Linux pour l'embarqué – TP N°2

Création d'une d'un Micro-OS Linux/GNU personnalisée avec BuildRoot

Application : **HVAC** systems in trains

Heating, Ventilation, and Air Conditioning

Salem BEN KEMMOUM
Ingénieur contrôle commande embarquée chez CCAMY Systèmes
Secteur Ferroviaire

Décembre 2024

Explication HVAC : https://www.youtube.com/watch?v=NUewC7ci4s0
HVAC unit : https://rail.knorr-bremse.com/en/de/portfolio/products-and-systems/climate-control-systems-hvac/
HVAC Controller : https://www.ebay.co.uk/itm/233512343491

Partie 1 : BuildRoot

Tutoriel : https://www.blaess.fr/christophe/buildroot-lab/index.html

I. Préparation de l'environnement BuildRoot

Dans cette partie nous allons procéder au téléchargement des codes sources BuildRoot ainsi qu'à l'installation des outils Linux/GNU nécessaires à son utilisation.

1. Mise à jour de la machine hôte (votre machine virtuelle Debian)

- Taper les commandes
  sudo apt-get update
  sudo apt-get upgrate
- Si le système indique que vous n'êtes pas sudoers taper les commandes suivantes
  su
  mot de passe : ****
  /sbin/usermod -aG sudo [votre nom d'utilisateur]
  /sbin/reboot

Une fois la machine virtuelle redémarrée, refaire les commande sudo apt-get pour mettre à jour votre système.

2. Installation des outils GNU/Linux nécessaires

- Outils nécessaires au fonctionnement de BuildRoot
  sudo apt-get install bison
  sudo apt-get install g++
  sudo apt-get install flex
  sudo apt-get install gettext
  sudo apt-get install texinfo
  sudo apt-get install patch
  sudo apt-get install git-core
  sudo apt-get install libtool
  sudo apt-get install autoconf
  sudo apt-get install build-essential
  sudo apt-get install libncurses5-dev
  sudo apt-get install rsync

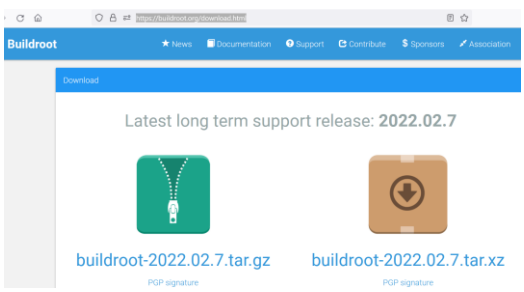
  Remarque : vous pouvez installer ces outils avec une seule ligne apt-get :
  sudo apt-get install bison g++ flex gettext texinfo patch git-core libtool …

- Outils nécessaires pour communiquer avec la Raspberry Pi
  sudo apt-get install ssh
  sudo apt-get install telnet

3. Téléchargement des codes sources BuildRoot

NB : les captures d'écran datent de 2022 (à l'époque c'était la version 2022.02.7)

- Aller sur le site officiel et regarder quelle est la « Latest long term support release »
  https://buildroot.org/download.html



- Télécharger la version « buildroot-2024.02.8.tar.gz»
  wget https://www.buildroot.org/downloads/buildroot-2024.02.8.tar.gz
- Créer le dossier tp2 dans votre répertoire utilisateur principal
  mkdir tp2

```
instru@ING-instru:~$ ls
Bureau     Images   Musique  Téléchargements  Vidéos
Documents  Modèles  Public   tp2
```

- Copier l'archive « buildroot-2024.02.8.tar.gz » dans tp2

```
instru@ING-instru:~$ cp Téléchargements/buildroot-2022.02.7.tar.gz tp2
```

- Vérifier si l'archive a été téléchargée correctement
  gunzip -tv buildroot-2024.02.8.tar.gz
- Décompresser l'archive
  tar -xzvf buildroot-2024.02.8.tar.gz
- Protéger le dossier buildroot-2024.02.8 en écriture
  chmod -R -w buildroot-2024.02.8
- Se positionner à l'intérieur du dossier buildroot-2024.02.8

```
instru@ING-instru:~$ cd tp2/buildroot-2022.02.7/
instru@ING-instru:~/tp2/buildroot-2022.02.7$
```

