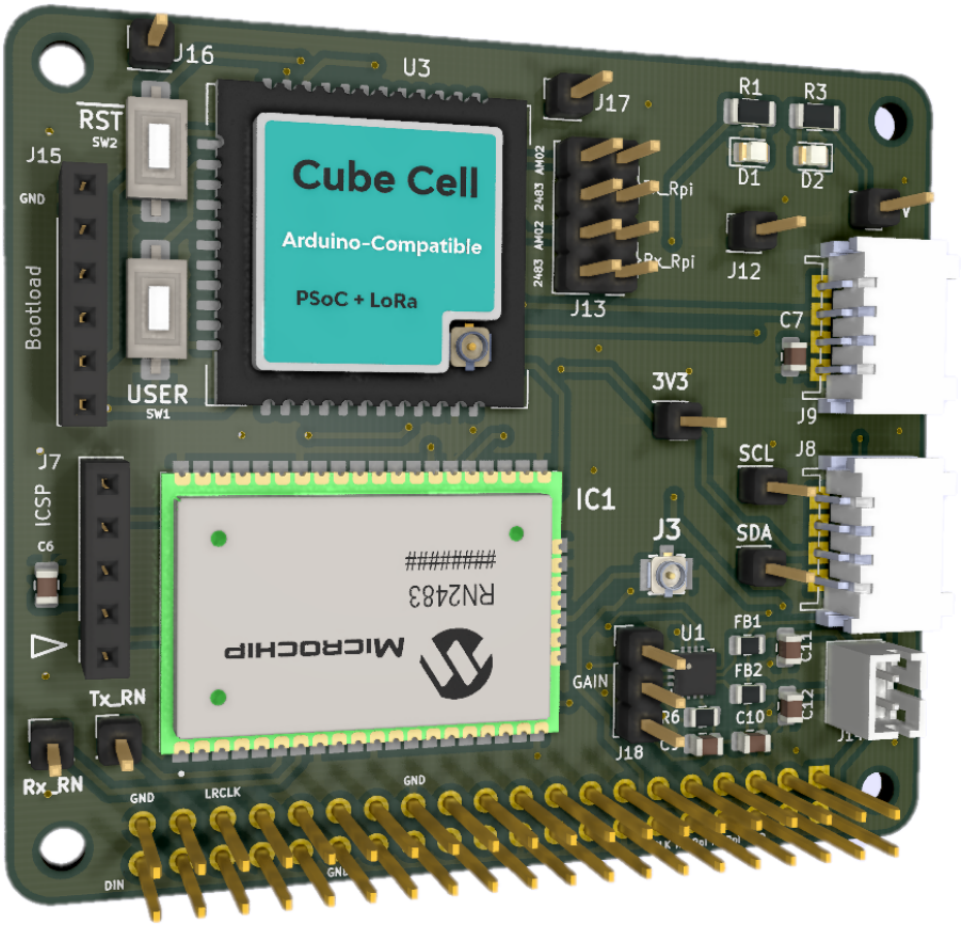
MyIAQ : Manuel d'installation

1. Installation du matériel

L'installation du matériel consiste à :

1. connecter la carte *hat* développée pour le projet à la *Raspberry Pi*



2. téléverser le *firmware* du transceiver LoRa HTCC-AM02 de chez *Heltec*
3. configurer les cavaliers présents sur la carte d'extension (→ *hat*)

1.1. Mise en place du *hat*

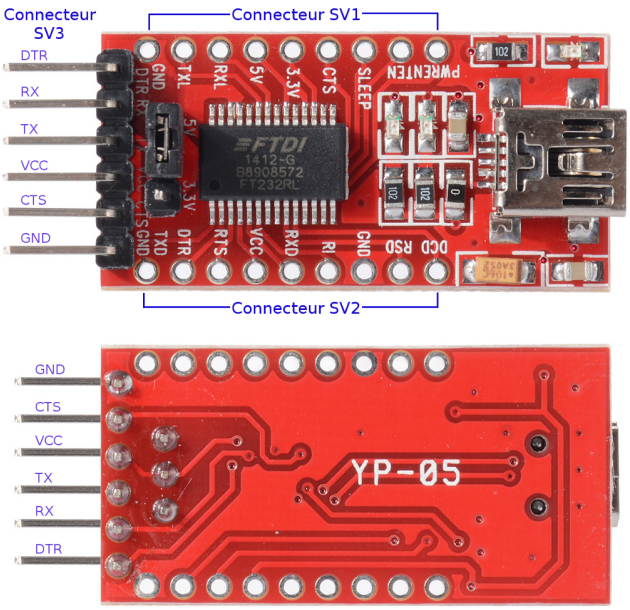
Insérer le *hat* sur le connecteur GPIO de la *Raspberry Pi* en veillant à bien aligner les 2 connecteurs.

