Pulseira localizadora via GPS e alarme localizador

Luiza Carneiro Cezário
Universidade de Brasília - UnB
Brasília-DF, Brasil
luiza_cezario@hotmail.com

Vinícius Ferreira Ribeiro
Universidade de Brasília - UnB
Brasília-DF, Brasil
ribeirovinicius08@gmail.com

***Resumo*— Devido ao grande número de idosos que se perdem por causa de doenças como Alzheimer, e de crianças que por distração dos responsáveis, se perdem em grandes multidões como por exemplo em praias durante o verão, é necessário o acompanhamento em tempo real desses indivíduos. Desta forma o projeto visa implementar esse monitoramento com o uso de uma pulseira localizadora com GPS, que envia a localização do indivíduo monitorado através de SMS para um número cadastrado, alarme controlado e configurado via bluetooth, com dados do responsável exibidos em um display acoplado a pulseira, para monitoramento de crianças e/ou idosos portadores de Alzheimer.**

Palavras-Chave — Pulseira localizadora, GPS, Bluetooth, Alzheimer.

I. Introdução

De acordo com o IBGE, o número de idosos cresce todo ano, porém na mesma proporção que em que a população idosa cresce, também cresce a incidência de doenças crônicas, como o Alzheimer. A partir dos 65 anos, o risco de desenvolvimento desta doença duplica a cada cinco anos. No Brasil, cerca de 1,2 milhão de pessoas são portadores da doença, a Organização Mundial da Saúde prevê que até 205 o número de casos aumente em até 500% em toda a América Latina. [1,2]

O número de casos de crianças perdidas durante o período de férias de verão é grande nas praias do litoral, de acordo com os Grupamentos de Praia da Guarda Municipal do Rio de Janeiro, registraram 482 casos no verão de 2015-2016, enquanto que no litoral paulista, foram 673 crianças, de acordo com o Grupamento de Bombeiros Marítimo do estado. [3]

Para prevenir o desaparecimento de idoso e crianças é necessário a máxima atenção dos responsáveis pelas mesmas, de qualquer modo, existem iniciativas para prevenir tal situação, e ajudar na localização dos desaparecidos. Existe um projeto português para ajudar os portadores de Alzheimer perdidos, a utilização de colar ou pulseira com um código, onde os responsáveis seriam contatados pelas autoridades. Este código contém informações sobre o idoso, e o contato de seus cuidadores, as quais somente as entidades autorizadas teriam acesso a esses dados. [4]

Uma forma parecida de identificação é utilizada nas praias brasileiras. Uma pulseira é distribuída em postos de guarda-vidas, onde nela são escritas as informações da criança e responsável. [5]

Tendo em vista toda pesquisa realizada, comprovou-se a necessidade de um método mais eficaz de monitoramento para ambos os casos, de forma a aliar a tecnologia e a necessidade de prevenção de desaparecimentos, ou ainda em caso de se perderem, os responsáveis terem a localização dos mesmos em tempo real, para que sejam encontrados de maneira rápida e segura. Partindo dessa análise, levantou-se o objetivo e os requisites do projeto.

II. Objetivo

A. Garantir segurança

A pulseira utilizará o mesmo material já utilizado em tornozeleiras eletrônicas, sendo um fio condutor responsável por verificar o rompimento da mesma, e enviar o alerta para o responsável, o alerta sonoro chamará a atenção das pessoas em volta, evitando um possível desaparecimento da criança ou idoso.

B. Informação em tempo real e configuração bluetooth

A informação quanto a localização do usuário, será disponibilizada em tempo real, através de mensagens de texto, enviadas utilizando um módulo GPRS SIM800L, no qual o responsável envia uma mensagem com o comando para obter a localização, e recebe a resposta da mesma, através de um link para o Google Maps. A configuração por Bluetooth visa a simplicidade para o usuário, de forma que possa controlar a distância limite de acordo com o local, e grave os dados do usuário monitorado e do responsável, podendo a mesma ser usada por outros indivíduos a serem monitorados, posteriormente.

III. Desenvolvimento

A. Descrição do Hardware

O Hardware do projeto é constituído por dois microcontroladores combinados com módulos GPS, GSM, Bluetooth e display, com o objetivo de monitorar indivíduos e evitar desaparecimento. Esse Sistema foi dividido em dois blocos:

Bloco 1: Este bloco é responsável pelas informações de localização do indivíduo, pertencente a pulseira que o mesmo utilizará, onde conterá suas informações e será configurado por Bluetooth.

