Guide pas-à-pas pour la communication infrarouge (émetteur et récepteur)

1. Objectif du système

La communication infrarouge a pour but d'assurer une liaison simple entre les deux robots. Plus précisément, dans le cadre du jeu de cache-cache, le robot qui se cache utilise un émetteur infrarouge pour diffuser un signal, tandis que le robot chercheur est équipé d'un récepteur infrarouge lui permettant de détecter la présence du robot caché.

2. Composants utilisés

Pour établir la communication infrarouge entre les deux robots, nous avons utilisé des modules de la gamme Grove.

Émetteur infrarouge

- **Nom du module** : *Grove – Infrared Emitter v1.2*
- **Description** : Ce module contient une LED infrarouge capable d'émettre un signal infrarouge. Il est conçu pour fonctionner sous une tension de 5V et se commande via une broche numérique du microcontrôleur.
- **lien** : https://wiki.seeedstudio.com/Grove-Infrared_Emitter/

Récepteur infrarouge

- **Nom du module** : *Grove – Infrared Receiver v1.0*
- **Description** : Ce module est équipé d'un capteur infrarouge capable de détecter un signal IR provenant de l'émetteur. Il délivre un signal logique en sortie, interprétable par le microcontrôleur pour déterminer la réception d'un signal.
- **lien** : https://wiki.seeedstudio.com/Grove-Infrared_Emitter/

3. Branchement utilisé

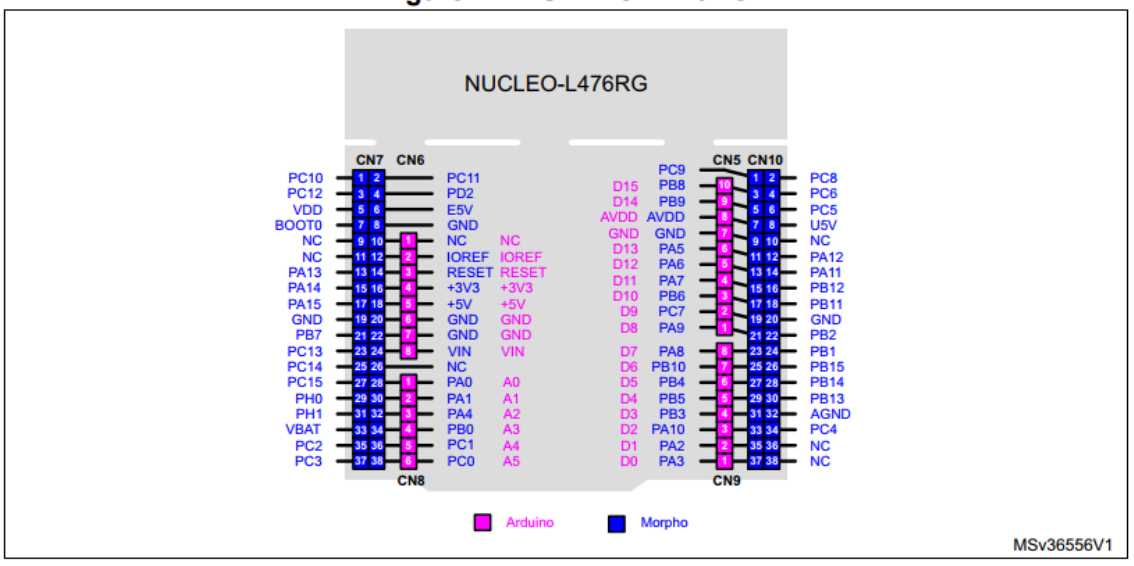
Émetteur infrarouge

STM32-L476RG	Grove - Infrared Emitter
5V	Red
GND	Black
Not connected	White
D2	Yellow