1.2. Téléversement du *firmware* du HTCC-AM02

Le *firmware* du HTCC-AM02 est un *sketch* de type *Arduino*.

Pour le téléverser dans le composant :

1. Ôter les cavaliers présents sur J13
2. relier le connecteur J15 du *hat* au port USB d'un PC — sur lequel est installé *VSCode* et *PlatformIO* — via un adaptateur USB ↔ série TTL 3.3V dont le brochage est compatible avec le modèle ci-dessous :

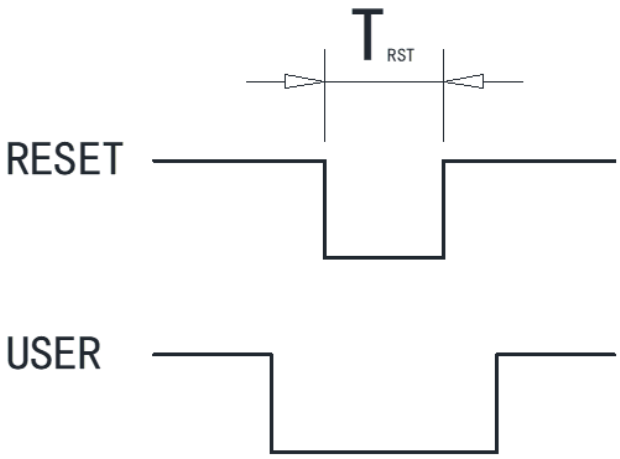


CAUTION	Attention au sens de montage de l'adaptateur sur le connecteur J15 (→ GND de l'adaptateur relié à la broche sérigraphiée GND sur le connecteur J15)

3. Mettre sous tension la *Raspberry Pi* pour alimenter le transceiver LoRa

NOTE	Le transceiver est alimenté à travers la *Raspberry Pi* et non à partir de l'alimentation présente sur une des broches de l'adaptateur ⇒ Il n'y a donc aucune crainte à avoir sur le niveau de tension (3.3V ou 5V) délivrée sur cette broche.

4. Mettre le transceiver en mode "BootLoader" en appliquant la séquence suivante au boutons poussoir "USER" et "RST" avec $T_{RST} \geq 10ms$:



Source : How to Access Bootloader Mode

5. Compiler et téléverser le *sketch Arduino* fourni dans le projet *PlatformIO* présent dans le dépôt *Framagit* (→ `src/platform.io/MyIaqHtccAb01Firmware/)

1.3. Configuration des cavaliers

Les cavaliers présents en J13 permettent de sélectionner le transceiver qui servira à la transmission LoRa des mesures.

Pour sélectionner le transceiver HTCC-AM02 de chez *Heltec*, relier sur J13 les broches suivantes avec des cavaliers :

- AM02 ↔ Tx_Rpi
- AM02 ↔ Rx_Rpi

Pour sélectionner le transceiver RN2483 de chez *Microchip*, relier sur J13 les broches suivantes avec des cavaliers :

- 2483 ↔ Tx_Rpi
- 2483 ↔ Rx_Rpi

NOTE	L'application logicielle fournie ne gère pas le transceiver RN2483.

2. Préparation de l'environnement d'exécution

2.1. Installation de *Raspberry Pi OS*

Préparer une µSD card avec la version "BullsEye" de *Raspberry Pi OS*.

