Bordeaux INP
ENSEIRB
MATMECA

Embedded system project

Persistence of vision

Presented by
BRISSAUD Cloé
COUZINIER Adrien
TALBI Anas

TABLE OF CONTENTS

01 **INTRODUCTION**

Architecture & Goal

02 **Analog Clock**

Timer & Bluetooth

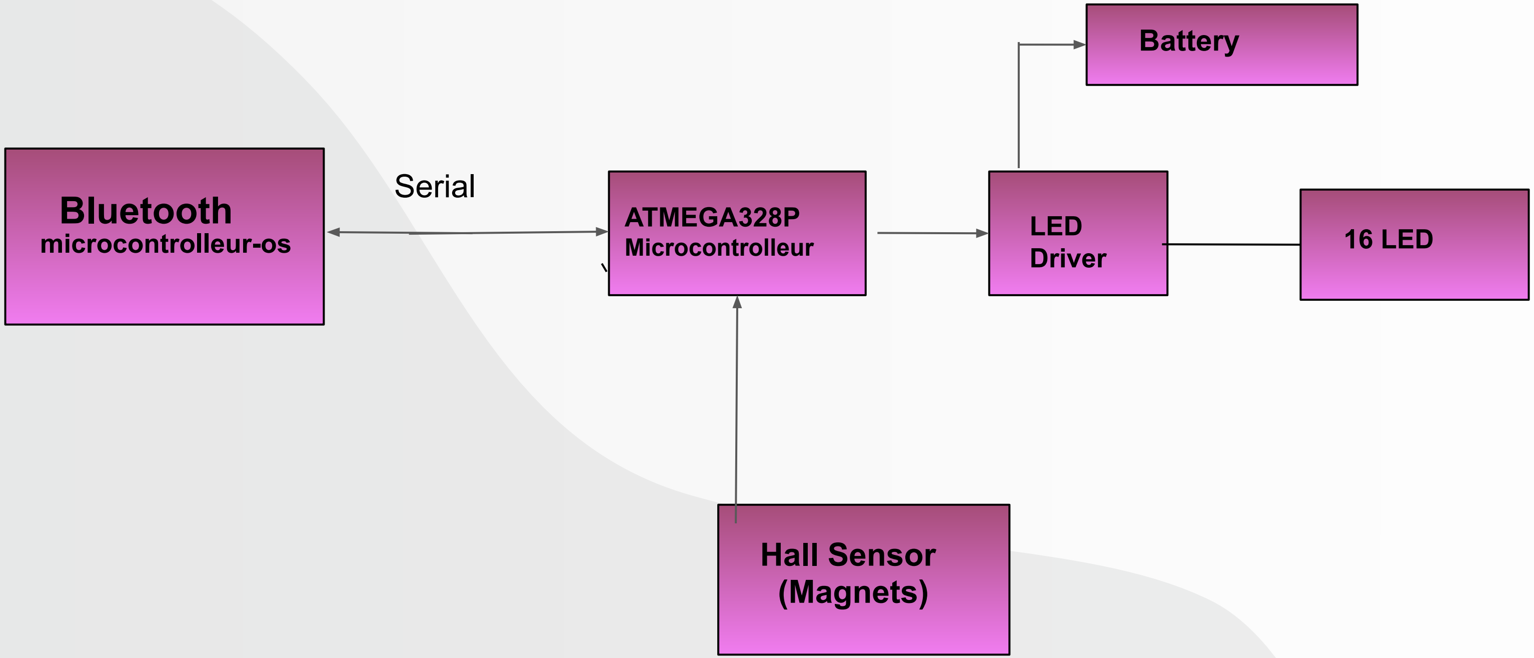
03 **Digital Clock**

04 **Conclusion**

Prospect for improvement

05 **Demonstration**

Projet architecture



GOAL

☑

☑

☒

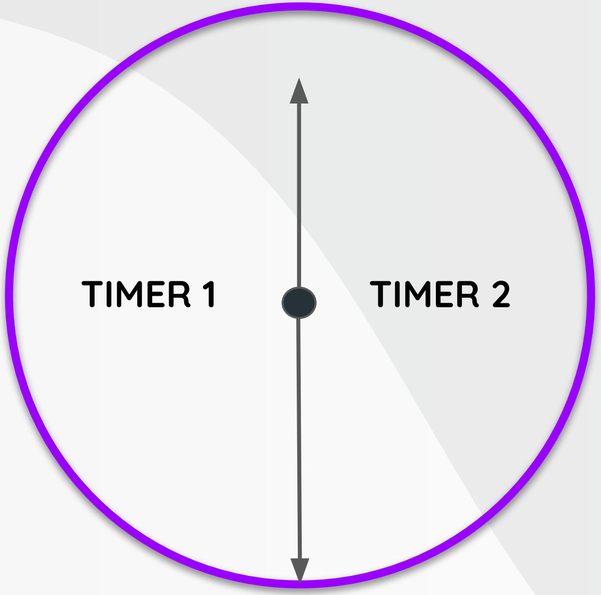
02

Analog Clock

TIMERS

Timer to count 1 second

- On 8 bits
- Reset Value = **250**