Para o microcontrolador utilizou-se o MSP430G2553IN20, que é um microcontrolador RISC de 16 bits, voltado para aplicações de baixo consumo de energia, fabricado pela Texas Instruments.

O Bluetooth (HC-05), recebe as informações enviadas por um celular com Sistema operacional Android, utilizando o aplicativo Terminal Bluetooth que pode ser encontrado na Play Store. Essas informações são o nome do indivíduo monitorado, a idade, o nome do responsável e o contato do mesmo.

De posse dessas informações, o microcontrolador as envia para o display (Nokia 5110), que utiliza comunicação SPI, como mostrado na figura abaixo:

Figura 1: Informações no display (Número borrado por questão de privacidade).

Ainda neste bloco, está presente um buzzer, que é ativado caso o indivíduo monitorado retire a pulseira, ou ultrapasse a distância pré-determinada, e somente é desativado quando o responsável reinicia o sistema.

Bloco 2: Este bloco é responsável pela transmissão de informação da localização do indivíduo monitorado. 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 MSP430G2553IN20, 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 do indivíduo monitorado, também utilizando comunicação UART.

Unidades	Materiais	Fabricante
02	MSP430G2553IN20	Texas Instruments
01	Módulo GPS GY-GPS6MV2 NEO6M Ublox	Ublox
01	Módulo SIM800L	Ublox
01	Módulo Bluetooth Serial HC-06 Escravo	
01	Buzzer 5V	
-	Jumpers	-
01	Protoboard	Hikari
01	Display Nokia 5110	
-	Resistores 4,7 KOhm	-
-	Resistores 10 KOhm	-
-	Resistores 220 Ohm	-

Tabela 1: Tabela de materiais utilizados no projeto.

O projeto será implementado em uma pulseira, que utiliza o mesmo material que tornozeleiras eletrônicas, que a deixa a prova de água, que seja ergonômica e não incomode o usuário.

Utilizou-se o software *Fritzing* para a montagem do esquemático completo do projeto, como mostrado na figura 2 abaixo.

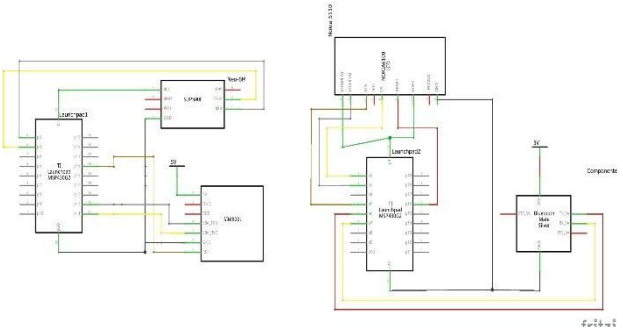

Figura 2: Esquemático completo do projeto.

De acordo com o explicado na descrição de hardware, na figura 3 e figura 4, pode-se observado de forma resumida o funcionamento do projeto de acordo com o fluxograma de funcionamento e o diagrama de blocos do mesmo.

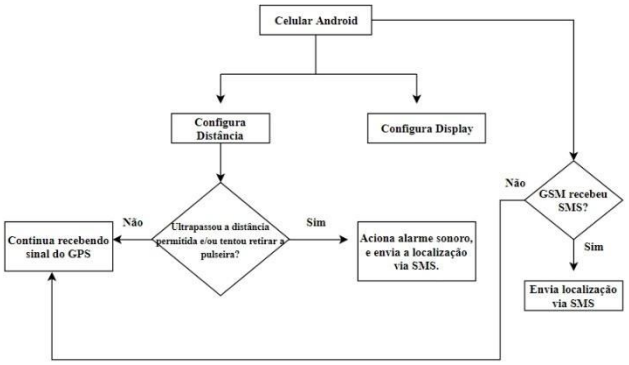

Figura 3: Fluxograma do Projeto.

Figura 4: Diagrama de blocos.

B. Descrição de Software

O software utilizado inicialmente no projeto foi a IDE Energia, devida à simplicidade na codificação por ser uma plataforma de código aberto, que visa ser próxima ao framework Arduíno, para a launchpad do MSP430.