NOTE	La procédure ci-dessous a été testée avec la version 32 bits de "Raspberry Pi OS with desktop" basée sur la version 11 de *Debian* (→ "BullsEye") publiée le 22/09/2022 avec le kernel en version 5.15

2.2. Configuration système

2.2.1. Activation des interfaces de communication

Exécuter `sudo raspi-config` pour activer les interfaces de communication suivantes :

- Le bus I^2C
- Le port série matériel

2.2.2. Désactivation de l'économiseur d'écran

Exécuter `sudo raspi-config` puis désactiver l'économiseur d'écran via l'option Display Options › Screen Blanking

2.2.3. Inversion du sens de l'écran

Ajouter l'option suivante au fichier `/boot/config.txt` :

```
lcd_rotate = 2
```

2.2.4. Configuration audio

1. Dans le fichier `/boot/config.txt` décommenter l'option `dtparam=i2s=on`
2. Modifier ce même fichier `/boot/config.txt` comme suit au niveau de la configuration de l'audio :

```
# [...]

# Enable audio (loads snd_bcm2835)
# MYIAQ2022 : Début config audio
# ligne *commentée* pour mise en service du dac I2S
# dtparam=audio=on
# lignes *ajoutées* pour mise en service du dac I2S
dtoverlay=hifiberry-dac
#dtoverlay=i2s-mmap (1)
# MYIAQ2022 : Fin config audio

# [...]
```

   1. Option apparemment plus nécessaire

3. créer un nouveau fichier `asound.conf` dans le répertoire `/etc` avec le contenu suivant (fichier fourni dans le dépôt dans le répertoire `rpios/etc`) :

```
pcm.hifiberry {
    type hw card 0
}

pcm.!default {
    type plug
    slave.pcm "dmixer"
}

pcm.dmixer {
    type dmix
    ipc_key 1024
    slave {
        pcm "hifiberry"
        channels 2
    }
}
```

4. Rebooter la *Raspberry Pi*
5. Exécuter `raspi-config` et sélectionner la valeur "HifiBerry DAC HiFi pcm5102a-hifi-0" dans le menu System Options › Audio

2.2.5. Désactivation du message d'avertissement sur le mot de passe SSH

Si on se connecte par SSH à la *Raspberry Pi* et que le mot de passe n'a pas été changé, une fenêtre d'avertissement s'affiche systématiquement au démarrage de la *Raspberry Pi*.

Pour désactiver cet avertissement sans avoir à changer le mot de passe, exécuter la commande suivante dans un terminal :

```
sudo rm /etc/xdg/lxsession/LXDE-pi/sshpwd.sh
```

Source : How to Disable the Annoying SSH Password Warning on Raspberry Pi

2.2.6. Configuration RTC DS3231

- ajouter la ligne `dtoverlay=i2c-rtc,ds3231` dans le fichier `/boot/config.txt`
- désactiver l'émulation d'horloge hardware :

  ```
  $ sudo systemctl stop fake-hwclock (1)
  $ sudo systemctl disable fake-hwclock (2)
  ```

  1. stoppe l'émulation
  2. désactive le démarrage automatique de l'émulation au prochain *reboot*

- Commenter les lignes suivantes dans le fichier `/lib/udev/hwclock-set` :

  ```
  if [ -e /run/systemd/system ] ; then
    exit 0
  fi
  ```

  Résultat attendu :

  ```
  # if [ -e /run/systemd/system ] ; then
  #   exit 0
  # fi
  ```

- Mettre en place un service lancé au démarrage pour forcer l'heure système sur l'heure de la RTC. Pour cela :

  a. Créer un fichier `/etc/systemd/system/hwclock-start.service` avec le contenu :

```
[Unit]
Description=Mise à l'heure du système depuis la RTC
After=network.target

[Service]
Type=oneshot
ExecStart=/sbin/hwclock -s
TimeoutSec=0

[Install]
WantedBy=multi-user.target
```

b. Vérifier le bon fonctionnement de ce service en l'exécutant et en parcourant son fichier journal :

```
$ sudo systemctl start hwclock-start.service
$ sudo journalctl -u hwclock-start.service
[...]
May 17 09:28:16 raspberrypi systemd[1]: Starting Mise à l'heure du système depuis la
RTC...
May 17 09:28:17 raspberrypi systemd[1]: hwclock-start.service: Succeeded. (1)
May 17 09:28:17 raspberrypi systemd[1]: Finished Mise à l'heure du système depuis la
RTC.
[...]
```

1. Ligne qui indique le bon déroulement de l'exécution du service

c. Activer ce service au démarrage du système avec la commande :

```
$ sudo systemctl enable hwclock-start.service
```

- Mettre en place un service lancé juste avant l'arrêt du système pour mettre à jour l'heure de la RTC à partir de l'heure système. Ceci peut être utile si le boitier MyIAQ a accès à internet — même provisoirement — dans la mesure où l'heure système sera mise jour par NTP ⇒ Ceci permet alors de synchroniser la RTC avec avec l'heure de référence.