- Prescaler of **64**



Timer to count time of a round/positions

- On 16 bits
- No prescaler
- From 250 to 0

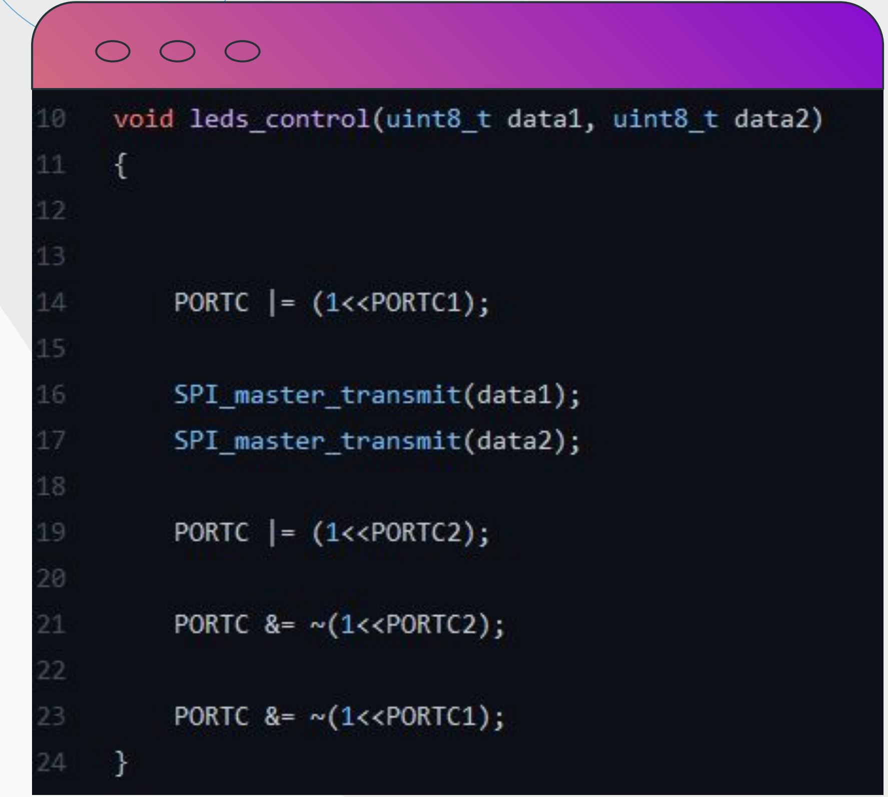
Handling leds

SPI bus : 16 leds +Driver (Master)

STEPS :

- Initialising master
- Transferring data

Controlling leds

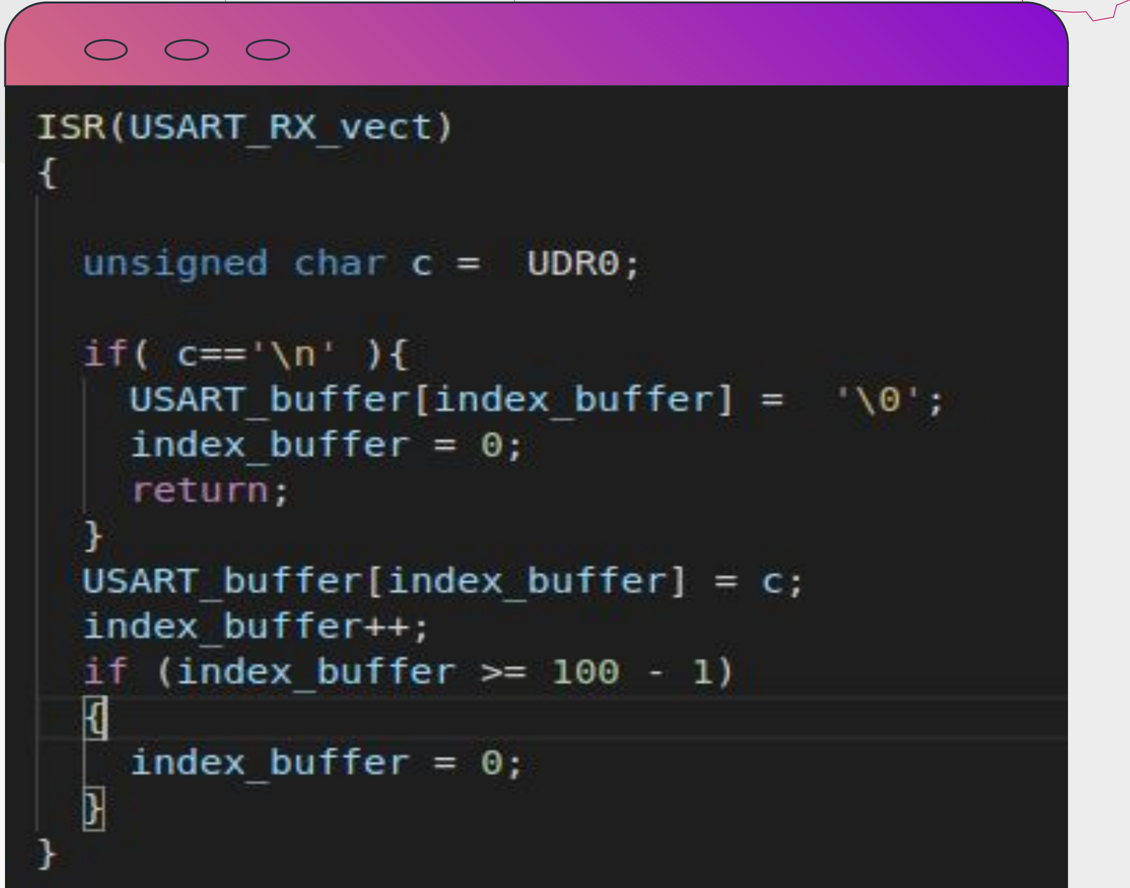


```
void leds_control(uint8_t data1, uint8_t data2)
{


    PORTC |= (1<<PORTC1);

    SPI_master_transmit(data1);
    SPI_master_transmit(data2);

    PORTC |= (1<<PORTC2);

    PORTC &= ~(1<<PORTC2);

    PORTC &= ~(1<<PORTC1);
}
```