Após o período de testes dos módulos, e o entendimento quanto a programação, foi realizada a adequação dos códigos para o software Code Composer Studio (CCS), que é utilizado para o desenvolvimento de sistemas embarcados e de baixo nível de consumo de energia, entretanto é um software que exige maior entendimento de codificação.

Para a comunicação do módulo Bluetooth HC-05, que utiliza comunicação serial UART, foi criado o código do **Anexo I,** onde a launchpad configurada para Hardware Serial, utiliza os pinos P1.1 como RX (recebe dados) e P1.2 como TX (transmite dados), através da saída TX do módulo, conectado ao pino P1.1, recebe os dados enviados de um celular android, (utilizando o aplicativo Bluetooth Terminal, disponível na Play Store), sendo esses dados o nome do indivíduo monitorado, sua idade, o nome do responsável e o contato do mesmo. [6,7]

Após o recebimento desses dados pelo microcontrolador, os mesmos são enviados para o display Nokia 5110, que utiliza comunicação SPI , que é um importante método de interface de comunicação, que envolvem principalmente RTC, controladores LCD e unidades de transmissão e recepção UART, que é o caso em questão. A comunicação entre módulo e bluetooth é mostrada no **Anexo I**, como já dito. [8]

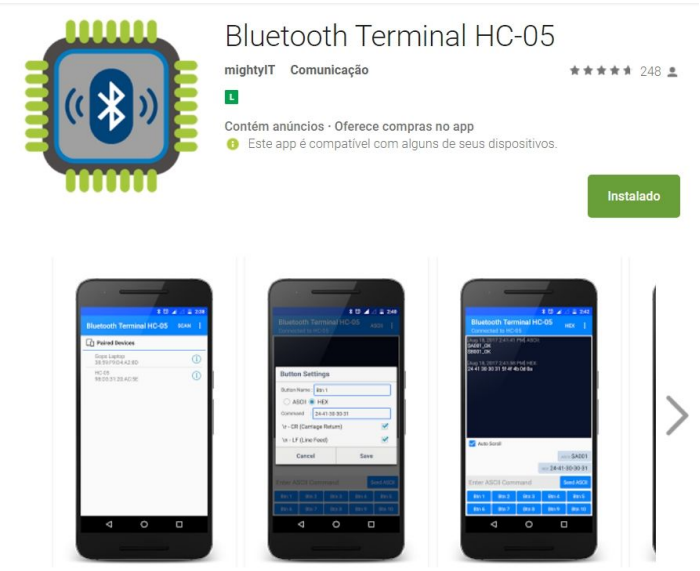

Figura 5: Aplicativo Terminal Bluetooth

Figura 6: Configuração Hardware Serial da launchpad.

O desenvolvimento do software do bloco 2, deu-se forma parecida. Os códigos foram desenvolvidos separadamente, na IDE Energia para o módulo GPS e no CCS para o GSM.

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 (**Anexo II**).

O código para o GSM (**Anexo III**) recebe o comando via SMS, através da comunicação UART, e a launchpad em configuração Hardware Serial (Figura 6), compara com o comando salvo no software e retorna a localização do usuário monitorado e/ou aciona o alarme sonoro, utilizando um chip de celular com acesso a rede 3G pelo menos, com um número de destinatário previamente definido. [9,10]

O código para o GPS (**Anexo IV**), foi desenvolvido utilizando a IDE Energia, 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 essas valores e retorna o que é pedido, latitude, longitude, altura, hora.[11]

Como o código do GPS foi desenvolvido utilizando a IDE Energia, foi necessária a adaptação do código GSM para a mesma plataforma.

