Eyegle

Scanner 3D sur Drone.

Projet de Terminale S SI

2013-2014

ARAUJO Carine
GUILLAMOT Vincent
PHELIPOT Pascal
ROIG Joan

Sommaire.

- Présentation
  - Documents techniques.
  - Notre projet.
- Contraintes et Solutions
  - Mécaniques.
  - Electroniques.
  - Informatiques.
  - Législation.

Présentation et problématique

Comment réaliser une cartographie de notre monde de manière automatique ?

Présentation

- Notre projet:
  - Un dispositif de **triangulation** via un **laser** et une **caméra** embarqué sur un drone.
  - Une **représentation 3D** de l'environnement du drone.

- Objectifs :
  - **Eviter les risques** inutiles aux hommes.
  - **Explorer des environnements** instables et humainement dangereux.

- Exemple:
  - Investigation d'un bâtiment **hostile** aux hommes (fumée, produits toxiques, risque électrique) dans le but de **préparer une opération** nécessitant des ressources **humaines**.

Les détails de notre projet.

- **Le programme de cartographie 3D:**
  - Assurer la communication entre le drone et l'ordinateur.
  - Gérer les calculs et la récupération des coordonnées des points
  - Mettre en place un système de rendu en 3D des points.

- **Le module de scanner :**
  - Capturer des données par le biais d'un système caméra/laser.
  - Renvoyer les données à un ordinateur.
  - Contrôler le laser avec une précision angulaire.

- **Le drone :**
  - Adapter le drone pour le système embarqué.
  - Créer plusieurs modules pour accorder le drone à son environnement.

Fonctionnement du système

Module Drone

- Compositions :
  - Servomoteur
  - 74HC595
  - nRF24L01+ (Transceiver)
- Fonctions :
  - Rotation du servomoteur
  - Commande des LEDs et du laser par le 74HC595
  - Communication

Le programme du module Drone

- Initialisation de diverses variables devant être réinitialisé à chaque boucle.
- Initialisation de variable lié au fonctionnement du servomoteur, du 74HC595 et du transceiver nRF24L01+
- Programme principal
- Fonction send_data()
- Fonction change_led()

Les librairies incluse

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

```
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

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 octects)
    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
}
```

Le programme principal | Algorithme

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 LEDs de réception

On l'éteint

On éteint la LEDs allumé lors de l'envoie de l'angle

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 réception

On l'éteint

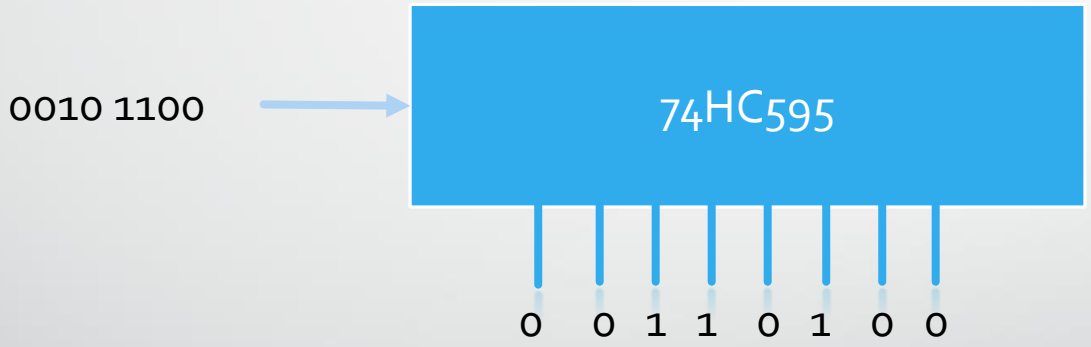
On remet la valeur exacte de 0 à la variable de position

Fonction send_data()

```
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 74GC595 sur 0 pour modifier les LEDs allumé
    binled = change_led(data, clock, true, 2, binled); // on envoie la nouvelle commande
    digitalWrite(locker, HIGH); // on rebloque
}
```

Fonction change_led()

Fonctionnement du 74HC595



Fonction change_led()

Addition

OR 0000 1001
0000 0010

0000 1011

Soustraction

NAND 0000 1011
0000 0010

0000 1001

Fonction change_led()

int change_led(int dataPin, int clockPin, boolean ope, char data, int data_comp)

Int dataPin	Broche « data » de la puce 74HC595
Int clockPin	Broche « clock » de la puce 74HC595
Boolean ope	Sélection de l'opération à effectuer
char data	Sélection de la LED à allumer
Int data_comp	Code binaire