  Pour cela :

  a. Créer un fichier `/etc/systemd/system/hwclock-stop.service` avec le contenu :

```
[Unit]
Description=Mise à l'heure de la RTC depuis l'heure système
DefaultDependencies=no
Before=shutdown.target

[Service]
Type=oneshot
ExecStart=/sbin/hwclock --systohc

[Install]
WantedBy=reboot.target halt.target poweroff.target
```

  b.

Vérifier le bon fonctionnement de ce service en l'exécutant et en parcourant son fichier journal :

```
$ sudo systemctl start hwclock-stop.service
$ sudo journalctl -u hwclock-stop.service
[...]
May 17 09:44:16 raspberrypi systemd[1]: Starting Mise à l'heure de la RTC depuis
l'heure système...
May 17 09:44:17 raspberrypi systemd[1]: hwclock-stop.service: Succeeded. (1)
May 17 09:44:17 raspberrypi systemd[1]: Finished Mise à l'heure de la RTC depuis
l'heure système.
[...]
```

1. Ligne qui indique le bon déroulement de l'exécution du service

c. Activer ce service au démarrage du système avec la commande :

```
$ sudo systemctl enable hwclock-start.service
```

Sources :

- Ajout d'un module RTC DS3231 au Raspberry Pi
- Tutorial: Add a Real Time Clock to the Raspberry Pi

2.3. Paquets logiciels requis

2.3.1. Applications

L'application logicielle de la station de mesure de qualité d'air intérieure fait appel à 2 logiciels standards qu'il faut installer sur la *Raspberry Pi* :

1. le *broker* MQTT "*Mosquitto*"
2. le streamer audio "*gstreamer*"

L'installation de ces paquets est réalisée avec les commandes suivantes :

```
sudo apt-get update; sudo apt-get install mosquitto gstreamer1.0-pulseaudio
```

2.3.2. *Framework Qt*

L'application logicielle de la station de mesure de qualité d'air intérieur, codée avec le framework *Qt*, s'appuie sur des librairies qu'il faut également installer.

Les dommaines couverts par ces librairies sont :

- les fonctionnalités de base
- la programmation QML
- le multimedia

- la communication série
- le protocole MQTT

Pour les fonctionnalités de base, installer les paquets :

- `libqt5core5a`
- `libqt5gui5`
- `libqt5qml5`
- `libqt5network5`

Pour les fonctionnalités de programmation QML, installer les paquets :

- `libqt5quick5`
- `libqt5quickwidgets5`
- `libqt5quickcontrols2-5`
- `qml-module-qtquick-dialogs`

Pour le domaine multimédia, installer les paquets :

- `libqt5multimedia5`
- `libqt5multimedia5-plugins`

Poour le domaine de la communication série, installer le paquet :

- `libqt5serialport5`

Pour le domaine du protocole MQTT, l'installation est plus délicate car cette librairie — ou plus exactement le module *Qt* — n'est pas disponible sous forme de paquet. Il faut donc l'installer depuis les fichiers source :

```
$ sudo apt-get install qtbase5-private-dev ❶
$ sudo apt-get install cmake
$ qmake --version ❷
QMake version 3.1
Using Qt version 5.15.2 in /usr/lib/arm-linux-gnueabihf
$ git clone git://code.qt.io/qt/qtmqtt.git --branch 5.15.2 ❸
$ cd qtmqtt ❹
$ qmake ❹
$ make ❹
$ sudo make install ❹
```

1. package **indispensable** pour avoir accès au fichier `qobject_p.h` requis lors de la compilation du module `qtmqtt`
2. récupère la version de *Qt* installée (ici **5.15.2**)
3.

clone les sources du module `qtmqtt` dans la version la plus proche de la version du framework *Qt* installé (se rendre par exemple sur https://code.qt.io/cgit/qt/qtmqtt.git/ pour lister les versions disponibles)

4. compile et installe le module

2.3.3. Librairie `bcm2835`

L'application logicielle de la station de mesure de qualité d'air intérieur nécessite l'accès aux GPIOs de la *Raspberry Pi*.