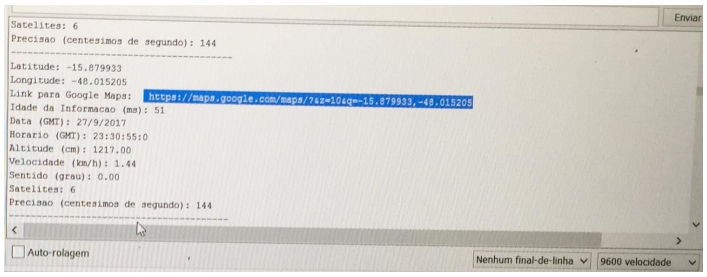

Figura 7: Resposta do módulo GPS no MSP

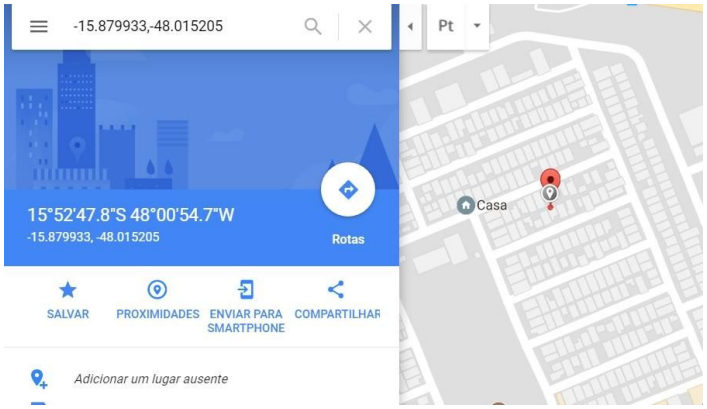

Figura 8: Localização dada pelo módulo GPS

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 (**Anexo V**).

O alarme que é representado como fio de segurança (Figura 4), é a prevenção caso o usuário tente retirar a pulseira, o mesmo depende apenas da comunicação de um pino de saída com um pino de entrada, onde se os valores forem iguais, o alarme fica desligado, caso sejam diferentes, o que indica o rompimento da pulseira, o alarme é disparado, e só é desligado caso o responsável reinicie o Sistema.

Todos os códigos e bibliotecas utilizadas também encontram-se no GitHub. [12]

IV. Resultados

Para o bloco 1 obteve-se o resultado esperado na proposta do projeto. O bluetooth envia os dados através do aplicativo Terminal Bluetooth, proveniente de um celular Android, esses dados são recebidos pelo microcontrolador MSP430, através da conexão UART, e são enviados para o display Nokia 5110, utilizando comunicação SPI, que exibe as mesmas.

Para a comprovação do envio dos dados através do bluetooth, o aplicativo utilizado, também exibe a informação enviada. Neste bloco o maior problema foi a migração da IDE Energia para o CCS, para a parte do bluetooth, onde encontrou-se dificuldade para a leitura das strings, resolvida através de um contador para separar cada informação.

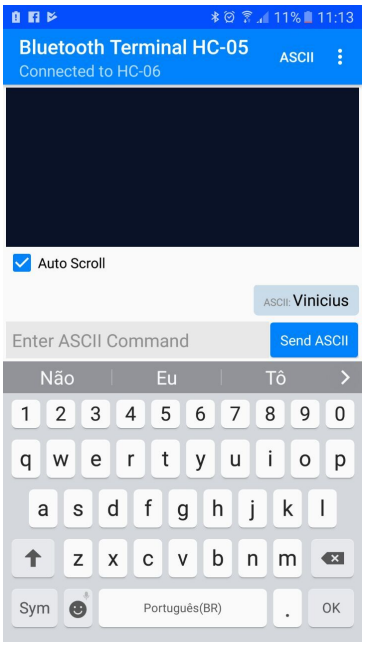

Figura 9: Aplicativo terminal bluetooth.

Para o bloco 2, o projeto não foi completado de acordo com a proposta. O módulo GPS necessita de um maior conhecimento de programação para sua utilização no CCS. Por meio de pesquisas realizadas em materiais, fóruns e espaços para discussão, disponibilizados e indicados pela Texas Instruments, encontrou-se a biblioteca utilizada e adaptada.

Para o módulo GSM conseguiu-se a migração para o CCS, com seu funcionamento comprovado através de mensagens de teste, onde uma mensagem foi enviada utilizada um celular Android, recebendo uma resposta de "OK".

Como a migração do GPS para o módulo GPS não foi possível, o bloco 2 foi desenvolvido na IDE Energia, entretanto, encontrou-se dificuldade para a comunicação dos módulos, visto que o GSM, funciona sem problemas com a configuração da placa MSP430 para SoftwareSerial, enquanto que o GPS funciona em Hardware Serial. Para essa comunicação, é necessário a criação de uma biblioteca SoftwareSerial, o que se torna inviável devido a problemas com a memória da placa.

