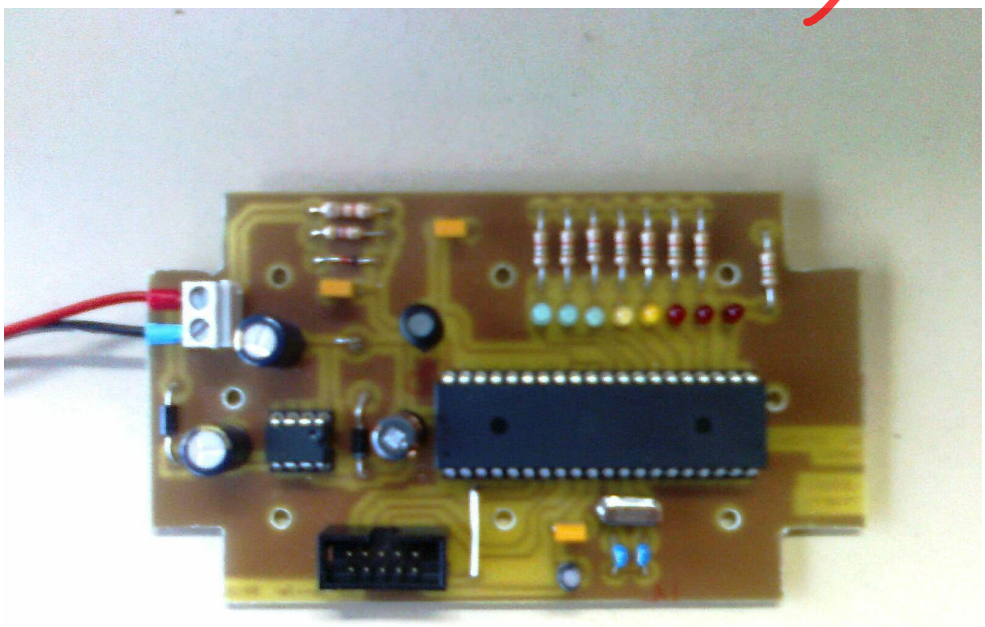
Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle

UNIVERSITE FRANCOIS-RABELAIS
TOURS
Institut Universitaire de Technologie
Département
GENIE ELECTRIQUE ET
INFORMATIQUE INDUSTRIELLE

Programmation de l'ATmega8535 pour mesure de tension batterie

Pas mal



JUNG Erik & RIDEAU Aurélien
2ème année groupe P1
Promotion 2011-2013

Enseignants :
Thierry LEQUEU
Sofi RODIER

Université François-Rabelais de Tours
Institut Universitaire de Technologie de Tours
Département Génie Électrique et Informatique Industrielle

UNIVERSITE FRANCOIS-RABELAIS
TOURS
Institut Universitaire de Technologie
Département
GENIE ELECTRIQUE ET
INFORMATIQUE INDUSTRIELLE

Programmation de l'ATmega8535

pour mesure de tension batterie

Illustr + source

2e

JUNG Erik & RIDEAU Aurélien
2ème année groupe P1
Promotion 2011-2013

Enseignants :
Thierry LEQUEU
Sofi RODIER

Sommaire

Aérer
Organiser visuellement aussi
= Metre TM1 en gras avec espace au dessus

Introduction

Au cours de ce quatrième semestre, dans le cadre du cours d'Étude et Réalisation, nous avons choisi en binôme un sujet lié à la programmation de l'ATmega8535. Ce projet donnera lieu à la remise d'un rapport écrit et à une présentation orale début avril.

Pour réaliser cela, nous allons utiliser une carte électronique de Monsieur LEQUEU sur laquelle est implémentée un microcontrôleur Atmel, de type Atmega8535, et 8 LED de couleurs différentes. Le but de ce projet est d'implémenter un programme dans le microcontrôleur, afin de mesurer la tension appliquée (par exemple, la charge d'une batterie) et de l'afficher en allumant des LED en fonction de la valeur de la tension.

Dans ce compte-rendu, nous allons d'abord présenter le cahier des charges de notre projet, puis nous ferons une présentation de l'ATmega et pour finir, la programmation du microcontrôleur et les résultats des tests effectués.

1. Présentation du projet

Le projet consiste à utiliser une carte électronique déjà réalisée, sur laquelle est implantée un microcontrôleur, et d'y implémenter un programme permettant de réaliser la mesure de la tension, d'une batterie par exemple. Cette mesure de tension sera ensuite affichée à l'aide de 8 LED de couleurs différentes situées sur la carte électronique. On écrira le programme en langage C à l'aide du logiciel CodeVisionAVR, un logiciel de programmation informatique.

