Table des matières

Fiche pédagogique

Objectifs pédagogiques	2.1 Conception fonctionnelle d'un système local
Connaissances visées	• Conversion d'une information (CAN CNA)
Prérequis	•
Matériel	• Carte "domoweb" • programmateur AVRISP mkll
Logiciel	AVRStudio
Évaluation	
webographie	

1 Présentation

La carte domoweb, est équipée un capteur de température LM19 relié aux bornes J13, J21 et J15. et qui fournit une tension Vo appliquée à l'entrée PA1 du micro contrôleur ATMEGA 32 (U2).

PA1 est l'entrée d'une fonction **C**onversion **A**nalogique **N**umérique (**CAN**): à partir de cette tension, la fonction CAN fournit un nombre binaire qui dépend de la tension Vo

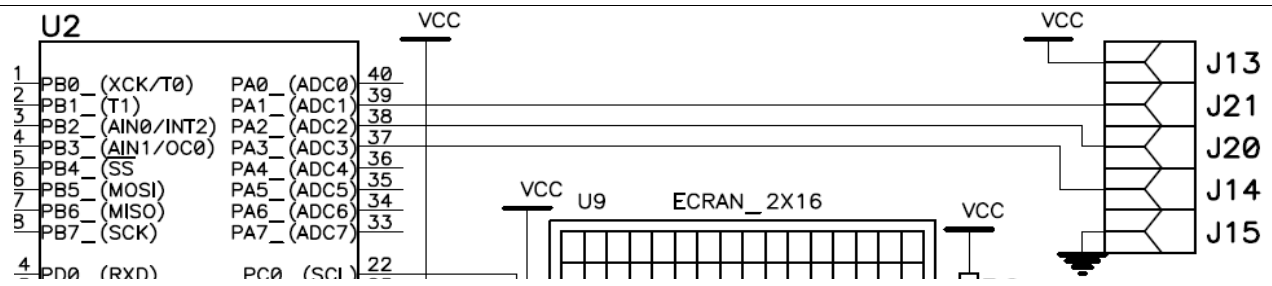


schéma structurel partiel de la carte domoweb

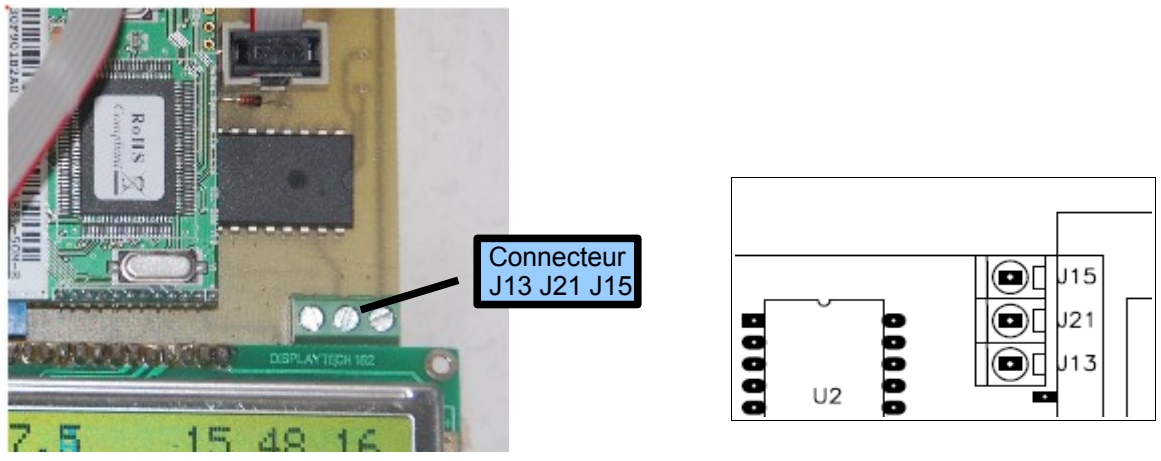


Photo et schéma d'implantation partiels de la carte domoweb

- Dessiner le capteur LM19 sur le schéma structurel et le schéma d'implantation
- Colorier sur le schéma structurel et le schéma, d'implantation la broche PA1 et la liaison entre cette broche et le capteur
- Préciser le le rôle des connecteurs:

J13	J21	J15

2 Programme can.c

2.1 Récupération d'un fichier existant:

Le programme **can.c** permet d'utiliser la fonction CAN du micro contrôleur ATMEGA 32. Vous allez récupérer ce programme dans votre projet **domoweb:**

- Ouvrir votre projet, domoweb (cliquer sur le fichier AVR.aps de **votre** dossier domoweb)
- Charger le fichier **can.c** situé dans **commun:\travail\@STI2D_SIN\ATMEL\domoweb** et sauvegarder ce fichier dans votre répertoire **ATMEL\domoweb** (utiliser File Open File... et File Save As...)
- Choisir le fichier **can.c** comme fichier source (voir Activité_ATMEGA_mise_en_oeuvre)
- Compiler, programmer le micro contrôleur(voir Activité_ATMEGA_mise_en_oeuvre) et noter vos observations.