Através dessas mesmas pesquisas realizadas pela dupla, descobriu-se com o decorrer do projeto (impossibilitando a troca do mesmo), que projetos que envolvem o uso de módulo GPS, não utilizam a placa MSP430G2553IN20 utilizada na matéria, devido a pouca memória, o que impossibilitou por exemplo, o cálculo da distância de um ponto ao outro para a ativação do alarme, e a conversão de char para string com os valores de longitude e latitude para retorno do link para o Google Maps, pronto para envio por SMS, mesmo com a dupla retirando do código fonte tudo que não seria usado, deixando

apenas partes do código utilizadas para obtenção e conversão da latitude, longitude e o link pretendido para o Google Maps.

Como os dois módulos utilizam comunicação UART, foi necessário criar uma porta para comunicação entre o GPS e a placa, essa comunicação foi criada, entretanto com os problemas acima citados, não foi possível seu teste por completo para a conclusão do projeto.

V. Conclusão

O objetivo principal do projeto não foi alcançado, que era o envio da localização do usuário monitorado através de SMS para o celular do responsável, entretanto vale ressaltar que apenas não foi completa a comunicação entre módulos, todo o restante do projeto foi concluído como proposto.

A principal limitação encontrada, foi a memória do microcontrolador utilizado, prejudicando o uso do módulo GPS com as funções pretendidas, além da codificação utilizando o software CCS, o que impossibilitou a comunicação entre o módulo GPS e o GSM, o que representaria a conclusão do projeto proposto em sua totalidade.

Os problemas não foram solucionados a tempo para a apresentação do projeto final, sendo apresentado um protótipo onde a funcionalidade do bloco 1, foi a proposta e cumprida, onde através de comunicação bluetooth, o responsável consegue gravar os dados no display da pulseira localizadora, enquanto que para o bloco 2, faltou a interação entre os módulos, onde os dados obtidos pelo GPS, retornaria um link para o Google Maps, enviado por SMS pelo módulo GSM, ultrapassando um limite pré-estabelecido ou pela vontade do responsável.

Verifica-se desta maneira que é necessário aprimoramento futuro do projeto em questão, para que o mesmo seja concluído em sua totalidade e cumpra com o prometido, e contribuir para a solução e/ou combate a um problema existente na sociedade.

VI. Referências Bibliográficas

[1] Os Números do Alzheimer no BRasil e no mundo. INNOVARE PESQUISA. Disponível em: <http://www.innovarepesquisa.com.br/blog/os-numeros-alzheimer-brasil-e-mundo/>. Acesso em: 24 de Novembro de 2017

[2] Instituto Brasileiro de Geografia e Estatística. Síntese de Indicadores Sociais Uma Análise das Condições de Vida da População Brasileira (2012). Rio de Janeiro: IBGE.

[3] "1155 crianças se perderam nas praias de SP e RJ verão passado: o que fazer?". UOL. Disponível em: <https://estilo.uol.com.br/gravidez-e-filhos/noticias/redacao/2017/01/20/1155-criancas-se-perderam-nas-praias-de-sp-e-rj-verao-passado-o-que-fazer.htm>. Acesso em: 24 de Novembro de 2017

[4] "Doença de Alzheimer: identificação em colar ou pulseira". PÚBLICO. Disponível em: <https://www.publico.pt/2008/12/06/sociedade/noticia/doenca-de-alzheimer-identificacao-em-colar-ou-pulseira-pode-ajudar-doentes-perdidos-1352299>. Acesso em: 25 de Novembro de 2017

[5] "Cresce em 40% número de crianças que se perdem nas praias". Jornal Noroeste. Disponível em: <http://www.jornalnoroeste.com/ExibeNoticia/91/5747/-cresce-em-40-n-mero-de-crian-as-que-se-perdem-nas-praias.html>. Acesso em: 25 de Novembro de 2017

[6] Terminal Bluetooth. Disponivel em: <https://play.google.com/store/apps/details?id=project.bluetoothterminal&hl=pt_BR>. Acesso em: 25 de Novembro de 2017

[7] Adding Bluetooth for MSP430 Project. Disponivel em: < http://www.msp430launchpad.com/2011/08/adding-bluetooth-to-your-m sp430-project.html>. Acesso em: 20 de Outubro de 2017

[8] Prática 8 – Comunicação SPI. Universidade Federal de Uberlândia. Disponível em: <http://www.alan.eng.br/disc_microprocessadores/pratica8_spi.pdf>. Acesso em: 24 de Novembro de 2017