Pour réaliser au mieux ce projet, nous avons commencé par établir un cahier des charges. Nous avons décrit dans celui-ci les fonctionnalités et les étapes de réalisation du projet, afin de bien mener nos recherches, et nous avons également établi un planning prévisionnel pour répartir le temps attribué à chaque tâche. Ce planning prévisionnel a également été complété par un planning réel, afin de montrer l'évolution du projet et les difficultés rencontrées.

1.1. Cahier des charges

Les contraintes et les fonctions liées au projet sont les suivantes :

- le système devra être capable de mesurer les différentes tensions,
- utilisation d'un microcontrôleur Atmega8535, limité à 5V par entrée,
- la mesure devra être la plus précise possible,
- la mesure de tension utilisera les 8 entrées du port analogique de l'ATmega,
- affichage de la tension sur des LED de couleurs différentes,
- utilisation du logiciel CodeVisionAVR pour programmer l'ATmega.

1.2. Planning prévisionnel et réel

Afin de garder une bonne organisation dans la réalisation de notre projet, nous avons dû faire un planning prévisionnel pour répartir le temps attribué à chaque tâche. Au cours des séances d'Étude et Réalisation, nous avons également complété ce document avec un planning réel pour vérifier l'évolution du projet.

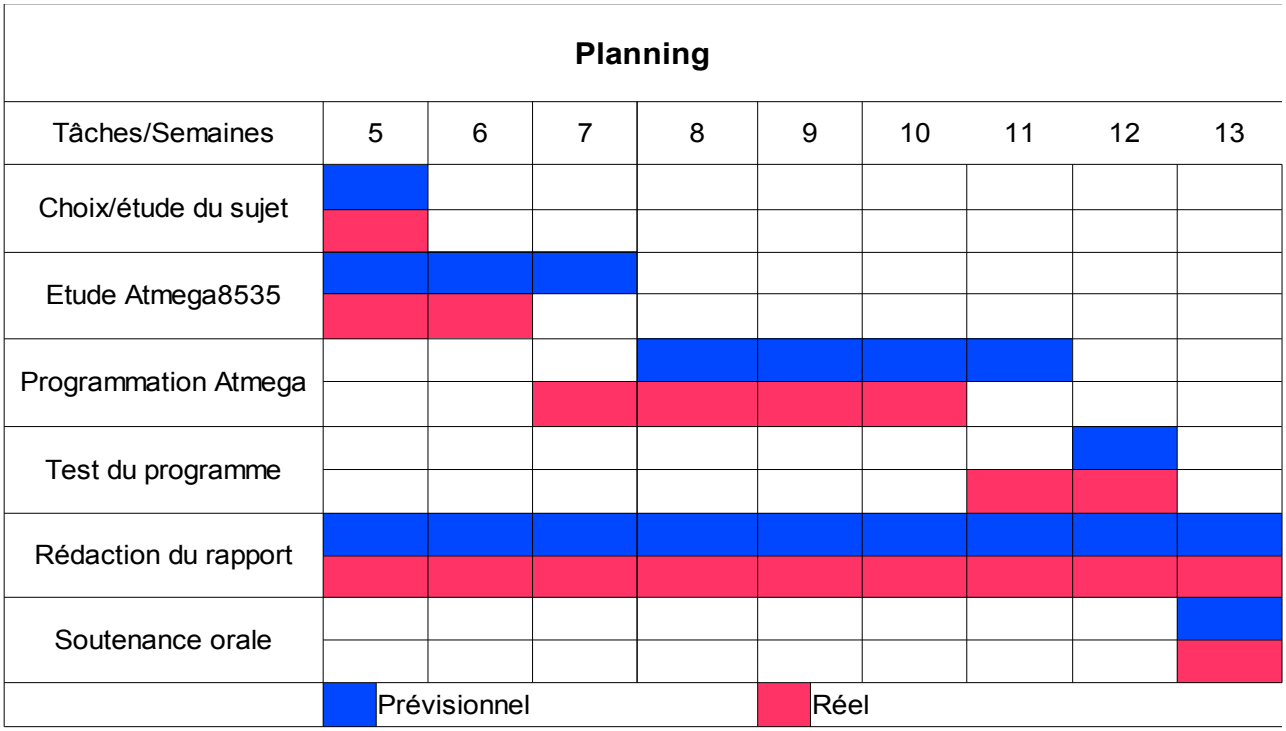


Illustration 1: Planning du projet

Comme nous pouvons le constater sur le planning réel ci-dessus, la programmation a été longue et les tests sur le programmes se sont étalés sur plusieurs séances. En effet, nous n'étions pas familier avec l'ATmega 8535 et nous n'avions encore jamais utilisé le logiciel CodeVision AVR pour programmer, ce qui nous a fait perdre du temps au début pour la prise en main du logiciel.

