Relatório Final
Coleira Inteligente

Hércules I. d. A. Santos

16/0124450
Faculdade do Gama
Universidade de Brasília
Gama - DF, Brasil
ismael-456@hotmail.com

Joselito Prado Marques da Silva

14/0023704
Faculdade do Gama
Universidade de Brasília
Gama - DF, Brasil
joselito.prado.marques@gmail.com

***Abstract*—This project consists in the development of a smart pet collar with GPS which sends localization data, in case of the pet get lost.**

***Keywords*—Smart Collar, MSP430, embedded systems, eletronic, low power, pet**

***Resumo* – Esse projeto consiste no desenvolvimento de uma coleira inteligente com rastreamento GPS que envia dados de localização do animal para, caso ele se perca, seja achado com facilidade.**

***Palavras-chaves* – coleira inteligente, MSP430, sistemas embarcados, eletrônica, baixa potência, animais**

I. Introdução

Quando um animal se perde, os únicos recursos que se pode contar são redes sociais e aplicativos nos quais se pode colocar os dados do animal doméstico perdido. Para que se acelere o processo de localização do animal, o intuito desse projeto é desenvolver uma coleira capaz de rastrear o animal.

Nos dispositivos inteligentes que monitoram animais, sejam domésticos ou selvagens, verificou-se que os parâmetros de localização foram acompanhados para obter informações do bando para saber quando ele estava próximo das plantações (CAMBROM, 2014), outro estudo desejava compreender o comportamento de deslocamento dos animais (SONG, 2011). Outro estudo utilizou os animais para capturar parâmetros ambientais como a temperatura, umidade, orientação da cabeça do animal e luz ambiente monitoradas pela coleira do animal (JAIN, 2008).

Há dispositivos focados em animais domésticos que monitoram batimentos cardíacos, temperatura e a localização cuja comunicação é realizada sem fio (BRYON, 2016).

Para dispositivos cujo intuito seja localizar o pet perdido, encontra-se disponíveis no mercado, porém com um preço exorbitante.

Segundo a revista LIDE, o mercado de pet no Brasil é o terceiro maior em faturamento mesmo em um período de recessão econômica e a estimativa é que mais de R$ 25 bilhões tenham circulado na área no segmento. Assim, acreditamos que esse é um projeto viável em um segmento que está em pleno crescimento e que ainda pode ser muito explorado.

II. Desenvolvimento

1. Solução

Para se ter a localização do pet, independente de uma conexão *bluetooth* ou uma rede *wifi*, projetou-se uma coleira que conta com um módulo GPS para receber a localização do animal, independente de onde esteja, e um comunicador GSM para que essa localização seja enviada ao dono, também, independente de onde o pet esteja.

2. Descrição do Hardware

2.1. Bill of Materials (BOM):

- 1 - MSP430G2553
- 1 - NEO-6M GPS
- 1 - GSM SIM 800L
- 1 - Potenciômetro de qualquer valor
- 1 - LCD 16x02
- 1 - LM7805

- 1 - Regulador de Tensão de 3V3

2.2. Esquemáticos

2.2.1. Esquemático Teórico

O esquemático teórico utilizando o GPS e o GSM pode ser visualizado na figura 1 (apêndice). Esse circuito utiliza dois multiplexadores 2x1 e um regulador de tensão de 3V3 para que se possa alimentar os módulos GSM e GPS, além do MSP430.

A pinagem utilizada pelo GPS e GSM podem ser visualizados também no código, em suas respectivas bibliotecas.