2.2 Observations:

2.3 Interprétation:

- Rechercher dans la documentation technique du circuit ATMEGA32, (page 201) la résolution du CAN (c'est le nombre de chiffres binaires du résultat n en sortie du CAN):

- Calculer les valeurs maxi et mini de de n:

	en binaire	en décimal
n mini		
n maxi		

🗅 Rechercher (page 214) la tension de référence (on suppose que REFS1 = REFS0 =1)

V_{REF}=

🗅 Rechercher (page 213, single ended conversion) l'expression de n (nombre résultat de la conversion) en fonction de Vo (tension en entrée du CAN)

n =

🗅 Déduire l'expression de Vo en fonction de n.

***Vo* =**

🗅 En déduire la valeur actuelle, la valeur mini et la valeur maxi de Vo

	n (décimal)	Vo (en volts)
mini		
actuel		
maxi		

3 Analyse du programme source

3.1 can.c:

🗅 Repérer dans le programme can.c (page suivante) le segment qui s'effectue en boucle indéfiniment.

🗅 Quel est l'effet de `#include <lcd2x16.h>`

🗅 Indiquer les noms et le type des variables utilisée dans ce programme.

nom	type

🗅 Quel est l'effet de

```
CursLCD (0,0);
AffChaine ("n=");
```

combien de fois est effectué ce segment de programme?

🗅 Quel est l'effet de

```
CursLCD (0,5);
AffDec (n,4);
```

combien de fois est effectué ce segment de programme?

```
/*-----------------------------can.c-------------------------------------------------------
-------------------------------------------------------------------------------------------*/
#define F_CPU 8000000UL
#include <avr/io.h>
#include <lcd2x16.h>
#include <math.h>                       // pour la fonction racine crrée:  sqrt();

//variables globales
unsigned int n;                         // nombre en sortie du CAN
float Vo;                               // tension en entrée du CAN
float t;                                // température

int main(void)
{
  InitLCD ();
  ADMUX=0b11000001;
  /*
    ADMUX:      REFS1   REFS0   ADLAR   MUX4    MUX3    MUX2    MUX1    MUX0
                  1       1       0       0       0       0       0       1
                  |       |       |       |_______________________________|
                  |       |       |                   entrée ADC1 (PA1)
                  |_______|       Left Adjust Result (1:résultat ajusté à gauche; 0: à droite)
                    00: vref= AREF
                    01: Vref= AVCC
                    11: Vref= 2.56v (ref. interne)
  */

  ADCSRA=0b10000111;
  /*
  ADSCRA         ADEN    ADSC    ADATE   ADIF    ADIE    ADSP2   ADSP1   ADSP0
                   1       0       0       0       0       1       1       1
                   |       |       |       |       |       |_______________|
                   |       |       |       |       |   freq. XTAL divisée par 2(000);4(010)
                   |       |       |       |       |   8(011);16(100); 32(101);64(110);128(111)
                   |       |       |       |       |
                   |       |       |       |     Pas d'interruption
                   |       |       |     Drapeau de fin de conversion
                   |       |     Pas de declenchement automatique de conversion
                   |     Start conversion non actif
                 Enable
  */
  CursLCD (0,0);
  AffChaine ("n=");

  while (1)
    {
      ADCSRA |= 0b01000000;             // Début de conversion (ADSC <-- 1)
      while ((ADCSRA & 0b00010000)==0)  // boucle d'attente:tant que ADIF=0
        {
        }
      ADCSRA|=0b00010000;               // ADIF <-- 0

      n=ADC;
      CursLCD (0,5);
      AffReel (n,4,2);
    }
}
```

3.2 *Registres:*

L'utilisation de la fonction CAN fait intervenir des registres: ce sont des "emplacements" dans lequel le programme peut écrire ou lire des octets ou des doubles octets. Ces registres ont des noms (ADMUX, ADCSRA et ADC) qui sont définis dans la librairie **io.h**. Il est donc nécessaire d'inclure cette librairie avec `#include <avr/io.h>`. Les chiffres (bits) de ces registres ont souvent des noms. Par exemple, le bit de poids fort de ADMUX s'appelle: REFS1.

