

Sistema de Sinalização para Ciclistas

Karine Valença

Engenharia de Software

Universidade de Brasília, FGA

Gama, Brasil

valenca.karine@gmail.com

Wilton Rodrigues

Engenharia de Software

Universidade de Brasília, FGA

Gama, Brasil

wiltonsr94@gmail.com

Abstract—A fim de diminuir a quantidade de acidentes envolvendo ciclistas nas vias brasileiras foi proposto o objetivo de desenvolver um sistema de sinalização luminosa para permitir aos ciclistas indicarem suas intenções para os demais participantes do trânsito através de uma matriz luminosa. Utilizando o microcontrolador MSP430 da Texas Instruments foi possível atingir o objetivo proposto.

Index Terms—msp430, matriz de led, max7219, microcontrolador, ciclista, sinalização

I. INTRODUÇÃO

A. Revisão Bibliográfica

Notícias sobre acidentes envolvendo bicicletas são comuns no Brasil. Recentemente, em São Paulo, um ciclista morreu logo após ser atropelado e arrastado [1]. Dados de 2014, mostram que 1.357 ciclistas morreram vítimas de acidentes de trânsito no Brasil, além disso, em 2016, ocorreram 11.741 internações de ciclistas vítimas de acidentes [2]. De acordo com Departamento Nacional de Infraestrutura de Transportes (DNIT) [3] só no ano de 2011 foram 1.698 casos de acidentes envolvendo ciclistas. Sendo que 246, equivalente a 14.5%, acabaram na morte.

O site hg.org apresenta uma lista de dicas para evitar acidentes ao utilizar bicicleta. O site sugere aos ciclistas que eles se façam visíveis aos demais usuários das vias, e que utilizem sinais de mão para mostrar intenção de parar ou de mudar de faixa [4].

Existe uma série de sinais que podem ser utilizados pelos ciclistas para indicar suas intenções. O site mapmyrun [5], apresenta um lista com 10 sinais que podem ser utilizados a fim de evitar acidentes. Pode-se notar que, de fato, os sinais auxiliam a diminuir os acidentes de trânsito envolvendo ciclistas. Porém, alguns desses sinais não são tão intuitivos e podem não fazer sentido para os motoristas. Além disso, a grande quantidade de sinais pode gerar confusão até mesmo aos ciclistas.

B. Justificativa

Pode-se notar que a visibilidade e sinalização por parte dos ciclistas é crucial para sua segurança no trânsito. Diante disso, este projeto tem como objetivo a criação de um sistema de sinalização eletrônico visando aumentar a segurança dos ciclistas. Espera-se que os usuários do sistema de sinalização eletrônico sofram menos acidentes causados por falta de visibilidade.

C. Objetivos

O objetivo do projeto é desenvolver um sistema de sinalização, utilizando o MSP430, a fim de aumentar a visibilidade dos ciclistas durante seu trajeto para aumentar a segurança e confiança dos utilizadores deste meio de transporte.

D. Requisitos

O sistema deve atender aos requisitos:

- Indicar sinal luminoso intermitente que fica ativo sempre que não houver outro sinal
- Indicar seta para a direita ou para a esquerda após clique do botão correspondente
- Indicar sobre parada quando o ciclista iniciar a freagem

O sistema não atende aos requisitos:

- Funcionar em dias chuvosos
- Possuir fonte de energia própria

E. Benefícios

O sistema proporciona um equipamento de sinalização que ajuda os demais condutores a ter uma melhor visão dos ciclistas. Baseado nisto, o principal benefício do sistema é a diminuição de ocorrências de acidentes envolvendo ciclistas.

II. DESCRIÇÃO DO HARDWARE

A. Lista de Materiais

Os materiais utilizados para a construção do Sistema de Sinalização para Ciclistas, foram:

- 1 MSP430 LaunchPad

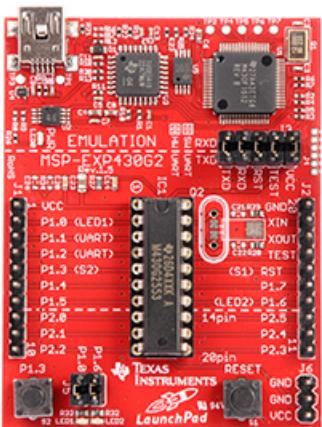


Fig. 1. MSP430 LaunchPad. Fonte: <http://e2e.ti.com/>

- 1 Matriz de LED 8x8

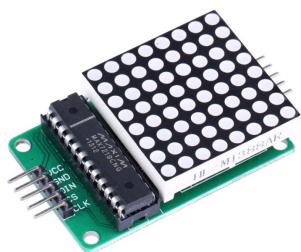


Fig. 2. Matriz de LED 8x8. Fonte: <http://www.huinfinito.com.br>

- 2 Protoboard

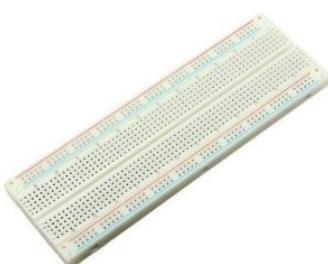


Fig. 3. Protoboard. Fonte: www.filipeflop.com/

- Jumpers Macho-Macho e Macho-Fêmea

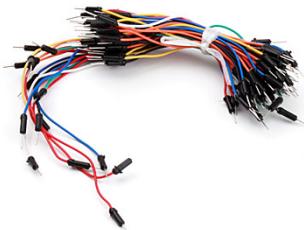


Fig. 4. Jumpers. Fonte: <http://www.msselettronica.com>

- 1 chave on-off-on

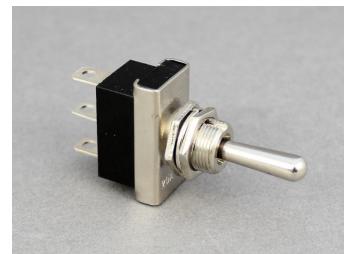


Fig. 5. Chave on-off-on. Fonte: <http://www.12voltplanet.co.uk/>

- 2 chaves push-botton sem trava



Fig. 6. Chave push-botton. Fonte: <http://www.huinfinito.com.br>

B. Verificação dos componentes

Com o intuito de verificar se os componentes estavam funcionando conforme o esperado, forem feitos alguns teste simples de funcionamento.

1) Verificação da Matriz de Led: Para verificar se a matriz de led estava funcionando corretamente, foi ligada uma tensão de aproximadamente 3.3 volts no VCC e no DIN, e no GND, uma tensão de 0 volts.

Esquemático: A figura abaixo mostra o esquemático montado para o teste da matriz de led:

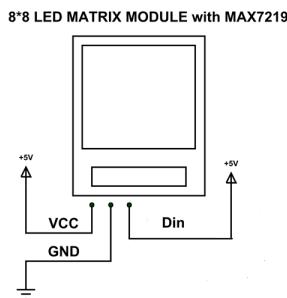


Fig. 7. Teste realizado com a matriz de led. Fonte: Autores

Demonstração: A figura abaixo mostra o teste realizado com a matriz de led:

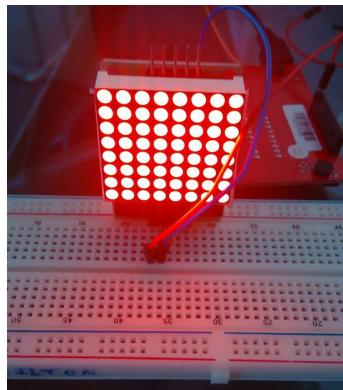


Fig. 8. Teste realizado com a matriz de led. Fonte: Autores

2) *Verificação da Chave on-off-on:* Para verificar se a chave on-off-on estava funcionando corretamente, foi ligada uma tensão de aproximadamente 3.3 volts nos pinos 1 e 3 do botão. O pino 2 funcionou como saída e foi ligado em um resistor de 1000 ohm, que estava ligado em série a um Led.

Demonstração: A figura abaixo mostra o teste realizado com a chave on-off-on:

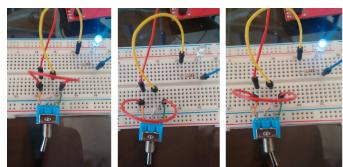


Fig. 9. Teste realizado com a chave on-off-on. Fonte: Autores

3) *Verificação da Chave Push-Botton:* Para verificar se a chave push-button estava funcionando corretamente, foi ligada uma tensão de aproximadamente 3.3 volts em um resistor de 1000 ohm, que estava ligado em série ao botão. O botão, por sua vez, estava ligado em série a um led. Dessa forma, era esperado que ao pressionar o botão, o led acendesse, e ao soltar o botão, o led apagasse.