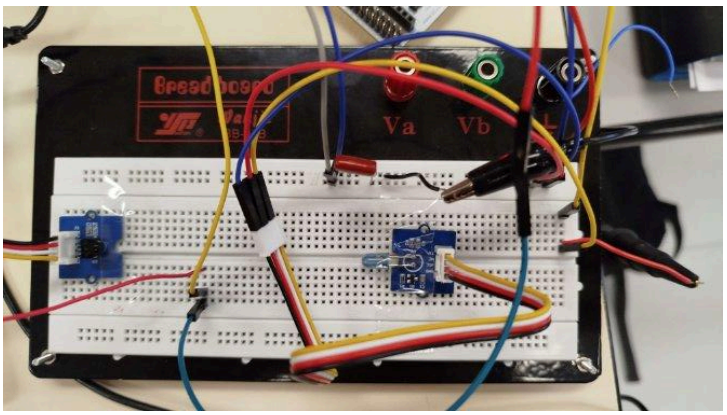
Récepteur infrarouge

STM32-L476RG	Grove - Infrared Emitter
5V	Red
GND	Black
Not connected	White
D4	Yellow

Figure 24. NUCLEO-L476RG



Nucleo-L476RG pin mapping



Récepteur et émetteur mis en face pour assurer le bon fonctionnement

4. Principe de fonctionnement

Du côté émetteur, un signal créneaux (succession d'états hauts et bas) est généré via la broche numérique choisie du microcontrôleur pour allumer et éteindre la LED infrarouge à une fréquence régulière.

Du côté récepteur, ce signal est interprété électriquement : à chaque front montant envoyé par l'émetteur, le récepteur réagit en produisant un front descendant.

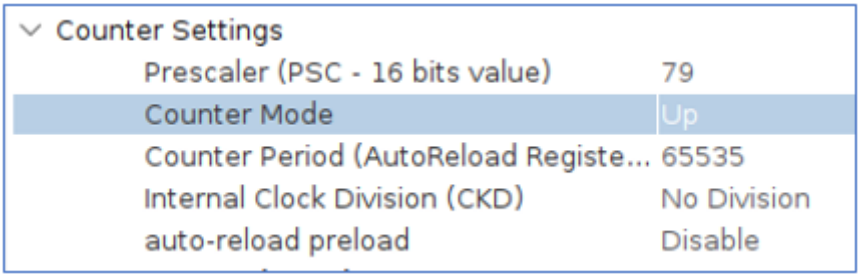
Observation experimentale à l'osscilloscope de la réaction du récepteur (rose) au signal de l'émetteur (jaune)

5. Gestion logicielle - Le timer

Afin de gérer la communication infrarouge en parallèle des déplacements du robot, nous avons mis en place un timer. L'objectif principal de ce timer est de permettre l'émission ou la lecture de signaux infrarouges sans bloquer le reste du programme. Le timer fonctionne en arrière-plan et déclenche régulièrement une routine permettant :

- D'envoyer un signal IR, ou
- De vérifier l'état du récepteur IR, ou
- D'agir en fonction de la lecture du récepteur

Activation du timer (choix arbitraire)



Paramètre du timer

Vous trouverez la démarche dans le tp ou dans le document Developer's guide for STM32CubeIDE v31 12/03/2025 à la section 8.2.

6. Implémentation des capteurs

La broche de l'émetteur est configurée en mode output et celle du récepteur en mode input. À noter que le choix des broches est arbitraire.

PA10	n/a	Low	Output Pu...	No pull-up...	Very High	n/a	EMITTER_PIN	☑
PA13 (JTM...	SYS_JTMS-...	n/a	n/a	n/a	n/a	n/a	TMS	☑
PA14 (JTCK...	SYS_JTCK-...	n/a	n/a	n/a	n/a	n/a	TCK	☑
PB0	n/a	Low	Output Pu...	No pull-up...	Low	n/a	MOTOR_R_...	☑
PB3 (JTDO-...	SYS_JTDO-...	n/a	n/a	n/a	n/a	n/a	SWO	☑
PB5	n/a	n/a	Input mode	No pull-up...	n/a	n/a	RECEIVER_...	☑

Activation des broches dans le bon mode

Puis pour le code on définit des macros qui permettent de pas mettre de valeur magique.

```
// Port D2
#define EMITTER_GPIO GPIOA
#define EMITTER_PIN GPIO_PIN_10
```

Définition d'une macro pour l'émetteur

```
// Port D4
#define RECEIVER_GPIO GPIOB
#define RECEIVER_PIN GPIO_PIN_5
```

