Centre professionnel du Nord vaudois
Filière d'automatique et d'électronique
Modules complémentaires techniques

CPNV

Trame de communication

P1704_Manettes

PICOTTE Alexandre
SANDOZ Pierre-Olivier
15/06/2018

Table des matières

Introduction :

L'état actuelle du code de la manette contient trois trames, qui ne peuvent être activés simultanément. Chaque trame est envoyée au moyen de la librairie PJON. Le code laisse le choix de la trame, selon la trame choisie l'arduino n'envoie pas les mêmes informations. Pour sélectionner la première ou la seconde trame, il suffit de mettre « TRAME » à 1 pour la première et à 0 pour la deuxième. La troisième trame envoie des messages, elle résulte d'une demande spécifique de la part du groupe « chevaliers et dragon », elle sera détaillée plus bas.

Il est important de noter que la libraire PJON envoie seulement des tableaux de charactères, rien d'autres.

```
/*************************************************************************
  Array of char who will be sent to the other arduino
*************************************************************************/
namespace {
// name of namespace left empty --> this is the anonymous namespace
// now the IDE will not mess with our stuff

#define TRAME 0 //defini quel type de trame le programme doit traiter
//0 = une liste avec toutes les valeurs
//1 = les valeurs sont traiter indpendamment des autres

#if TRAME // if TRAME = 1
char to_send_joystick[5]; // [0] = joystick
// [1] = first byte val_joystick[0] // [2] = second byte val_joystick[0]
// [3] = first byte val_joystick[1] // [4] = second byte val_joystick[1]

char to_send_bouton1[2] = {bouton1, 0};
// [0] = bouton1 // [1] = the state of the button -> 2 if pressed, else 1

char to_send_bouton2[2] = {bouton2, 0};
// [0] = bouton2 // [1] = the state of the button -> 2 if pressed, else 1

char to_send_bouton3[2] = {bouton3, 0};
// [0] = bouton3 // [1] = the state of the button -> 2 if pressed, else 1

char to_send_bouton4[2] = {bouton4, 0};
// [0] = bouton4 // [1] = the state of the button -> 2 if pressed, else 1

char to_send_mode[2] = {mode_de_jeu, 0};
// [0] = mode_de_jeu // [1] = current menu on the arduino

#else // if TRAME = 0
char to_send[9] = {0, 0, 0, 0, 1, 1, 1, 1, 1};
//[0,1] = value X from the joystick //[2,3] = value Y from the joystick
//[4] = state of bouton 1 -> 2 if pressed, else 1
//[5] = state of bouton 2 -> 2 if pressed, else 1
//[6] = state of bouton 3 -> 2 if pressed, else 1
//[7] = state of bouton 4 -> 2 if pressed, else 1
//[8] = current menu of the arduino
//menu: 1 = bluetooth / 2 = main menu / 3 = in game / 4 = pause / 5 = game over
#endif
}
```

Première trame :

Envoi de données :

```
#if TRAME // if TRAME = 1
char to_send_joystick[5]; // [0] = joystick
// [1] = first byte val_joystick[0] // [2] = second byte val_joystick[0]
// [3] = first byte val_joystick[1] // [4] = second byte val_joystick[1]

char to_send_bouton1[2] = {bouton1, 0};
// [0] = bouton1 // [1] = the state of the button -> 2 if pressed, else 1

char to_send_bouton2[2] = {bouton2, 0};
// [0] = bouton2 // [1] = the state of the button -> 2 if pressed, else 1

char to_send_bouton3[2] = {bouton3, 0};
// [0] = bouton3 // [1] = the state of the button -> 2 if pressed, else 1

char to_send_bouton4[2] = {bouton4, 0};
// [0] = bouton4 // [1] = the state of the button -> 2 if pressed, else 1

char to_send_mode[2] = {mode_de_jeu, 0};
// [0] = mode_de_jeu // [1] = current menu on the arduino
```

La première méthode qui est active que lorsque « TRAME » vaut 1, envoie les informations indépendamment des autres. L'arduino ne communique pas toutes les informations relatives à ses inputs et à son mode actuelle lorsqu'il envoie des données, l'arduino transmet que les informations qui ont changé d'un cycle à l'autre.