Bluetooth/USART communication

- **BUFFER** Used to store data received

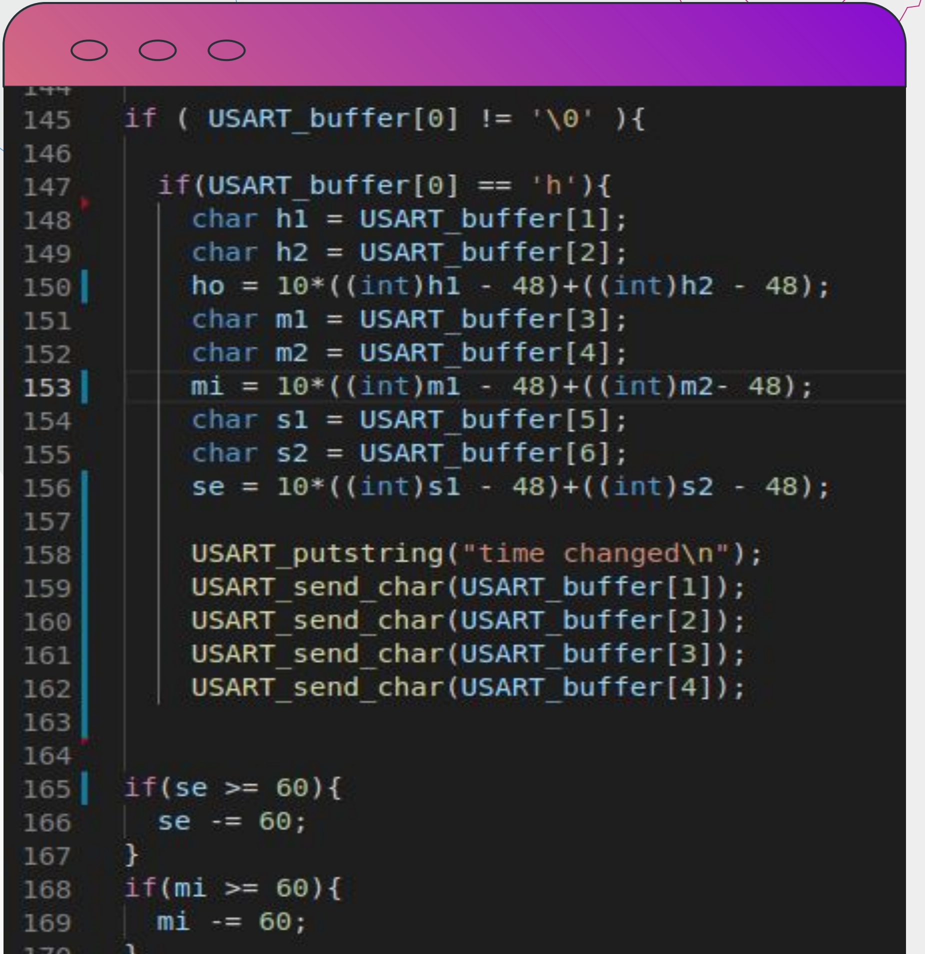
- **Global variable** to browse the buffer



```
ISR(USART_RX_vect)
{

   unsigned char c =  UDR0;

   if( c=='\n' ){
      USART_buffer[index_buffer] =  '\0';
      index_buffer = 0;
      return;
   }
   USART_buffer[index_buffer] = c;
   index_buffer++;
   if (index_buffer >= 100 - 1)
   {
      index_buffer = 0;
   }
}
```

Setup/Change Time

- Verify if buffer not empty
- Test if the command is valid
- Convert from ASCII to int(Hours, minutes, seconds)
- Send message to user



