PONTO DE CONTROLE 4
Módulo Eletrônico Para Bicicletas

Fábio Barbosa Pinto
Programa de Engenharia Eletrônica
Faculdade do Gama
Universidade de Brasília – UnB
Brasília, Brasil
fabio_bbarbosa@hotmail.com

João Pedro Moreira da Silva
Programa de Engenharia Eletrônica
Faculdade do Gama
Universidade de Brasília – UnB
Brasília, Brasil
jptekc@gmail.com

Larissa Aidê Araújo Rocha
Programa de Engenharia Eletrônica
Faculdade do Gama
Universidade de Brasília – UnB
Brasília, Brasil
larissa.aide@hotmail.com

***Resumo*— O projeto visa desenvolver um módulo eletrônico para bicicletas, que terá a funcionalidade de rastreador e contará com um sistema de setas. Ele irá mostrar através de mensagens, via celular, as coordenadas de localização da bicicleta. Além disso, irá indicar, através de setas, a intenção do ciclista de mudar de direção na via.**

Palavras-chave— rastreador; bicicleta; sinalização; MSP430.

I. JUSTIFICATIVA

Com o excesso de carros nas ruas e lentidão no trânsito, cada vez mais pessoas buscam meios de transporte alternativos. A preferida é a bicicleta, que promove uma grande qualidade de vida. Mas infelizmente, o aumento da violência urbana e acidentes são mais rápidos que uma busca pela sustentabilidade.

Apesar de ser um ótimo transporte alternativo, sabe-se que a bicicleta não é um investimento barato, podendo custar até 35 mil reais dependendo do modelo ou marca. Independente do valor encontrado no mercado, certamente, se o proprietário usa sua bicicleta com frequência, ficar sem ela fará muita falta.

De acordo com o Cadastro Nacional de Bicicletas Roubadas [1], dentre os 30 municípios que divulgaram as ocorrências de furtos e roubos nos últimos seis anos, as quatro cidades que lideram o ranking são respectivamente: São Paulo, Rio de Janeiro, Curitiba e Brasília. Nesse período, foram cerca de 1.940 casos, sem contar com as inúmeras ocorrências que não foram registradas.

Além da preocupação com o furto de bicicletas, é imprescindível zelar também pela segurança do ciclista. Notícias sobre acidentes envolvendo bicicletas são comuns no Brasil. 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 o 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 em morte.

Há diversos modelos no mercado de rastreadores para veículos, o site Vox Popi [4] fez uma lista com mais de 17 modelos, onde mostra a diferença entre os produtos.

O diferencial deste projeto é ir além de ser um mero rastreador, dando importância também a segurança do ciclista enquanto trafega em vias com outros veículos. Assim, o módulo eletrônico para bicicletas poderá torna-se o investimento ideal para aumentar as chances de recuperá-la, caso seja roubada e provê maior segurança para o ciclista no trânsito.

II. OBJETIVO

Construir um módulo eletrônico capaz de transmitir as coordenadas geográficas de uma bicicleta através de mensagens, via celular, e implementar um sistema para indicar a intenção do ciclista de mudar de direção na via.

III. REQUISITOS

Para o correto funcionamento do módulo eletrônico, será necessária uma leitura das coordenadas geográficas da bicicleta, através de um GPS. Os dados recebidos serão enviados via SMS, através do módulo GSM em conjunto com um chip SIM. É preciso que o usuário possua um número de celular para cadastrar e utilizar as funcionalidades do rastreador. Para o sistema de setas, será necessária uma matriz de LEDs e uma chave seletora.

Os dados e comandos recebidos serão interpretados através do microcontrolador MSP430, esse dispositivo possui uma poderosa CPU RISC de 16 bits, que alia uma boa eficiência com um baixo consumo de energia. Além disso, será imprescindível uma bateria recarregável.