Ces accès sont réalisés à travers de la librairie `bcm2835` développée par Mike McCauley et qui n'est pas distribuée sous forme de paquet mais sous forme de fichiers sources.

Cette librairie et la procédure pour l'installer sur la *Raspberry Pi* sont disponibles sur le site de l'auteur

La procédure d'installation est reproduite ci-dessous :

```
# download the latest version of the library, say bcm2835-1.xx.tar.gz, then:
tar zxvf bcm2835-1.xx.tar.gz
cd bcm2835-1.xx
./configure
make
sudo make check
sudo make install
```

2.4. Installation de l'application *MyIAQ*

L'installation de l'application consiste simplement à copier les 3 exécutables de l'application dans le système de fichiers de la *Raspberry Pi*.

En effet, outre le *broker MQTT Mosquitto* et le streamer audio *gstreamer*, l'application logicielle de la station de mesure de qualité d'air intérieure s'appuie sur 3 exécutables *Qt* :

1. une application qui s'occupe de l'acquisition des capteurs (CO_2, température, humidité, luminosité) et qui les publie sur le *broker* MQTT
2. une application qui se charge essentiellement de l'affichage de ces mesures sur l'écran 7" après les avoir extraites du *broker* MQTT
3. une dernière application dont le rôle est de transmettre périodiquement par *LoRa* les mesures présentes dans le *broker* MQTT sur *The Things Network*.

Ces 3 exécutables, présents dans les fichiers `DaqApp`, `GuiApp` et `LoRaApp`, sont à récupérer depuis l'environnement de compilation (voir plus loin le chapitre Compilation de l'application pour la mise en place de cet environnement sur la *Raspberry Pi*) et à copier dans le répertoire de votre choix (→ ex. : `/home/pi/myiaq`)

2.5. Configuration locale de l'application

La configuration de l'application ne concerne que le paramétrage de la communication LoRa avec le transceiver HTCC-AM02 de chez *Heltec*.

Ce paramétrage s'effectue à partir du fichier `/home/pi/.config/CERI/MyIAQ.ini`.

S'il n'existe pas au démarrage de l'exécutable `LoRaApp`, il est créé automatiquement avec le contenu suivant :

```
[LoRa]
AppEUI=4D794941512D3232 (1)
AppKey=00112233445566778899AABBCCDDEEFF (2)
DevEUI= (3)
DutyCycle=60 (4)
```

1. Identifiant de l'application "The Things Network"
2. Clé de l'application "The Things Network"
3. Identifiant du nœud LoRa (i.e. la station de mesure de qualité d'air intérieur). Si celui-ci est vide (comme c'est la cas ici), il est généré automatiquement à partir du *ChipID* du transceiver LoRa. La façon de récupérer sa valeur est expliqué plus loin.
4. l'intervalle de temps — exprimé en secondes — au bout duquel les mesures sont transmises périodiquement à "The Things Network"

2.5.1. Récupération du *DevEUI* par défaut

Si on ne spécifie pas le *DevEUI* du nœud LoRa à travers le fichier de configuration (→ `/home/pi/.config/CERI/MyIAQ.ini`), celui-ci est généré automatiquement par le *firmware* du transceiver LoRa HTCC-AM02 (→ appel à la fonction `LoRaWAN.generateDeveuiByChipID()` dans le *sketch Arduino*) à partir du *ChipID* présent par construction dans le circuit.