- Créer un projet pour une Raspberry Pi 4
  make O=../build-pi4 raspberrypi4_defconfig
- Créer un projet pour une Raspberry Pi 3
  make O=../build-pi3 raspberrypi3_defconfig
- Créer un projet pour une architecture x86 (processeur Intel 32 bits)
  make O=../build-x86 pc_x86_64_bios_defconfig
- Constater la présence des trois projets dans le dossier tp2

```
instru@ING-instru:~/tp2/buildroot-2022.02.7$ cd ..
instru@ING-instru:~/tp2$ ls
build-pi3  build-pi4  buildroot-2022.02.7  buildroot-2022.02.7.tar.gz  build-x86
```

II. Première compilation d'un projet (Raspberry Pi 4 dans ce cas)

Dans cette partie vous allez voir comment on personnalise notre système Linux/GNU avec le menu de configuration de BuildRoot et comment lancer la compilation.

La première compilation prend beaucoup de temps surtout sur une machine virtuelle (jusqu'à 4h !!).

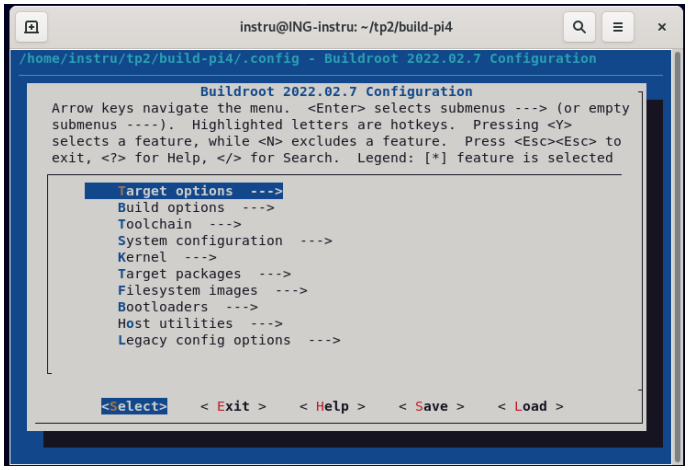
1. Configuration

N'oubliez pas de sauvegarder la configuration à chaque modification !

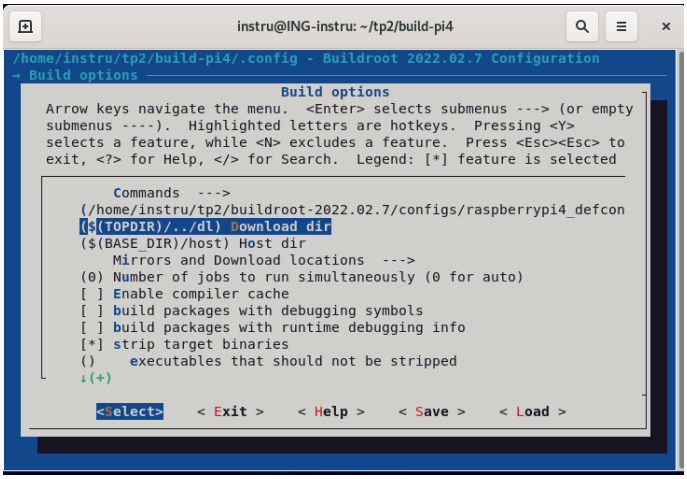
- Accéder au répertoire projet (build-pi4 dans ce cas) et lancer la commande suivante
  make menuconfig

```
instru@ING-instru:~/tp2$ cd build-pi4
instru@ING-instru:~/tp2/build-pi4$ make menuconfig
```

Le menu suivant s'affiche :



- Pendant la compilation BuildRoot télécharge plein de paquets, il est recommandé de mutualiser ces derniers entre les différents projets, nous allons donc déplacer le répertoire de téléchargement « dl » vers le dossier tp2.
  Cliquer sur « Build options », le menu suivant s'affiche, remplacer « dl » par « /../dl »