2. Programmation de l'ATmega8535

2.1. Présentation de l'ATmega8535

L'ATmega8535 est le microcontrôleur que nous allons programmer afin qu'il puisse gérer la mesure de la tension et l'affichage de celle-ci sur les LED.

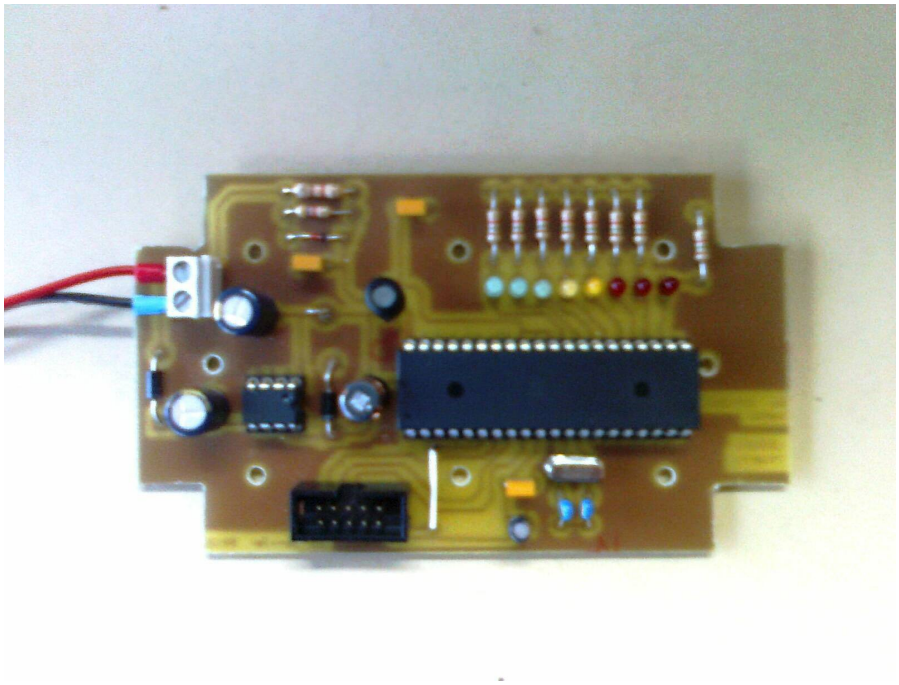


Illustration 2: Carte de programmation de l'ATmega et affichage

Dans ce projet, l'ATmega8535 est déjà installé sur une carte mise à notre disposition. Pour implémenter le programme sur le microcontrôleur, il suffit de raccorder l'ATmega au bus du PC avec un connecteur de type J-tag/HE10.

Illustration 3: Connecteur J-tag

Il suffit ensuite de compiler le programme à l'aide d'un logiciel comme AVR Studio ou CodeVision AVR. Une fois la compilation effectuée, le programme est chargé dans le microcontrôleur et est exécuté.

L'ATmega est un composant CMOS[1] 8 bits microcontrôleur qui est cadencé par une horloge de 16 MHz. Ce composant permet de programmer la partie « intelligente » de la carte. Les données échangées sont stockées dans des mémoires :

- 512 octets EEPROM[2]
- 512 octets SRAM[3]

2.2. Description des pattes de l'ATmega

L'ATmega comprend plusieurs ports bidirectionnels qui peuvent être des entrées comme des sorties.