2.2.2. Esquemático Experimental

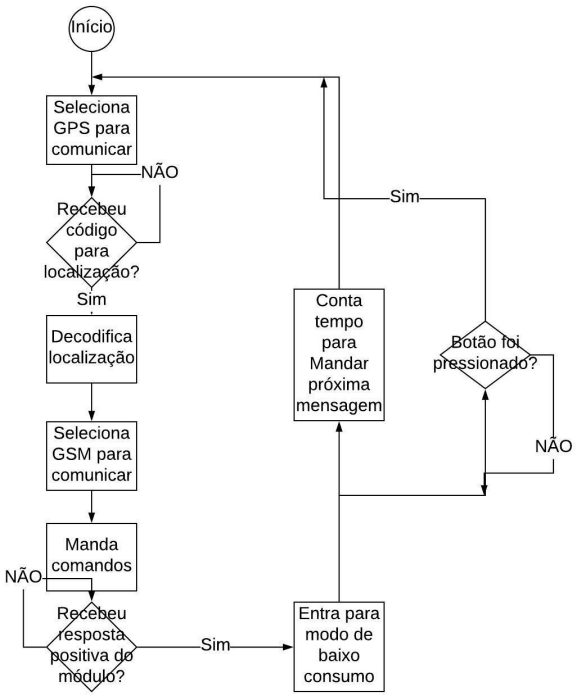
Como não se utilizou o GSM no circuito prático, utilizou-se um display LCD 16x02 para visualizar os dados de latitude e longitude para validar os dados advindos do sensor GPS.

Esse circuito utilizou uma pilha de 9V e dois reguladores de tensão, um de 5V para o display LCD e outro de 3V3 para alimentar o módulo GPS e o microcontrolador MSP430. Além disso, o potenciômetro utilizado não precisa ter um valor pré determinado pois a função dele é apenas ajustar o nível de contraste do display LCD.

A pinagem utilizada pelo GPS e display LCD podem ser visualizados também no código, em suas respectivas bibliotecas.

3. Descrição do SOFTWARE

O código para este projeto foi pensado com a intenção de se economizar o máximo de energia possível, isso por causa da natureza do projeto, o fato da coleira inteligente não estar conectada a energia, e a necessidade de se ter um caminho para procurar o animal perdido demanda uma eficiência energética elevada. Para isto, foi utilizado o modo de baixo consumo na MSP430, de modo que no tempo em que o microcontrolador estiver ocioso, não se precisará de outra função além de contar o tempo para a CPU ser ligada. No caso deste projeto, a idéia que se teve foi colocar intervalos entre 1h e 3h entre os envios da localização do animal, assim o microcontrolador não precisa ficar completamente ativo todo o tempo, mas apenas em pequenos intervalos durante o dia, passando a maior parte do tempo em modo de baixo consumo.



Ao ativar a CPU para mandar a localização para o dono, o microcontrolador deve se comunicar com o módulo de GPS, e com o módulo GSM. Esta comunicação é serial, e é feita pelo protocolo UART, para ambos os módulos. A MSP430 conta com apenas uma interface para comunicação UART, com uma porta de transmissão de dados (TX), e uma porta de recepção de dados(RX), porém, precisa-se comunicar com dois dispositivos. A solução para isto foi utilizar um multiplexador de dados, para selecionar o dispositivo que se quer comunicar em cada momento. A seleção do sinal é feita pelo controlador, por uma porta de saída digital que funcionará como seletora do multiplexador.

Então a sequência que teremos para o código quando a MSP430 estiver ativa será, primeiro, adquirir a localização com o GPS, decodificar os dados obtidos, e então mandar uma mensagem ao dono por meio do módulo GSM. É importante salientar que ambos os módulos possuem protocolos específicos para a comunicação, sendo para o GPS o protocolo NMEA, e para o GSM os comandos AT. Então para cada um desses protocolos, é necessário validar as informações, garantindo que elas respeitem as especificações destes protocolos, para então decodificar os dados do GPS e mandar a mensagem pelo GSM.

Também foi implementado um botão para mandar a localização ao dono, afora os horários já implementados

para ser mandada a localização automaticamente. Assim o projeto permite não só uma operação automática, mas uma operação manual de mandar a localização ao dono. Esta funcionalidade permite também conferir a integridade do dispositivo, sem que se precise esperar longos períodos em que ele ligue automaticamente.

Foi concebido também um código que simula os mesmos efeitos do código descrito anteriormente, porém com o módulo GSM substituído por um display LCD. Este código foi feito como uma forma mais simples do projeto.