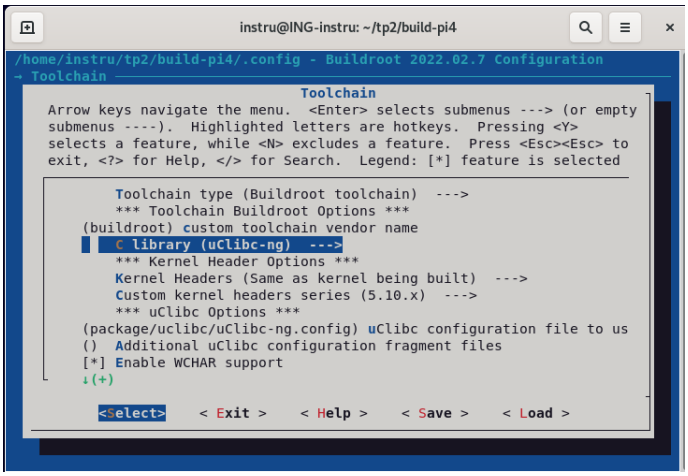
- Nous avons vu dans le cours que la librairie standard du langage C est très importante pour les systèmes Linux/GNU (vu qu'ils sont codés en C).

  Cette librairie a été modifiée et complétée pour le projet GNU « glibc » et nous avons vu également qu'une version dédiée aux systèmes embarqués a été implémentée « uclibc », cette dernière est plus légère que la « glibc ».

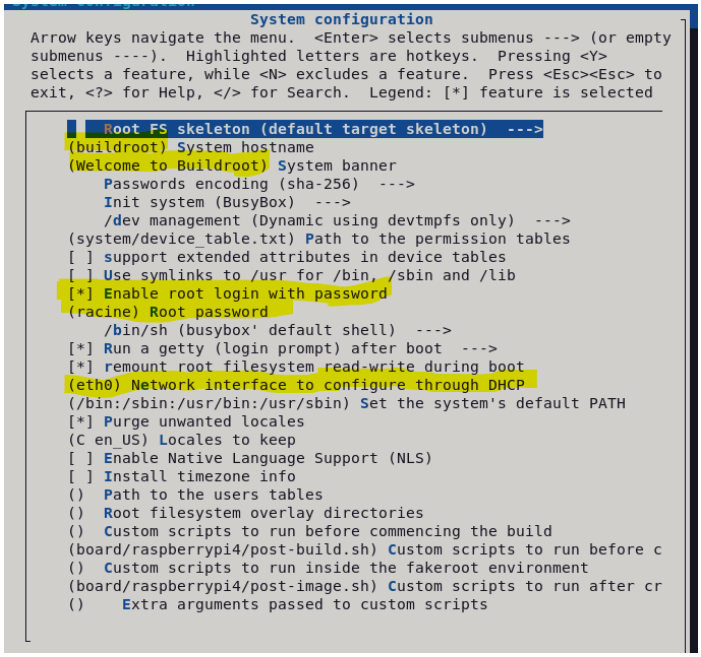
  BuildRoot nous laisse la liberté de choisir entre la « glibc » et la « uclibc ».

  Nous laissons le paramètre par défaut, c'est-à-dire « uclibc ».

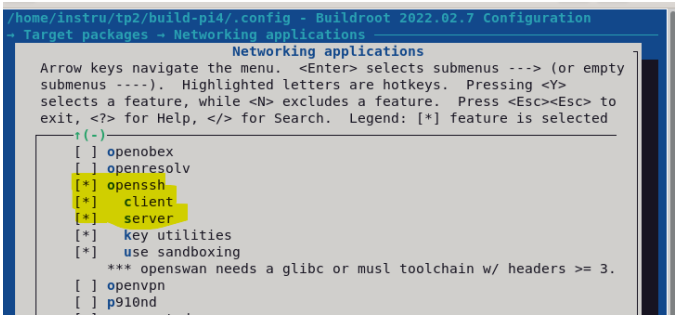
  Toujours sur le menu, aller dans Toolchain puis vérifier que uclibc est bien sélectionnée dans la ligne « C library ».