NOTE

Cette fonctionnalité est pilotée par l'option `board_build.arduino.lorawan.deveui = Generate By ChipID` dans le fichier `platform.ini`

Si on veut fixer le *DevEUI* à la valeur spécifiée dans le *sketch* (→ `uint8_t devEui[] = { … };`), il faut alors changer la valeur de cette option en : `board_build.arduino.lorawan.deveui = CUSTOM`

Pour récupérer la valeur du *DevEUI* par défaut, il suffit de consulter le fichier journal du service associé à l'application :

```
$ sudo journalctl -u myiaq-loraapp
[...]
Jan 08 15:07:59 raspberrypi LoRaApp[1375]: transaction  "AT+DevEui=?" "emitted at 15:07:59"
Jan 08 15:07:59 raspberrypi LoRaApp[1375]: Rx waiting delay :  2000
Jan 08 15:07:59 raspberrypi LoRaApp[1375]: AppEui :  "4D794941512D3232"
Jan 08 15:07:59 raspberrypi LoRaApp[1375]: Response cmd< "AT+DevEui" > ->
"+OK\r\n+DevEui=0003222F143A26DB(For OTAA Mode)\r\n" (1)
Jan 08 15:07:59 raspberrypi LoRaApp[1375]: transaction
"AT+AppKey=00112233445566778899AABBCCDDEEFF" "emitted at 15:07:59"
Jan 08 15:07:59 raspberrypi LoRaApp[1375]: Rx waiting delay :  2000
[...]
```

1. Valeur du *DevEUI* par défaut

2.5.2. Configuration de la valeur du *DutyCycle*

La valeur du *duty cycle* doit être en accord avec la politique d'utilisation équitable du site "The Things Network" (voir Fair Use Policy) à savoir le temps d'émission accordé par jour à chaque nœud LoRa.

Ce dernier est fixé à 30s d'émission par 24h.

Sachant que le payload LoRa envoyé par l'application fait 4 octets de long, cela autorise de 22 à 583 transmissions par 24h selon le *data rate* (Voir Airtime calculator for LoRaWAN).

La valeur du *duty cycle* doit alors théoriquement être compris entre 148s et 3927s dans les conditions normales d'utilisation (voir la console TTN pour obtenir le *data rate* réel relevé pour le nœud LoRa considéré).

2.6. Démarrage automatique des services de l'application

Ces 3 applications peuvent être lancées automatiquement au démarrage de *Raspberry Pi OS* en mettant en place des services *Linux* de type *SystemD*.

Pour cela:

1. copier les 3 fichiers présents dans le répertoire `rpios/etc/systemd/system` (`myiaq-daqapp.service`, `myiaq-guiapp.service`, `myiaq-lorapp.service`) du dépôt *Framagit* dans le répertoire `/etc/systemd/system` de *Linux*.

   myaq-daqapp.service

   ```
   [Unit]
   Description=MyIAQ DaqApp - Application locale d'acquisition des mesures de qualité d'air
   intérieur
   Requires=mosquitto.service
   After=mosquitto.service

   [Service]
   Type=simple
   WorkingDirectory=/home/pi/projet-2022/myiaq/src/qt/build-MyIAQ-Desktop-Debug/DaqApp (1)
   User=root
   Group=root
   Restart=on-failure
   RestartSec=10
   ExecStart=/home/pi/projet-2022/myiaq/src/qt/build-MyIAQ-Desktop-Debug/DaqApp/DaqApp (2)

   [Install]
   WantedBy=multi-user.target
   ```

   1. chemin à adapter
   2. chemin à adapter

myiaq-guiapp.service

```
[Unit]
Description=MyIAQ GuiApp - Application locale de visualisation des mesures de qualité
d'air intérieur
Requires=mosquitto.service
After=mosquitto.service

[Service]
Type=simple
Environment="DISPLAY=:0.0"
Environment="XDG_RUNTIME_DIR=/run/user/1000"
Environment="QT_QPA_PLATFORM=eglfs"
Environment="QT_QPA_FONTDIR=/usr/share/fonts/truetype/dejavu/"
Environment="QT_QPA_EGLFS_HIDECURSOR=1"
Environment="QT_QPA_EGLFS_PHYSICAL_WIDTH=154"
Environment="QT_QPA_EGLFS_PHYSICAL_HEIGHT=86"
WorkingDirectory=/home/pi/projet-2022/myiaq/src/qt/build-MyIAQ-Desktop-Debug/GuiApp (1)
User=pi
Group=pi
Restart=on-failure
RestartSec=10
ExecStart=/home/pi/projet-2022/myiaq/src/qt/build-MyIAQ-Desktop-Debug/GuiApp/GuiApp (2)

[Install]
WantedBy=multi-user.target
```