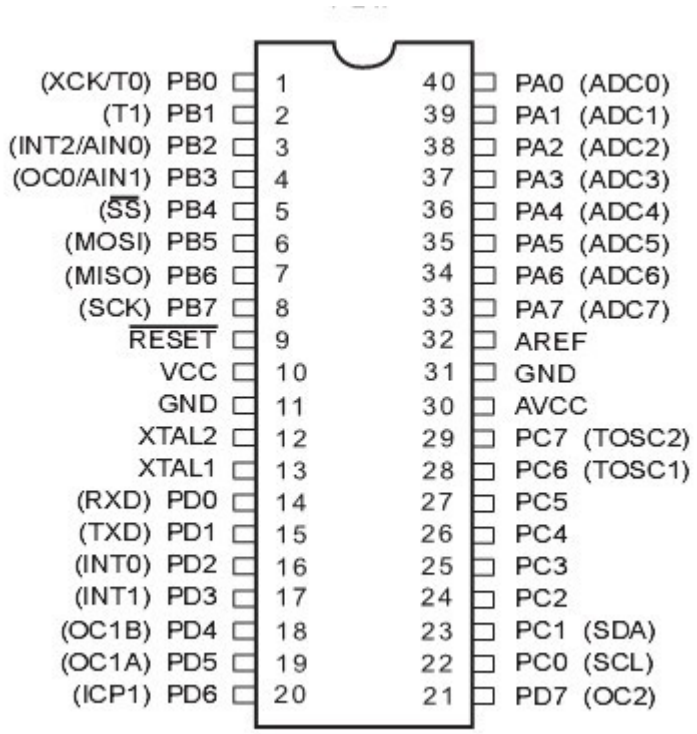


Illustration 4: Broches de l'ATmega8535

1 Complementary Metal Oxyde Semi-conductor
2 Type de mémoire utilisée lorsque les données ne doivent pas être perdues quand le composant n'est plus alimenté
3 Type de mémoire utilisée lorsque les données sont temporaires, perdues lorsque le composant n'est plus alimenté

Vcc est la broche d'alimentation principale de l'ATmega.

GND est la masse de l'ATmega.

Port A (PA0 à PA7)

Ports 8 bits bidirectionnels, ils peuvent être des entrées ou des sorties.Ce sont des ports analogiques utilisés pour la conversion analogique-numérique. Cependant, ils sont toujours alimentés entre 0 et 5V.

Port B (PA0 à PB7)

Ports 8 bits bidirectionnels. Ces ports sont utilisés pour la programmation.

Port C (PC0 à PC7)

Ports 8 bits bidirectionnels.

Port D (PD0 à PD7)

Ports 8 bits bidirectionnels. Ces ports sont utilisés pour connecter les LED qui témoignent de la charge de la batterie.

Reset peut générer une remise à zéro du système.

XTAL1 est une entrée d'horloge qui permet le fonctionnement de l'ATmega.

XTAL2 est une sortie de la patte inverseuse de l'amplificateur de l'oscillateur.

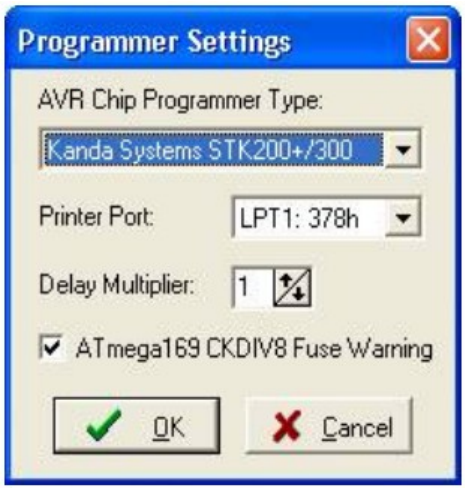
AVCC est une tension d'alimentation du port A et du convertisseur analogique-numérique. Cette patte doit être connectée à Vcc même si le convertisseur n'est pas utilisé.

AREF est une patte analogique de référence pour le convertisseur.

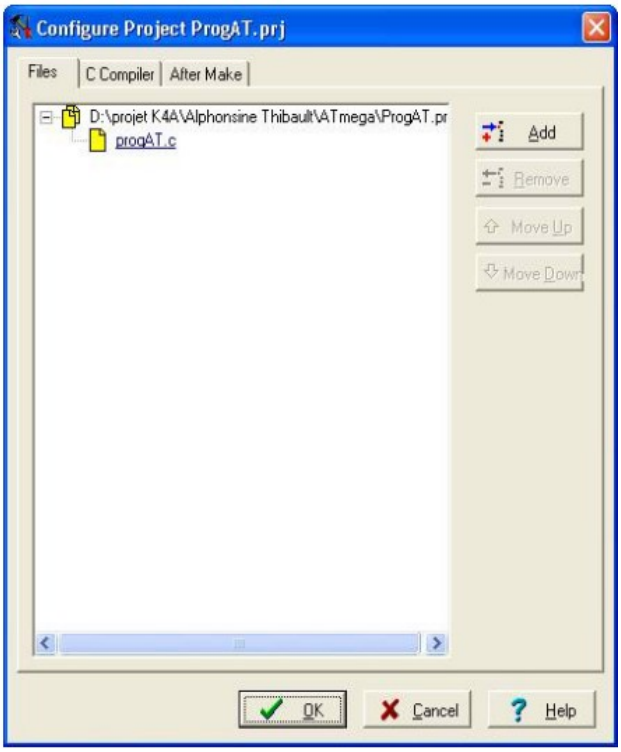
2.3. Configuration du logiciel CodeVision AVR

Pour réaliser la programmation de l'ATmega, nous avons utilisé le logiciel CodeVision AVR qui permet d'écrire des programmes informatiques en langage C et de programmer des microcontrôleurs.