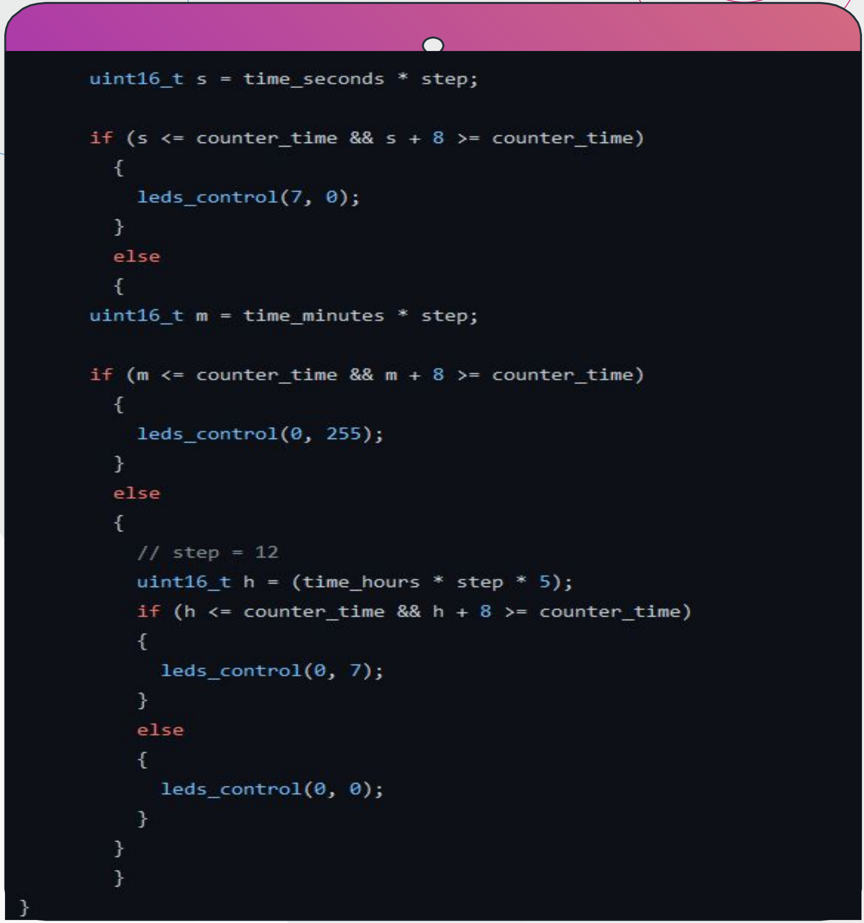
```
if ( USART_buffer[0] != '\0' ){

  if(USART_buffer[0] == 'h'){
    char h1 = USART_buffer[1];
    char h2 = USART_buffer[2];
    ho = 10*((int)h1 - 48)+((int)h2 - 48);
    char m1 = USART_buffer[3];
    char m2 = USART_buffer[4];
    mi = 10*((int)m1 - 48)+((int)m2- 48);
    char s1 = USART_buffer[5];
    char s2 = USART_buffer[6];
    se = 10*((int)s1 - 48)+((int)s2 - 48);

    USART_putstring("time changed\n");
    USART_send_char(USART_buffer[1]);
    USART_send_char(USART_buffer[2]);
    USART_send_char(USART_buffer[3]);
    USART_send_char(USART_buffer[4]);


if(se >= 60){
  se -= 60;
}
if(mi >= 60){
  mi -= 60;
}
```

Loop

Calculate the time to make a clock round (hall detect)

Step = Time / 60

Actual position of POV by comparing counter_time to 2 values



```
      uint16_t s = time_seconds * step;

      if (s <= counter_time && s + 8 >= counter_time)
        {
          leds_control(7, 0);
        }
        else
        {
      uint16_t m = time_minutes * step;

      if (m <= counter_time && m + 8 >= counter_time)
        {
          leds_control(0, 255);
        }
        else
        {
          // step = 12
          uint16_t h = (time_hours * step * 5);
          if (h <= counter_time && h + 8 >= counter_time)
          {
            leds_control(0, 7);
          }
          else
          {
            leds_control(0, 0);
          }
        }
        }
}
```

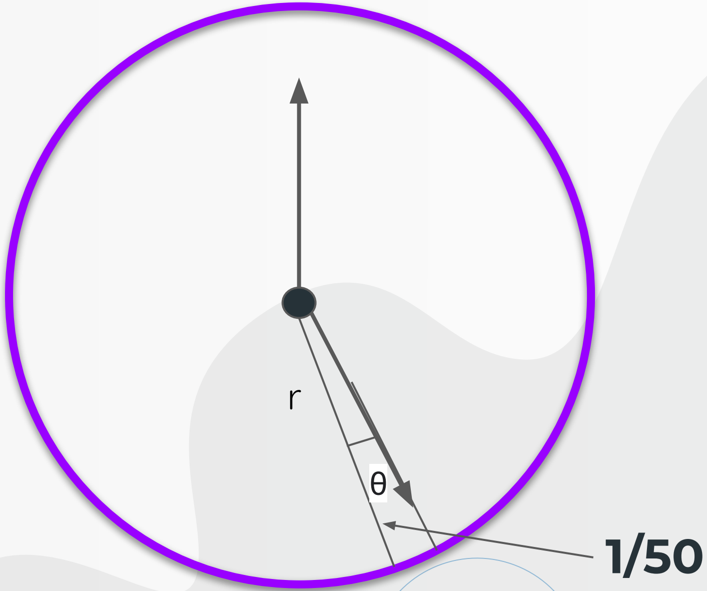
03

DIGITAL CLOCK

Roughly 50 ms by clock turn

First idea :

- Divide the circle into 50 sections of 1ms
- Display on the 8 first DEL
- Can display until 10 letters