Le tableau suivant concerne les adresses attribuées :

```
/*****************************************************************************
   Define to where is adressed the first byte of the data sent
*****************************************************************************/
#define joystick 1 //Pin A0 for x and A1 for y
#define bouton1 2
#define bouton2 3
#define bouton3 4
#define bouton4 5
#define mode_de_jeu 6
```

La première valeur de chaque tableau contient l'adresse à laquelle la donnée doit être enregistrée. Selon cette valeur l'arduino qui réceptionne les données peut déterminer à quelle variable enregistrer le changement.

Trame de communication

```
void bouton_pressed() {

  if (digitalRead(bouton_1) == LOW) {
    delay(1);
    if (etat_bouton_1 == false) {

      val_bouton1 = true;

        #if TRAME
        to_send_bouton1[1] = 2;
        #else
        to_send[4] = 2;
        #endif

      change_bouton1 = true;
    }
    etat_bouton_1 = true;
  } else {
    etat_bouton_1 = false;
    val_bouton1 = false;

      #if TRAME
      to_send_bouton1[1] = 1;
      #else
      to_send[4] = 1;
      #endif
  }
```

```
void check_change() {

    #if TRAME
    if (change_joystick) {
      change_joystick = false;
      bus.send_packet(44, to_send_joystick, 5);
    }
    if (change_bouton1) {
      change_bouton1 = false;
      bus.send_packet(44, to_send_bouton1, 2);
    }
    if (change_bouton2) {
      change_bouton2 = false;
      bus.send_packet(44, to_send_bouton2, 2);
    }
    if (change_bouton3) {
      change_bouton3 = false;
      bus.send_packet(44, to_send_bouton3, 2);
    }
    if (change_bouton4) {
      change_bouton4 = false;
      bus.send_packet(44, to_send_bouton4, 2);
    }
```

Lorsque l'arduino détecte un changement, il va remplir le tableau correspondant avec les nouvelles valeurs puis l'envoyé ; par exemple si l'arduino détecte que le bouton 1 est appuyé, alors il va modifier le tableau correspondant au bouton 1, dans ce cas la deuxième valeur du tableau « to_send_bouton1 », puis l'arduino communiquera seulement ce tableau à l'autre arduino. Le choix de la valeur représentant l'état des boutons a été attribué arbitrairement, 2 pour symboliser que le bouton est pressé sinon 1.

Pour le joystick, l'arduino stocke lors de son *setup* la première valeur de l'axe X et Y qu'il lit, ces deux valeurs deviennent alors des valeurs de base qui servent à déterminer s'il y a eu un changement, en comparant les valeurs actuelles aux valeurs de base

```
/*****************************************************************************
   Function Name: setup
   Description:
   Parameters:
   Return:
*****************************************************************************/
void setup() {
  // put your setup code here, to run once:

val_joystick_base[0] = analogRead(joystick_X); //neutral state X
val_joystick_base[1] = analogRead(joystick_Y); //neutral state Y
```

```
/*********************************************************************
   Function Name: check_joystick
   Description:  check the value of the joystick and store the news values
   Parameters:
   Return:
*********************************************************************/
void check_joystick() {
  //Check the value of the joystick
  val_joystick[0] = analogRead(joystick_X);
  val_joystick[1] = analogRead(joystick_Y);

  //Check if the axe X
  if (val_joystick[0] != val_joystick_base[0]) {
    change_joystick = true;
    change = true;
#if TRAME
    to_send_joystick[1] = highByte(val_joystick[0]);
    to_send_joystick[2] = lowByte(val_joystick[0]);
#else
    to_send[0] = highByte(val_joystick[0]);
    to_send[1] = lowByte(val_joystick[0]);
#endif

  }
```

Les valeurs des joysticks sont stockées dans des integers mais comme on peut que faire transiter des tableaux de charactères il faut séparer les valeurs des axes en deux bytes chacun pour les enregistrer dans des charactères.