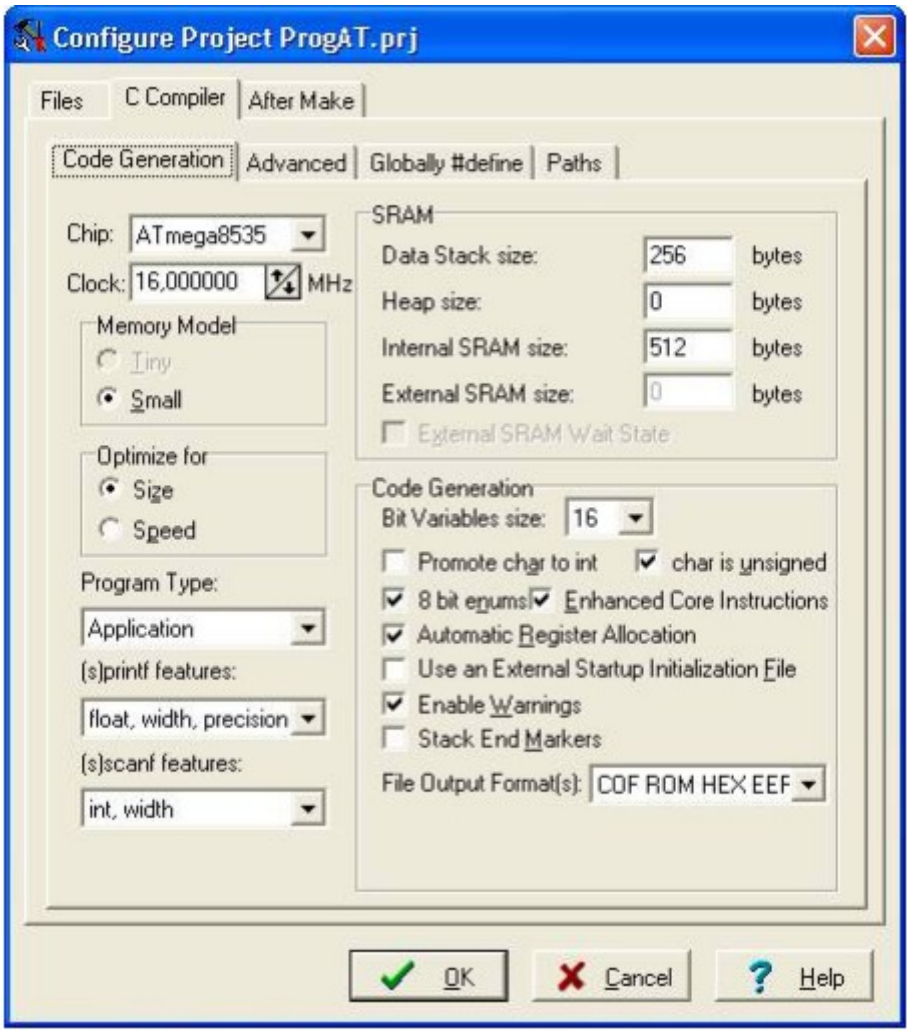
Dans la barre de menu → Settings → Programmer, on choisit la puce Kanda Systems STK200+/300 pour la programmation de l'ATmega8535.