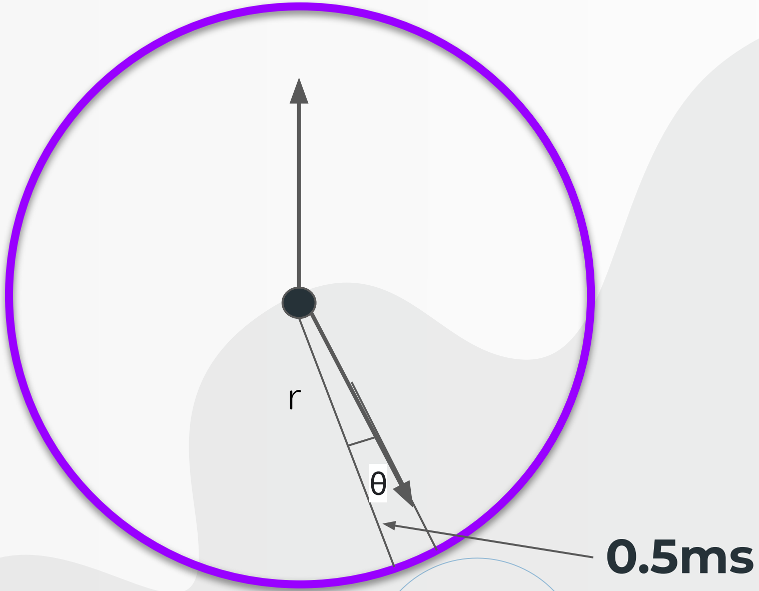
Issues :

- Forgot the spaces
- Letters too wide
- Letters too close from center

Roughly 50 ms by clock turn

Second idea :

- Column displaying on 0.5ms
- Spaces noticed by pauses of 1ms
- Display on the 8 last DEL
- Can display far more characters

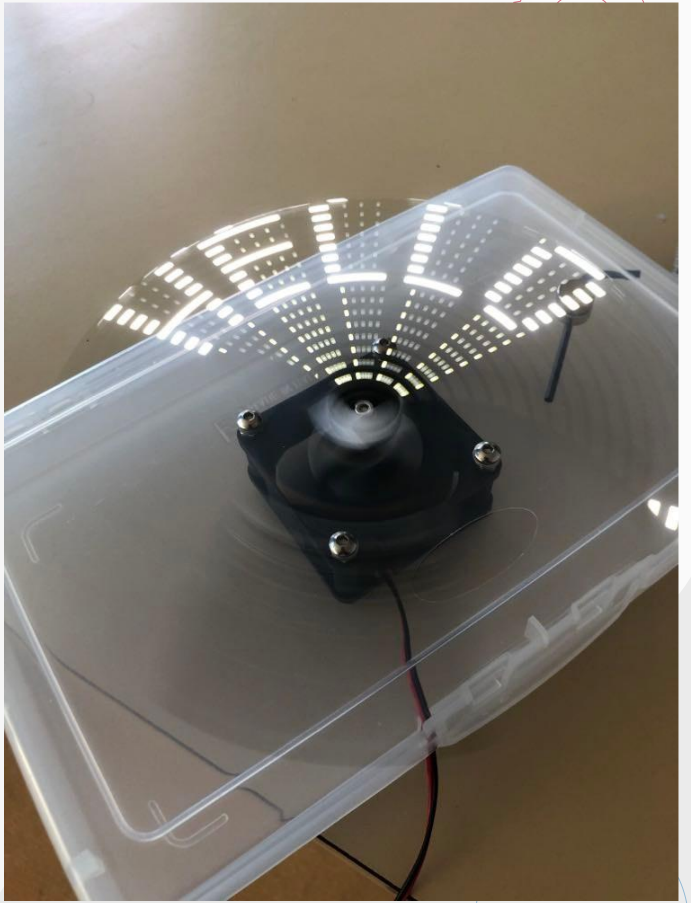


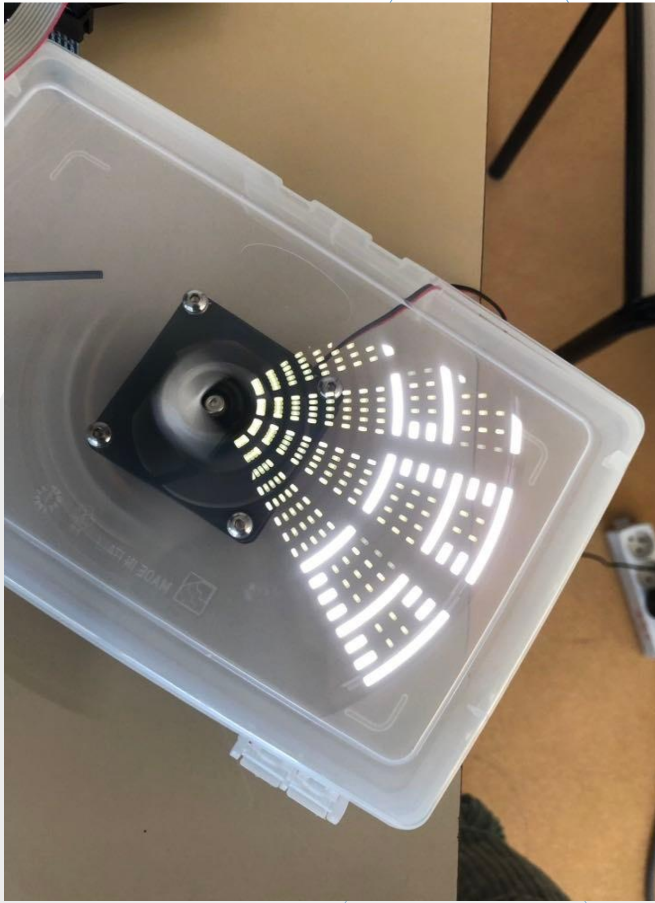
Issues Noticed :