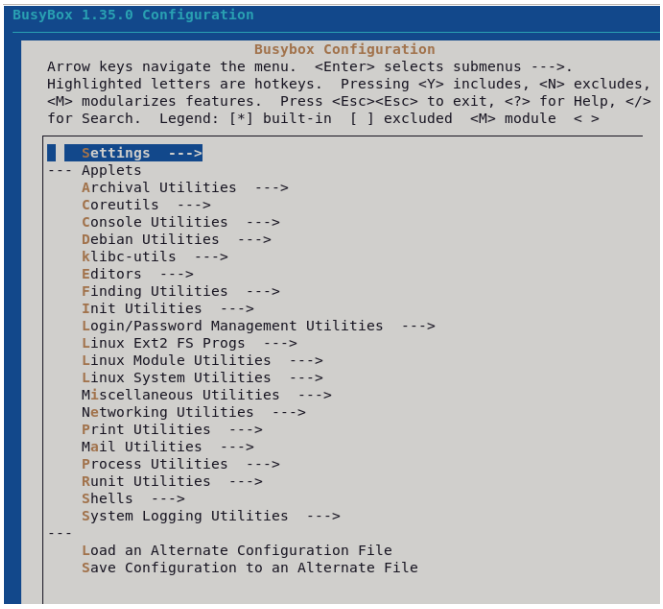
  Toujours dans le menu Toolchain, activer les paramètres suivants :
  - Enable WCHAR support
  - Thread library debug
  - Build cross gdb for the host
- Par défaut le système ne possède aucun utilisateur, il y a que le superutilisateur root et en plus sans mot de passe, il faut sécuriser le système en mettant un mot de passe.
  Aller dans « system configuration » et mettre « racine » pour le « Root password ».
  Modifier le « System hostname » (utilisez celui qui correspond à votre Controller HVAC dans le train, exemple: car1-hvac1).
  Vous pouvez faire la même chose pour le « System banner » (le message d'accueil au démarrage).



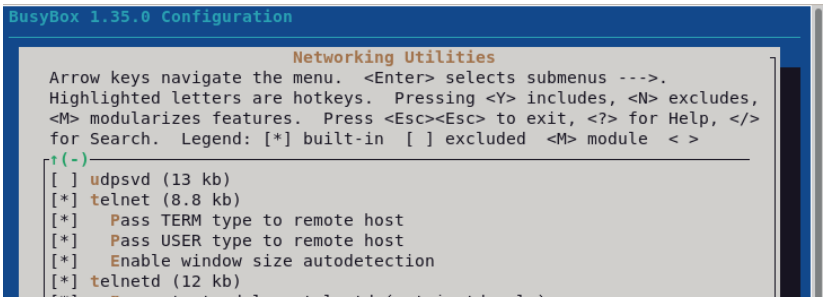
- Dans un premier temps également, sélectionner l'interface Ethernet « eth0 » de la Raspberry Pi comme l'interface configurable en DHCP (le routeur lui attribue dynamiquement une adresse IP). Il suffit de laisser le paramètre par défaut (voir image précédente).
- Pour pouvoir se connecter à la Raspberry Pi en ssh ou telnet à l'aide d'un câble Ethernet branché à notre PC, il faut intégrer open ssh et telnet dans le système.
  Toujours sur le menu BuildRoot, aller dans Target packages, ensuite dans Networking applications. Sélectionner openssh.



Sauvegarder la configuration et quitter.
Ensuite lancer la commande :
make busybox-menuconfig
Le menu suivant s'affiche :



Aller dans networking utilities et activer le protocole telnet (attention ce protocole n'est pas sécurisé ! on en a besoin juste pour la connexion Ethernet PC-RPi pendant la phase de prototypage de l'application).



Sauvegarder et quitter.

2. Compilation

Attention il faut être connecté à internet !
Pour compiler il suffit de taper la commande suivante :
make

```
instru@ING-instru:~/tp2/build-pi4$ make
```

Cette opération va prendre beaucoup de temps (jusqu'à 4h sur machine virtuelle non performante !).

3. Préparation de la carte SD

Le résultat de compilation se trouve dans le répertoire **/images**, ce dernier contient :

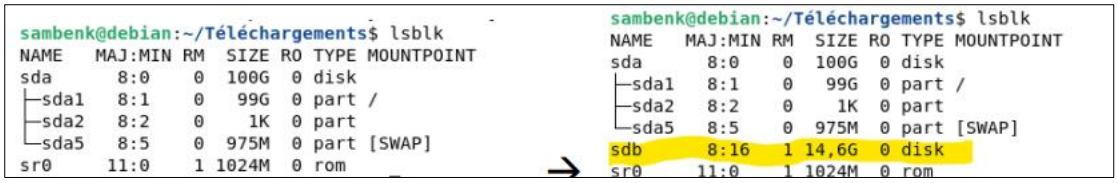
```
[build-pi4]$ ls images/
bcm2711-rpi-4-b.dtb  rootfs.ext2  rpi-firmware  zImage
boot.vfat            rootfs.ext4  sdcard.img
[build-pi4]$
```

«zImage» le noyau Linux
«bcm2711-rpi-4-b.dtb » le device tree décrivant le matériel,
«rpi-firmware/» les fichiers précompilés du bootloader spécifique au Raspberry Pi,
«boot.vfat» une copie binaire de la partition de boot,
«rootfs.ext2» image binaire de la partition du système de fichiers,
«sdcard.img» une image binaire contenant une table des partitions et les deux images de partitions précédentes. C'est ce fichier que nous allons copier sur une carte micro-SD.

