

Project Summary (English Translation)

1. Introduction

This semester project was carried out in the framework of the *Microcontrollers and Digital Systems* course taught by Alexandre Schmid at EPFL. Using the STK-300 development board and an Atmega128L microcontroller programmed in assembly, we were asked to design a system based on a 1-Wire temperature sensor and at least two additional applications. In our implementation, we developed a complete control system for a skewer oven, including temperature measurement, LCD display, servo-motor control, adjustable reference temperature, and a short melody played at the end of the cooking process.

2. General Description of the Application

The application emulates a small electric skewer oven with three operating states based on measured temperature:

“Off” mode: when the temperature is below 25 °C, the oven is considered off. The LCD displays “off T = X” on the first line, where X is the real-time temperature, and shows the reference temperature (T_{ref}) on the second line.

“Pre-heating” mode: once the temperature exceeds 25 °C, the first line switches to “pre-heating T = X”, while the second line continues to display (T_{ref}).

“Cooking” mode: when the temperature goes above (T_{ref}), the skewer motor starts rotating to simulate cooking. The LCD displays “cooking T = X” on the first line and keeps showing (T_{ref}) on the second line. After one full rotation of the motor, the system plays the beginning of *Take Five* by Dave Brubeck and indicates that the cooking phase is finished.

The reference temperature (T_{ref}) (initially set to 30 °C) defines the transition between pre-heating and cooking and can be adjusted by the user via buttons.

3. User Operation – System and Interface

From power-up, the system runs continuously and displays the current mode, the real-time temperature, and the reference temperature on the LCD, with a resolution of 0.25 °C.

The user can interact with the system using three buttons:

- **SW0:** decrease (T_{ref}) by 0.25 °C,
- **SW1:** increase (T_{ref}) by 0.25 °C,
- **SW2:** reset (T_{ref}) to its default value of 30 °C.

This provides a simple and intuitive interface similar to a conventional oven, allowing the user to set the desired cooking temperature.

4. Technical Description – Hardware and Interfaces

The system is implemented on the STK-300 board using the following hardware configuration:

Temperature sensor: a 1-Wire digital temperature sensor connected to **Port E**.

Motor control: a servo motor connected to **Port F**. Due to the address of Port F (0x62) being outside the range accessible by a simple **OUT** instruction, we use **STS** and SRAM addressing to control this port.

LCD display: driven via **Ports A and C** using a provided **printf.asm** library.

Buzzer: connected to **Port B** to play the melody at the end of the cooking process.

Buttons: three external interrupts are configured to detect button presses on SW0, SW1, and SW2 for adjusting or resetting (T_{ref}).

5. Program Operation and Software Architecture

The program is structured around a main loop managing three states corresponding to the modes described in Section 2. Temperature acquisition is handled by a dedicated module **wire1**, which communicates with the 1-Wire sensor and stores the measurement across two 8-bit variables (**a0** and **a1**). A second routine, **mouv**, extracts the relevant bits from these variables (using shifts and bit masks) and combines them into a single 8-bit value **W** representing the temperature. In the main program, three subroutines implement the system states:

inf for the “off” mode,
supp1 for the “pre-heating” mode,
supp2 for the “cooking” mode.

Transitions between states are determined by comparing the measured temperature with fixed thresholds (e.g., 25 °C) or with (T_{ref}) stored in **W**, using **CP** and **CPI** instructions. The motor is controlled via a dedicated module containing the subroutine **motorun** and the macro **MOTOR**. **motorun** uses nested loops and a counter to drive the motor step-by-step, allowing multiple invocations of **MOTOR** to complete one full rotation while still periodically updating the temperature. The sound module implements the melody using a lookup table addressed by pointer registers (**Z**).

Finally, we rely on several course-provided libraries: **definition.asm** (register and variable definitions), **macro.asm** (common macros), and **printf.asm** (LCD printing routines). Together, these modules form a complete embedded control system for a temperature-controlled skewer oven with user interaction, motor actuation, and audio feedback.