- Find a pattern to display reversed text
- Use of 4 to 12 DEL to center displaying

Reversed text issues

Light Saber effect on purpose :-)

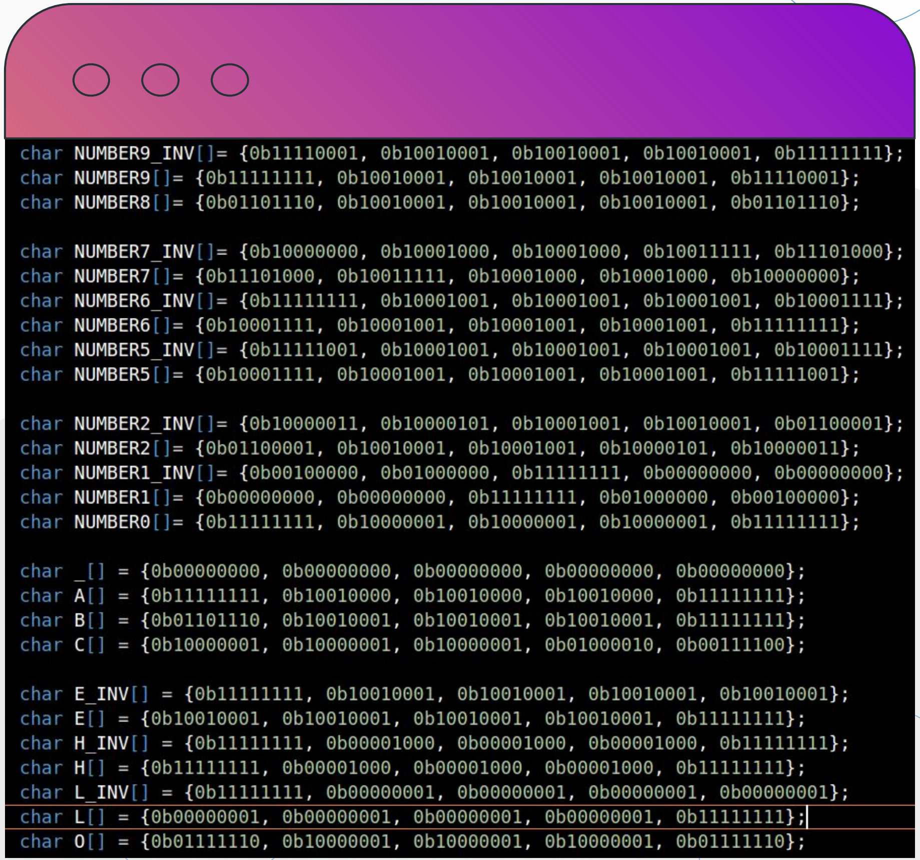




Characters representation

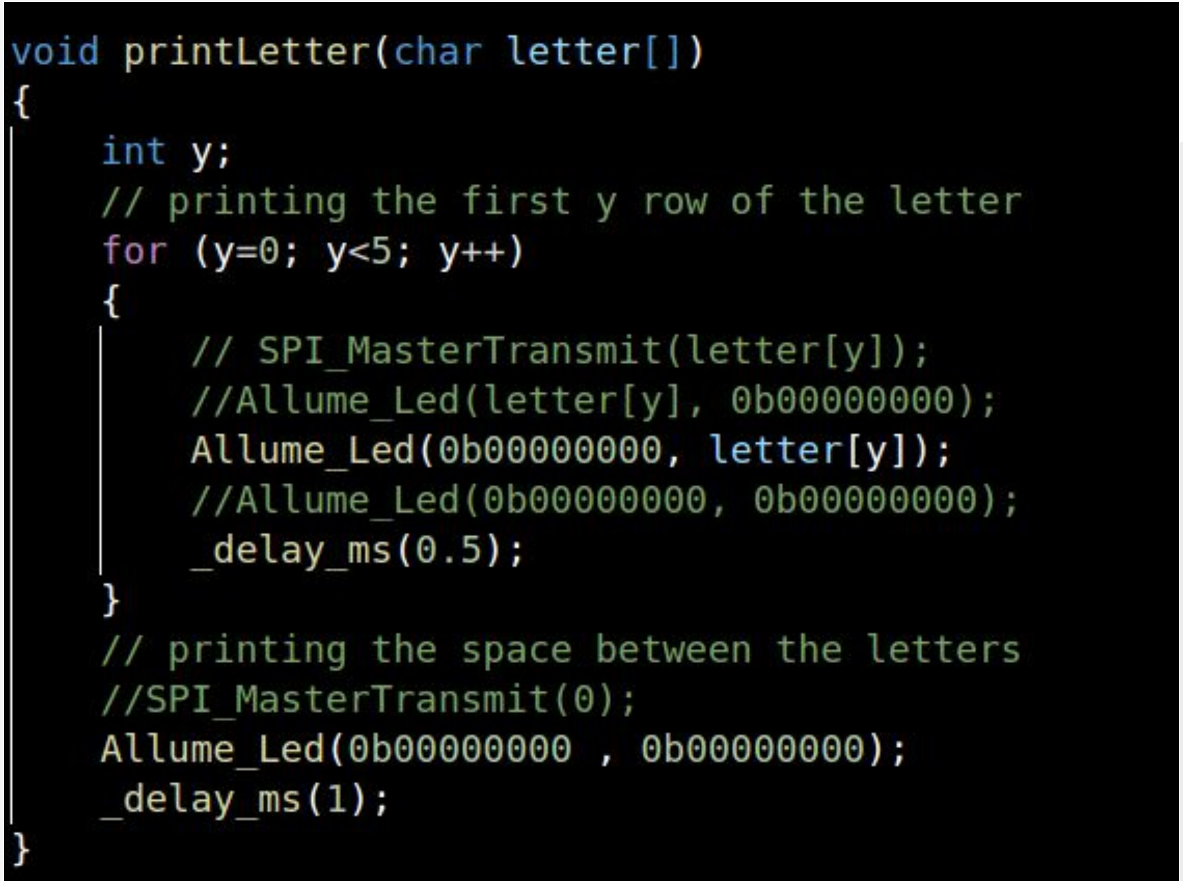
Info :