[9] Sending SMS using MSP430. Texas Instruments–Fórum. Disponivel em:<https://e2e.ti.com/support/microcontrollers/msp430/f/166/t/37959>.Acesso em: 22 de Outubro de 2017

[10] Research Design Lab. GSM Interfacing with MSP430. Disponivel em: < https://researchdesignlab.com/gsm-interfacing-msp430.html>. Acesso em: 23 de Outubro de 2017

[11] GPS Library – Fórum 43oh. Disponivel em: < http://forum.43oh.com/topic/3643-gps-library-for-msp430g2553/>. Acesso em: 15 de Outubro de 2017

[12] GitHub. Vinicius Ribeiro. Disponivel em: < https://github.com/viniciusribeiro95/Microcontrolador/tree/master/3_Trabalho/C%C3%B3digos>.

Anexo I

Código para comunicação bluetooth e display Nokia 5110.

```
#include <msp430.h>
#include "PCD8544.h"

#define LCD5110_SCLK_PIN BIT5
#define LCD5110_DN_PIN BIT7
#define LCD5110_SCE_PIN BIT0
#define LCD5110_DC_PIN BIT3
#define LCD5110_SELECT P1OUT &= ~LCD5110_SCE_PIN
#define LCD5110_DESELECT P1OUT |= LCD5110_SCE_PIN
#define LCD5110_SET_COMMAND P1OUT &=
~LCD5110_DC_PIN
#define LCD5110_SET_DATA P1OUT |= LCD5110_DC_PIN
#define LCD5110_COMMAND 0
#define LCD5110_DATA 1

#define SPI_MSB_FIRST UCB0CTL0 |= UCMSB // or UCA0CTL0
|= UCMSB (USCIA) or USICTL0 &= ~USILSB (USI)
#define SPI_LSB_FIRST UCB0CTL0 &= ~UCMSB // or
UCA0CTL0 &= ~UCMSB or USICTL0 |= USILSB (USI)

void writeStringToLCD(const char *string);
void writeCharToLCD(char c);
//void writeBlockToLCD(char *byte, unsigned char length);
//void writeGraphicToLCD(char *byte, unsigned char transform);
void writeToLCD(unsigned char dataCommand, unsigned char data);
void clearLCD();
void clearBank(unsigned char bank);
void setAddr(unsigned char xAddr, unsigned char yAddr);
void initLCD();
```

```
unsigned char currXAddr = 0; //TODO this will be used for tracking current addr
unsigned char currYAddr = 0; //not implemented

//char testBlock[8] = {0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3F, 0x7F, 0xFF};
char testBlock[8] = {0x00, 0x7F, 0x7F, 0x33, 0x33, 0x03, 0x03, 0x03};
char testBlock2[8] = {0x00, 0x18, 0x18, 0x18, 0x7E, 0x3C, 0x18, 0x00};

int uart_getchar(void)
{
  int chr = -1;

  if (IFG2 & UCA0RXIFG) {
    chr = UCA0RXBUF;
 }

  return chr;
}

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;         // Stop Watchdog
  DCOCTL = 0;                       // Select lowest DCO settings
  BCSCTL1 = CALBC1_1MHZ;            // Set DCO to 1 MHz
  DCOCTL = CALDCO_1MHZ;

  P1SEL = BIT1 + BIT2 ;             // Select UART RX/TX function on P1.1,P1.2
  P1SEL2 = BIT1 + BIT2;

  UCA0CTL1 |= UCSSEL_2;             // UART Clock -> SMCLK
  UCA0BR0 = 104;                    // Baud Rate Setting for 1MHz 9600
  UCA0BR1 = 0;                      // Baud Rate Setting for 1MHz 9600
  UCA0MCTL = UCBRS_1;               // Modulation Setting for 1MHz 9600
  UCA0CTL1 &= ~UCSWRST;             // Initialize UART Module

  P1OUT |= LCD5110_SCE_PIN + LCD5110_DC_PIN; // + ALARME;
    P1DIR |= LCD5110_SCE_PIN + LCD5110_DC_PIN;// + ALARME;