MICROCONTRÔLEURS ET SYSTÈMES NUMÉRIQUES

PROJET DE FIN DE SEMESTRE - RAPPORT GROUPE

96

Gabriel Le Guay et Blaise Depauw

Pour le 1er juin 2021

1 Introduction

Dans le cadre du cours de Microcontrôleurs et systèmes numériques dispensé par Alexandre Schmid, un projet de fin de semestre était proposé afin de permettre d'approfondir et étendre les connaissances théorique acquises, ainsi que de développer des compétences de gestion de projet et de développement de projet en groupe. Ayant accès au kit STK-300 et faisant usage du MCU Atmega128L, nous devions coder en assembleur un système utilisant un capteur de température 1-Wire et deux autres applications. Nous avons choisis d'implémenter en plus la rotation d'un moteur servo, la possibilité de modifier les températures de référence et de la reset, un affichage LCD et une musique à la fin.

2 Description générale de l'application

Notre application serait utilisée pour un four à brochettes. Les températures, et le temps de cuisson ont été vues à la baisse en raison des limites du capteur et pour le bien de la démonstration vidéo. Notre implémentation comprend trois états :

- le mode « éteint », le four est éteint lorsque la température est inférieure à 25 degrés. Il est alors affiché « éteint T=X » sur la première ligne avec X la température en temps réel. Et sur la seconde ligne, la température de référence (T_{ref}).
- Le mode « préchauffage », dès que la température excède les 25 degrés, l'affichage de la première ligne change, il est alors marqué « préchauffage T=X » et la même qu'avant pour la seconde ligne.
- Le mode « cuisson ». Enfin, lorsque la température passe la barre des T_{ref} degrés, le moteur à brochette se met alors à tourner afin de les faire cuire, pendant qu'il sera affiché « cuisson T=X » en première ligne (et toujours la température de référence en seconde ligne). Il va effectuer un tour à la suite duquel sonnera une musique. Cette musique est le début d'un morceau de jazz nommé Take five de l'artiste Dave Brubeck.

Quand l'état du mode cuisson est finit, il sera alors affiché « cuisson fini ». Il est possible de régler la température de référence (T_{ref}), température où la cuisson et le moteur commenceront. Ici elle sera initialement à 30 degrés, puis pourra être réglée en appuyant sur des boutons.

3 Mode d'emploi - mise en opération système et utilisateur

A partir de son lancement, le système sera toujours actif et affichera le mode (selon la température), la température actuelle et la température de référence. Elle sera affichée avec une précision de 0,25 degrés.

L'utilisateur pourra lui modifier la température de référence (qui sera initialement à 30 degrés). En effet, en appuyant sur le bouton 0, T_{ref} sera alors décrémentée de 0,25 degrés. En appuyant sur le bouton 1, T_{ref} sera alors incrémentée de 0,25 degrés. Afin de reset la température de référence, l'utilisateur pourra appuyer sur le bouton 2 : T_{ref} retournera alors directement à 30 degrés. T_{ref} correspond au passage du mode « préchauffage » au mode « cuisson »(comme indiqué dans la description du programme). Ce qui nous donne un système indépendant et simple d'utilisation, à l'image d'un four classique.

4 Description technique de l'application et du matériel

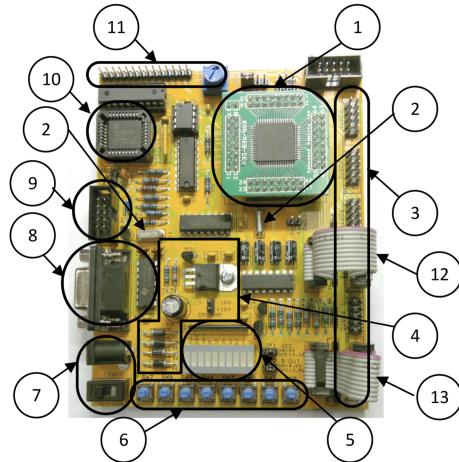


FIG. 1 – Principaux modules de la carte STK-300

Sur la figure on a les ports A à E indiqué par la bulle 3. Pour le buzzer, qui va effectuer la musique, nous utilisons le port B.

Le capteur de température sera branchée sur le port E.

Pour le port F, on a du utiliser l'accès à la mémoire Sram avec la commande STS car l'adresse du port F (0X62) est trop lointaine pour un simple out (qui n'a accès qu'à des adresses comprises entre 0X20 et 0X60). C'est pourquoi nous y avons branché le moteur.