Esquemático: A figura abaixo mostra o esquemático montado para o teste da chave push-button:

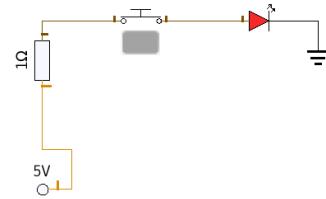


Fig. 10. Teste realizado com o Push Button. Fonte: Autores

Demonstração: A figura abaixo mostra o teste realizado com a chave push-button:

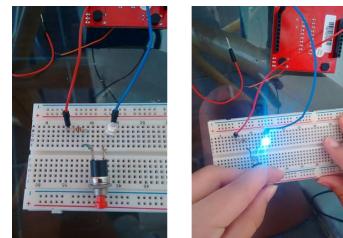


Fig. 11. Teste realizado com a chave push-button. Fonte: Autores

C. Montagem

Para auxiliar no procedimento de montagem, foi feito um esquemático utilizando a ferramenta Fritzing¹. A figura 12 apresenta o diagrama simplificado com os componentes de hardware que são utilizados no sistema.

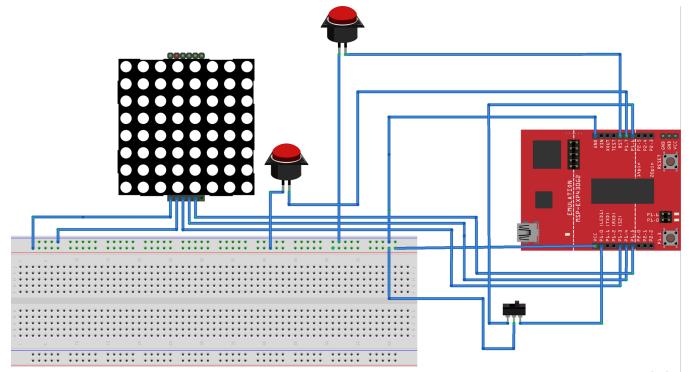


Fig. 12. Esquemático do hardware. Fonte: Autores

A matriz de led, que possui o multiplexador MAX7219, tem seu primeiro pino, VCC, ligado na entrada de energia de 3,3 Volts. O segundo pino, GND, é ligado no aterramento. O terceiro pino, DIN, que corresponde ao DATA IN, é ligado no BIT3 do MSP. O quarto pino, CS, é ligado no BIT4 do MSP. E finalmente o quinto pino, CLK, que corresponde ao CLOCK, é ligado no BIT5 do MSP.

A chave on-off-on tem o pino central ligado no VCC. O pino da direita é ligado no BIT0 do MSP e o pino da esquerda é ligado no BIT6 do MSP.

¹<http://fritzing.org/home/>

Uma chave push-button tem um de seus pinos ligados no VCC, enquanto o outro pino fica ligado no BIT7 do MSP. A outra chave push-button tem um de seus pinos ligados no GND e o outro no pino RST.

III. DESCRIÇÃO DO SOFTWARE

A. Diagrama UML

A figura 13 apresenta o diagrama UML do sistema. O sistema consiste de 3 arquivos, o arquivo main e 2 bibliotecas. O main realiza a chamada das funções das bibliotecas, realizando a execução do sistema em si. A biblioteca Desenho contém os desenhos que são exibidos pela matriz 8x8. A biblioteca MAX7219 contém todas as funções relacionadas à manipulação do multiplexador MAX7219.

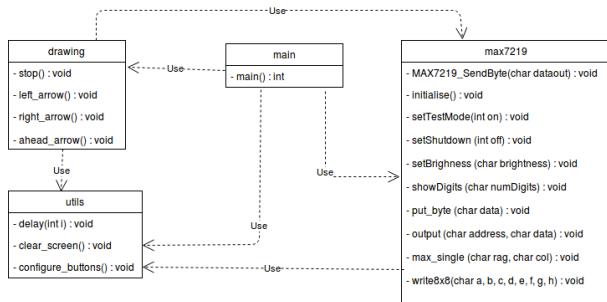


Fig. 13. Diagrama UML do sistema. Fonte: Autores

B. Descrição do funcionamento

1) *Max7219.h*: Nesta biblioteca estão as funções de comunicação dos dados enviados pela placa MSP430, o multiplexador MAX7219 e a matriz de led.

```

void initialize(){
    Leve o pino CS para nível alto;
    Sete os pinos CS, DIN, CLK para saída;
}
  