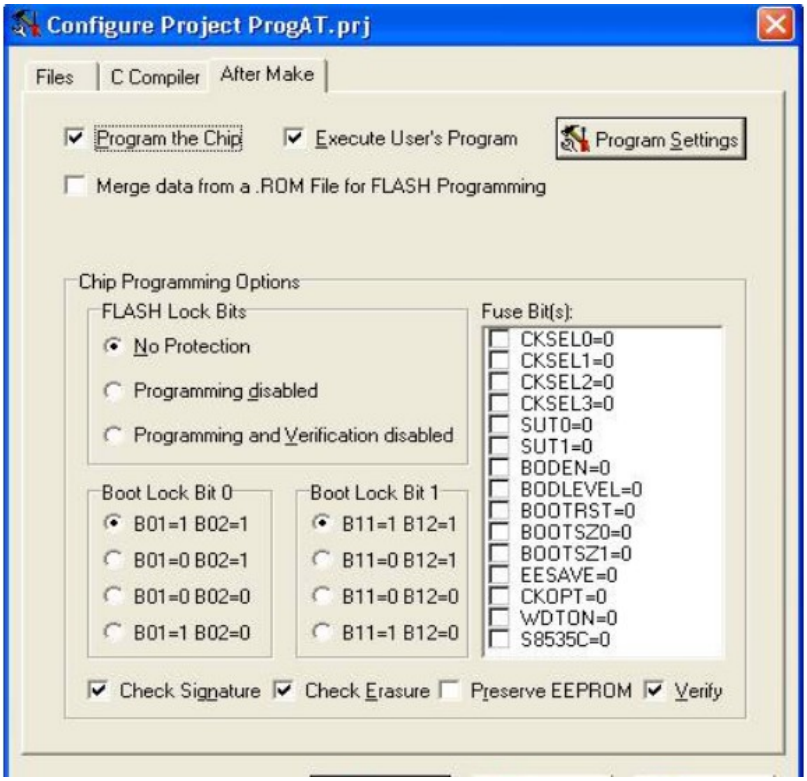
Ensuite, Tools → Configure → Configure project. Ceci permet de configurer l'emplacement où on veut enregistrer notre programme.



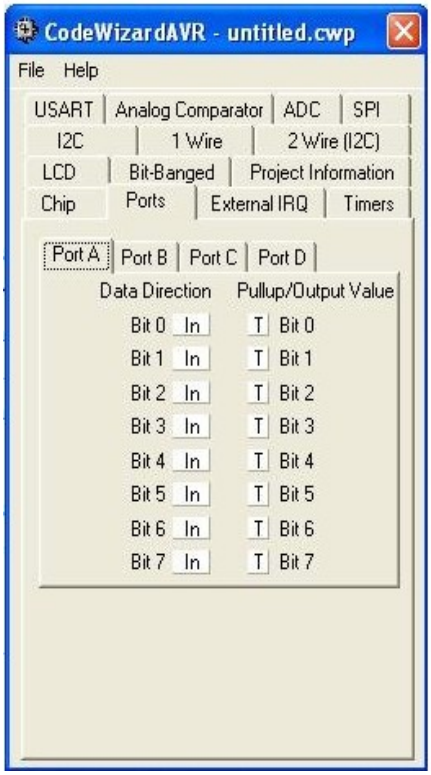
Une fois l'emplacement défini, on paramètre le composant Atmega8535 à une fréquence de fonctionnement de 16 MHz. Dans la mémoire SRAM, les données doivent être contenues dans 256 octets. La mémoire interne SRAM de l'ATmega est de 512 octets. Pour le codage, on peut utiliser des variables de taille maximum 16 bits, de type caractère non signé (unsigned char).



Dans la configuration de l'ATmega, on transfert le programme dans la puce et on autorise l'exécution du programme.



On configure les ports de l'ATmega comme des entrées et des sorties selon les besoins du programme :



2.4. Commande des LED

Le contrôle des LED pour signaler le niveau de tension s'effectue selon l'ordinogramme suivant :