Pour l'affichage LCD, nous utilisons les portes A et C.

Nous avons 3 interruptions qui vont se déclencher lorsque l'on va appuyer sur les boutons suivants (6 sur la figure) :

SW0 Pour décrémenter T_{ref}

SW1 Pour incrémenter T_{ref}

SW2 Pour reset la valeur de T_{ref}

5 fonctionnement du programme

Pour capter la température et pouvoir la comparer, nous appelons le modules « wire1 » qui contient la sous-routine « mouv » et « temp ». Wire1 va utiliser le capteur de température, et attribuer la valeur à a0 et a1 (codés sur 8bits chacun). Tandis que mouv va placer les bits de a0 et a1 dont nous avons besoins dans W. Dans notre main, nous avons 3 états qui ont été précisés dans la sections 2. On peut voir sur la figure ci-dessous l'architecture du main et l'interaction entre les trois états. Après que le mode cuisson soit resté suffisamment longtemps pour que le moteur fasse un tour complet, on jump dans « fin » qui va afficher « cuisson fini » et lancer la musique implémentée dans le module sound.

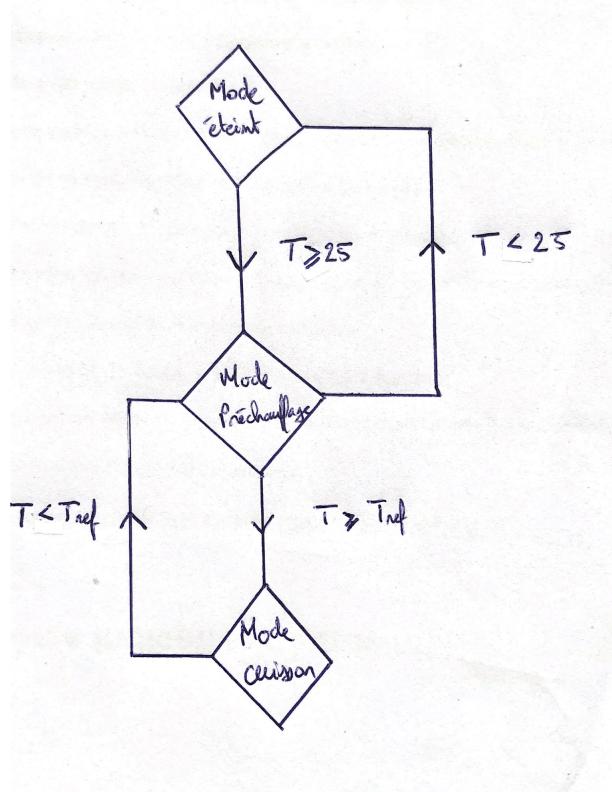


FIG. 2 – schéma de la structure du main

6 présentation des modules : (parties indépendantes du programme)

Le module Wire1 utilise donc la carte et capte la température. Celle-ci est placé dans la sous-routine temp, dans les variables a0 et a1. Cependant, le format des deux variables n'était pas appropriées. Nous n'avions besoin uniquement de 8bits en tout. C'est pourquoi, dans Wire1, il y a également la sous-routine Mouv. Celle-ci prend alors les deux premiers bit de a1 et les 6 derniers de a0. Pour cela, on a utilisé la commande « lsl » et « lsr »(cf figure ci-dessous). On aurait pu également utiliser des masques par exemple: avec a0 and 0011111 et a1 and 11000000, puis les ajouter. Cette valeur est placée dans la variable W. (cf figure 4)

Dans notre main, nous avons implémenté trois sous-routines correspondant aux trois états expliqués précédemment. La sous-routine « inf » correspondant au mode éteint, « supp1 » pour le mode de préchauffage et « supp2 » pour le mode cuisson.

Une fois arrivée dans une de ces sous-routines, le programme alterne entre elles et reste dedans. Pour passer de l'une à l'autre, on a comparé la température avec des constantes (ici 25) ou avec la valeur de Tref placée dans la variable « W ». Ces comparaisons se font avec les commandes « cp » lorsque l'on compare la température en temps réel avec la variable Tref (W), et « cpi » lorsque l'on compare avec une constante (ici 25).