III. Resultados

Para a primeira fase do projeto testamos os sensores necessários para a execução do projeto, de modo a conseguir fazer a comunicação entre os sensores e o microcontrolador, garantindo que as informações sejam extraídas dos sensores e possam ser utilizadas de forma adequada, garantindo um bom funcionamento do equipamento.

Foi utilizada nesta fase, a placa MSP430G2553, e foram testados 2 sensores, o primeiro foi o GPS GY-NEO6MV2, que irá indicar a posição do animal de estimação a ser encontrado, e o sensor MPU6050, um acelerômetro conjugado com um giroscópio. Este sensor tem a funcionalidade de indicar a orientação do animal, e monitorar alterações na rotina de movimentação do animal.

Para extrair os dados destes sensores utilizamos a ferramenta Energia, do qual escreve-se um código para Arduino, e este é traduzido para o MSP430, de modo que pode-se ter uma análise prática do funcionamento dos sensores.

Os códigos se basearam em simplesmente extrair os dados dos sensores, e escrevê-los na tela de um computador, para que se possa visualizar os resultados dos estímulos nos sensores.

Foram utilizadas bibliotecas específicas aos sensores para se obter os dados dos sensores.

Na segunda fase, implementou-se a interface entre o GPS e o MSP430G2553, por código em C para por meio do IAR Embedded Workbench IDE para msp430.

Foi testada a comunicação entre o GPS e o msp430, e teve-se sucesso em obter-se informações do GPS. Foi utilizado o protocolo UART para fazer a comunicação. Percebeu-se, porém, uma instabilidade em se obter os valores para lugares fechados, recebendo-se neste caso apenas o horário GMT.

Foi implementado o interrupt na linha 161 do Código GPS (PARA MS430). Essa sessão do código descrever como foi executado o processo de temporização do código, que, no caso, definirá com qual frequência o GSM irá mandar o sinal de localização do GPS da mesma forma que define de quanto em quanto tempo o GPS coletará os seus dados. O intervalo escolhido foi de um minuto.

Além da temporização do envio do sinal do GPS, fez-se a definição dos modos de baixo consumo para os intervalos nos quais o GPS não coleta os seus dados nem o GSM envia o seu sinal, assim, otimizando o consumo de potência para o sistema, fazendo com que o sistema dure mais tempo sem uma necessária recarga. Essa sessão pode ser encontrada na linha 48 do arquivo Código GPS (PARA MSP430).

Para a última fase implementou-se o GPS, com um display LCD, de modo que a localização foi mostrada no display, junto com a hora para o meridiano de Greenwich, e a data. Esta adaptação ocorreu pois não conseguiu-se fazer a comunicação correta com o módulo do GSM, problema este que pode se derivar de fatores como a alimentação do projeto, falha na configuração da comunicação em si, ou falha no próprio módulo.

Assim, conseguiu-se implementar com sucesso a localização com display, utilizando modos de baixo consumo, e interrupções com botão, adquirindo a posição do GPS corretamente no display

IV. Conclusão

Com isso, a problemática de achar animais domésticos perdidos, demonstrou-se um problema minimizável com o uso de uma coleira inteligente para rastrear animal, e com respaldo bibliográfico que mostra ser viável a proposta. Por limitações da equipe, e de equipamentos, foi necessário simplificar o projeto com o intuito de mostrar resultados concretos para os prazos definidos, porém, para o projeto simplificado adquiriu-se sucesso na aplicação com display LCD.

Referências

[1] Cambron, Mark E., and Michael Stokes. "Design of elephant collars to reduce crop foraging." *SOUTHEASTCON 2014, IEEE*. IEEE, 2014.

[2] Jain, Vishwas Raj, et al. "wildCENSE: GPS based animal tracking system." *Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008. International Conference on*. IEEE, 2008.

[3] Song, Yang, Yu-xi Liu, and Ming-hao Gu. "Design of new electrical collar based on the technology of Zigbee." *Computational Intelligence and Design (ISCID), 2011 Fourth International Symposium on*. Vol. 1. IEEE, 2011.