- Characters tab
- 5 columns display



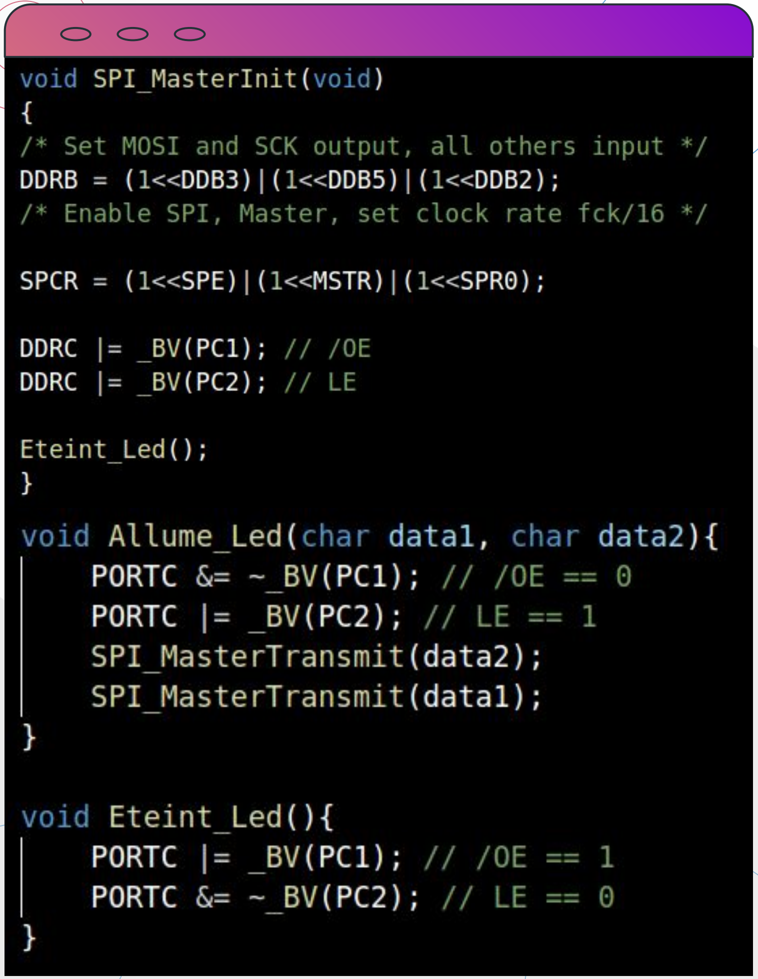
```
char NUMBER9_INV[]= {0b11110001, 0b10010001, 0b10010001, 0b10010001, 0b11111111};
char NUMBER9[]= {0b11111111, 0b10010001, 0b10010001, 0b10010001, 0b11110001};
char NUMBER8[]= {0b01101110, 0b10010001, 0b10010001, 0b10010001, 0b01101110};

char NUMBER7_INV[]= {0b10000000, 0b10001000, 0b10001000, 0b10011111, 0b11101000};
char NUMBER7[]= {0b11101000, 0b10011111, 0b10001000, 0b10001000, 0b10000000};
char NUMBER6_INV[]= {0b11111111, 0b10001001, 0b10001001, 0b10001001, 0b10001111};
char NUMBER6[]= {0b10001111, 0b10001001, 0b10001001, 0b10001001, 0b11111111};
char NUMBER5_INV[]= {0b11111001, 0b10001001, 0b10001001, 0b10001001, 0b10001111};
char NUMBER5[]= {0b10001111, 0b10001001, 0b10001001, 0b10001001, 0b11111001};

char NUMBER2_INV[]= {0b10000011, 0b10000101, 0b10001001, 0b10010001, 0b01100001};
char NUMBER2[]= {0b01100001, 0b10010001, 0b10001001, 0b10000101, 0b10000011};
char NUMBER1_INV[]= {0b00100000, 0b01000000, 0b11111111, 0b00000000, 0b00000000};
char NUMBER1[]= {0b00000000, 0b00000000, 0b11111111, 0b01000000, 0b00100000};
char NUMBER0[]= {0b11111111, 0b10000001, 0b10000001, 0b10000001, 0b11111111};

char _[] = {0b00000000, 0b00000000, 0b00000000, 0b00000000, 0b00000000};
char A[] = {0b11111111, 0b10010000, 0b10010000, 0b10010000, 0b11111111};
char B[] = {0b01101110, 0b10010001, 0b10010001, 0b10010001, 0b11111111};
char C[] = {0b10000001, 0b10000001, 0b10000001, 0b01000010, 0b00111100};

char E_INV[] = {0b11111111, 0b10010001, 0b10010001, 0b10010001, 0b10010001};
char E[] = {0b10010001, 0b10010001, 0b10010001, 0b10010001, 0b11111111};
char H_INV[] = {0b11111111, 0b00001000, 0b00001000, 0b00001000, 0b11111111};
char H[] = {0b11111111, 0b00001000, 0b00001000, 0b00001000, 0b11111111};
char L_INV[] = {0b11111111, 0b00000001, 0b00000001, 0b00000001, 0b00000001};
char L[] = {0b00000001, 0b00000001, 0b00000001, 0b00000001, 0b11111111};
char O[] = {0b01111110, 0b10000001, 0b10000001, 0b10000001, 0b01111110};
```