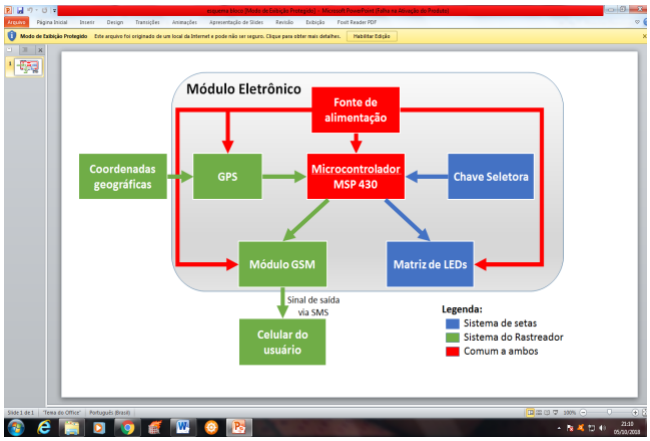

Fig. 1. Diagrama de blocos do módulo eletrônico para bicicletas.

Na Figura 1, através do diagrama de blocos do módulo eletrônico, é possível ver com maior facilidade os componentes eletrônicos e suas interações descritas neste tópico.

IV. BENEFÍCIOS

Por meio do módulo eletrônico, o usuário poderá localizar sua bicicleta em qualquer lugar, caso a mesma tenha sido roubada ou furtada e trará maior segurança ao ciclista, ao trafegar em vias públicas.

O rastreador seria uma contraproposta ao uso do seguro de bicicletas, já que, geralmente, eles não cobrem bicicletas desprotegidas, ou seja, sem a presença do seu proprietário. Se você deixar a sua bicicleta presa ao poste ou esquecer no parque e a levaram, infelizmente, você não terá mais como recuperá-la se optar apenas pelo seguro.

O sistema de setas alerta a intenção dos atos do ciclista para outros condutores, visando garantir maior segurança ao trafegar em vias públicas.

Dessa forma, o proprietário da bicicleta terá uma segurança maior em deixá-la presa em algum lugar, como postes, árvores ou paraciclos e até mesmo ao andar com ela pela rua.

V. HARDWARE

Os materiais utilizados até o ponto de controle 3 foram:

- 2 MSP-EXP430G2553LP
- 1 módulo GSM800L
- 1 módulo GPS GY-NEO6M
- 1 módulo Matriz LED 8x8 com MAX7219
- 3 chaves seletoras
- Jumpers

O Hardware do projeto é constituído por dois microcontroladores combinados com o módulo GPS, GSM e o módulo matriz LED.

Visando uma melhoria no projeto, o sistema foi dividido em duas partes:

Parte 1: Essa parte é responsável pela sinalização da bicicleta, onde será integrado o módulo LED 8x8 MAX7219 ao MSP430.

A matriz LED pode ser conduzida de duas maneiras (paralela ou serial). Aqui o conduzimos de maneira serial, para simplificar a utilização e reduzir o tamanho do circuito. O CI MAX7219 é um driver de exibição de entrada /saída de cátodo comum de série, que interage com microcontroladores como arduinos, MSP430, Raspberry e outros, para um LED numérico de 7 segmentos de até 8 dígitos, exibições de gráfico de barras ou 64 LEDs individuais.

Na figura 5.1 é possível observar o modo como o circuito foi montado já com as entradas e saídas do microcontrolador definidas.

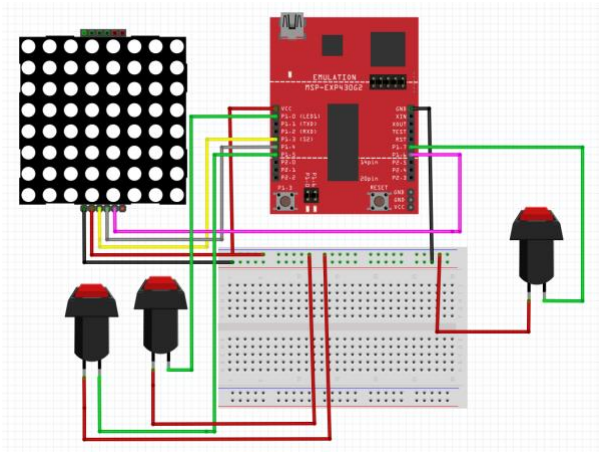

Figura 5.1 Circuito montado para a sinalização da bicicleta

Parte 2: Esta parte é responsável pela transmissão de informação da localização da bicicleta. Um microcontrolador receberá as informações provenientes do módulo GPS, e os comandos do módulo GSM, através de SMS.

