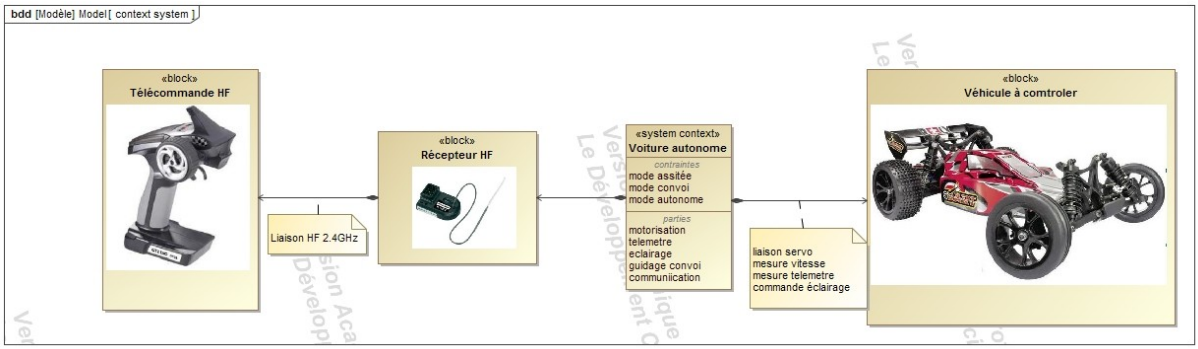
VOITURE AUTONOME - CARTE TÉLÉMÈTRE : DOSSIER FABRICATION

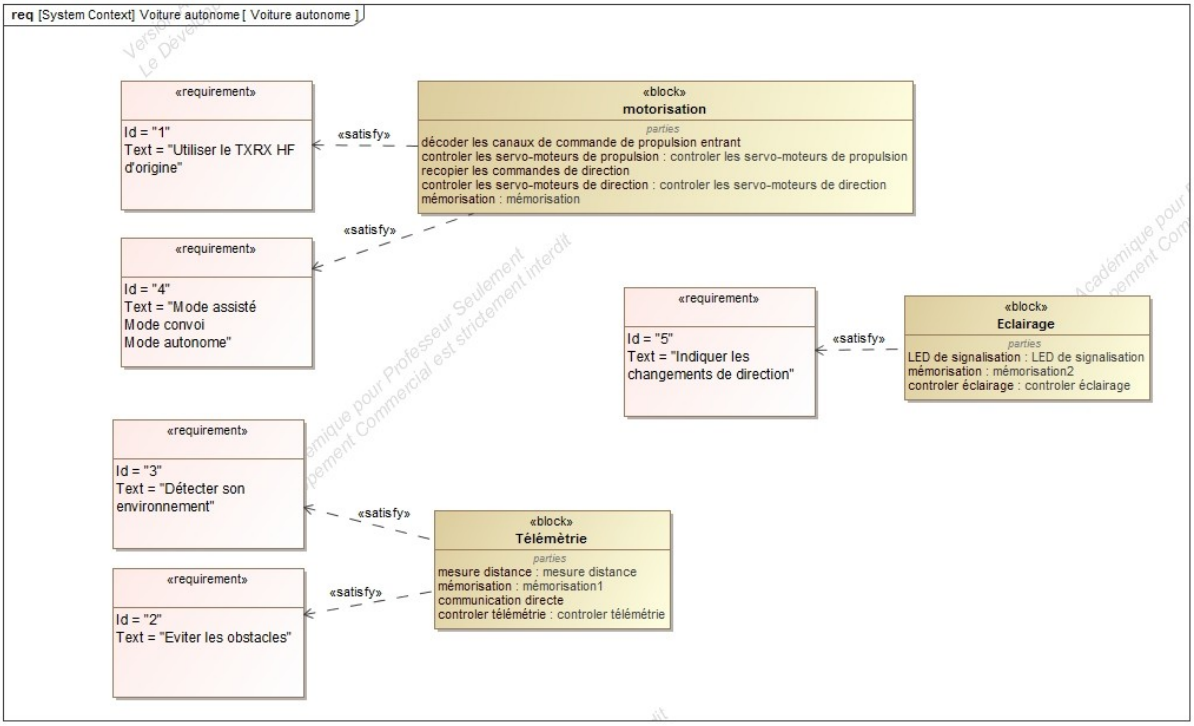
1. . Cahier des charges de l'objet technique