```

```

void shift_out(entrada, clock, valor){
    Itere nos 8 bits da entrada;
    Se a entrada != 0
        Leve o valor da entrada para a matriz;
    Se não
        Limpe o valor da entrada da matriz;
    Sete o clock para nível alto;
    Sete o clock para nível baixo;
}
  
```

```

void put_byte(dado){
    Enquanto i > 0{
        Gere a máscara de ativação do led;
        Leve o CLK para o nível baixo;
        Se dado & máscara
            Leve o DIN para o nível alto;
        Se não
            Leve o DIN para o nível baixo;
}
  
```

```

        Leve o CLK para o nível alto;
        Decremento o i;
    }
}
  
```

```

void max_single(registrador, coluna){
    Leve o estado de CS para nível baixo;
    Execute put_byte(registrador);
    Execute put_byte(coluna);
    Leve o estado de CS para nível alto;
    Leve o estado de CS para nível baixo;
}
  
```

```

void write8x8(a, b, c, d, e, f, g, h){
    Execute max_single(a);
    Execute max_single(b);
    Execute max_single(c);
    Execute max_single(d);
    Execute max_single(e);
    Execute max_single(f);
    Execute max_single(g);
}
  
```

2) *Desenho.h*: A biblioteca Desenho.h é a responsável pelo apoio na criação de funções para a exibição de imagens na matriz de led 8x8. Nesta biblioteca estão as funções right_arrow(), left_arrow(), ahead_arrow() e stop(). O funcionamento básico de todas as funções desta biblioteca é o seguinte:

```

void left_arrow(){
    Executa write8x8(8 valores hexadecimais);
}
  
```

3) *Utils.h*: A biblioteca Utils.h é onde armazenamos as funções de uso geral do sistema. Lá se encontram as funções delay(), clear_screen() e configure_buttons();

```

void configure_buttons(){
    Configura todos os botões para entrada;
}
  
```

```

void delay(dado){
    Itere ate de dado ate zero;
}
  
```

```

void clear_screen(dado){
    Executa write8x8(0x00 em todos os leds);
}
  
```

IV. RESULTADOS

Durante o desenvolvimento do sistema, foram feitos vários testes para verificar o funcionamento correto do sistema. Os primeiros testes foram feitos apenas com a matriz de led, chamando as funções de desenho e verificando se o desenho estava correto. As figuras a seguir ilustram o funcionamento de cada uma das funções baseado no funcionamento descrito.

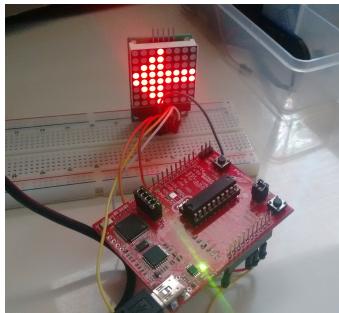


Fig. 14. Função left_arrow() em funcionamento. Fonte: Autores

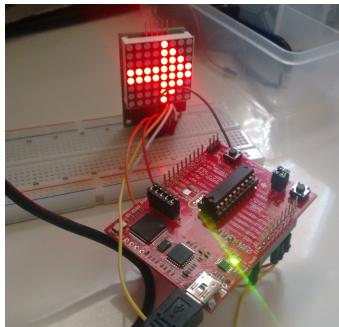


Fig. 15. Função right_arrow() em funcionamento. Fonte: Autores

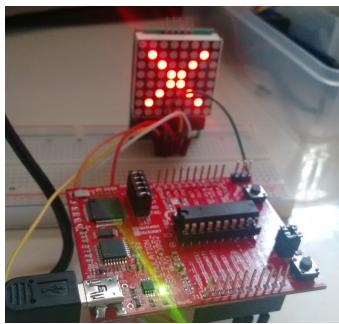


Fig. 16. Função stop() em funcionamento. Fonte: Autores

Logo após conseguir desenhar corretamente as imagens na matriz de led, os botões foram adicionados às portas do MSP e foi verificado se o acionamento do botão chamava a função correta.

Com o passo anterior concluído, o próximo passo foi passar o sistema para a bicicleta. A matriz de led foi fixada atrás banco, a chave on-off-on foi fixada do lado direito do guidão e foi feito um sensor de contato, utilizando alumínio, que ficou dentro dos freios. Então, foi feita a ligação conforme a figura 12.