Réception des données :

La réception de donnée s'effectue au moyen de la fonction « receiver_function » de la librairie PJON.

Dans le code actuel nous utilisons seulement le tableau payload.

```
/*********************************************************************
   Function Name: receiver_function
   Description:  function where the data sent by the other arduino are readed
   Parameters: payload is an array of char containing the data
   Return:
*********************************************************************/
void receiver_function(uint8_t *payload, uint16_t length, const PJON_Packet_Info &packet_info) {
  /* Make use of the payload before sending something, the buffer where payload points to is
     overwritten when a new message is dispatched */
```

```
#if TRAME
switch (payload[0]) {
case 1:
  x1 = payload[1];
  x2 = payload[2];
  val_X = (x1 << 8 | x2);
  y1 = payload[3];
  y2 = payload[4];
  val_Y = (y1 << 8 | y2);
  change_joy = true;
  break;
case 2:
  val_bouton1 = payload[1];
  change_bou1 = true;
  break;
case 3:
  val_bouton2 = payload[1];
  change_bou2 = true;
  break;
case 4:
  val_bouton3 = payload[1];
  change_bou3 = true;
  break;
case 5:
  val_bouton4 = payload[1];
  change_bou4 = true;
  break;
case 6:
  val_mode = payload[1];
  break;
}
```

Comme la première valeur de chaque tableau envoyé représente l'adresse, il est aisé de trier les données reçus. Il suffit alors de stocker les données dans les variables allouées et d'indiquer un changement dans les informations réceptionnées.

Les boutons ne nécessitent pas d'opérations autre qu'enregistrer la valeur reçue contrairement au joystick. Les valeurs X et Y du joystick sont des integers, il convient alors de séparer les deux bytes de l'integer pour enregistrer chaque byte dans un charactère, highByte et lowByte le permette facilement. Puis une fois envoyés il faut reformer l'integer, il est à noter que les variables charactère sont *signed* ainsi pour éviter d'avoir des valeurs négatives quand on lit les données reçues il faut enregistrer les deux charactères dans des *unsigned* charactère.

Ainsi chaque tableau est composé d'une adresse et d'une donnée représentant la valeur de l'input. L'adresse sert à l'arduino réceptionnant les données pour savoir à qui est attribué la valeur de la deuxième entrée du tableau.

Deuxième trame :

Envoi de données :

```
#else // if TRAME = 0
char to_send[9] = {0, 0, 0, 0, 1, 1, 1, 1, 1};
//[0,1] = value X from the joystick //[2,3] = value Y from the joystick
//[4] = state of bouton 1 -> 2 if pressed, else 1
//[5] = state of bouton 2 -> 2 if pressed, else 1
//[6] = state of bouton 3 -> 2 if pressed, else 1
//[7] = state of bouton 4 -> 2 if pressed, else 1
//[8] = current menu of the arduino
//menu: 1 = bluetooth / 2 = main menu / 3 = in game / 4 = pause / 5 = game over
#endif
}
```

La deuxième méthode consiste à envoyer toutes les données. Cette méthode ne nécessite plus d'adresse, car l'arduino réceptionnant les données à besoin de recevoir qu'un seul tableau, il ne reste alors qu'à préciser dans quelles variables enregistrer ces valeurs,

Cette méthode envoie un unique tableau composé de toutes les valeurs des inputs et du menu actuel, chacune de ces valeurs sont attribuées une place précise dans le tableau à envoyer : les deux premières entrées sont pour l'axe x du joystick, les deux suivantes pour l'axe y, la cinquième entrée est réservée au bouton 1, la sixième au bouton 2 et ainsi de suite, la dernière entrée est attribuée au menu de l'arduino. L'arduino qui reçoit les données peut alors attribuer les valeurs du tableau reçu à ses variables.

