Lycée Pablo Picasso, PERPIGNAN

2013-2014

EYE-GLE

Projet Personnel Encadré

DOSSIER PERSONNEL

GUILLAMOT VINCENT

TG6

Table des matières.

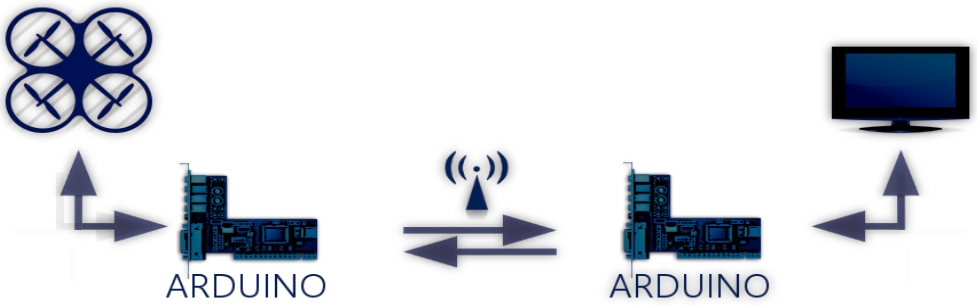
Introduction

Le projet est un grand exemple de la réalisation d'une idée. Les objectifs du projet sont simples : créer un module de scan 3D pour un drone. Cette fonctionnalité est destinée aux unités de secours (pompiers, gendarmerie, sécurité civile, etc…). Plus précisément, le scan permettra de restituer un environnement en 3D en temps réel, donc lors des déplacements du drone l'image affichée sur le moniteur sera toujours actualisée.

En résumé, les caractéristiques du projet sont les suivantes :
- Création d'un module caméra/laser faisant un balayage vertical et communiquant avec un ordinateur sur un système sans-fil.
- Création d'une interface pour l'ordinateur permettant l'affichage du scan en temps réel, d'un programme de configuration du module, et d'une console de commande évitant les erreurs du programme, l'exécution de tâche de maintenance (remise à zéro, etc…) et l'exécution de tâches spécifiques.

Notre projet, le scanner laser embarqué sur un drone, requiert de nombreuses compétences : une grande partie d'électronique (radiocommunication, encodage et acquisition vidéo), une part non négligeable de programmation de systèmes numériques (Arduino, OpenGl et Qt en C++), la mécanique n'est pas absente et assure la cohésion de l'ensemble (fonctionnement du drone).
Ce projet a pour but d'éviter des risques inutiles aux hommes en permettant à ces derniers d'explorer des environnements instables et potentiellement dangereux comme par exemple des immeubles insalubres et inaccessibles à l'homme. En reprenant cet exemple, notre projet permettra par exemple de disposer de la cartographie d'un immeuble pour organiser des secours.

Autre particularité de fonctionnement, le système est composé de deux module, un situé sur le drone (on le nommera module D) et un autre connecté à l'ordinateur par une liaison USB (nommé module P).



« Schéma communication entre le pc et le drone »

Composition des modules :

Module D :

Le module D est composé d'une arduino mini (dû à la contrainte de poids) d'un transceiver, d'un servomoteur, d'un transistor, d'un laser, d'un CI (Circuit intégré) 74HC595 et de quatre LED.

Module P :

Ce module est quant à lui très léger, en effet il est uniquement constitué d'une arduino UNO (dispose d'un connecteur USB lui permettant d'être alimenté et de communiquer avec l'ordinateur) et d'un transceiver.

Les deux modules communiquent entre eux pour un système sans-fil utilisant les radiofréquences (en l'occurrence sur la fréquence 2,4Ghz) au moyen des transceivers (émetteur-récepteur).
Le module D envoie par le biais du système sans-fil uniquement l'angle du servomoteur (et donc du laser) en temps réel et peut recevoir des commandes de l'ordinateur.

Nous allons maintenant voir quelque donnée sur la carte électronique permettant la programmation du module D.

