Tristan de Fournas
Benjamin Lacroix
Tristan Leuillier
Hugo Pastre

Projet système embarqué :
MiniWorld - Communication RF avec les véhicules.

1. Initialisation / installation de la Beaglebone[1]

Le projet commence par l'installation de debian sur la carte SD qu'utilisera la Beaglebone pour démarrer.
→ http://www.nathael.net/wiki_cpe contient toutes les informations nécessaires à cette installation.

Après avoir mis à jour de nombreux paquets tels que le compilateur C des processeurs ARM sur la clé USB, on crée des partitions sur la carte SD, l'une contenant une petite initialisation permettant de charger la seconde plus complète.

On utilise minicom (à installer) et un port série afin de communiquer avec la Beaglebone. Il faut enfin connecter au réseau la Beaglebone pour qu'elle soit accessible par plusieurs PC et non seulement celui connecté au port série.

Le projet comporte trois étapes :

- Espion

Il est utilisé pour détecter tous les messages RF[2] qui sont envoyés par la carte et le camion, sans faire de traitement. Il est également utilisé pour le développement.

- Carte_RF (≈ communication série http://fr.wikipedia.org/wiki/Communication_série)

La structure des paquets envoyés par la carte RF est la suivante:

```
struct tx_packet {
  uint8_t length;    /* (1 octet, = N + 2) La taille totale du paquet envoyé*/
  uint8_t addr low;   /* (un octet) L'adresse basse du véhicule*/
  uint8_t addr high;   /* (un octet) L'adresse haute du véhicule afin d'avoir plus de véhicule*/
  uint8_t[] data;   /* (N octets) Les données ou commandes à envoyer */
 }
```

La structure des paquets envoyés par le camion (réceptionné par la carte RF) :

```
struct rx_packet {
  uint8_t length;   /* Taille du paquet - 1 : toujours 15 */
  uint8_t bcast;    /* Toujours 0xFF */
  uint16_t addr;     /* Big endian */
  uint8_t status;      /* Etat des leds du Véhicule , 8 bits */
  uint8_t anticolision;      /* Statut de l'anti-collision*/
  uint8_t speed_target;    /* vitesse cible */
  uint8_t motor_speed;   /* vitesse actuelle */
  uint8_t Vbat;     /* tension batterie */
```

[1] Beaglebone : Micro ordinateur de la taille d'une carte bleu (équivalent raspberry Pi pour les connaisseurs)
[2] RF : Radio fréquence

```
  uint8_t magnets_to_action;    /* nombre d'aimants avant la prochaine instruction*/
  uint16_t nb_magnets;    /* nombre d'aimants comptés*/
  uint32_t message_num;   /* Numéro du dernier message reçu */
} __attribute__ ((__packed__));
```