2. . Étude SYSML

2.1 Diagramme de contexte

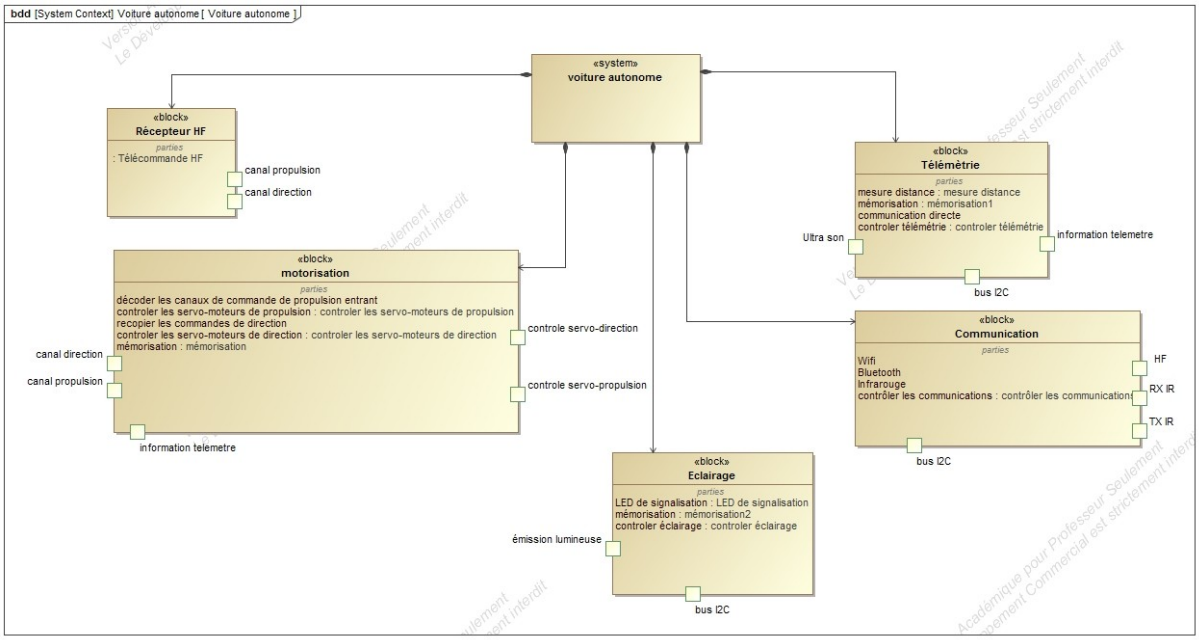


2.2 Diagramme d'exigences



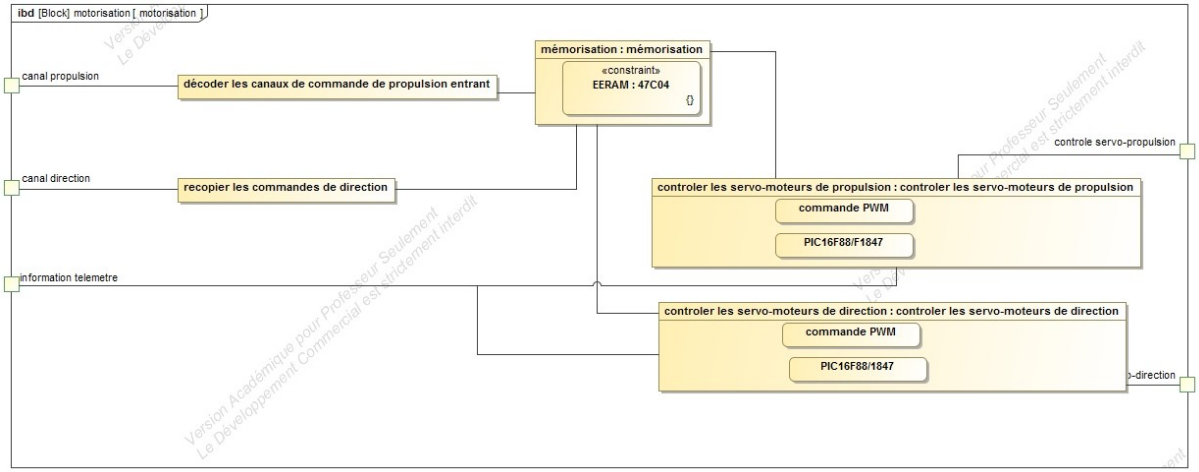
2.3 Diagramme de cas d'utilisation

2.4 BDD

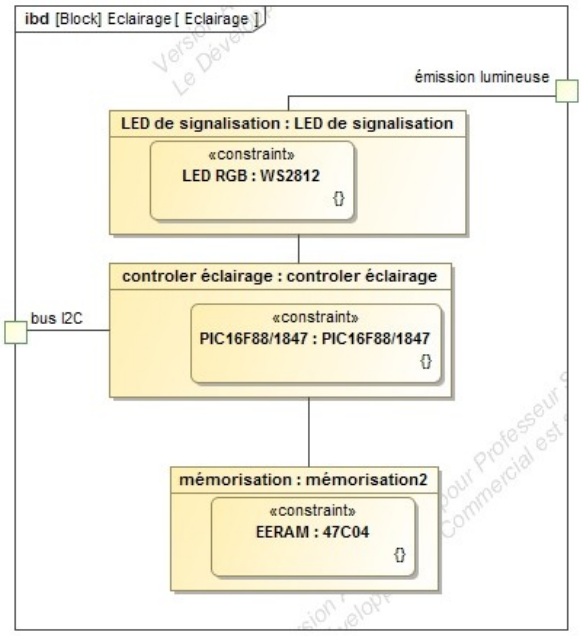


2.5 IBD