1. chemin à adapter
2. chemin à adapter

myiaq-loraapp.service

```
[Unit]
Description=MyIAQ LoRaApp - Application locale de transmission des mesures de qualité
d'air intérieur
Requires=mosquitto.service
After=mosquitto.service

[Service]
Type=simple
WorkingDirectory=/home/pi/projet-2022/myiaq/src/qt/build-MyIAQ-Desktop-Debug/LoRaApp (1)
User=pi
Group=pi
Restart=on-failure
RestartSec=10
ExecStart=/home/pi/projet-2022/myiaq/src/qt/build-MyIAQ-Desktop-Debug/LoRaApp/LoRaApp (2)

[Install]
WantedBy=multi-user.target
```

1. chemin à adapter
2. chemin à adapter

2. ajuster si nécessaire les chemins des 3 exécutables *Qt* présents dans ces fichiers
3. activer ces services pour les lancer au démarrage de *Raspberry Pi OS* :

```
sudo systemctl enable myiaq-daqapp.service myiaq-guiapp.service myiaq-loraapp.service
```

2.7. Configuration de l'application sur "The Things Network"

Une fois créée l'application sur "The Things Network" et le nœud LoRa enregistré, il faut mettre en place le *Payload Formatter* de façon à pouvoir décoder les données transmises dans le *payload*

La structure du *payload* est la suivante :

```
+----------+----------+----------+----------+----------+----------+---------+---------+
| Tx10°C[8]| Tx10°C[7]| Tx10°C[6]| Tx10°C[5]| Tx10°C[4]| Tx10°C[3]|Tx10°C[2]|Tx10°C[1]|
|   (msb)  |          |          |          |          |          |         |         |
+----------+----------+----------+----------+----------+----------+---------+---------+
| Tx10°C[0]|  HR%[6]  |  HR%[5]  |  HR%[4]  |  HR%[3]  |  HR%[2]  |  HR%[1] |  HR%[0] |
|   (lsb)  |   (msb)  |          |          |          |          |         |         |
+----------+----------+----------+----------+----------+----------+---------+---------+
|CO2ppm[15]|CO2ppm[14]|CO2ppm[13]|CO2ppm[12]|CO2ppm[11]|CO2ppm[10]|CO2ppm[9]|CO2ppm[8]|
|   (msb)  |          |          |          |          |          |         |         |
+----------+----------+----------+----------+----------+----------+---------+---------+
| CO2ppm[7]| CO2ppm[6]| CO2ppm[5]| CO2ppm[4]| CO2ppm[3]| CO2ppm[2]|CO2ppm[1]|CO2ppm[0]|
|          |          |          |          |          |          |         |  (lsb)  |
+----------+----------+----------+----------+----------+----------+---------+---------+
```

avec :

- Tx10°C : température exprimée en dizièmes de °C (valeur entière)
- HR% : humidité en % (valeur entière)
- CO2ppm : taux de CO2 exprimé en ppm (valeur entière)

Le code du *payload formatter* (fourni dans le dépôt *Framagit* dans `/src/ttn/payload_formatter.js`) pour extraire les différentes mesures est alors :

```
function Decoder(bytes, port) {
    var decoded = {};
    if (port === 1) {
      var temp = (bytes[ 0 ] << 1) | (bytes[ 1 ] >> 7);
      decoded.temperature = temp / 10;

      var hum = bytes[ 1 ] & ~(1 << 7); // val & ~0x80 -> val & 0x7f
      decoded.humidite = hum;

      var co2 = (bytes [ 2 ] << 8) | bytes[ 3 ]; // 2222 2222 0000 0000 | 2222 2222
      decoded.co2 = co2;
    }
    return decoded;
}
```

3. Compilation de l'application

La compilation de l'application nécessite de :

- récupérer le dépôt *Framagit* associé au projet
- mettre en place l'environnement de compilation sur la *Raspberry Pi*
- lancer la compilation

3.1. Récupération du dépôt logiciel

```
git clone https://framagit.org/projets-2022/myiaq.git myiaq-repository
```

3.2. Mise en place de l'environnement de compilation

```
sudo apt-get install qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools qtcreator (1)
sudo apt-get install qtdeclarative5-dev (2)
sudo apt-get install qtmultimedia5-dev (3)
sudo apt-get install libqt5serialport5-dev (4)
```