C'est pour ça notre carte SD une fois programmée, elle contient deux partitions :

```
sdb      8:16   1 14,6G  0 disk
├─sdb1   8:17   1  256M  0 part /media/sambenk/boot
└─sdb2   8:18   1 14,4G  0 part /media/sambenk/rootfs
```

Afficher les lecteurs avant et après l'insertion de votre carte microSD avec la commande :
lsblk



Il est important de faire cela pour éviter d'endommager un autre lecteur ou disque. Il faut toujours être sûr qu'on copie vers le bon disque.

Noter le nom de votre carte SD et taper les commandes suivantes :

umount /dev/sdb

Pour copier l'image « sdcard.img » vers la carte SD, utiliser la commande suivante :

sudo cp images/sdcard.img /dev/sdb

Avant de retirer la carte SD, il faut configurer l'interface Ethernet pour qu'on puisse nous connecter à la Raspberry Pi via un câble Ethernet (en ssh ou en telnet).

Taper les commandes suivantes :

sudo mkdir media/boot

sudo mkdir media/rootfs

sudo mount dev/sdb1 /media/instru/boot

sudo mount dev/sdb2 /media/instru/rootfs

remplacer « instru » par votre nom d'utilisateur.

Une fois les deux partitions montées, nous allons modifier des fichiers de configuration.

Le fichier de configuration des interfaces réseau :

Pendant la configuration de BuildRoot nous avons choisi de laisser « eth0 » comme interface configurable en DHCP (un routeur réseau lui donne une IP dynamiquement dès qu'on branche le câble Ethernet). Nous n'avons pas suffisamment de prises Ethernet sur mon modem 4G, donc nous allons essayer de nous connecter autrement à la Raspberry Pi directement depuis le PC (sans passer par un réseau Ehternet).

Le fichier à modifier est le « /etc/network/interfaces »

Taper la commande suivante pour créer une copie du fichier original :

cp /media/instru/rootfs/etc/network/interfaces /media/instru/rootfs/etc/network/interfacesORI

remplacer « instru » par votre nom d'utilisateur.

Ensuite modifier le fichier « interfaces » en utilisant l'éditeur **nano**

sudo nano /media/instru/rootfs/etc/network/interfaces

- Avant modification

```
instru@ING-instru:/media/instru/rootfs/etc/network$ more interfacesORI
# interface file auto-generated by buildroot

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
  pre-up /etc/network/nfs_check
  wait-delay 15
  hostname $(hostname)
instru@ING-instru:/media/instru/rootfs/etc/network$
```

- Après modification :

```
instru@ING-instru:/media/instru/rootfs/etc/network$ more interfaces
# interface file auto-generated by buildroot

auto lo
iface lo inet loopback


iface eth0
  pre-up /etc/network/nfs_check
  wait-delay 15
  hostname $(hostname)
```

Le fichier de configuration « cmdline.txt » :

Ce fichier permet de passer des paramètres de configuration au noyau Linux pendant la phase de démarrage du système. Nous voulons lui demander de lier l'interface Ethernet eth0 de notre carte à une adresse IP fixe « 169.254.25.25 ».

De même pour modifier le fichier il faut utiliser un éditeur de texte tel que nano.