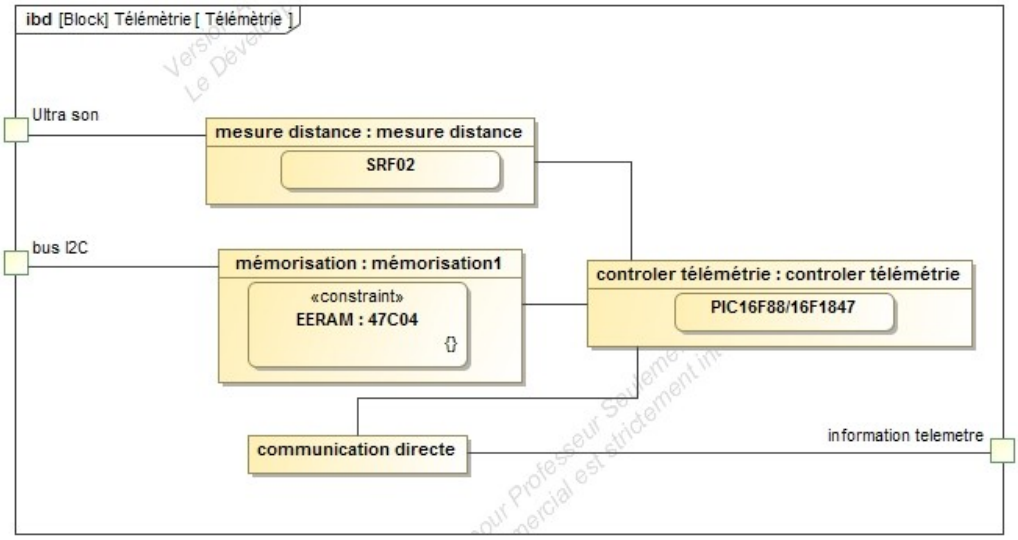
2.5.1 Motorisation



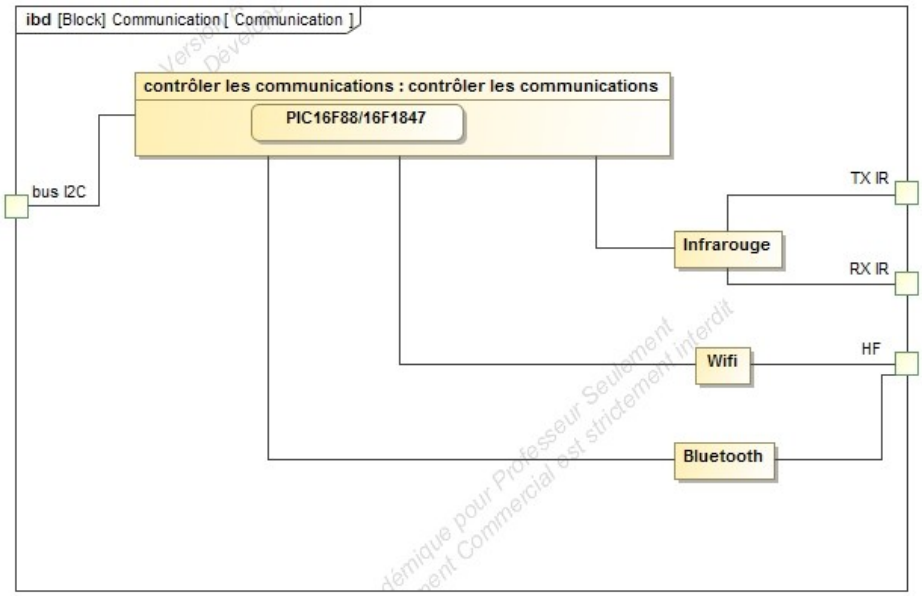
2.5.2 Eclairage



2.5.3 Télémétrie



2.5.4 Communication



2.6

3. . Étude Structurelle

3.1 Études théoriques : détails, justifications des calculs et des choix de composants