Para o microcontrolador, utiliza-se outra placa MSP430, que recebe as informações do módulo GPS (GPS NEO6M), através de comunicação UART. Este mesmo módulo recebe o comando proveniente do módulo GSM (SIM800L), através de SMS, e envia a localização da bicicleta, também utilizando comunicação UART.

VI. CÓDIGO

Para a Parte 2 do projeto tanto o módulo GPS como o módulo GSM utilizam comunicação UART, dessa forma foi necessária a criação de uma nova porta UART para um dos módulos, sendo escolhido o módulo GPS, pois o mesmo apenas envia dados para o microcontrolador, ou seja, só utiliza uma porta RX, enquanto o módulo GSM recebe e envia dados para o microcontrolador, utilizando uma porta RX e uma TX.

6.1. *Módulo GSM*

O código gerado para a comunicação da placa MSP430 com módulo GSM800l utiliza o software Energia. O intuito do código é enviar uma sms para o destinatário ao qual o numero de celular foi adicionado no próprio código (futuramente o corpo da mensagem será a localização da bicicleta fornecida pelo módulo GPS).

A comunicação entre o Microcontrolador e o módulo GSM é feita através da porta Serial (UART), dessa forma uma vez que as comunicações tenham sido declaradas e estabelecidas, basta somente enviar comandos AT do datasheet para que o módulo possa processá-los e agir de acordo.

Um fator importante analisado é que módulo pode utilizar muita corrente de sua fonte em picos de transmissão (de até 3A e devido a isso não se pode alimentá-lo diretamente na MSP), então embora o código funcione perfeitamente se a fonte não for suficiente para alimentá-lo, ele irá reiniciar aleatoriamente durante as transmissões.

6.2 *Módulo GPS*

Para realizar a montagem do projeto, deve-se conectar os pinos TX e RX do módulo GPS aos pinos TX e RX correspondentes na MSP430. É possível utilizar o próprio microcontrolador para suprir a alimentação do módulo GPS, conectando os cabos Vcc e Gnd desse aos respectivos equivalentes.

O código gerado para a comunicação da placa MSP430 com módulo GPS GY-NEO6M utiliza o software Energia. De maneira geral o código simplesmente ler as strings originais do módulo GPS e mostrá-las na tela serial. Ele também usa uma biblioteca para tratar os dados e usar funções pré-definidas para obter apenas as informações que você precisa. Essa biblioteca é a A TinyGPS++, que permite extrair as informações fornecidas pelo módulo GPS, fornecendo apenas as informações que desejamos.

Devido a complexibilidade do módulo, o mesmo retorna a localização do indivíduo monitorado na forma de um link para o Google Maps, e envia a mesma para o responsável em forma de SMS pelo módulo GSM. O módulo utiliza os dados do padrão NMEA para coordenadas, o software separa esses

valores e retorna o que é pedido, latitude, longitude, altitude, hora.

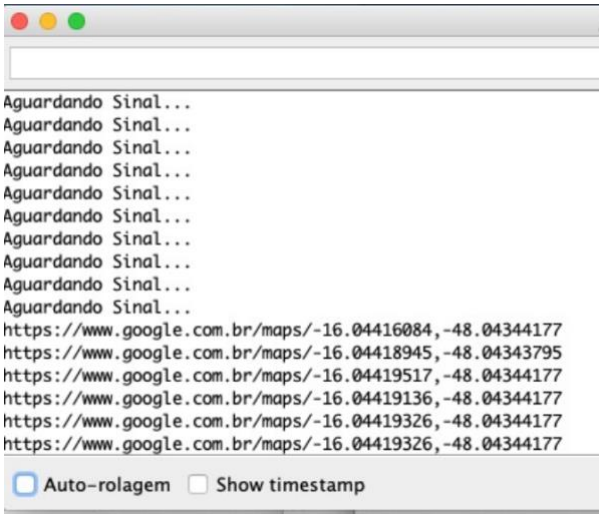

Figura 6.2.1 Resposta do módulo GPS no MSP

Para que o GPS e o GSM retornem um link para o Google Maps, foi necessário a criação de uma string que concatena, uma string inicial para o link, e os valores obtidos para latitude e longitude. Para o cálculo da distância de um ponto a outro utilizou-se funções já prontas para a biblioteca do módulo GPS.