O MSP foi ligado em um *Power Bank*. Este *Power Bank* tem capacidade de fornecer energia para todo o sistema por aproximadamente 8h.

Todo o sistema funcionou conforme os requisitos definidos, uma vez que a matriz indica o sinal intermitente sempre que não há outros sinais. Quando ocorre a freagem, a matriz de led

mostra um sinal de parada e quando o ciclista utiliza a seta, para direita ou esquerda, a matriz mostra o sinal adequado.

V. CONCLUSÃO

A quantidade de ciclista está cada vez maior, porém ainda não há respeito suficiente aos ciclistas. Por conta disso, inúmeros acidentes são contabilizados todos os anos. Muitos ciclistas investem caro em equipamentos para melhorar sua visibilidade enquanto pedalam. Estes equipamentos ajudam a reduzir os acidentes de trânsito.

O sistema de sinalização para ciclistas, mostrado neste relatório, auxilia os ciclistas a terem maior visibilidade no trânsito. Além disso, ele também ajuda o ciclista a indicar melhor suas intenções, por meio de sinais luminosos de parada e de direção. Desta forma, é possível que os motoristas consigam entender e visualizar com mais clareza qual é a intenção dos ciclistas. Os sinais luminosos mostrados por este sistema são intuitivos e seguem padrões de trânsito.

Os objetivos definidos no trabalho foram atingidos. Considera-se que o projeto tem potencial comercial, porém são necessários testes de validação no trânsito, para confirmar se os ciclistas apreciam a ideia e se os motoristas conseguem visualizar melhor os ciclistas.

Para trabalhos futuros, é possível a adição de novos sinais à matriz de led e também sinais sonoros para avisar a aproximação de carros.

REFERENCES

- [1] G1, “Ciclista morre após ser atropelado e arrastado em SP”. Disponível em: <http://g1.globo.com/sao-paulo/noticia/ciclista-morre-apos-ser-atropelado-e-arrastado-em-sp.ghtml>.
- [2] G1, “Brasil tem, em média, 32 ciclistas internados por dia devido a acidentes”. Disponível em: <http://g1.globo.com/bom-dia-brasil/noticia/2017/03/brasil-tem-em-media-32-ciclistas-internados-por-dia-devido-acidentes.html>.
- [3] DNIT, “NÚMERO DE VITIMADOS ENVOLVIDOS POR TIPO DE USUÁRIO”, 2011.
- [4] Mesriani Law Group, “Safety Tips to Avoid Bicycle Accidents”. Disponível em: <https://www.hg.org/article.asp?id=7752>.
- [5] Marc Lindsay, “10 Cycling Hand Signals You Need to Know”. Disponível em: <http://blog.mapmyrun.com/10-cycling-hand-signals-need-know/>.

1 Anexos

1.1 Main

```
#include "max7219.h"
#include "utils.h"
#include "drawing.h"
#include <msp430g2553.h>
#include <legacymsp430.h> // Para rodar interrupcoes

/*
   MSP430G2553
   -----
   | VCC          GND|
RIGHT_BTN-->|P1.0          XIN|
               | P1.1          XOUT|
               | P1.2          TST|
DIN<--|P1.3          RST|
CS<--|P1.4          P1.7<--STOP_BTN
CLK<--|P1.5          P1.6<--LEFT_BTN
               | P2.0          P2.5|
               | P2.1          P2.4|
               | P2.2          P2.3|
   -----
*/
int main(){
    WDTCTL = WDTPW + WDTHOLD;      // Desabilita WDT
    DCOCTL = CALDCO_1MHZ;         // 1 Mhz DCO
    BCSCTL1 = CALBC1_1MHZ;

    initialise();
    setTestMode(0);
    setShutdown(0);
    setBrightness(0xff);
    showDigits(8);

    configure_buttons();

    __enable_interrupt();
}
```

```

    clear_screen();

    while(1){
        ahead_arrow();
    }

    return 0;
}

#pragma vector=PORT1_VECTOR
_interrupt void Port_1(void){
    if(P1IFG & RIGHT_BTN){
        while (!(P1IN & RIGHT_BTN)==0){
            right_arrow();
        }
        P1IFG &= ~RIGHT_BTN;
    }
    if(P1IFG & LEFT_BTN){
        while (!(P1IN & LEFT_BTN)==0){
            left_arrow();
        }
        P1IFG &= ~LEFT_BTN;
    }
    if(P1IFG & STOP_BTN){
        while((P1IN & STOP_BTN)==0){
            delay(20);
            stop();
        }
        P1IFG &= ~STOP_BTN;
    }
    return 0;
}