Arduino

L'arduino est une carte électronique dites « open-hardware » et « open-source », c'est-à-dire que tout le projet « Arduino » est ouvert et totalement accessible. Elle dispose d'un circuit de programmation intégrer, et d'un « bootloader » qui permet de programmer la puce ATMega328 avec un langage de programmation propre à Arduino. Celui-ci est un dérivé du C disposant des librairies et des fonctions permettant une programmation simplifiée de micro-contrôleur.

Servomoteur

Le servomoteur est un moteur à courant continue disposant de son électronique d'asservissement et de commande, très utilisé en modélisme. Il dispose d'un potentiomètre rotatif lui permettant de régler l'angle suivant la résistance du potentiomètre, du fait de sa présence les servomoteurs sont souvent bridés à une rotation de 180°.

Tout d'abord le servomoteur dispose de trois broches, la première (en partant du haut) reçoit la commande d'angle, la seconde est la borne « + » alimenté en 5V et la troisième la masse.

74HC595

Le Circuit intégré 74HC595 est un registre à décalage, il permet de rajouter des sorties à l'Arduino. Il fonctionne avec un « locker », une horloge et un pin de donnée.

nRF24L01+

Le nRF24L01+ est une « mise à jour » du nRF24L01 qui sont des transceivers (émetteur-récepteur), très utilisé dans le monde professionnel cette puce est simple à programmer avec arduino du fait de la présence de librairie spécialisé, notamment les librairies : SPI.h, Mirf.h,

nRF24L01.h et MirfHardwareSpiDriver.h.

Programmation

La programmation arduino est presque similaire à de la programmation en C mais dispose d'une syntaxe de base spécifique

```
void setup()
{
...
}

void loop()
{
...
}
```

Module D:

Ce programme est constitué donc de la fonction setup(), de la fonction loop(), mais aussi de send_data() qui s'occupe de l'envoie des informations et de change_led qui contrôle la puce 74HC595 où est branché les leds.

Nous commencerons à expliquer les déclarations des librairies et des variables. Nous « includons » cinq librairies à notre premier programme arduino :

Librairie	Utilité
SPI.h	Communication Sans-fil
Mirf.h	Communication Sans-fil
nRF24L01.h	Communication Sans-fil
MirfHardwareSpiDriver.h	Communication Sans-fil
Servo.h	Commande du servomoteur

Puis nous initialisons les variables, les objets (Servo) nécessaire au fonctionnement du programme.

(cf. Programme Module D1)

Ensuite nous avons donc la fonction setup() où nous configurons les paramètres d'envoie pour la liaison avec l'autre arduino et on attache la commande du servo moteur au « pin » 2.

(cf. Programme Module D2)

La fonction loop() est quand à elle le corps principal du programme, il

fonctionne suivant la logique suivante :

Allume les leds de fonctionnement
Si l'angle est de 0° on tourne jusqu'à 90°
    On change l'angle
    On envoie l'angle
    On lit ce que le Module P nous envoie.
        On allume la led de reception
        On l'éteint

Si l'angle est de 90° on tourne jusqu'à 0°
    On change l'angle
    On envoie l'angle
    On lit ce que le Module P nous envoie.
        On allume la led de reception
        On l'éteint
On remet la valeur exacte de 0 à la variable de position

(cf. Programme Module D3)

La fonction send_data() a pour objectif de gérer l'envoie et l'allumage des led d'envoi. L'envoi est soutraité par la librairie Mirf.

(cf. Programme Module D4)

Pour terminer ce programme, la fonction change_led gère toute la puce 74HC595, elle demande un code binaire où le chaque bit équivaut à l'état logique d'une entré. Elle prend commande paramètre, les pins de donnée, d'horloge et le choix de l'opération (par une valeur boolean), data (choix de la led) et le binaire actuel (data_comp). L'addition est faite par une opération logique OR (OU) et la soustraction par une opération logique ET (NAND).

(cf. Programme Module D5)

Module P:

Le programme gère l'envoie et la réception entre le Module D et P mais aussi la communication entre le module P et l'ordinateur par une communication série émulé.