L'arduino envoie les données que lorsqu'il détecte un changement, que ce soit les boutons, le joystick, les changements de menu sont traités indépendamment et sont envoyés lorsque l'arduino configure le nouveau menu.

```
if (etat_bouton_4 || etat_bouton_3 ||
    etat_bouton_2 || etat_bouton_1) {
  if (!etat_change) {
    change = true;
  }
  etat_change = true;
}
else {
  etat_change = false;

}
```

```
  //Check if the axe X
  if (val_joystick[0] != val_joystick_base[0]) {
    change_joystick = true;
    change = true;
#if TRAME
    to_send_joystick[1] = highByte(val_joystick[0]);
    to_send_joystick[2] = lowByte(val_joystick[0]);
#else
    to_send[0] = highByte(val_joystick[0]);
    to_send[1] = lowByte(val_joystick[0]);
#endif
  }
```

Ainsi lorsque la variable change passe à true l'arduino enverra un tableau de charactère contenant les nouvelles valeurs.

```
/*****************************************************************************
   Function Name: check_change
   Description:  verifie s'il y a eu un changement de valeur
                 pour le joystick et les boutons
                 si oui alors il envoie les nouvelles informations
   Parameters:
   Return:
*****************************************************************************/
void check_change() {

...

#else
  if (change) {

    change = false;
    bus.send_packet(44, to_send, 9);
  }
#endif
```

Recetion des données :

Comme expliqué plus haut l'arduino recevant les données doit juste trier les données, chaque entrée du tableau reçu est attibué à une variable.

Trame de communication

```
/**************************************************************************
   Function Name: receiver_function
   Description:  function where the data sent by the other arduino are readed
   Parameters: payload is an array of char containing the data
   Return:
**************************************************************************/
void receiver_function(uint8_t *payload, uint16_t length, const PJON_Packet_Info &packet_info) {
  /* Make use of the payload before sending something, the buffer where payload points to is
     overwritten when a new message is dispatched */
```

…

```
#else

x1 = payload[0];
x2 = payload[1];
if (val_X != (x1 << 8 | x2)) {
val_X = (x1 << 8 | x2);
change_joy = true;
}
y1 = payload[2];
y2 = payload[3];
if (val_Y != (y1 << 8 | y2)) {
val_Y = (y1 << 8 | y2);
change_joy = true;
}
if (val_bouton1 != payload[4]) {
val_bouton1 = payload[4];
change_bou1 = true;
}
if (val_bouton2 != payload[5]) {
val_bouton2 = payload[5];
change_bou2 = true;
}
if (val_bouton3 != payload[6]) {
val_bouton3 = payload[6];
change_bou3 = true;
}
if (val_bouton4 != payload[7]) {
val_bouton4 = payload[7];
change_bou4 = true;
}
if (val_mode != payload[8]) {
val_mode = payload[8];
}

#endif
}
```

La première méthode permet de limiter les informations envoyées, afin d'éviter que l'arduino communique des données superflues, la deuxième méthode permet de s'affranchir des adresses et de nos craintes que l'arduino perde un des packet à envoyer.

Troisième trame :

Cette troisième trame résulte d'une demande du groupe « chevaliers et dragon ». Contrairement aux deux trames précédentes, celle-ci n'envoie pas à proprement parler de données, il ne s'agit pas de tableau à lire pour extraire les états des boutons. Cette trame envoie un message selon les informations de l'arduino, c'est-à-dire qu'il envoie des informations spécifiques à son état :

Il y a 4 adresses :

- 0 pour « Game »
- 1 pour « Contact »
- 2 pour « Motors »
- 3 pour « Head »