Ajouter à la fin de la ligne ip=169.254.25.25

sudo nano /media/instru/boot/cmdline.txt

- Avant la modification

```
instru@ING-instru:/media/instru/1992-BD9A$ more cmdlineORI.txt
root=/dev/mmcblk0p2 rootwait console=tty1 console=ttyAMA0,115200
```

- Après la modification

```
instru@ING-instru:/media/instru/1992-BD9A$ more cmdline.txt
root=/dev/mmcblk0p2 rootwait console=tty1 console=ttyAMA0,115200 ip=169.254.25.25
```

Une fois les modifications apportées, démonter les partitions (éjecter la carte SD) en utilisant les commandes suivantes :

sudo umount /media/instru/boot

sudo umount /media/instru/rootfs

4. **Test de notre premier système GNU/Linux personnalisé**

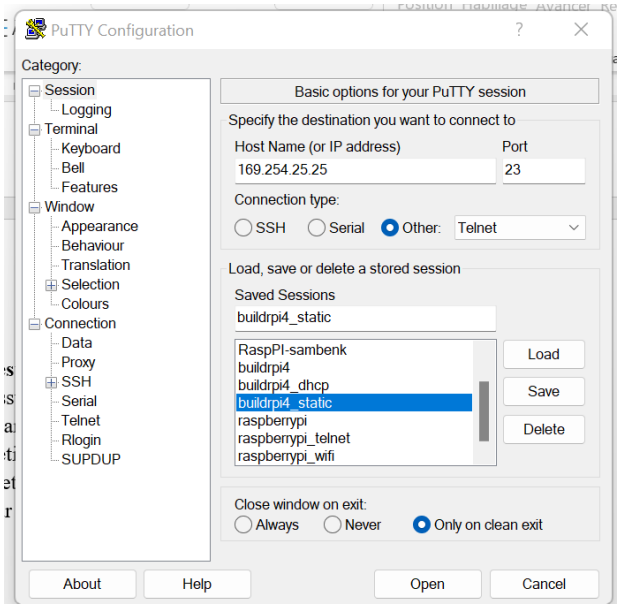
Assurer vous que la Raspberry Pi est éteinte !

Brancher la Raspberry Pi à votre PC en utilisant le câble Ethernet.

Retirer la carte SD du lecteur SD et brancher la dans la Raspberry Pi.

Mettre sous tension la Raspberry Pi et attendre 2 minutes.

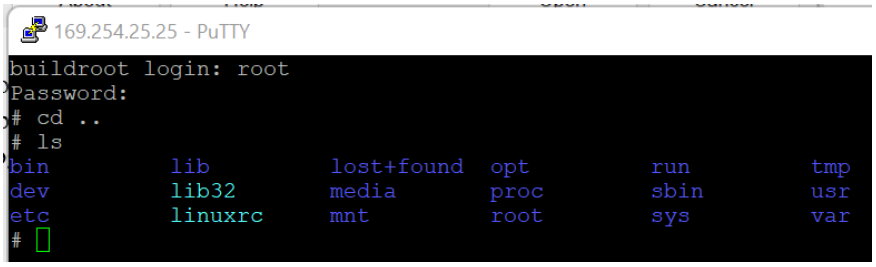
Sur votre PC Windows ouvrir PuTTY et lancer une connexion Telnet :



Noter bien que nous avons mis l'adresse IP statique 169.254.25.25 comme Host name.

Appuyer sur Open, une fenêtre s'ouvre et vous demande les identifiants.