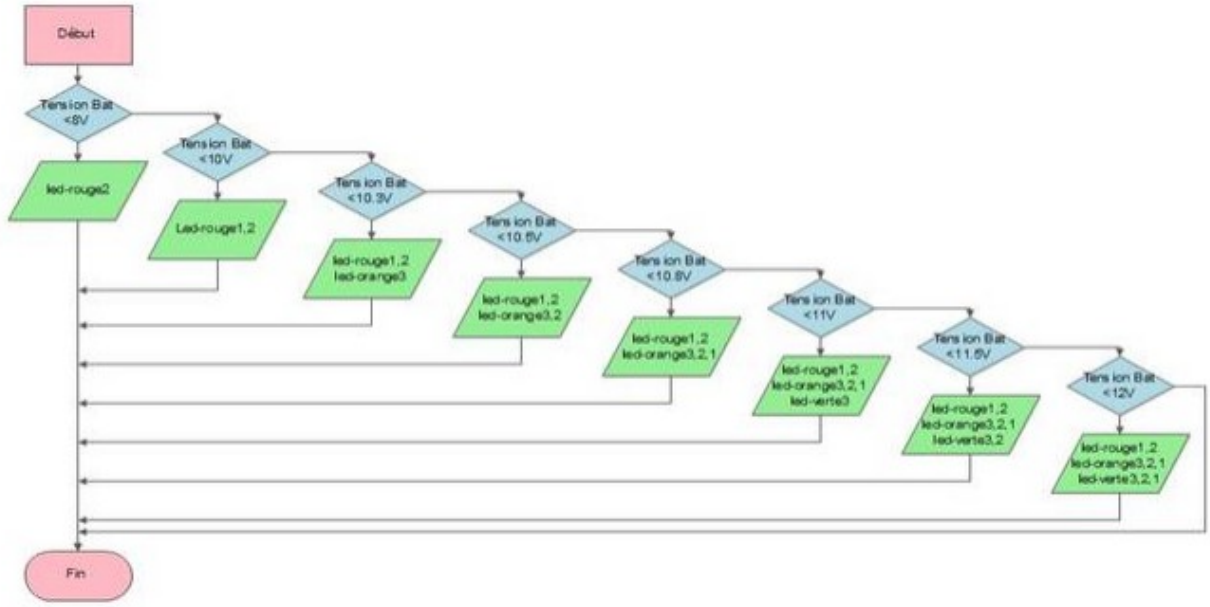


Illustration 5: Organigramme de commande des LED

Les tensions délivrées étant des tensions analogiques, il nous a fallu les convertir en valeurs numériques pour pouvoir les traiter dans notre programme. Pour cela, nous avons utilisé le convertisseur analogique numérique présent dans l'ATmega8535. Le programme complet est disponible en annexe, à la fin du dossier.

Conclusion

Au cours de ce projet, nous avons surtout étudié la programmation d'un micro contrôleur type ATmega. Cette étude nous a beaucoup apporté en connaissance de logiciel de programmation industrielle, nous avons dû faire des recherches sur CodeVisionAVR car nous n'avions jamais utilisé ce logiciel. En effet, nous avons rencontré des problèmes lors de la programmation et de la configuration, et notamment lors de la compilation du programme dans l'ATmega8535. Le site de Thierry Lequeu nous a été très utile pour avancer dans notre projet, grâce à la documentation sur l'ATmega et CodeVision.

Ce projet nous a aussi permis de mettre en oeuvre d'autres compétences, comme l'organisation de notre travail et la gestion du temps. Nous avons du effectivement réaliser en septembre un cahier des charges et un planning prévisionnel, puis le tenir a jour pour en faire un planning réel. Ce travail nous a permis de nous rendre compte de l'importance de l'organisation pour mener a bien notre projet, et cette expérience nous servira pour d'autres projets et le stage de fin d'études.

Résumé

Dans le cadre de l’Étude et Réalisation au semestre 4, nous avons à mener un projet, puis à rédiger un rapport écrit et à le présenter lors d'un oral. Nous avons choisi comme projet de travailler sur la programmation d'un microcontrôleur, l'ATmega8535, afin de mesurer la tension simulée d'une batterie et de l'afficher en allumant des LED. Pour cela, nous avons utilisé le logiciel de programmation informatique CodeVision AVR. Nous avons utilisé le langage C pour écrire le programme. On a ensuite implémenté le programme dans l'ATmega, en le raccordant au bus du PC avec un connecteur J-tag. Une fois exécuté, le programme traite la mesure de la tension analogique en la convertissant en valeur numérique et allume les LED en fonction de cette valeur. Lors des tests, nous avons rencontré des problèmes avec la compilation du programme, qui comportait au début des erreurs. Puis, nous avons résolu les erreurs, ce qui nous a permis de compiler et d'exécuter le programme dans l'ATmega.

≈ 180 mots

Bibliographie

Sites internet :

[1] Thierry LEQUEU. La documentation de Thierry LEQUEU sur OVH, 2012. (Page consultée le 01/02/2012) <http://www.thierry-lequeu.fr/>

[2] Atmel. ATmega8535(L), 2006. <http://www.atmel.com/images/doc2502.pdf>

[3] Atmel. AVR033: Getting Started with the CodeVisionAVR C Compiler, 2008. <http://www.atmel.com/images/doc2500.pdf>

