Table des matières

Les exercices suivants ne sont pas notés. Il y aura à la fin du mode une application rendre à l'aide des TP que vous aurez réalisés.

1 Trace d'exécution

Il est toujours pratique de savoir où on en est dans l'avancement d'un programme et de pouvoir placer des traces pour cerner le moment où le programme plante ou bien une variable change de valeur.

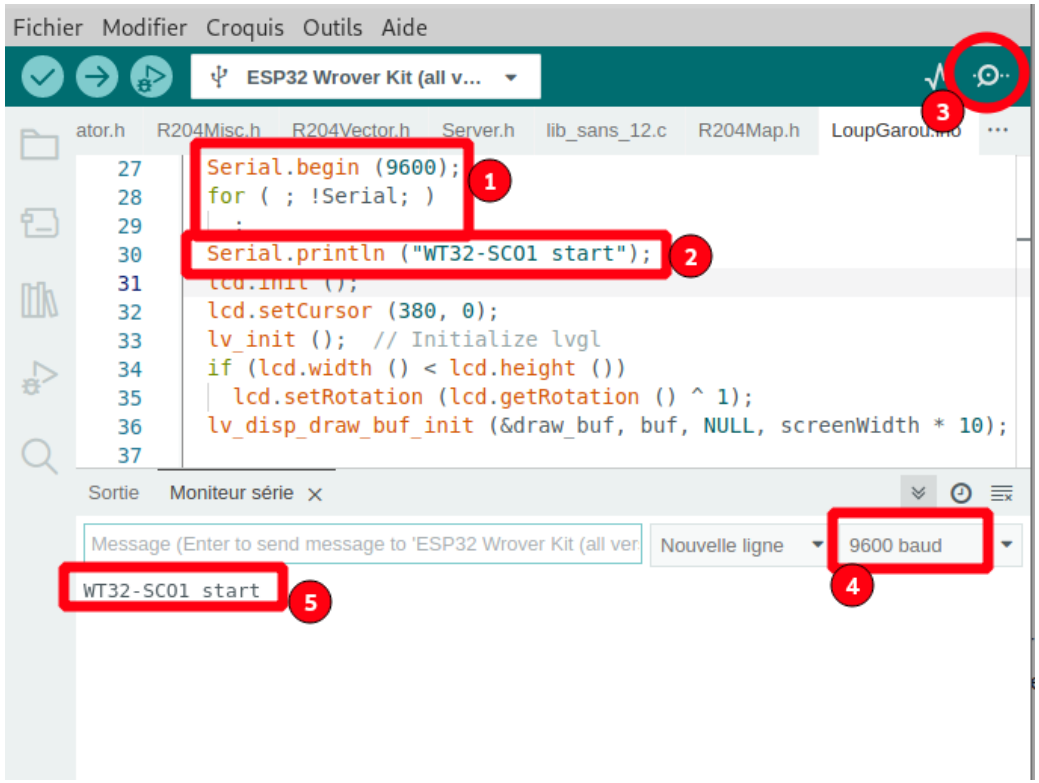
Le port USB sert à programmer la carte, il sert également à communiquer. Pour cela, on utilise la fonction d'écriture sur le port série. Il y a 2 côtés à traiter.

Dans votre programme qui sera exécutée sur la carte,

- ❶ il faut ouvrir le port série à une certaine cadence (ici 9 600 bauds = 9 600 bits par seconde) et attendre que le port série soit prêt (Serie retourne vrai dans ce cas).
- ❷ Ensuite, vous pourrez utiliser la méthode « print » ou « println » qui prennent une chaîne de caractères ou un objet de la classe String.

Côté simulateur,

- ❸ il faut ouvrir la fenêtre « moniteur série »
- ❹ vérifier que la cadence d'écrire est la même (ici 9600 bauds)
- ❺ et vous pourrez vérifier les traces



En partant d'un exemple de label, affichez un pseudo.

```
String monPseudo = "Lucien" ;
…
    statusBar = lv_label_create (lv_scr_act ());
    lv_label_set_recolor (statusBar, true);
    lv_label_set_text (statusBar, monPseudo.c_str ());
    lv_obj_set_size (statusBar, 480, 20);
    lv_obj_align (statusBar, LV_ALIGN_TOP_MID, 0, 0);
```

2 Entraînement de la semaine

Cette semaine nous allons choisir le mode serveur et développer les fonctions d'un serveur de jeux. La semaine prochaine nous traiterons la partie cliente.

L'avantage de traiter la partie serveur est que nous pourrons facilement tester les fonctions à partir d'un ordinateur portable se connectant sur le Wifi de la carte et utiliser un navigateur Firefox pour communiquer avec le jeu.

Nous allons chercher à développer un serveur web embarqué dans la carte qui gère les situations suivantes :

1. Sélection du SSID (réseau wifi)
2. Lancement du serveur
3. Acceptation d'un client web
4. Inscription d'un pseudo
5. Gestion du maître de jeu
6. Répartition des joueurs dans des groupes de rôle (loup-garou, villageois, voyante, voleur, chasseur, cupidon, sorcière, petite fille capitaine)
7. Gestion d'un vote pour un groupe

On peut se référer par exemple aux règles sur https://www.regledujeu.fr/loup-garou-regle/

Les étapes de 1 à 4 sont déjà réalisé.

Vous placerez dans un répertoire LoupGaroup les sources trouvées sur r204.merciol.fr. Cela comprend :

- LoupGarou.ino : le schéma général
- lib_sans_12.c : la police de caractères
- SetupNet.cpp SetupNet.h : les définitions communes
- Client.h, Client.cpp : la gestion d'un participant
- Server.cpp Server.h : la gestion du maître de jeu

Mais également

- R204Iterator.h : itérateur de tableaux et de tables associatives
- R204Vector.h : les tableaux dynamiques
- R204Map.h : les tables associatives
- R204Misc.h : des traitements utiles

2.1 Validation

Commencez par compiler et tester.

2.2 Gestion du maître de jeu

Il peut être judicieux d'ajouter provisoirement une commande « /admin » qui permettra de déclencher certaines actions (lancement de la répartition). Cet accès pourra être supprimé lorsque vous aurez développé l'interface graphique su serveur.

2.3 Répartition des rôles

Séparez le traitement en 2 actions séparer :

- L'administrateur qui lance la répartition aléatoire
- Les joueurs qui demandent quel est leur rôle

2.4 Gestion de vote

Écrire une fonction qui comptabilise des votes sur un groupe et indique (uniquement au groupe) le score provisoire.

L'administrateur figera la fin du vote.