Définition d'une macro pour le récepteur

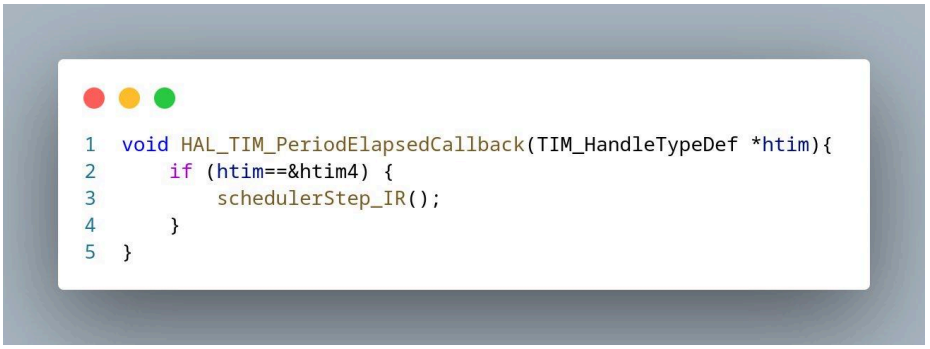
7. Exemple de code

```
typedef enum TASK_IR{
    TASK_EMITTER,
    TASK_EVOLVE,
    TASK_RECEIVER,
    TASK_RAW
}task_IR_t;
```

Utilisation d'un type énuméré pour différencier les différentes tâche à réaliser (TASK_RAW sert à faire un modulo, et permet donc d'ajouter des tâches si besoin est)

```
void schedulerStep_IR(){
    currentTask = (currentTask + 1) % TASK_RAW;
    switch (currentTask){
        case TASK_EMITTER:
            TASK_Emitter();
            break;
        case TASK_EVOLVE:
            TASK_Evolve();
            break;
        case TASK_RECEIVER:
            TASK_Receiver();
            break;
        default :
            break;
    }
}
```

Utilisation d'un scheduler avec un switch, les fonctions appelées sont définis dans un autre fichier. Vous pouvez choisir ce qu'elle font



```c
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
    if (htim==&htim4) {
        schedulerStep_IR();
    }
}
```

C'est une fonction que l'on réécrit (attribut weak) pour appeler le scheduler lorsque notre timer (timer 4) fait une interruption.

```c
void TASK_Emitter(){
    EMITTER_State = (((int)(HAL_GetTick()/EMITTER_PERIOD))%EMITTER_DUTY != 0) ? 0 : 1;
    setEmitter(EMITTER_State);
}
```

Code de TASK_Emitter, qui permet de faire clignoter la led

```c
void TASK_Receiver(){
    if(searchSuccess != 1){
        searchSuccess = RECEIVER_GetActivate();
    }
    return;
}
```

Code de TASK_Receiver, qui interroge le récepteur si jamais il a été activé. Pour le code de GetActivate référez vous au tp avec le bouton.

```c
void TASK_Evolve(){
    if(searchSuccess == 1){
        GREEN_LED_State = 1 - GREEN_LED_State;
        setGreenLed(GREEN_LED_State);
        RECEIVER_Pressed = 0;
    }
    return;
}
```

Ici c'est un exemple de code pour TASK_Evolve, la led clignote si l'autre robot à été trouvé.

8. Conseil pratique

Pour éviter les erreurs et maximiser l'efficacité, voici quelques recommandations issues de notre expérience :

- **Vérifiez soigneusement vos branchements** : assurez-vous que chaque fil est connecté à la bonne broche (VCC, GND, signal).

- **Paramétrez correctement l'IOC** : une erreur arrive vite ici

- **Écrivez un code lisible et commenté** : cela facilite la compréhension pour les autres et lors des sessions suivantes et surtout limite les oublis.

- **Alignez bien les capteurs** : pour une détection fiable, l'émetteur et le récepteur doivent être à **la même hauteur** et **placés en face l'un de l'autre**.la LED infrarouge émet dans un cône d'environ **±17°**, donc un mauvais alignement réduit fortement la portée du signal.