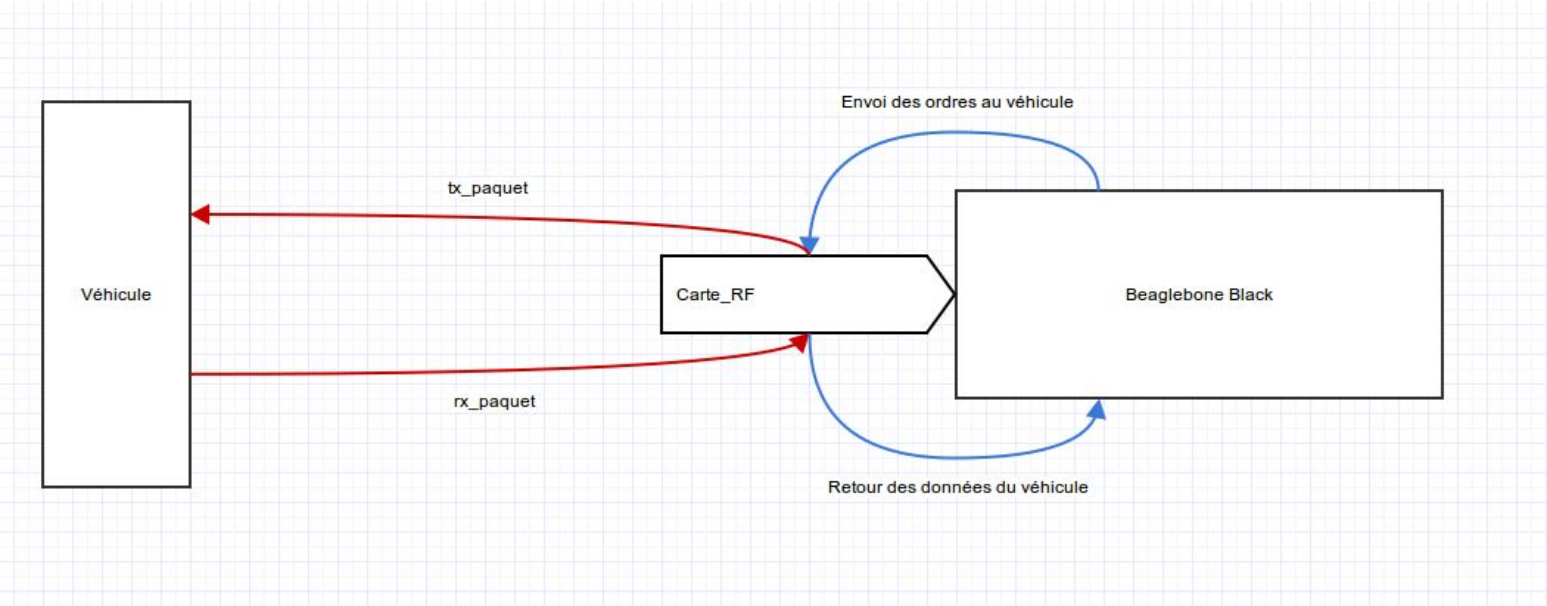

Schéma d'échange de données

Dans le sens ordinateur → véhicule, la carte RF lis les données en entrée du port USB, utilisé comme port série, et les transforme en onde RF. Pour le sens véhicule → ordinateur, la carte RF transforme les ondes en données physique et les écrits dans l'USB.

- Application de communication

Exécutable qui est lancé sur le Beaglebone.
La Beaglebone utilise le port USB comme port série UART.

De façon algorithmique simplifiée:

```
while (1) {
        data = lire_clavier;
        msg = creer_msg(taille(data),addresse_véhicule,data);

        ecrire_msg_UART(msg);

        /* On attend une réponse du véhicule car le véhicule renvoi toujours sont état en réponse à une commande */

        tant_que ( estVide( entree_UART )) {}
        dataVehicule =recuperer(entree_UART);
        afficherData( dataVehicule)
}
```

2. Utilisation

Tous les fichiers nécessaires au bon fonctionnement de ce projet sont dans le dossiers “sources” livrés avec le document ici présent.
Il y a donc :

1. carte_rf_bbb.c → firmware de la carte RF. A compiler avec le makefile[3]
2. espion_rf.c -> Firmware de la seconde carte RF. A compiler avec le Makefile.
3. appli_bbb_communication.c -> Utiliser le compilateur arm-linux-gnueabihf-gcc-5 pour une utilisation avec un processeur ARM
4. .makefile pour créer les exécutables correspondants (via la commande make).

Afin d’utiliser le maximum de puissance et donc avoir des compilations plus rapides, la compilation est faite en mode “cross-compilation[4]”. Tous les codes sont donc compilés sur l’ordinateur avant d’être copiés au bon endroit.
Pour flasher les cartes, il faut utiliser lpctools

⇒ lpcprog -d /dev/ttyXXX -c flash _fichier_

La carte RF doit être en mode ISP pour être flashée. Pour ce faire, il faut appuyer sur les deux boutons de la carte en même temps. Le flash de la carte RF de la Beaglebone peut être flashé à partir du PC.
La carte RF espion est positionnée sur le PC. On écoute les communications de la carte espion grâce à minicom.
La carte RF de la Beaglebone n’a pas besoin de minicom[5] car ses données sont utilisées par l’appli *appli_bbb_communication.c.*

[3] Makefile : fichier appelé pour la compilation, regroupe toutes les étapes de compilations et permet un seul appel : “make”.
[4] Cross-Compiler : compilateur capable de générer un exécutable pour plusieurs plateformes.
[5] Minicom : programme de contrôle de modem et d'émulation de terminal pour les Unix-like