Adresse	Donnée	Description
0	ConRequest	Permet de savoir si le Bluetooth est bien connecté
	Pause	Indique que le mode Pause est activé.
	Restart	Indique que le mode Pause est désactivé
	Over	Indique que la partie est perdue
1	ShieldUp	Activation du bouclier
	ShieldDown	Désactivation du bouclier
	SpecialUp	Activation de l'attaque spéciale
	SpecialDown	Désactivation de l'attaque spéciale
2	x	Position X du joystick
	,	Séparation entre X et Y
	y	Position Y du joystick
3	LoseLife	Perte de point de vie
	GainLife	Gain de point de vie

Les cellules en gris signifient que la donnée n'est pas utilisée dans la version de 2018 du projet P1631.
Les cellules en bleu signifient que la donnée est ajoutée pour la version 2018 du projet.

Le but étant de pouvoir implémenter la libraire PJON au protocole de communication réalisé par le groupe « chevaliers et dragon ». Malheureusement même si la trame n'a aucun problème à être envoyer et lu, le manque de connaissance du code du robot et le manque de temps font que la trame a peu d'utilité.

Envoi de donnée :

Les données à envoyer sont des chaines de charactères composé d'une adresse et de lettres

```
//NEW TRAME
char p0[6] = {0, 'P', 'a', 'u', 's', 'e'};
char r0[8] = {0, 'R', 'e', 's', 't', 'a', 'r', 't'};
char o0[5] = {0, 'O', 'v', 'e', 'r'};

char su1[9] = {1, 'S', 'h', 'i', 'e', 'l', 'd', 'U', 'p'};
char sd1[11] = {1, 'S', 'h', 'i', 'e', 'l', 'd', 'D', 'o', 'w', 'n'};
char sup1[10] = {1, 'S', 'p', 'e', 'c', 'i', 'a', 'l', 'U', 'p'};
char sdo1[12] = {1, 'S', 'p', 'e', 'c', 'i', 'a', 'l', 'D', 'o', 'w', 'n'};

char joy2[8] = {2, 0, 0, 0, ',', 0, 0, 0};

char ll3[9] = {3, 'L', 'o', 's', 'e', 'L', 'i', 'f', 'e'};
char gl3[9] = {3, 'G', 'a', 'i', 'n', 'L', 'i', 'f', 'e'};
```

Ces données sont à envoyer à des moments spécifiques, il ne s'agit plus d'envoyer seulement des données sur l'état des boutons ou la valeur du joystick mais des informations spécifiques au code chevalier.

Pour le joystick il a fallu convertir des integers en chaine de lettres, non pas comme dans la trame 1 et 2 où il s'agissait de rentrer un integer dans deux charactères.

```
char b[4];
String str;
str = String(analogRead(joystick_X));
str.toCharArray(b, 4);
joy2[1] = str[0]; joy2[2] = str[1]; joy2[3] = str[2];
```

Ce bout de code sert à convertir des integer en chaine de charactère. Pour ce faire on utilise des *String* qui vont servir de « passerelle ». On commence par créer une chaine de caractère vide de taille 4, puis on crée une *String* dans laquelle on met notre integer. Ensuite on utilise la fonction *toCharArray* pour transformer la valeur en charactère, attention la chaine de caractère doit posséder un espace vide pour la *null-terminaison*.

Réception des données :

```
/*************************************************************************
   Function Name: receiver_function
   Description:  function where the data sent by the other arduino are readed
   Parameters: payload is an array of char containing the data
   Return:
*************************************************************************/
void receiver_function(uint8_t *payload, uint16_t length, const PJON_Packet_Info &packet_info) {
  /* Make use of the payload before sending something, the buffer where payload points to is
     overwritten when a new message is dispatched */
```

```
    for (int i = 1; i < length; i++) {
      char c = payload[i];
      Serial.print(c);
    }
    Serial.println(" ");
```

Ce code n'enregistre pas de donnée il écrit simplement le texte qu'il a reçu.