Ensuite, on a le module moteur. Celui-ci comprend la sous-routine motorun et la macro MOTOR. Ainsi, dans le main et plus précisement dans Supp2, on appelle la sous-routine motorun 18 fois qui elle appelle 10 fois 6 fois (6 fois pour faire un tour) MOTOR. La macro motor prend pour paramètre le signal à envoyer sur les bornes du moteur.

En effet, la sous routine motorrun utilise une boucle loop qui appelle 6x la macro motor, celle-ci repete 10x dans la sous routine a l'aide d'un compteur qui s'incrémente à chaque passage.

Le moteur a été fait pas à pas afin de récupérer la température entre temps.

Ensuite, il y a le module sound, dans lequel il y a la sous routine musique qui joue la musique implémentée. Pour implémenter la musique dans le programme, nous utilisons une look-up table, avec des pointeurs(z).

Enfin, nous utilisons également les librairies données dans le cour, à savoir : définition.asm qui va attribuer les différentes variables aux registres, macro.asm qui va nous permettre d'utiliser les macros usuelles et printf.asm qui va nous permettre d'effectuer l'affichage LCD.

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

FIG. 3 – Attribution des bit a0 et a1

$$A_0 = X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0$$

$$A_1 = Y_7 Y_6 Y_5 Y_4 Y_3 Y_2 Y_1 Y_0$$

$$\text{mou } w, A_0 \rightarrow w = A_0$$

lsl w

$$lsl w \rightarrow w = 00 X_7 X_6 X_5 X_4 X_3 X_2$$

$$\text{mou } w, A_1 \rightarrow w = A_1$$

lsl w
x6

$$\rightarrow w = Y_1 Y_0 000000$$

$$\text{add } w, w_- \Rightarrow w = Y_1 Y_0 X_7 X_6 X_5 X_4 X_3 X_2$$

FIG. 4 - schéma explicatif du module mouv

Annexe :

Main :

```
.include "macros.asm" ; include macro definitions
.include "definitions.asm" ; include register/constant definitions

; === interrupt table ===
.org 0
jmp reset
jmp int_0 ; PIND0..3
jmp int_1
jmp int_2
jmp int_3
jmp int_4 ; PINE4..7
jmp int_5
jmp int_6
jmp int_7

; === interrupt service routines ===
int_0:
    in _sreg,SREG; sauvegarder du contexte
WAIT_US 100
    inc b0
    out SREG,_sreg
    reti
int_1:
    in _sreg,SREG; sauvegarder du contexte
WAIT_US 100
    dec b0 ; incrémenté la température
    out SREG,_sreg
    reti
int_2:
    in _sreg,SREG
    ldi b0,0b01111000 ;reset la température de cuisson
    out SREG,_sreg
    reti
int_3: reti
int_4: reti
int_5: reti
int_6: reti
int_7: reti

; === initialisation (reset) ===
reset :
    LDSP RAMEND; load stack pointer (SP)
    sbi DDRB,SPEAKER ;définit le port B pour le speaker
```

```

OUTEI DDRF,0x0f ; définit le port F pour le moteur
    OUTI EIMSK,0b00011111 ; définit les interruptions
    OUTEI EICRA,0b11111111
    OUTI EICRB,0b11111111
    sei ;set global interrupt
rcall wire1_init ; initialize 1-wire(R) interface
rcall lcd_init ; initialize LCD
rjmp main

.include "lcd.asm" ; include LCD driver routines
.include "printf.asm" ; include formatted printing routines
.include "wire1.asm" ; include Dallas 1-wire(R) routines
.include "sound.asm" ; module pour la musique
.include "moteur.asm" ; module pour le moteur

main :
ldi b0,0b01111000 ; initialise la t° de cuisson
    rcall lcd_home ; initialise le lcd
    rcall LCD_clear
rcall temp ;met la température dans w
cpi w, 0b01100100 ;compare la température
brge supp1; emmène dans le préchauffage si t°>30°
cpi w, 0b01100100
brlt inf; emmène dans le "four éteint" si t°<30°
rjmp main

supp1:
rcall temp
cpi w, 0b01100100 ;25
brlt inf
cp w, b0;30
brge supp2
rcall LCD_clear
PRINTF LCD
.db "prech",FFRAC2+FSIGN,a,4,$42,"C ",0; affiche la première ligne du lcd
PRINTF LCD
.db LF,"Tcui = ", FFRAC,b,2,$22,0; affiche la deuxième ligne du lcd
rjmp supp1

inf:
rcall temp
cpi w, 0b01100100 ;25
brge supp1
rcall LCD_clear
PRINTF LCD
.db "eteint",FFRAC2+FSIGN,a,4,$42,"C ",0
PRINTF LCD
.db LF,"Tcui = ", FFRAC,b,2,$22,0
rjmp inf