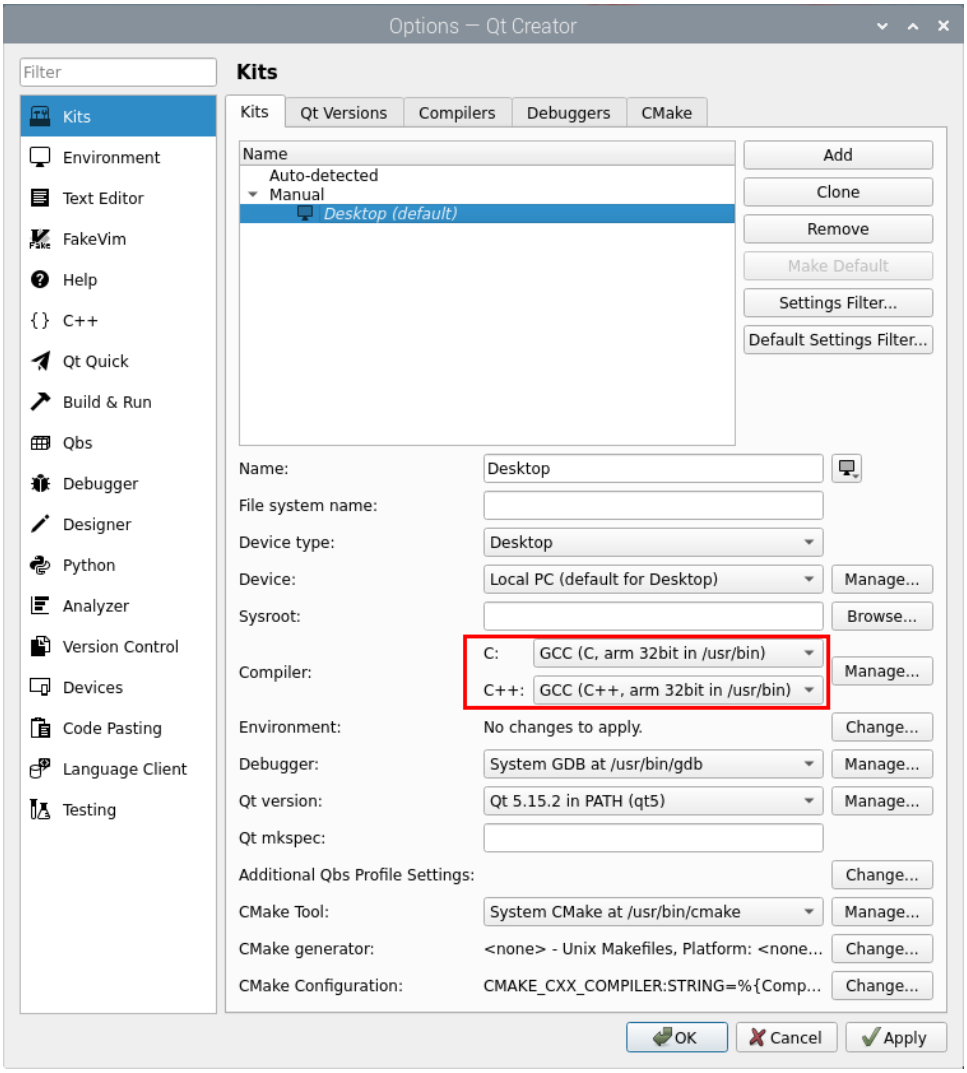
1. installation de l'environnement de développement *Qt* de base
2. fichiers de développement pour la programmation QML/Quick
3. fichiers de développement pour les librairies multimédia
4. fichiers de développement pour la liaison série

3.3. Lancement de la compilation

1. Lancer *Qt Creator*
2.

Ouvrir le projet `~/myiaq-repository/src/qt/MyiaqApp/MyIAQ.pro`

3. Ajuster les paramètre du kit *Qt* (menu **Tools › Options › Kits › Desktop(default)**) pour utiliser les compilateurs `gcc` et `g++` plutôt que ceux de la suite `Clang 11`



4. Lancer le *build*

5. Les exécutables sont alors disponibles dans les sous-répertoires `GuiApp/`, `LoRaApp/` et `GuiApp/` de `~/myiaq-repository/src/qt/build-MyIAQ-Desktop-Debug`