    //  P1DIR &= ~BTN;
     //P1REN |= BTN;
     //P1IES |= BTN;
     //P1IE |= BTN;
     _BIS_SR(GIE);
     // setup USIB
     P1SEL |= LCD5110_SCLK_PIN + LCD5110_DN_PIN;
       P1SEL2 |= LCD5110_SCLK_PIN + LCD5110_DN_PIN;

     UCB0CTL0 |= UCCKPH + UCMSB + UCMST + UCSYNC; // 3-pin, 8-bit SPI master
     UCB0CTL1 |= UCSSEL_2; // SMCLK
     UCB0BR0 |= 0x01; // 1:1
     UCB0BR1 = 0;
     UCB0CTL1 &= ~UCSWRST; // clear SW

     _delay_cycles(500000);
     initLCD();
     clearLCD();

     // LCD test
     int j = 0;
     const char *nome;
     const char *idade;
     const char *responsavel;
     const char *contato;

for(j =0; j < 4; j++)
{
   char chr;
  const char *text;
     while (chr != '\0' )
   {

       chr = uart_getchar();
       if(chr != -1)
       {
        text += chr;
       }
   }


     if(j==0)
     {
       nome = text;
     }
     if(j==1)
     {
       idade = text;
     }
     if(j==2)
     {
      responsavel = text;
     }
     if(j==3)
```

```
      {
       contato = text;
      }
}
for(;;){
 clearBank(0);
 writeStringToLCD("Nome:");
 clearBank(1);
 writeStringToLCD(nome);
 clearBank(2);
 clearBank(3);
 writeStringToLCD("Idade:");
 clearBank(4);
 writeStringToLCD(idade);
 _delay_cycles(2500000);

 clearLCD();
 clearBank(0);
 writeStringToLCD("Responsavel:");
 clearBank(1);
 writeStringToLCD(responsavel);
 clearBank(2);
 clearBank(3);
 writeStringToLCD("Contato:");
 clearBank(4);
 writeStringToLCD(contato);
 _delay_cycles(2500000);

 }
}



void writeStringToLCD(const char *string) {
   while(*string) {
      writeCharToLCD(*string++);
   }
}

void writeCharToLCD(char c) {
   unsigned char i;
   for(i = 0; i < 5; i++) {
      writeToLCD(LCD5110_DATA, font[c - 0x20][i]);
   }
   writeToLCD(LCD5110_DATA, 0);
}

void writeToLCD(unsigned char dataCommand, unsigned char data) {
   LCD5110_SELECT;
   if(dataCommand) {
      LCD5110_SET_DATA;
   } else {
      LCD5110_SET_COMMAND;
   }
   UCB0TXBUF = data;
   while(!(IFG2 & UCB0TXIFG));
   LCD5110_DESELECT;
}

void clearLCD() {
   setAddr(0, 0);
   int c = 0;
   while(c < PCD8544_MAXBYTES) {
      writeToLCD(LCD5110_DATA, 0);
      c++;
   }
   setAddr(0, 0);
}

void clearBank(unsigned char bank) {
   setAddr(0, bank);
   int c = 0;
   while(c < PCD8544_HPIXELS) {
      writeToLCD(LCD5110_DATA, 0);
      c++;
   }
   setAddr(0, bank);
}

void setAddr(unsigned char xAddr, unsigned char yAddr) {
   writeToLCD(LCD5110_COMMAND, PCD8544_SETXADDR |
xAddr);
   writeToLCD(LCD5110_COMMAND, PCD8544_SETYADDR |
yAddr);
}

void initLCD() {
   writeToLCD(LCD5110_COMMAND, PCD8544_FUNCTIONSET
| PCD8544_EXTENDEDINSTRUCTION);
   writeToLCD(LCD5110_COMMAND, PCD8544_SETVOP |
0x3F);
   writeToLCD(LCD5110_COMMAND, PCD8544_SETTEMP |
0x02);
   writeToLCD(LCD5110_COMMAND, PCD8544_SETBIAS |
0x03);
   writeToLCD(LCD5110_COMMAND,
PCD8544_FUNCTIONSET);
   writeToLCD(LCD5110_COMMAND,
PCD8544_DISPLAYCONTROL | PCD8544_DISPLAYNORMAL);
}
```