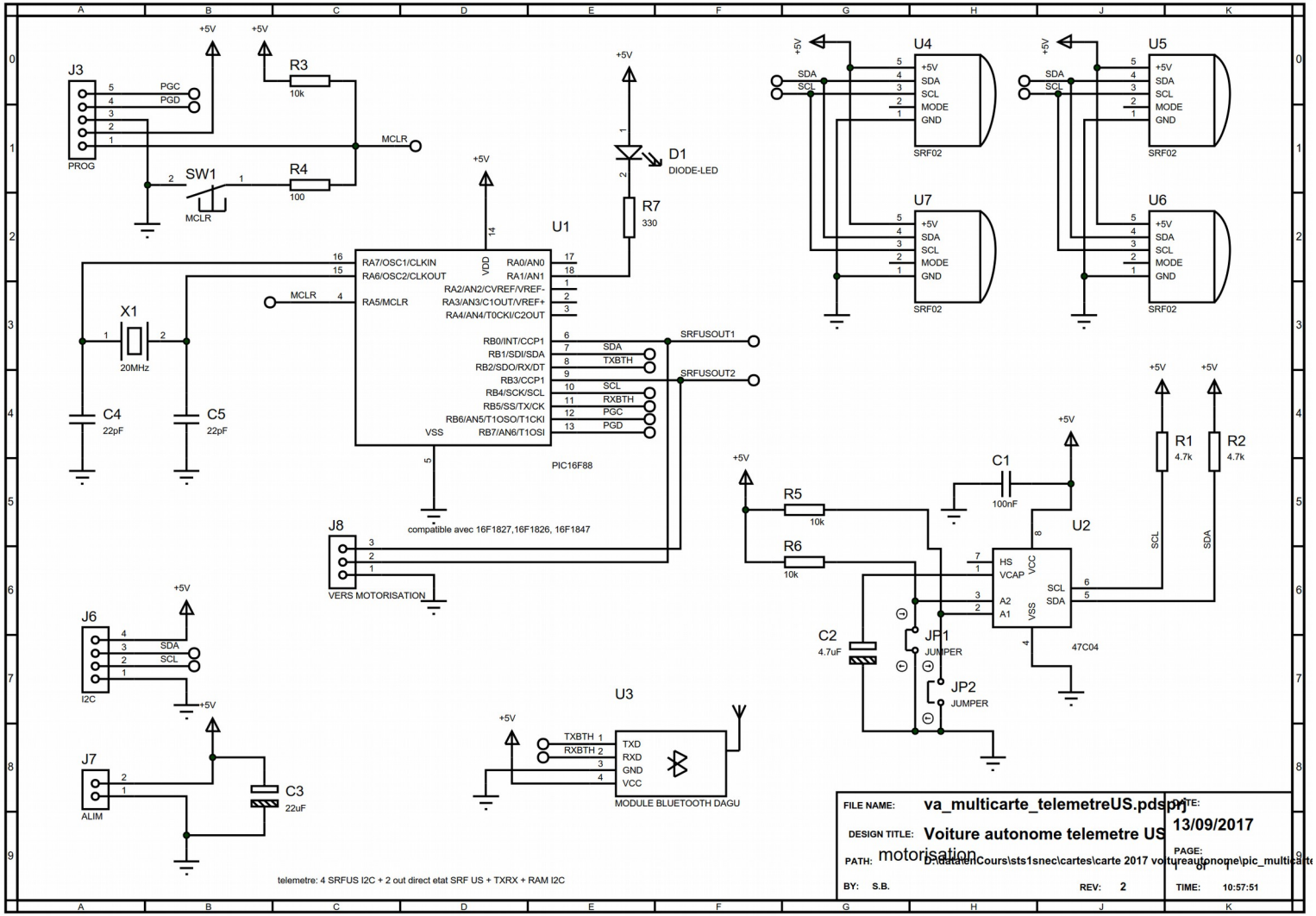
A FAIRE

3.2 Validation de l'étude théorique : documents de simulation et de test

A FAIRE

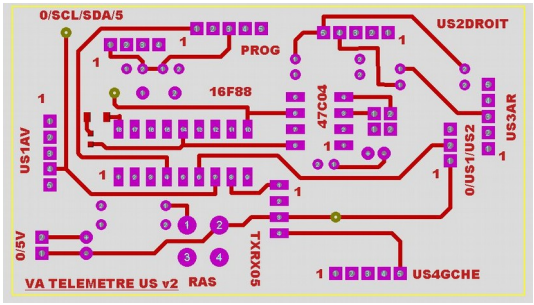
4. . Documents de fabrication