6.3 Setas de Sinalização

Para a execução do código usamos 3 desenhos diferentes seta para a esquerda, seta para a direita, stop e o de sinalização (que são todos os LEDs do módulo em alto piscando)

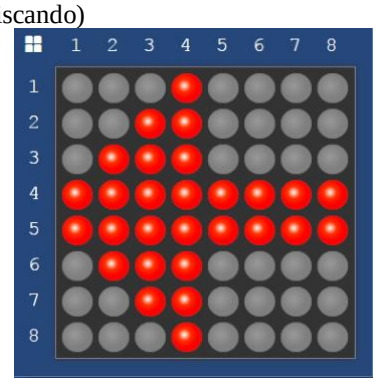

Figura 6.3.1 Desenha de referência seta para esquerda

Cuja a função no código foi descrita da seguinte maneira:

```
// BITS SETADOS EM CADA LINHA P/ FORMAR A SETA P/ ESQUERDA
/*
  B00010000,
  B00110000,
  B01110000,
  B11111111,
  B11111111,
  B01110000,
  B00110000,
  B00010000
 */
void seta_esquerda(){

   led8x8(0x18,0x3c,0x7e,0xff,0x18,0x18,0x18,0x18);
}
```

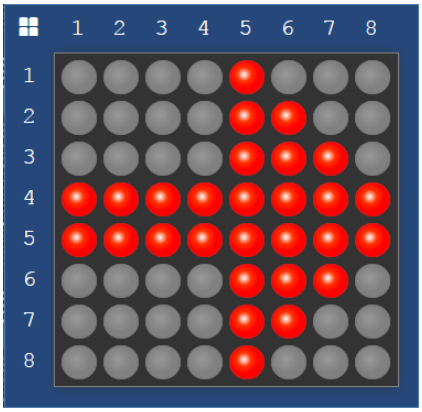

Figura 6.3.2 Desenha de referência seta para direita

Onde a função no código foi descrita da seguinte maneira:

```
//BITS SETADOS EM CADA LINHA P/ FORMAR A SETA P/ DIREITA
/*
  B00001000,
  B00001100,
  B00001110,
  B11111111,
  B11111111,
  B00001110,
  B00001100,
  B00001000
 */

// FUNÇAO SETA P/ DIREITA

void seta_direita(){

     led8x8(0x18,0x3c,0x7e,0xff,0x18,0x18,0x18,0x18);
```

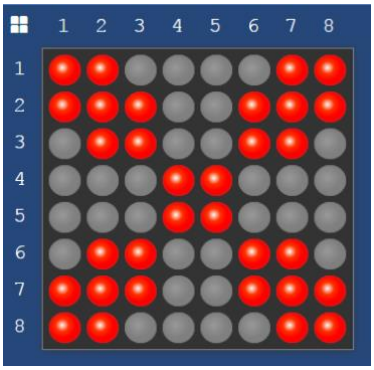

Figura 6.3.3 Desenha de referência para stop

Cuja a função no código foi descrita da seguinte maneira:

```
void stop(){
```

```
    led8x8(0xc3,0xe7,0x66,0x18,0x18,0x66,0xe7,0xc3);
}
```

E por fim a sinalização, onde todos os bits da matriz foram setados, seguidos de um delay para que ficasse piscando.

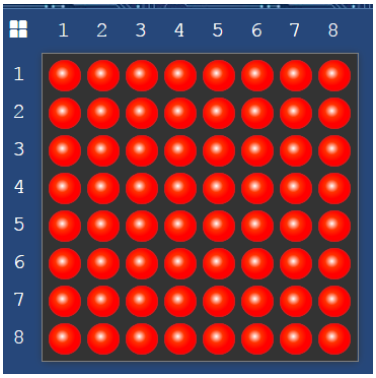

Figura 6.3.3 Desenha de referência seta para esquerda

Cuja a função no código foi descrita da seguinte maneira:

```
void asinalizaçao(){
```

```
write8x8(0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff);
delay (10000);
```