Anexo II

Comunicação UART criada para receber dados do GPS.

```
#include "msp430g2231.h"
```

```
#include "stdbool.h"

#define		TXD		BIT1
#define		RXD		BIT2
#define		Bit_time		104
#define		Bit_time_5		52

unsigned char BitCnt;
unsigned int TXByte;
unsigned int RXByte;
bool isReceiving;
bool hasReceived;

void Transmit(void);

void main(void)
{
	WDTCTL = WDTPW + WDTHOLD;

	BCSCTL1 = CALBC1_1MHZ;
	DCOCTL = CALDCO_1MHZ;

	P1SEL |= TXD;
	P1DIR |= TXD;

	P1IES |= RXD;
	P1IFG &= ~RXD;
	P1IE |= RXD;
	isReceiving = false;
	hasReceived = false;

	__bis_SR_register(GIE);
	while(1)
	{
	   if (hasReceived)
	   {
			hasReceived = false;
			TXByte = RXByte;
			Transmit();
	   }
	   if (~hasReceived)
			__bis_SR_register(CPUOFF + GIE);

	}
}

// Function Transmits Character from TXByte
void Transmit()
{
	while(isReceiving);
	CCTL0 = OUT;
	TACTL = TASSEL_2 + MC_2;
	BitCnt = 0xA;
	CCR0 = TAR;

	CCR0 += Bit_time;
	TXByte |= 0x100;
	TXByte = TXByte << 1;

	CCTL0 =  CCIS0 + OUTMOD0 + CCIE;
	while ( CCTL0 & CCIE );
}

// Port 1 interrupt service routine
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{
	isReceiving = true;

	P1IE &= ~RXD;
	P1IFG &= ~RXD;

	TACTL = TASSEL_2 + MC_2;
	CCR0 = TAR;
	CCR0 += Bit_time_5;
	CCTL0 = OUTMOD1 + CCIE;

	RXByte = 0;
	BitCnt = 0x9;


#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
	if(!isReceiving)
	{
	   CCR0 += Bit_time;
	   if ( BitCnt == 0)
			TACTL = TASSEL_2;
			CCTL0 &= ~ CCIE ;
	   }
	   else
```

```
        {
                CCTL0 |=  OUTMOD2;
                if (TXByte & 0x01)
                        CCTL0 &= ~ OUTMOD2;
                TXByte = TXByte >> 1;
                BitCnt --;
        }
    }
    else
    {
        CCR0 += Bit_time;
        if ( BitCnt == 0)
        {
                TACTL = TASSEL_2;
                CCTL0 &= ~ CCIE ;

                isReceiving = false;

                P1IFG &= ~RXD;
                P1IE |= RXD;

                if ( (RXByte & 0x201) == 0x200)
                {
                        RXByte = RXByte >> 1;
                        RXByte &= 0xFF;
                        hasReceived = true;
                }
                __bic_SR_register_on_exit(CPUOFF);
        }
        else
        {
                if ( (P1IN & RXD) == RXD)
                        RXByte |= 0x400;
                RXByte = RXByte >> 1;
                BitCnt --;
        }
    }
}
```

Anexo III

Código para módulo GSM no Code Composer.

```
#include "msp430g2553.h"

#include "uart.h"

void gsm();

int main(void)

{

WDTCTL = WDTPW + WDTHOLD;

BCSCTL1 = CALBC1_8MHZ;

DCOCTL = CALDCO_8MHZ;

 uart_init();

__enable_interrupt();

__delay_cycles(100000);

gsm();

}

void gsm()

{

 uart_puts((char *)"AT"); // COMMAND FOR INITIALIZING GSM

          uart_putc(0x0A);//ENTER

          uart_putc(0x0D);//CARRIAGE RETURN

  __delay_cycles(10000000);//DELAY...WAIT FOR OK FROM GSM

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

          uart_putc(0x0A);

          uart_putc(0x0D);

         __delay_cycles(10000000);//WAIT FOR OK

          uart_puts((char *)"AT+CMGS=\"+5561993852020\"");
uart_putc(0x0A);

          uart_putc(0x0D);

          uart_puts((char *)"TEST");

         uart_putc(0x1A);

}
```

Anexo IV

Código para módulo GPS na IDE Energia.

```
#include <TinyGPS++.h>

TinyGPSPlus gps;

float latitude;
float 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);
       }
}

                  static void smartDelay(unsigned long ms)
{
  unsigned long start = millis();
  do
  {
    while (Serial.available())
      gps.encode(Serial.read());
  } while (millis() - start < ms);
}
```

Anexo V

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 =\"061995523211\"\r");
    delay(500);
    Serial.print(latitude);
   Serial.print(longitude);

   Serial.print("\r");
   delay(500);
    Serial.print(0x1A);
}
```