3.2.1 Registre ADMUX

Dans le programme, la ligne `ADMUX=0b11000001;` permet d'écrire le nombre binaire $(1100\ 0001)_2$ dans le registre ADMUX.

Remarquer que la base 2 est indiquée par le préfixe **0b** et que les quartets **ne sont pas séparer par un espace**

🗅 Rechercher dans la documentation technique du circuit ATMEGA32, (page 214) les noms des bits du registre ADMUX et compléter l'état de ces bit après que `ADMUX=0b11000001;` ait été effectuée:

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
nom								
état								

🗅 Rechercher (page 214) le rôle des bits REFS1 et REFS0 du registre ADMUX et en conclure.

- Rechercher (page 215) le rôle des bits MUX4 à MUX0 et en conclure.

3.2.2 Registre ADCSRA

- Rechercher dans la documentation technique du circuit ATMEGA32, (page 216) les noms des bits du registre ADCSRA et compléter l'état de ces bit après que la ligne `ADCSRA=0b10000111;` ait été effectuée:

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
nom								
état								

- Rechercher le rôle du bit ADEN et conclure

- Rechercher le rôle du bit ADSC et conclure

- Rechercher le rôle du bit ADIF

3.2.3 Registre ADC

- Rechercher (page 217) le rôle du registre ADC

- Quel est l'effet de l'instruction `n=ADC;` ?

3.2.4 Modification d'un seul bit d'un registre:

La ligne de programme: `ADCSRA |= 0b01000000;` permet de ne mettre à 1 que le bit6 de ADCSRA sans modifier les autres bits de ce registre.

Si nous avions utilisé ADCSRA= 0b01000000; cela aurait mis à 1 le bit 6 et à 0 tous les autres bits.

Pour mettre à 0 le bit6 de ADCSRA sans modifier les autres bit, il faudrait utiliser: ADCSRA &= 0b10111111;

- Indiquer l'effet de l'instruction `ADCSRA |= 0b01000000;` dans le programme (justifier)

3.2.5 Boucle conditionnelle

Le segment de programme:

```
while ((ADCSRA & 0b00010000)==0)
  {
  }
```

a pour rôle d'effectuer en boucle les ligne entre accolades , (donc rien dans ce cas), tant que le bit4 de ADCSRA est à 0.

- Quel est l'effet de de ce segment dans le programme?

4 Réalisation d'un thermomètre

4.1 Fonction AffReel();

- Rechercher dans la librairie lcd2x16.h le rôle et la syntaxe de la fonction AffReel (r,e,d);

🗅 Rappeler l'expression de Vo en fonction de n

🗅 Insérer dans le programme (can.c), en fin de boucle infinie, le segment suivant:

```
Vo = 2.5e-3*n;
CursLCD (0,14);
AffReel (Vo,1,3);
```

et noter vos observations.

Remarque : différence entre le langage C et le langage mathématique.

	langage C	mathématiques
multiplication	*	x
notation scientifique	2.5e-3	2,5 x 10^{-3}
racine carrée	sqrt (*expression*)	$\sqrt{expression}$
quotient	a/b	a/b ou a:b

🗅 Proposer un programme (sauvegardé sous le nom can1.c) permettant d'obtenir l'affichage suivant:

n	**=**			x	x	x	x								
V	**o**	**=**	y	.	y	y	y	**v**							

où xxxxx représente la valeur de n sur 4 chiffres et y,yyy représente la valeur de la tension Vo sous forme décimale avec 3 chiffres après la virgule.
Les cases grisées correspondent à des chaines de caractères constantes (à n'afficher donc qu'une seule fois)

4.2 Thermomètre

🗅 Rechercher dans la documentation du LM19 l'expression de la température T en fonction de la tension Vo.