VII. ANEXOS

A. ***Código GSM***

```
#include <SoftwareSerial.h>
 // Incluimos a livraria SoftwareSerial
SoftwareSerial mySerial(1, 2);  // Declaramos os pinos RX(1) y TX(2) que vamos a usar

void setup(){
Serial.begin(9600);       // Iniciamos a comunicação serial
mySerial.begin(9600);     // Iniciamos uma segunda comunicação serial
delay(1000);              // Pausa de 1 segundo
EnviaSMS();               // Chamada a função que envia o SMS
}

void loop(){
if (mySerial.available()){          // Se a comunicação SoftwareSerial tem dados
  Serial.write(mySerial.read());    // Obtemos por comunicação serie normal
}

if (Serial.available()){            // Se a comunicação serie normal tem dados
  while(Serial.available()) {       // e enquanto tenha dados para mostrar
    mySerial.write(Serial.read());  // Obtemos pela comunicação SoftwareSerial
  }
  mySerial.println();               // Enviamos um fim de linha
}
}

// Função para o envio de um SMS
void EnviaSMS(){
 mySerial.println("AT+CMGF=1");                // Ativamos a função de envio de SMS
 delay(100);                                  // Pequena pausa
 mySerial.println("AT+CMGS=\"+5561995--------\"");  // Definimos o número do destinatário em formato internacional
 delay(100);                                  // Pequena pausa
 mySerial.print("Comunicaçao GSM + MSP430");    // Definimos o corpo da mensagem
 delay(500);                                  // Pequena pausa
 mySerial.print(char(26));                        // Enviamos o equivalente a Control+Z
 delay(100);                                  // Pequena pausa
 mySerial.println("");                           // Enviamos um fim de linha
 delay(100);                                  // Pequena pausa
}
```

B. Código GPS

```
#include <TinyGPS++.h>
/* Cria um objeto chamado gps da classe TinyGPSPlus */
TinyGPSPlus gps;

volatile float minutes, seconds;
volatile int degree, secs, mins;

void setup() {
  Serial.begin(9600);       /* Define a taxa de transmissão para comunicação serial */
}
```

```
void loop() {
      smartDelay(1000); /* Gerar atraso preciso de 1ms */
      if (!gps.location.isValid())
      {
        Serial.print("Latitude : ");
        Serial.println("*****");
        Serial.print("Longitude : ");
        Serial.println("*****");
      }
      else
      {
        //DegMinSec(gps.location.lat());
        Serial.print("Latitude em graus decimais : ");
        Serial.println(gps.location.lat(), 6);
//        Serial.print("Latitude em graus minutos em segundos : ");
//        Serial.print(degree);
//        Serial.print("\t");
//        Serial.print(mins);
//        Serial.print("\t");
//        Serial.println(secs);
        //DegMinSec(gps.location.lng());  /* Converte o valor do grau decimal em graus minutos, segundo
formulário */
        Serial.print("Longitude em decimais degraus : ");
        Serial.println(gps.location.lng(), 6);
//        Serial.print("Longitude em degraus minutos segundos : ");
//        Serial.print(degree);
//        Serial.print("\t");
//        Serial.print(mins);
//        Serial.print("\t");
//        Serial.println(secs);
      }
      if (!gps.altitude.isValid())
      {
        Serial.print("Altitude : ");
        Serial.println("*****");
      }
      else
      {
        Serial.print("Altitude : ");
        Serial.println(gps.altitude.meters(), 6);
      }
      if (!gps.time.isValid())
      {
        Serial.print("Time : ");
        Serial.println("*****");
      }
      else
      {
```

```
        char time_string[32];
        sprintf(time_string, "Time : %02d/%02d/%02d \n",gps.time.hour(), gps.time.minute(),
gps.time.second());
        Serial.print(time_string);
       }
}

static void smartDelay(unsigned long ms)
{
  unsigned long start = millis();
  do
  {
    while (Serial.available())        /* Codifica dados lidos do GPS enquanto os dados estão disponíveis na
porta serial  */
     gps.encode(Serial.read());
            /* Encode basicamente é usado para analisar a cadeia de caracteres recebida pelo GPS e armazená-
la em um buffer para que as informações possam ser extraídas dela */
  } while (millis() - start < ms);
}

//void DegMinSec( double tot_val)                  /* Converte dados em graus decimais em graus minutos
segundos */
//{
// degree = (int) tot_val;
// minutos = tot_val - degree;
// segundos = 60 * minutos;
// minutos = (int) segundos;
// mins = (int) minutos;
// segundos = segundos - minutos;
// segundos = 60 * segundos;
// segundos = (int) segundos;
//
```