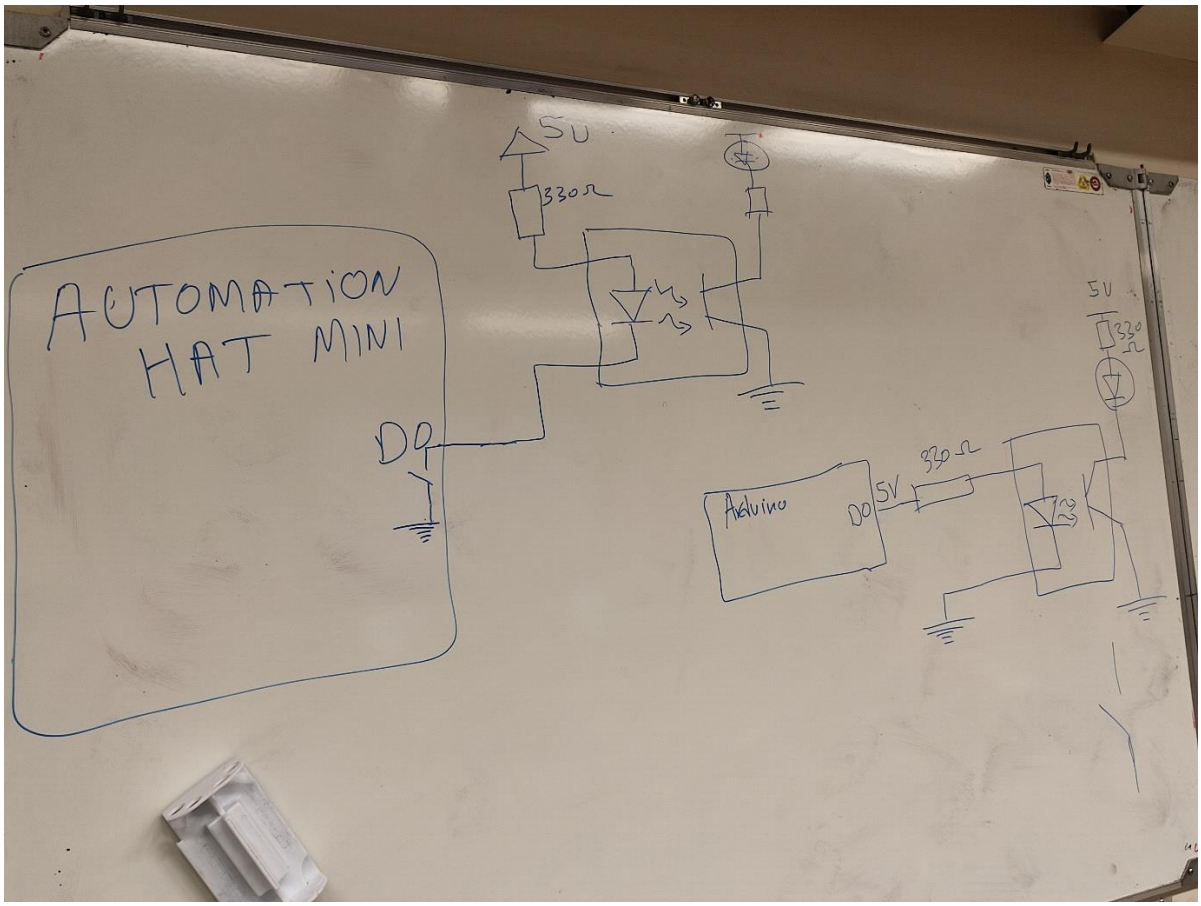
Login : root mot de passe : racine



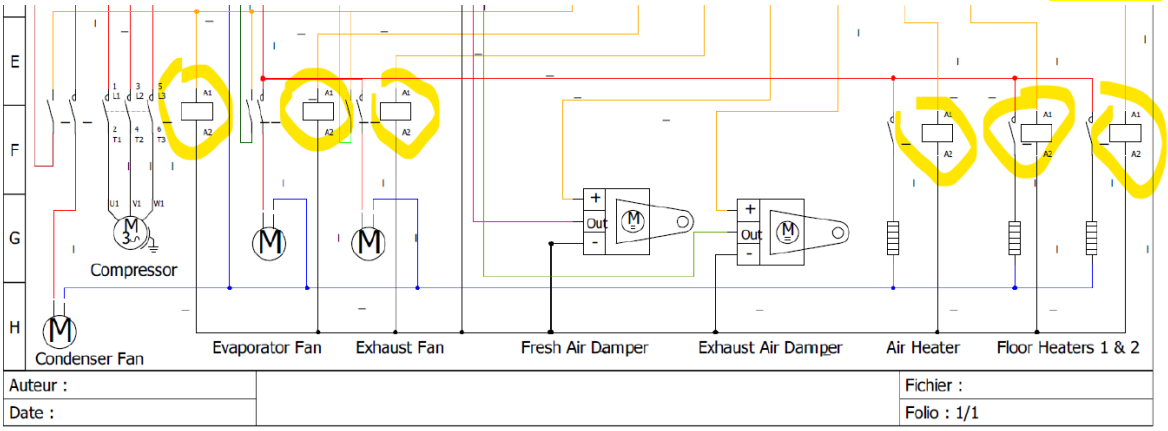
Partie 2 : Application « HVAC systems in trains »