4.1 Schéma structurel.

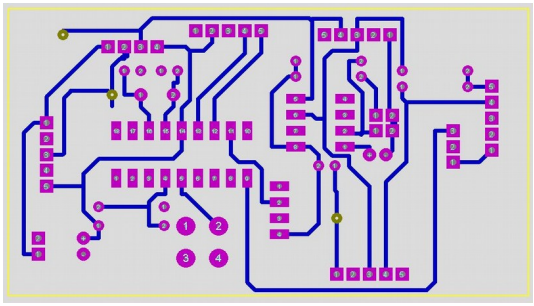


4.2 Typons (coté cuivre et coté composants)

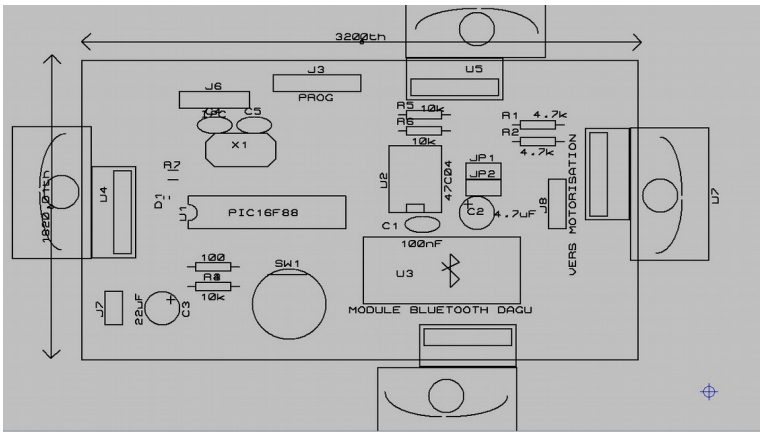
Coté cuivre (BOTTOM)