C. Código Módulo Matriz de LED 8x8

1.1 Main

```
#include "max7219.h"
#include "comando.h"
#include "figura.h"
#include <msp430g2553.h>


int main(){

  WDTCTL = WDTPW + WDTHOLD;   // Desabilita WDT
  DCOCTL = CALDCO_1MHZ;     // 1 Mhz DCO
  BCSCTL1 = CALBC1_1MHZ;

  limpa_tela();

 setIntensidade(0xff);
```

```
  setTestMode(0);
  setDesliga(0);
  showDigit(8);

  conf_botao();

  __enable_interrupt();

   initialise();  // Funçao definida em modulo que habilita as entradas CS, DIN e CLK do módulo

//CRIAR LAÇO EM QUE O PRIMEIRO ESTADO DOS LEDS SEJA A SINALIZAÇÃO
            while(1){
   ahead_arrow();
  }

}

// interrupção que habilita o display com o botão

#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void){
if(P1IFG & BTN_ESQUERDA){
    while(!(P1IN & BTN_ESQUERDA)==0){
      seta_esquerda();
    }
    P1IFG &= ~BTN_ESQUERDA;
  }
if(P1IFG & BTN_DIREITA){
    while(!(P1IN & BTN_DIREITA)==0){
      seta_direita();
    }
    P1IFG &= ~BTN_DIREITA;
  }
  if(P1IFG & STOP_BTN){
    while((P1IN & STOP_BTN)==0){
      delay(20);
      stop();
    }
    P1IFG &= ~STOP_BTN;
  }
}
```

1.2 Comando

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

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

// limpa o display
void limpa_display(){
  write8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
}

// Funçao que configura os botoes
void conf_botao(){
  P1DIR &= ~(BTN_DIREITA + BTN_ESQUERDA + STOP_BTN); // configura os botoes como entrada
```

```
  P1OUT &= ~(BTN_DIREITA + BTN_ESQUERDA + STOP_BTN; //Desliga ambos os leds
  P1IE |= (BTN_DIREITA + BTN_ESQUERDA + STOP_BTN);
  P1IFG &= ~(BTN_DIREITA + BTN_ESQUERDA + STOP_BTN);
  P1REN = (BTN_DIREITA + BTN_ESQUERDA + STOP_BTN);
  P1IES |= (BTN_DIREITA + BTN_ESQUERDA + STOP_BTN);
}
```

1.3 MAX7219

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

#define MAX7219_DIN BIT3
#define MAX7219_CS  BIT4
#define MAX7219_CLK BIT5


//Laço responsavel em fazer a comunicaçao do clk do modulo
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;
  }
}

// Funçao que Leva o pino CS para nivel alto e seta os pinos CS, DIN, CLK para a saida

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;
}

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

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

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


// funçao brilho do modulo

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

// Funçao que leva DIN para alto (DIN é utilizado para inserir os bits)
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;
  }
}

//Define a matriz de LED (registrador, coluna)
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);
}

//Configuraçao da matriz
void led8x8(char a, char b, char c, char d, char e, char f, char g, char h){
  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);
}
```