Index des illustrations

Annexe – Programme complet

```
/*****************************************************
Project : Mesure de tension de batterie avec ATmega
Version :
Date    : 06/03/2013
Author  : Erik JUNG & Aurélien RIDEAU
Company : IUT GEII
Comments:


Chip type           : ATmega8535
Program type        : Application
Clock frequency     : 16,000000 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 256
*****************************************************/

// Declare your global variables here
#include <mega8535.h>
#include <stdio.h>
#include <delay.h>
#include <math.h>
#define ADC_VREF_TYPE 0x00

//Déclaration des entrées
#define MesureBatterie PINA.0

//Déclaration des sorties
#define led_vert1 PORTC.5
#define led_vert2 PORTC.6
#define led_vert3 PORTC.7
#define led_jaune1 PORTC.3
#define led_jaune2 PORTC.4
#define led_rouge1 PORTC.0
#define led_rouge2 PORTC.1
#define led_rouge3 PORTC.2

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=(adc_input | (ADC_VREF_TYPE && 0xff));
// Delay needed for the stabilization of the ADC input voltage
// Start the AD conversion
ADCSRA=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA && 0x10)==0);
ADCSRA=0x10;
return ADCW;

// ADC initialization
// ADC Clock frequency: 1000,000 kHz
// ADC Voltage Reference: AREF pin
```

```
// ADC High Speed Mode: Off
// ADC Auto Trigger Source: None
ADMUX=(ADC_VREF_TYPE && 0xff);
ADCSRA=0x84;
SFIOR=0xEF;
}

void main(void)
{
// Declare your local variables here
float V_Batterie;
float Tens_Bat_equi;
unsigned int Bat_equi;

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTC=0x00;
DDRC=0xFF;

// Port D initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTD=0x00;
DDRD=0xFF;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x01;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
```

```
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x02;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

while (1)
    {
          //MESURE DE LA TENSION DE LA BATTERIE


          Bat_equi = read_adc(MesureBatterie);       //récupère la tension MesureBatterie
          Tens_Bat_equi = (5*(float)Bat_equi)/1024 ; //conversion de la tension batterie (8-14V) en
valeur numérique (0-5V)
          V_Batterie = Tens_Bat_equi*218/68;        //recalcule la vraie tension batterie
          delay_ms (50);                            //attente de 50 ms

          //LEDS EN FONCTION DE LA TENSION DE LA BATTERIE
```

```
if (Bat_equi <=8) //Si tension de batterie inférieure à 8V
{
	led_rouge1=1; //Allume led rouge1
}
else
{
	if (Bat_equi <=10) //Si tension de batterie  inférieure à 10V
	{
		led_rouge1=1; //Allume led rouge1
		led_rouge2=1; //Allume led rouge2
	}


else
{
	if (V_Batterie <=10.3) //Si tension de batterie inférieure à 10.3V
	{
		led_rouge1=1; //Allume led rouge1
		led_rouge2=1; //Allume led rouge2
		led_rouge3=1; //Allume led rouge3
	}
	else
	{
		if (V_Batterie <=10.5) //Si tension de batterie inférieure à 10.5V
		{
			led_rouge1=1; //Allume led rouge1
			led_rouge2=1; //Allume led rouge2
			led_rouge3=1; //Allume led rouge3
			led_jaune1=1; //Allume led orange1
		}
		else
		{
			if (V_Batterie<=10.8) //Si tension de batterie inférieure à 10.8V

			{
				led_rouge1=1; //Allume led rouge1
				led_rouge2=1; //Allume led rouge2
				led_rouge3=1; //Allume led rouge3
				led_jaune1=1; //Allume led orange1
				led_jaune2=1; //Allume led orange2
			}
			else
			{
				if (V_Batterie<=11) //Si tension de batterie inférieure à 11V

				{
					led_rouge1=1; //Allume led rouge1
					led_rouge2=1; //Allume led rouge2
					led_rouge3=1; //Allume led rouge3
					led_jaune1=1; //Allume led orange1
					led_jaune2=1; //Allume led orange2
					led_vert1=1; //Allume led verte1
				}
				else
				{
```

```
                                    if (V_Batterie <=11.5) //Si tension de batterie inférieure à 11.5V
                                    {
                                        led_rouge1=1; //Allume led rouge1
                                        led_rouge2=1; //Allume led rouge2
                                        led_rouge3=1; //Allume led rouge3
                                        led_jaune1=1; //Allume led orange1
                                        led_jaune2=1; //Allume led orange2
                                        led_vert1=1; //Allume led verte1
                                        led_vert2=1; //Allume led verte2

                                    }
                                    else
                                    {
                                        if(V_Batterie<=12) //Si tension de batterie inférieure à 12V
                                        {
                                            led_rouge1=1; //Allume led rouge1
                                            led_rouge2=1; //Allume led rouge2
                                            led_rouge3=1; //Allume led rouge3
                                            led_jaune1=1; //Allume led orange1
                                            led_jaune2=1; //Allume led orange2
                                            led_vert1=1; //Allume led verte1
                                            led_vert2=1; //Allume led verte2
                                            led_vert3=1; //Allume led verte3
                                        }
                                    }
                                }
                            }
                        }
                    }

                }
                }
            }
        }
```