PrintLetter function



```
void printLetter(char letter[])
{
    int y;
    // printing the first y row of the letter
    for (y=0; y<5; y++)
    {
        // SPI_MasterTransmit(letter[y]);
        //Allume_Led(letter[y], 0b00000000);
        Allume_Led(0b00000000, letter[y]);
        //Allume_Led(0b00000000, 0b00000000);
        _delay_ms(0.5);
    }
    // printing the space between the letters
    //SPI_MasterTransmit(0);
    Allume_Led(0b00000000 , 0b00000000);
    _delay_ms(1);
}
```

SPI functions

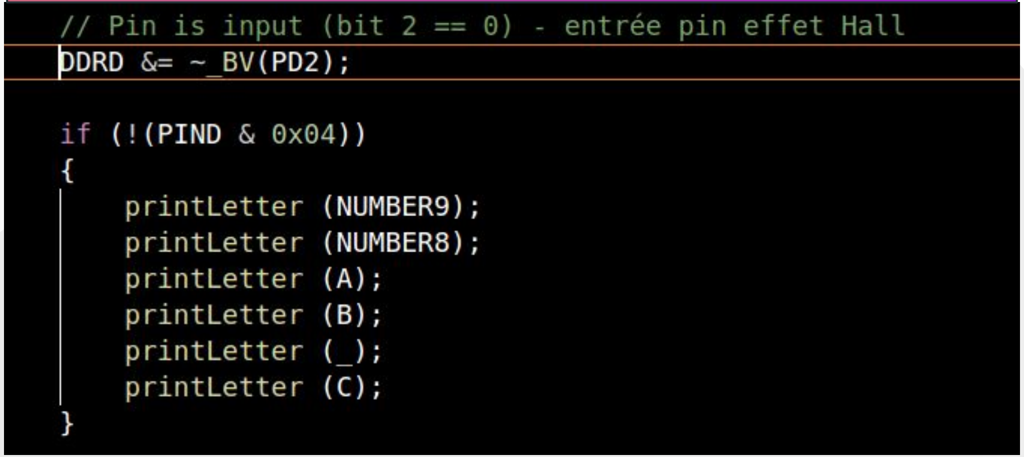


```
void SPI_MasterInit(void)
{
/* Set MOSI and SCK output, all others input */
DDRB = (1<<DDB3)|(1<<DDB5)|(1<<DDB2);
/* Enable SPI, Master, set clock rate fck/16 */

SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);

DDRC |= _BV(PC1); // /OE
DDRC |= _BV(PC2); // LE

Eteint_Led();
}

void Allume_Led(char data1, char data2){
    PORTC &= ~_BV(PC1); // /OE == 0
    PORTC |= _BV(PC2); // LE == 1
    SPI_MasterTransmit(data2);
    SPI_MasterTransmit(data1);
}

void Eteint_Led(){
    PORTC |= _BV(PC1); // /OE == 1
    PORTC &= ~_BV(PC2); // LE == 0
}
```

Hall effect and characters displaying

Pins :

- Initialize Hall sensor as input
- Check on magnet presence



```
// Pin is input (bit 2 == 0) - entrée pin effet Hall
DDRD &= ~_BV(PD2);

if (!(PIND & 0x04))
{
    printLetter (NUMBER9);
    printLetter (NUMBER8);
    printLetter (A);
    printLetter (B);
    printLetter (_);
    printLetter (C);
}
```

04

Conclusion

Conclusion & Propect for improvement

Skill gain

Understanding datasheet, coding in C and electronic notions

Analog Clock & Bluetooth

- Add divisons for the analog clock
- Improve the bluetooth commands
- Have two hands that light up at the same time

Digital clock

Implement the second digital clock ⇒

Make a function taking in input a pixelized image and a led buffet that we cross to display

Pass coordinate from cartesian to polar

Benchmark

the execution time of the functions, the global memory required, the evaluation o the efficiency of our programs

05

Let’s demonstrate