Une particularité de la communication avec l'ordinateur est le fait que la réception ce fait bit par bit, donc nous devons « reconstruire » le code initiale en utilisant des puissances de 10.

(cf. Programme module P en Annexe)

L'interface et librairies de gestion

L'interface ce fait à l'aide de Qt qui est une librairie d'interface. Cette librairies dispose de plusieurs avantage, d'abord il est totalement multi-plateforme et donc fonctionne sous Linux, Windows et même Mac OS, de plus il est disponible sous licence GPL et donc les sources sont totalement libre et finalement la documentation est très complète et bien réalisé.

Le « main » inclus toutes les librairies nécessaires ainsi que nos propre « classe » (mywindow et serialarduino).

Il créer un objet mywindow et utilise les méthode MyWindowBloc(), AffichageMenu(), AffichageShow().

main.cpp

```
#include <QApplication>
#include <QPushButton>
#include <QWidget>
#include <QMainWindow>
#include <qextserialport.h>
#include <qextserialenumerator.h>
#include "mywindow.h"
#include "serialarduino.h"

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    mywindow fenetre;
    fenetre.MyWindowBloc();
    fenetre.AffichageMenu();
    fenetre.AffichageShow();
    return app.exec(); }
```

Liste des méthodes de mywindow

`mywindow()`	Constructeur
`mywindow(int x, int y)`	Constructeur surchargé
`void MyWindowBloc();`	Code de l'interface
`void AffichageShow();`	Affiche en plein écran l'interface
`void AffichageMenu();`	Code du menu
`void MyConnectInterface();`	Liste de «connect » permettant de réalisé des actions lors d'événement
`void NewScanRaz(bool effect);`	Efface et relance tout
`void setAnswer(bool effect);`	Modifie la valeur indiquant s'il y a une réponse
`bool getAnswer() const;`	Renvoie un booléen indiquant si il y a une réponse
`void CommandHelp();`	Affiche dans la console tous les commandes

Les Slots

`void AnswerDialogueCommande(QString reponse);`	Affiche la réponse du module P en passant par des paramètres
void `AnswerDialogueCommandeArduino`();	Affiche la réponse en récupérant la réponse par un accesseur
void dialogboxmessagestop();	Arrête le scan
`void dialogboxmessagestart();`	Lance le scan
`void dialogboxquestion();`	Boite de dialogue pour vérification de relance
`void dialogboxnewscan();`	Boite de dialogue pour vérification de nouveau scan
`void AddDialogueCommande();`	Envoie à arduino et gère la commande « help »
`void ConnectInPort();`	Connecte mywindow à serialarduino

Serialarduino gère toute la communication série émulé avec le module P et utilise la librairie qextserialport

La liste des méthodes de serialarduino

`serialarduino();`	Constructeur
`void connected();`	Fenetre de gestion de la connexion
bool `protocol`(QString cmd);	Vérifie la commande
void `ConfigConnect`();	Configure la connexion
`QByteArray getArray();`	Accesseur pour la réponse

`void modificationPort();`	(Non fonctionnel) Modifie le port de lecture

Slots et Signal

`void Connection(QString cmd);`	Gère l'envoie des commandes à l'arduino
`void readData();`	Slot qui génère un signal pour mywindow
`void readOk();`	Signal

ANNEXE :

Programme module D1 :

```
#include <Servo.h>
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>

Servo monServo;
int pos=0;
const int locker = 12;
const int clock = 10;
const int data = 11;
int binled = 0b00000000;
int comget = 0; // stock la commande reçu
```

Programme module D2 :

```
void setup()
{
    monServo.attach(2, 1000, 2000); // moteur
    pinMode(locker, OUTPUT); // pin configuré en sortie pour le 74HC595
    pinMode(clock, OUTPUT);
    pinMode(data, OUTPUT);

    Serial.begin(9600);

    /*** nRF24L01+ ***/
    Mirf.cePin = 8; // CE sur D8
    Mirf.csnPin = 7; // CSN sur D7
    Mirf.spi = &MirfHardwareSpi; // on utilise la surcouche de SPI
    Mirf.init();

    Mirf.channel = 42; // canal 42 car c'est le bien (aucune raison) ...
    Mirf.payload = 2; // taille du message à transmettre (2 octets)
    Mirf.config();

    Mirf.setTADDR((byte *)"drone"); // Le 1er module va envoyer ses info au
2eme module
    Mirf.setRADDR((byte *)"pcard"); // On définit ici l'adresse du 1er
module
   }
```