```

```

supp2:
ldi r25,0 ; initialise le compteur
mvtmot:
rcall temp
cp w, b0 ;30
brlt supp1
rcall motorun ;fait faire 1/18 tour au moteur après réduction
rcall LCD_clear
PRINTF LCD
.db "cuisson",FFRAC2+FSIGN,a,4,$42,"C ",0
PRINTF LCD
.db LF,"Tcui = ", FFRAC,b,2,$22,0
inc r25
cpi r25,0x12
breq PC+2; si le moteur a finit son tour complet, on va dans fin
rjmp mvtmot
rjmp fin

```

```

fin:
rcall LCD_clear
PRINTF LCD
.db "cuisson terminee",CR,0
rcall music ; lance la musique

```

reinitialisation:
rjmp main ; à la fin de la musique, on retourne dans le main et on réinitialise

Moteur:

```

.macro MOTOR ;macro guidant le moteur
ldi w,@0
sts PORTF,w ; output motor pin pattern
WAIT_US 1200 ; wait period
.endmacro

```

```

motorun : ;sous routine qui fait faire 1/18 tour au moteur après réduction
ldi r27,0
loop:; boucle qui fait faire 1 tour au moteur avant réduction
MOTOR 0b0101 ; suite d'instruction pour faire le tour complet avant réduction
MOTOR 0b0001
MOTOR 0b1011
MOTOR 0b1010
MOTOR 0b1110
MOTOR 0b0100
inc r27
cpi r27,0x0A
breq PC+2; quitter la sous routine si le compteur est à 10
rjmp loop
ret

```

Sound :

```
takefive: ;suite de notes de la musique
.db lam,rem2,fam2,som2
.db la2,lam2,la2,som2
.db fam2,fam2,som,som
.db dom2,dom2,rem2,rem2
.db rem2,rem2,rem2,rem2
.db fa2,fa2,dom2,dom2
.db rem2,rem2,rem2,rem2
.db rem2,rem2,dom2,dom2
.db som,som,rem2,rem2
.db rem2,rem2,rem2,rem2
.db lam,rem2,fam2,som2
.db la2,lam2,la2,som2
.db fam2,fam2,som,som
.db dom2,dom2,rem2,rem2
.db rem2,rem2,rem2,rem2
.db dom2,dom2,som,som
.db rem2,rem2,rem2,rem2
.db rem2,rem2,fa2,fa2
.db dom2,dom2,rem2,rem2
.db rem2,rem2,rem2,rem2
.db 0;
```

```
music: ;joue la musique
```

```
ldi zl,low(2*takefive);
```

```
ldi zh,high(2*takefive)
```

```
play: lpm
```

```
adiw zl,1
```

```
tst r0
```

```
breq end
```

```
mov a0,r0
```

```
ldi b0,100
```

```
rcall sound
```

```
rjmp play
```

```
end : ret
```

Wire1:

```
temp: ; lis la temperature et la met dans w
rcall wire1_reset      ; send a reset pulse
CA wire1_write, skipROM ; skip ROM identification
CA wire1_write, convertT ; initiate temp conversion
WAIT_MS 750             ; wait 750 msec

rcall lcd_home          ; place cursor to home position
rcall wire1_reset      ; send a reset pulse
CA wire1_write, skipROM
CA wire1_write, readScratchpad
rcall wire1_read        ; read temperature LSB
mov c0,a0
rcall wire1_read        ; read temperature MSB
mov a1,a0
mov a0,c0
rcall mouv
ret
```

```
mouv:; prend 6 bits avant et 2 bits après la virgule de la temperature et les met dans w
mov w,a0
lsl w
lsl w
mov _w,a1
lsl _w
lsl _w
lsl _w
lsl _w
lsl _w
add w,_w
ret
```