```

1.2 MAX7219

```

#include "max7219.h"
#include "utils.h"
#include <msp430g2553.h>

#define MAX7219_DIN BIT3
#define MAX7219_CS BIT4

```

```

#define MAX7219_CLK BIT5

static void MAX7219_SendByte (unsigned char dataout)
{
    char i;
    for (i=8; i>0; i--) {
        unsigned char mask = 1 << (i - 1);
        P1OUT &= ~(MAX7219_CLK);
        if (dataout & mask)
            P1OUT |= MAX7219_DIN;
        else
            P1OUT &= ~(MAX7219_DIN);
        P1OUT |= MAX7219_CLK;
    }
}

void initialise(){
    P1OUT |= MAX7219_CS;

    P1DIR |= MAX7219_DIN;

    P1DIR |= MAX7219_CS;

    P1DIR |= MAX7219_CLK;

    output(0x0b, 7);
    output(0x09, 0x00);
}

void output(char address, char data){
    P1OUT |= MAX7219_CS;
    MAX7219_SendByte(address);
    MAX7219_SendByte(data);
    P1OUT &= ~(MAX7219_CS);
    P1OUT |= MAX7219_CS;
}

void setTestMode(int on){
    output(0x0f, on ? 0x01 : 0x00);
}

```

```

void setShutdown(int off){
    output(0x0c, off ? 0x00 : 0x01);
}

void showDigits(char numDigits){
    output(0x0b, numDigits - 1);
}

void setBrightness(char brightness){
    output(0x0a, brightness);
}

void put_byte(char data) {
    char i = 8;
    char mask;
    while(i > 0) {
        mask = 0x01 << (i - 1);
        P1OUT &= ~(MAX7219_CLK);
        if (data & mask){
            P1OUT |= MAX7219_DIN;
        } else{
            P1OUT &= ~(MAX7219_DIN);
        }
        P1OUT |= MAX7219_CLK;
        —i;
    }
}

void max_single(char reg, char col) {
    P1OUT &= ~(MAX7219_CS);
    put_byte(reg);
    //asm("mov.w reg, R15");
    //asm("call #putByte");
    //asm("pop R15");
    put_byte(col);
    P1OUT &= ~(MAX7219_CS);
    P1OUT |= (MAX7219_CS);
}

void write8x8(char a, char b, char c, char d, char e, char f, char g, char

```

```

max_single(1,a);
max_single(2,b);
max_single(3,c);
max_single(4,d);
max_single(5,e);
max_single(6,f);
max_single(7,g);
max_single(8,h);
delay(5000);
}

1.3 Drawing

#include "drawing.h"
#include "utils.h"
#include "max7219.h"

void stop(){
    write8x8(0x0,0x42,0x24,0x18,0x18,0x24,0x42,0x0);
    delay(10000);
    write8x8(0xff,0xbd,0xdb,0xe7,0xe7,0xdb,0xbd,0xff);
    delay(10000);
}

void left_arrow(){
    write8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    write8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x18);
    write8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x18,0x3c);
    write8x8(0x0,0x0,0x0,0x0,0x0,0x18,0x3c,0x7e);
    write8x8(0x0,0x0,0x0,0x0,0x18,0x3c,0x7e,0xff);
    write8x8(0x0,0x0,0x0,0x18,0x3c,0x7e,0xff,0x18);
    write8x8(0x0,0x0,0x18,0x3c,0x7e,0xff,0x18,0x18);
    write8x8(0x0,0x18,0x3c,0x7e,0xff,0x18,0x18,0x18);
    write8x8(0x18,0x3c,0x7e,0xff,0x18,0x18,0x18,0x18);
    write8x8(0x18,0x3c,0x7e,0xff,0x18,0x18,0x18,0x18);
    write8x8(0x3c,0x7e,0xff,0x18,0x18,0x18,0x18,0x0);
    write8x8(0x7e,0xff,0x18,0x18,0x18,0x18,0x0,0x0);
    write8x8(0xff,0x18,0x18,0x18,0x0,0x0,0x0,0x0);
    write8x8(0x18,0x18,0x18,0x18,0x0,0x0,0x0,0x0);
    write8x8(0x18,0x18,0x0,0x0,0x0,0x0,0x0,0x0);
}