Coté composant (TOP)



4.3 Plan d'implantation et de perçage



4.4 Nomenclature des composants (A COMPLETER)

Catégorie	Références	Valeur	Stock Code	Unit Cost
Capacitors	C1	100nF		
Capacitors	C2	4.7uF		
Capacitors	C3	22uF		
Capacitors	C4	22pF		
Capacitors	C5	22pF		
Resistors	R1	4.7k		
Resistors	R2	4.7k		
Resistors	R3	10k		
Resistors	R4	100		
Resistors	R5	10k		
Resistors	R6	10k		
Resistors	R7	330		
Integrated Circuits	U1	PIC16F88		
Integrated Circuits	U2	47C04		
Integrated Circuits	U3	MODULE BLUETOOTH DAGU		
Integrated Circuits	U4	SRF02		
Integrated Circuits	U5	SRF02		
Integrated Circuits	U6	SRF02		
Integrated Circuits	U7	SRF02		
Diodes	D1	DIODE-LED		
Miscellaneous	J3	PROG		
Miscellaneous	J6	I2C		
Miscellaneous	J7	ALIM		
Miscellaneous	J8	VERS MOTORISATION		
Miscellaneous	JP1	JUMPER		
Miscellaneous	JP2	JUMPER		

Miscellaneous	SW1	MCLR		
Miscellaneous	X1	20MHz		

5. . Etude de mise en conformité

5.1 Les protocoles de test détaillés

A FAIRE

5.2 Les chronogrammes et valeurs de mesurage obtenus

A FAIRE

6. . Programmes de test (A faire)

6.1 Mise en œuvre générale des tests

Une carte à tester (va_motorisation v??) avec 16F88

un module SRF02

une EERAM 47C04

un pickit3 (ou 2)

un cable USB/TTL branché sur RXBTH et la masse

une alim externe +5V + un adaptateur pour branchement sur carte à tester.

Chaque partie testée comprendra :

6.1.1 protocole de test

6.1.2 Code C

6.1.3 Résultat du test

6.1.4 Remarque sur le déroulement du test

7. Détail du coût

Le coût totale est calculé en incluant :

le matériel

le temps de test

le temps de fabrication

le coût horaire inclura toutes les charges : salaire + patronnale + fiscale + machine (voir avec le prof d'éco-gestion)

8. . Correction des protocoles de test hard

8.1 Tests statiques

Voir document test de carte.

Voir le TP voiture RC.

8.2 Tests dynamiques

8.2.1 Test de la communication UART

Créer un programme envoyant un mot vers le TXPIC et visualiser le résultat à l'aide d'un analyseur logique en mode décodeur UART ainsi qu'un cable USB/TTL adapté.

8.2.2 Test de la mesure au télémètre US

Lire la documentation du SRF02.

Utiliser le programme de configuration du SRF02 afin de fixer une adresse I2C.

On programmera un adresse cohérente avec son numéro de table (exemple : table B0 = $E0, table B1 = $E2, table B2 = $E4, …)

Utiliser le driver fournit par l'enseignant (stssnsb.free.fr/telecharger/SN1/microntroleur) pour tester le bon fonctionnement du télémètre. Les valeurs seront visualisées par l'analyseur logique en mode I2C et par une liaison USB/TTL avec le PC.

8.2.3 Test de la mémorisation 47C04

Créer un programme écrivant dans la EERAM (utiliser comme exemple le programme EEPROM I2C)

Créer un programme lisant les valeur écrites en EERAM. Utiliser la liaison UART pour faciliter le débuggage. (cable USB/TTL)

Visualiser les échanges à l'aide de l'analyseur logique en mode I2C.