Programme module D3 :

```
void loop()
{
  digitalWrite(locker, LOW);
  binled = change_led(data, clock, true, 1, binled); // on allume la led de
fonctionnement
  digitalWrite(locker, HIGH);

  if(pos == 0)
  {
      for(pos = 0; pos<91;pos++)
      {
        monServo.write(pos);
        send_data(pos);

        if(Mirf.dataReady()){ // On vérifie si on a quelque chose de
nouveau
            Mirf.getData((byte *) &comget);
            digitalWrite(locker, LOW); // on allume la led de reception
            binled = change_led(data, clock, true, 3, binled); // on envoie
la nouvelle commande
            digitalWrite(locker, HIGH);
        }

        delay(10);

        digitalWrite(locker, LOW); // on éteint les LEDs de réception et
d'envoie
        binled = change_led(data, clock, false, 3, binled); // on envoie la
nouvelle commande
        digitalWrite(locker, HIGH);
        digitalWrite(locker, LOW); // on met le cadenas de la 74GC595 sur 0
pour modifier les led allumé
        binled = change_led(data, clock, false, 2, binled); // on envoie la
nouvelle commande
        digitalWrite(locker, HIGH); // on rebloque
      }

  }
  else
  {
   for(pos = 90; pos>=0;pos--)
      {
        monServo.write(pos);
        send_data(pos);

        if(Mirf.dataReady()){ // On vérifie si on a quelque chose de
nouveau
            Mirf.getData((byte *) &comget);
            digitalWrite(locker, LOW); // on allume la LED de réception
            binled = change_led(data, clock, true, 3, binled); // on envoie
la nouvelle commande
            digitalWrite(locker, HIGH);
        }
```

```
delay(10);

        digitalWrite(locker, LOW); // on éteint les LEDs de réception et
d'envoie
        binled = change_led(data, clock, false, 3, binled); // on envoie la
nouvelle commande
        digitalWrite(locker, HIGH);
        digitalWrite(locker, LOW); // on met le cadenas de la 74HC595 sur 0
pour modifier les LEDs allumé
        binled = change_led(data, clock, false, 2, binled); // on envoie la
nouvelle commande
        digitalWrite(locker, HIGH); // on bloque le cadenas (locker)
       }
      pos=0;
  }

}
```

Programme module D4 :

```
void send_data(int pos)
{
  Mirf.send((byte *)&pos); // On envoi la valeur de la position
  while(Mirf.isSending()); // On boucle (attend) tant que le message n'as
pas était envoyé
  digitalWrite(locker, LOW); // on met le cadenas de la 74HC595 sur 0 pour
modifier les led allumé
  binled = change_led(data, clock, true, 2, binled); // on envoie la
nouvelle commande
  digitalWrite(locker, HIGH); // on rebloque
}
```