Module Ordinateur

Le programme du module Ordinateur

Boucle tant qu'on a pas 8 bit d'un message provenant d'un message de l'ordinateur

on vérifie la présence d'un message de l'ordinateur

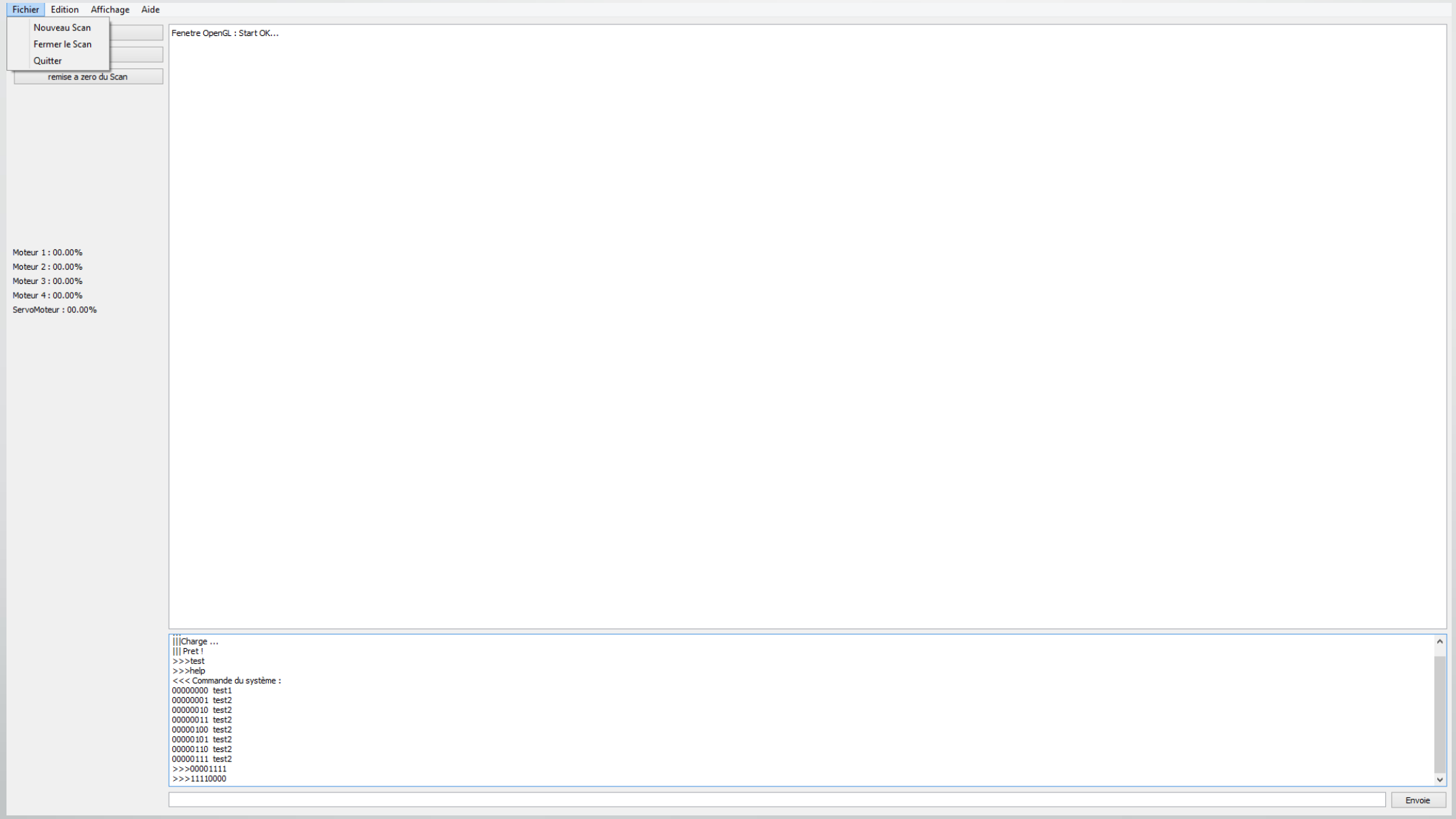
on récupère le message du module Drone

Si on a un message de l'ordinateur on enregistre chaque bit dans un tableau

On recompose grâce à des puissance de 10 la commande

On transmet le message au module Drone

L'interface


Fichier
Edition
Affichage
Aide
Nouveau Scan
Fermer le Scan
Quitter
remise a zero du Scan
Moteur 1 : 00.00%
Moteur 2 : 00.00%
Moteur 3 : 00.00%
Moteur 4 : 00.00%
ServoMoteur : 00.00%
Fenetre OpenGL : Start OK...
|||Charge ...
||| Pret !
>>>test
>>>help
<<< Commande du système :
00000000 test1
00000001 test2
00000010 test2
00000011 test2
00000100 test2
00000101 test2
00000110 test2
00000111 test2
>>>00001111
>>>11110000
Envoie

Détails techniques

- Programmation orienté objet
- Créer à l'aide de la bibliothèque logicielle multiplateforme Qt
- Communication série par le biais de QExtSerialPort

Fichier	Fonction
main.cpp	Code principale
mywindow.h	Définition de la classe mywindow
mywindow.cpp	Classe mywindow gérant l'interface
serialarduino.h	Définition de la classe serialarduino
serialarduino.cpp	Classe mywindow gérant la communication série

FIN