9. . Correction programmes de test

9.1 Test du télémètre

```
/*
test des télémetres : 4 ok
test du port série : TX : ok, RX : ?
test BP reset : ok
test envoi SRFOUT1 SRFOUT2 : OK OK
test RAM : ecriture : OK lecture OK

*/


#include <16F88.h>

#device adc=8

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES HS                       //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES NOPUT                    //No Power Up Timer
#FUSES MCLR                     //Master Clear pin enabled
#FUSES NOBROWNOUT               //No brownout reset
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOCPD                    //No EE protection
#FUSES NOWRT                    //Program memory not write protected
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOPROTECT                //Code not protected from reading
#FUSES FCMEN                    //Fail-safe clock monitor enabled
#FUSES IESO                     //Internal External Switch Over mode enabled

#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_B5,rcv=PIN_B2,bits=8,errors)
#use i2c(Master,Fast,sda=PIN_B1,scl=PIN_B4)

#include "SRF02 driver_telemetreUS_I2C.c"

#define SRFUS1 PIN_B0
```

```
#define SRFUS2 PIN_B3

#define SRFUS_AVANT_ADR 0xE4

#define ADRSRAM_read 0xAC;
#define ADRSRAM_write 0xAD;
#define ADRUS1 0x00;
#define ADRUS2 0x02;
#define ADRUS1 0x04;
#define ADRUS2 0x06;

int16 valSRFUSAvant=0;


#int_RDA
void  RDA_isr(void)
{

}


void RAMwrite(int16 adrData,int8 ldata){
//sauve les mesures dans la RAM : OK  testé
/*
entrées : mesure des 4 telemetres
sorties : aucune

ADR RAMI2C = 1010 110X avec X = 1 si READ et 0 si WRITE

mode WRITE un octet : ADRHIGH(8bits) ADRLOW(8bits) DATA(8bits) STOP
mode WRITE x octet : ADRHIGH(8bits) ADRLOW(8bits) DATA0(8bits) DATA1.x fois...STOP
*/


//printf("\r\nEcriture RAM :");
i2c_start();
i2c_write(0xAC);     // Device address
i2c_write(adrData/256);i2c_write(adrDAta%256);  // Data to device 2 octets
i2c_write(ldata);     //  data
i2c_stop();
```

```
}//

int8 RAMread(int16 adrData){
//lit les mesures dans la RAM :  OK testé
/*
entrées : mesure des 4 telemetres
sorties : aucune

ADR RAMI2C = 1010 110X avec X = 1 si READ et 0 si WRITE
*/
int8 ldata;

i2c_start();
i2c_write(0xAC);     // Device address
i2c_write(adrData/256);i2c_write(adrData%256);  // Data to device 2 octets
i2c_start();
i2c_write(0xAD);
ldata=i2c_read(0);    // Now read from slave
i2c_stop();
return ldata;

}//RAMread


void main()
{

int8 data;

   port_b_pullups(TRUE);
   setup_adc_ports(NO_ANALOGS|VSS_VDD);
   setup_adc(ADC_OFF);
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   set_tris_a(0b11000100);
   set_tris_b(0b11111111);
//   enable_interrupts(INT_RDA);
//   enable_interrupts(GLOBAL);
```

```
  printf("debut telemetre");

  // TODO: USER CODE!!
while(1){
  printf(".");
  valSRFUSAvant=TelemetreUS_mesure(SRFUS_AVANT_ADR);//ok
  printf("\r\nvalSRFSAvant = %lu\r\n",valSRFUSAvant);//ok
  data=valSRFUSAvant/256;printf("dataMSB=%u",data);RAMwrite(0,data);//delay_ms(1000);
  data=valSRFUSAvant%256;printf("dataLSB=%u",data);RAMwrite(1,data);//delay_ms(1000);
//   printf("valeur lue à 0: %u\r\n",RAMread(0));
//   printf("valeur lue à 1: %u\r\n",RAMread(1));
  //ecriture vers SRF1 et SRF2
  if (valSRFUSAvant < 25) //arrêt
  {
    output_high(SRFUS1);
    output_high(SRFUS2);
  }
  else if (valSRFUSAvant <50)//ralentir 50%
       {
         output_low(SRFUS1);
         output_high(SRFUS2);
       }
       else //rien
         {
           output_low(SRFUS1);
           output_low(SRFUS2);
         }

}//while
}
```