Programme module D5:

```
int change_led(int dataPin, int clockPin, boolean ope, char data, int
data_comp) // gestion des led et du laser
{

  /*** operateur utilisé : OR pour l'addition et NAND pour soustraction
(principe d'un masque) ***/
  if(data == 1) // led fonctionnement
  {
    if(ope == true)
    {
    data_comp = data_comp | 0b00000010;
    }
    if(ope == false)
    {
    data_comp = data_comp & ~0b00000010;
    }
  }
  if(data == 2) // Led envoi
  {
    if(ope==true)
    {
```

```
        data_comp = data_comp | ~0b00000100;
      }
      if(ope==false)
      {
        data_comp = data_comp & ~0b00000100;
      }

    }
    if(data == 3) // Led reception
    {
      if(ope==true)
      {
        data_comp = data_comp | 0b00001000;
      }
      if(ope==false)
      {
        data_comp = data_comp & 0b00001000;
      }
    }
    if(data == 4) // Led moteur
    {
      if(ope==true)
      {
        data_comp = data_comp | 0b00010000;
      }
      if(ope==false)
      {
        data_comp = data_comp & 0b00010000;
      }
    }
    if(data == 5) // Laser
    {
      if(ope==true)

        data_comp = data_comp | 0b01000000;
      }
      if(ope==false)
      {
        data_comp = data_comp & 0b01000000;
      }
    }
     //octet inversée avec '~' pour piloter les LED à l'état bas (arduino
n'aime pas donné du courant ^^)
     shiftOut(dataPin, clockPin, LSBFIRST, ~data_comp );
     return data_comp;
}
```

Programme module P :

```
#include <Servo.h>
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>

void setup()
{
    Serial.begin(19200); // correspond au début de la communication
    /*** nRF24L01+ ***/
    Mirf.cePin = 8; // CE sur D8
    Mirf.csnPin = 7; // CSN sur D7
    Mirf.spi = &MirfHardwareSpi; // on utilise la surcouche de SPI
    Mirf.init();

    Mirf.channel = 42; // canal 42 car c'est le bien (aucune raison) ...
    Mirf.payload = 2; // taille du message à transmettre (2 octects)
    Mirf.config();

    Mirf.setTADDR((byte *)"pcard"); // Le 1er module va envoyer ses info au
2eme module
    Mirf.setRADDR((byte *)"drone"); // On définit ici l'adresse du 1er
module
}

void loop()
{
    int cardispo = 0;
    int cmd = 0;
    char comp[3];
    int i = 0;

          while(i<8)
          {
            cardispo = Serial.available();
            if(Mirf.dataReady())
            { // On vérifie si on a quelque chose de nouveau
                  Mirf.getData((byte *) &cmd); // on récupère ce que le
l'arduino mini nous a envoyé
                  Serial.println(cmd);
            }

             if(cardispo > 0)
            {
              comp[i] = Serial.read();
              i++;
            }

          }
    long total = (comp[0]-48)*1000+(comp[1]-48)*100+(comp[2]-
48)*10+(comp[3]-48); //Problème avec les caractères ASCII donc -48
    Mirf.send((byte *)&total); // la commande
    while(Mirf.isSending()); // On boucle (attend) tant que le message n'a
pas était envoyé

//fin du programme
}
```

Synthèse personnelle

I) Projet de Terminale

En début d'année, nous avons procédé à l'élaboration des groupes. Nous étions 4 avec Carine ARAUJO, Pascal PHELIPOT, Joan ROÏG et moi-même. Nous avons quelques idées mais elle ce sont vite montrée irréalisable

Suite à de longues journées de réflexion par rapport au sujet que nous allions choisir pour cette année nous avons finalement choisit le drone avec scanner 3D (EyeGle). Le travail nécessaire pour l'élaboration de ce projet est un des points qui m'a séduit. De plus son côté « original » et « nouvelle technologie » ont grandement contribué à ce choix.

II) Séances de Projet

Les séances de projet au sein du lycée permettent un « débriefing » de l'équipe, ou nous abordons les avancées réalisées, les problèmes rencontrés, ainsi que les possibles solutions. Elles permettent aussi une avancé sur des points techniques (programmation, conception du circuit). La plus grosse partie du travail qui m'était attribué a était réalisé en dehors des heures de projet sur mon temps personnel avec mon propre matériel.

III) Apports personnels.

Ce projet m'a apporté de l'autonomie, du fait de la réalisation entière d'un projet en groupe, sans aide des professeurs, du moins dans l'organisation du travail et de l'avancé. Ça m'a permis aussi d'appliqué des connaissances que je n'avais jamais appliqué. Ce projet a était une expérience très enrichissante, sur le domaine des connaissances, mais aussi dans le domaine du travail en groupe qui n'est pas simple.