$T =$

🗅 Proposer un programme (sauvegardé sous le nom: thermometre.c) qui fournisse l'affichage suivant:

n	**=**	x	x	x	x		**V**	**o**	**=**	y	.	y	y	y	**v**
T	**=**	**z**	z	.	z	°	C								

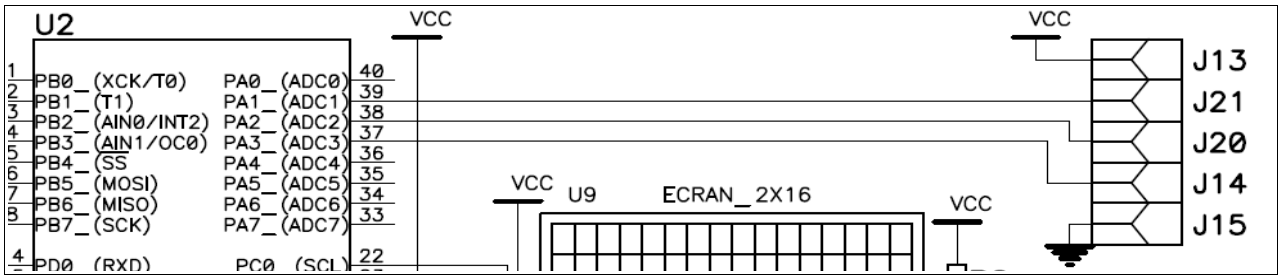
où xxxx est la valeur de n en décimal, y.yyy est la valeur de Vo et zz.z est la valeur de la température T

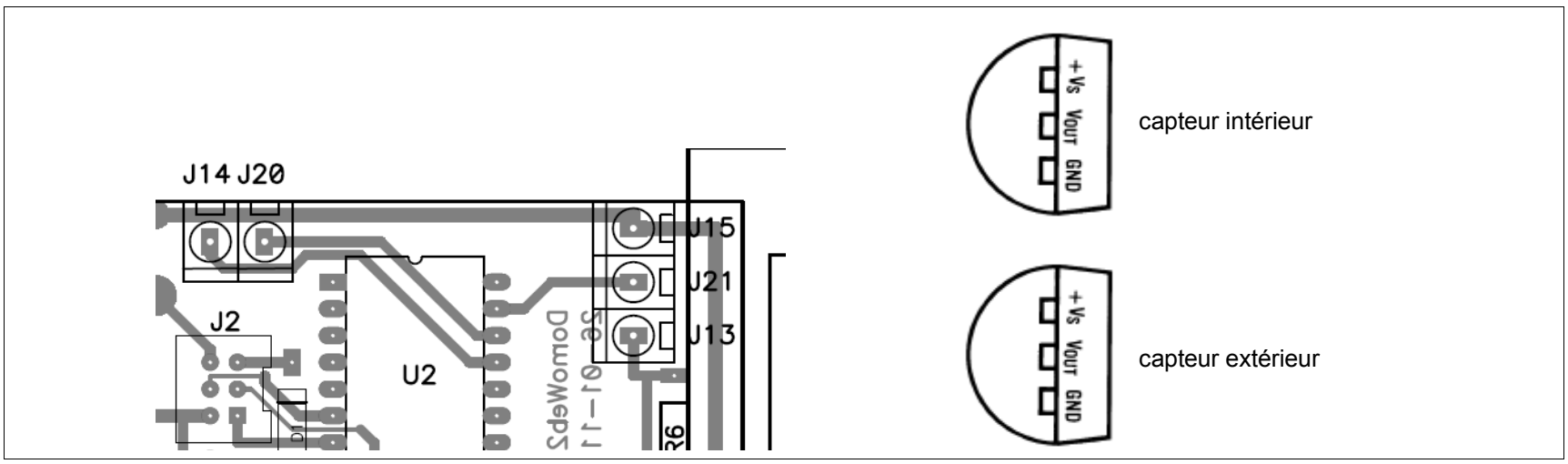
5 Exercices

5.1 Thermomètre intérieur extérieur

On se propose de réaliser un double thermomètre (intérieur, extérieur). On utilise pour cela deux capteurs LM19, l'un à l'extérieur, relié à ADC1 et l'autre à l'intérieur, relié à ADC2.

🗅 Compléter sur le schéma structurel et le schéma d'implantation, le branchement de ces capteurs





🗅 Rechercher dans la documentation du LM19 (page 5/8) l'expression simplifiée de la tension Vo en fonction de la température T (utilisable sur l'intervalle -55°C à 130°C)

🗅 En déduire l'expression simplifiée de la température T en fonction de V0

🗅 Proposer un programme(sauvegardé sous le nom: double_thermometre.c) utilisant l'expression simplifiée de T pour obtenir l'affichage suivant:

T	**e**	x	t	=		x	**x**	**.**	**x**	°	C				
T	**i**	n	t	=		y	**y**	**.**	**y**	°	C				

xx.x : température extérieure en degrés Celsius et yy.y : température intérieure en degrés Celsius.