Résultat du test :

Ok

9.2 Test de la liaison série.

9.2.1 Protocole de test

Utilisation d'un module Bluetooth ou un cable USB/TTL

En ajoutant un "printf()" on envoie du texte sur la liaison série.

L'utilisation du module DAGU Bluetooth nécessite une appli Android développée avec AppInventor.

Le cable USB/TTL nécessite l'utilisation d'un terminal : terraterm ou esscom32...

9.2.2 Code C

```
printf("debut") ;
```

9.2.3 Résultat du test

OK

Attention l'envoi d'un texte dans la boucle principale perturbe le servomoteur qui nécessite un timer réglé sur 20ms afin de garder du couple.

9.3 Test de la mémoire RAM

La RAM permet de sauvegarder les données du véhicules :

direction et vitesse

distance des obstacles

réglage des éclairages…

L'écriture et la lecture sur la EERAM a été testé sur la carte "télémètre US" avec écritue et lecture de la mesure du télémetre.

9.3.1 Protocole de test

Ecrire un valeur en EERAM, afficher un info sur UART, attendre un peu,

Relire la valeur en EERAM et l'afficher sur l'UART.

La mémoire EERAM est à l'adresse :

1010 110X avec X = 1 si READ et 0 si WRITE

9.3.2 Code

```
void RAMwrite(int16 adrData,int8 ldata){
//sauve les mesures dans la RAM : OK  testé
/*
entrées : mesure des 4 telemetres
sorties : aucune

ADR RAMI2C = 1010 110X avec X = 1 si READ et 0 si WRITE

mode WRITE un octet : ADRHIGH(8bits) ADRLOW(8bits) DATA(8bits) STOP
mode WRITE x octet : ADRHIGH(8bits) ADRLOW(8bits) DATA0(8bits) DATA1.x fois...STOP
*/



//printf("\r\nEcriture RAM :");
i2c_start();
i2c_write(0xAC);     // Device address
i2c_write(adrData/256);i2c_write(adrDAta%256);  // Data to device 2 octets
i2c_write(ldata);     //  data
```

```
i2c_stop();

}//

int8 RAMread(int16 adrData){
//lit les mesures dans la RAM :  OK testé
/*
entrées : mesure des 4 telemetres
sorties : aucune

ADR RAMI2C = 1010 110X avec X = 1 si READ et 0 si WRITE
*/
int8 ldata;

i2c_start();
i2c_write(0xAC);     // Device address
i2c_write(adrData/256);i2c_write(adrData%256);  // Data to device 2 octets
i2c_start();
i2c_write(0xAD);
ldata=i2c_read(0);    // Now read from slave
i2c_stop();
return ldata;

}//RAMread



extrait du main(){

   valSRFUSAvant=TelemetreUS_mesure(SRFUS_AVANT_ADR);//ok
   printf("\r\nvalSRFSAvant = %lu\r\n",valSRFUSAvant);//ok
   data=valSRFUSAvant/256;printf("dataMSB=%u",data);RAMwrite(0,data);//delay_ms(1000);
   data=valSRFUSAvant%256;printf("dataLSB=%u",data);RAMwrite(1,data);//delay_ms(1000);
   printf("valeur lue à 0: %u\r\n",RAMread(0));
   printf("valeur lue à 1: %u\r\n",RAMread(1));

}//main()
```

9.3.3 Résultat du test

OK

Remarque : le fait de mettre un valeur #define ADRRAM pour l'adresse à la place de 0xAC (read) 0xAD (write) entraine une erreur à la compilation !!! à voir