```

```

        write8x8(0x18,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    }

void right_arrow(){
    write8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    write8x8(0x18,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    write8x8(0x3c,0x18,0x0,0x0,0x0,0x0,0x0,0x0);
    write8x8(0x7e,0x3c,0x18,0x0,0x0,0x0,0x0,0x0);
    write8x8(0xff,0x7e,0x3c,0x18,0x0,0x0,0x0,0x0);
    write8x8(0x18,0xff,0x7e,0x3c,0x18,0x0,0x0,0x0);
    write8x8(0x18,0x18,0xff,0x7e,0x3c,0x18,0x0,0x0);
    write8x8(0x18,0x18,0x18,0x18,0xff,0x7e,0x3c,0x18,0x0);
    write8x8(0x18,0x18,0x18,0x18,0x18,0xff,0x7e,0x3c,0x18);
    write8x8(0x18,0x18,0x18,0x18,0x18,0x18,0xff,0x7e,0x3c,0x18);
    write8x8(0x0,0x18,0x18,0x18,0x18,0x18,0xff,0x7e,0x3c);
    write8x8(0x0,0x0,0x18,0x18,0x18,0x18,0x18,0xff,0x7e);
    write8x8(0x0,0x0,0x0,0x18,0x18,0x18,0x18,0x18,0xff);
    write8x8(0x0,0x0,0x0,0x0,0x18,0x18,0x18,0x18);
    write8x8(0x0,0x0,0x0,0x0,0x0,0x18,0x18,0x18);
    write8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x18,0x18);
}

void ahead_arrow(){
    write8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    write8x8(0x0,0x0,0x0,0x80,0x80,0x0,0x0,0x0);
    write8x8(0x0,0x0,0x80,0xc0,0xc0,0x80,0x0,0x0);
    write8x8(0x0,0x80,0xc0,0xe0,0xe0,0xc0,0x80,0x0);
    write8x8(0x80,0xc0,0xe0,0xf0,0xf0,0xe0,0xc0,0x80);
    write8x8(0x40,0x60,0x70,0xf8,0xf8,0x70,0x60,0x40);
    write8x8(0x20,0x30,0x38,0xfc,0xfc,0x38,0x30,0x20);
    write8x8(0x10,0x18,0x1c,0xfe,0xfe,0x1c,0x18,0x10);
    write8x8(0x8,0xc,0xe,0xff,0xff,0xe,0xc,0x8);
    write8x8(0x8,0xc,0xe,0xff,0xff,0xe,0xc,0x8);
    write8x8(0x4,0x6,0x7,0x7f,0x7f,0x7,0x6,0x4);
    write8x8(0x2,0x3,0x3,0x3f,0x3f,0x3,0x3,0x2);
    write8x8(0x1,0x1,0x1,0x1f,0x1f,0x1,0x1,0x1);
    write8x8(0x0,0x0,0x0,0xf,0xf,0x0,0x0,0x0);
    write8x8(0x0,0x0,0x0,0x7,0x7,0x0,0x0,0x0);
    write8x8(0x0,0x0,0x0,0x3,0x3,0x0,0x0,0x0);
    write8x8(0x0,0x0,0x0,0x1,0x1,0x0,0x0,0x0);
}

```

```
}
```

1.4 Utils

```
#include "utils.h"
#include "max7219.h"
#include <msp430g2553.h>

void delay(volatile unsigned int i){
    while((i--)>0);
}

void clear_screen(){
    write8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0); // Cleaning screen
}

void configure_buttons(){
    P1DIR &= ~(RIGHT_BTN + LEFT_BTN + STOP_BTN); //Seta como entrada 0 =
    P1OUT &= ~(RIGHT_BTN + LEFT_BTN + STOP_BTN); //Desliga ambos os leds
    P1IE |= (RIGHT_BTN + LEFT_BTN + STOP_BTN);
    P1IFG &= ~(RIGHT_BTN + LEFT_BTN + STOP_BTN);
    P1REN = (RIGHT_BTN + LEFT_BTN + STOP_BTN);
    P1IES |= (RIGHT_BTN + LEFT_BTN + STOP_BTN);
}
```