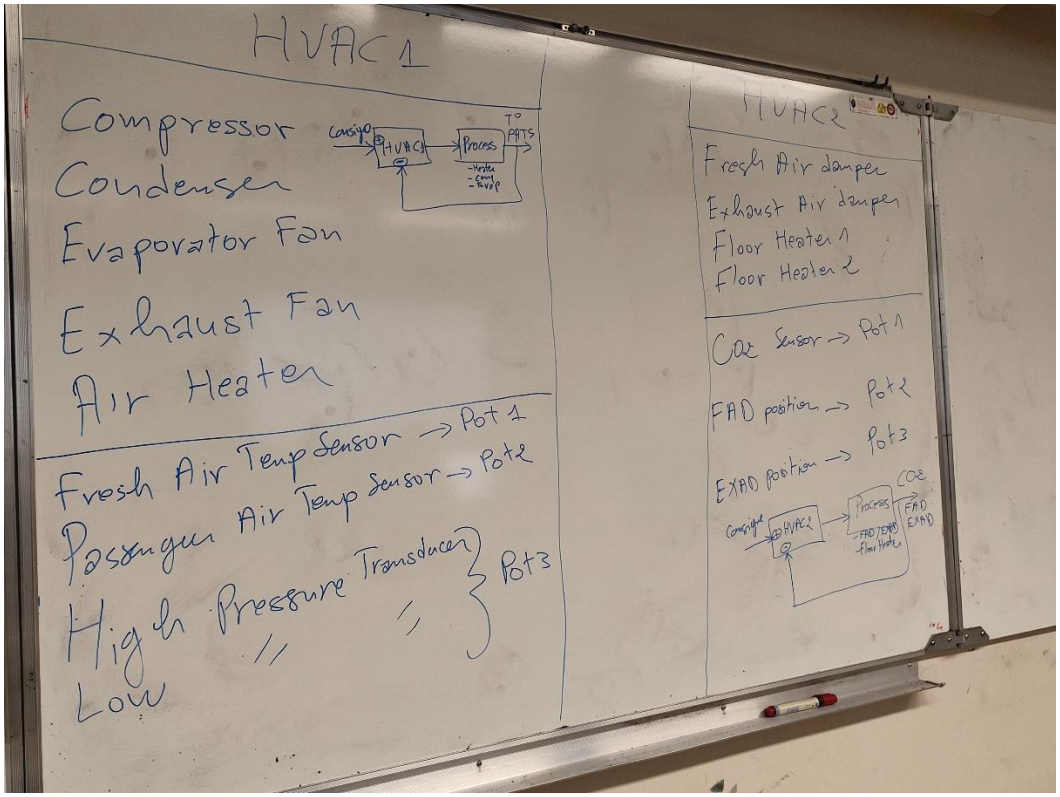
Le but de cette partie est de retester vos codes sources du TP1, relancer le menu de configuration de BuildRoot et ajouter les modules que vous jugez nécessaires à cette application.

Rappel : explication montage



➔ Vous pouvez améliorer votre montage en utilisant les optocoupleurs pour simuler les relais.





HVAC1 : boucle régulation de température.

HVAC2 : boucle régulation de CO2.

Mais, vu que /

- Le feedback de l'Air Heater est lu par le HVAC2.
- Les Floors heaters 1 et 2 sont commandés par le HVAC 2.

Les deux Controllers doivent coopérer !

Comment faire ça le plus simplement possible ? → On utilise MQTT !

Le Broker MQTT sera implémenté dans l'ordinateur central du train (tcms.mpu1). celui-ci contiendra une arborescence de tout les HVAC et leurs variable ! chaque cycle de calcul, les Controllers hvac doivent envoyer l'état de leur variables internes.

Donc il suffit que le HVAC1 écrit une valeur égal à 1 ou à 0 dans les variables FloorHeater1_Cmd et FloorHeater2_Cmd (pour démarrer/arrêter un floor heater). Et le HVAC2 va se connecter à ces variables et vérifier leur état à chaque cycle de calcul afin de savoir s'il doit les mettre à 1 ou à 0 (allumer ou éteindre les LEDs).

De même pour le feedback de l'Air Heater, le HVAC1 se connecte à la variable AirHeater_Fbk et vérifie son état à chaque cycle de calcul.

NB : le HVAC1 peut demander aussi la fermeture ou l'ouverture des dampers FAD et EXAD au HVAC2, suivant les valeurs de température (pour garder la chaleur à l'intérieur du train). Mais pour faciliter l'algorithme, le HVAC2 peut simplement surveiller les valeurs des températures FATS et PATS transmises par le HVAC1 au tcms.mpu1 et décider de lui-même quand il faut ouvrir ou fermer les dampers.