1.4 Figura

```
#include "figura.h"
#include "comando.h"
#include "max7219.h"
```

```
void stop(){

    led8x8(0xc3,0xe7,0x66,0x18,0x18,0x66,0xe7,0xc3);
    led8x8(0xc3,0xe7,0x66,0x18,0x18,0x66,0xe7,0xc3);
    led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
}
// BITS SETADOS EM CADA LINHA P/ FORMAR A SETA P/ ESQUERDA
/*
  B00010000,
  B00110000,
  B01110000,
  B11111111,
  B11111111,
  B01110000,
  B00110000,
  B00010000
 */
void seta_esquerda(){

    led8x8(0x18,0x3c,0x7e,0xff,0x18,0x18,0x18,0x18);
    led8x8(0x18,0x3c,0x7e,0xff,0x18,0x18,0x18,0x18);
    led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
}

//BITS SETADOS EM CADA LINHA P/ FORMAR A SETA P/ DIREITA
/*
  B00001000,
  B00001100,
  B00001110,
  B11111111,
  B11111111,
  B00001110,
  B00001100,
  B00001000
 */

// FUNÇAO SETA P/ DIREITA

void seta_direita(){

      led8x8(0x18,0x3c,0x7e,0xff,0x18,0x18,0x18,0x18); //direita
      led8x8(0x18,0x3c,0x7e,0xff,0x18,0x18,0x18,0x18);
      led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
      led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
      led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
      led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);

   //write8x8(0x18,0x18,0x18,0x18,0xff,0x7e,0x3c,0x18);
}

//FUNÇAO SINALIZAÇAO - TODOS OS LED LIGADOS PARA INDICAR O CICLISTA

void sinalizaçao(){
```

```
    led8x8(0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff); //sinalizaçao
    led8x8(0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff);
    led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);
    led8x8(0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0);

}
```

D. Código para interação entre GSM e GPS na IDE Energia

```
#include <TinyGPS++.h>
TinyGPSPlus gps;
double latitude;
double longitude;
void setup() {
Serial.begin(9600);
}
void loop() {
smartDelay(1000); /* Generate precise delay of 1ms */
if (gps.location.isValid())
{
latitude = gps.location.lat(),8;
longitude = gps.location.lng(),8;
Serial.println(latitude);
Serial.println(longitude);
sendsms();
}
else{
latitude = 1;
longitude = 1;
sendsms();
}
}
static void smartDelay(unsigned long ms)
{
unsigned long start = millis();
do
{
while (Serial.available()) /* Encode data read from GPS while data
is available on serial port */
gps.encode(Serial.read());
/* Encode basically is used to parse the string received by the GPS
and to store it in a buffer so that information can be extracted from it
*/
} while (millis() - start < ms);
}
void sendsms()
{
Serial.print(" AT+CMGF=1\r");
delay(500);
Serial.print("AT+CMGS =\"+5561995--------\"\r");
delay(500);
Serial.print(latitude);
Serial.print(longitude);
Serial.print("\r");
delay(500);
Serial.print(0x1A);
}
```

E. Código do GSM para receber mensagem no CCS

```
#include "msp430g2553.h"
#include "uart.h" // Arquivar o arquivo uart com o codigo main


void gsm(); // Prototipo de funçao
int main(void){

   WDTCTL = WDTPW + WDTHOLD;// Para o o whatchdogtimer
   BCSCTL1 = CALBC1_8MHZ;// Faz a frequencia de 8Mhz com o launchpad
   DCOCTL = CALDCO_8MHZ;



   uart_init(); // CHAMAR A FUNÇÃO UART INIT QUE ESTÁ DISPONÍVEL NO ARQUIVO
   __enable_interrupt();//  Interrupçao ENABLE
   __delay_cycles(100000);
   gsm();// Chama a funçao gsM
}

void gsm(){

     uart_puts((char *)"AT"); // Comando para inicializar o GSM

     uart_putc(0x0A);

     uart_putc(0x0D);//retorna carriege

     __delay_cycles(10000000);// ATRASO ... ESPERE OK DE GSM

     uart_puts((char *)"AT+CMGF=1");//Comunicaçao

     uart_putc(0x0A);

     uart_putc(0x0D);

    __delay_cycles(10000000); // espera pelo Ok

     uart_puts((char *)"AT+CMGS=\"+5561995094992\""); //manda a mensagem para esse numero
     uart_putc(0x0A);
     uart_putc(0x0D);
     uart_puts((char *)"LOCALIZAÇAO: ... "); //MANDA A LOCALIZAÇAO PARA O NUMERO ... AINDA FALTA
INTREGRAR
     uart_putc(0x1A);// "ctrl Z"


}
```