[4] Bryon Walsh, Dustin DeCarlo, Steve Heagney, Stephanie Heagney. "*Doggy Pal Collar*". University of Central California, 2016.

[5] Em crise: mercado de pets no Brasil é o terceiro do mundo em faturamento. Revista LIDE, São Paulo. V. 68, 2018

Apêndice

Código para o teste do GPS(Energia):

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

//Pinos utilizados para conexao do modulo
GY-NEO6MV2
static const int RXPin = 4, TXPin = 3;

//Objeto TinyGPS++
TinyGPSPlus gps;

//Conexao serial do modulo GPS
SoftwareSerial Serial_GPS(RXPin, TXPin);


//Ajuste o timezone de acordo com a regiao
const int UTC_offset = -3;

void setup()
{
 //Baud rate Arduino
 Serial.begin(115200);
 //Baud rate Modulo GPS
 Serial_GPS.begin(9600);

 //Mostra informacoes iniciais no serial monitor
 Serial.println(F("Data, Hora, Latitude e Longitude"));
 Serial.println(F("Modulo GPS GY-NEO6MV2"));
 Serial.print(F("Biblioteca TinyGPS++ v. "));
 Serial.println(TinyGPSPlus::libraryVersion());
 Serial.println();

void loop()
{
 //Conexao com modulo GPS
 while (Serial_GPS.available() > 0)
   if (gps.encode(Serial_GPS.read()))
     displayInfo();

 if (millis() > 5000 && gps.charsProcessed() < 10)
 {
  Serial.println(F("No GPS detected: check wiring."));
  while (true);
 }


void displayInfo()
{
 //Mostra informacoes no Serial Monitor
 Serial.print(F("Location: "));
 if (gps.location.isValid())
 {
  Serial.print(gps.location.lat(), 6); //latitude
  Serial.print(F(","));
  Serial.print(gps.location.lng(), 6); //longitude
 }
 else
 {
  Serial.print(F("INVALID"));
 }

 Serial.print(F("  Date/Time: "));
 if (gps.date.isValid())
 {
  Serial.print(gps.date.day()); //dia
  Serial.print(F("/"));
  Serial.print(gps.date.month()); //mes
  Serial.print(F("/"));
  Serial.print(gps.date.year()); //ano
 }
 else
 {
  Serial.print(F("INVALID"));
 }

 Serial.print(F(" "));
 if (gps.time.isValid())
 {
  if (gps.time.hour() < 10) Serial.print(F("0"));
  Serial.print(gps.time.hour()); //hora
  Serial.print(F(":"));
```

```
    if (gps.time.minute() < 10) Serial.print(F("0"));
    Serial.print(gps.time.minute()); //minuto
    Serial.print(F(":"));
    if (gps.time.second() < 10) Serial.print(F("0"));
    Serial.print(gps.time.second()); //segundo
    Serial.print(F("."));
    if (gps.time.centisecond() < 10) Serial.print(F("0"));
    Serial.print(gps.time.centisecond());
   }
   else
   {
    Serial.print(F("INVALID"));
   }
   Serial.println();
}

void GPS_Timezone_Adjust()
{
  while (Serial_GPS.available())
  {
   if (gps.encode(Serial_GPS.read()))
   {
    int Year = gps.date.year();
    byte Month = gps.date.month();
    byte Day = gps.date.day();
    byte Hour = gps.time.hour();
    byte Minute = gps.time.minute();
    byte Second = gps.time.second();

    //Ajusta data e hora a partir dos dados do GPS
    setTime(Hour, Minute, Second, Day, Month, Year);
    //Aplica offset para ajustar data e hora
    //de acordo com a timezone
    adjustTime(UTC_offset * SECS_PER_HOUR);
   }
  }
}
```

Código GPS (C para MSP430):

```
#include "msp430g2553.h"
#include "uart.h" // ATTACH THE UART FILE WITH THE MAIN CODE
#include <string.h>

#define RMC_SENT_LEN 57
#define BTN BIT3

typedef struct{
 char Time[11];
 char Lat[10];
 char Long[11];
 char Date[7];
} gps;

int gsm(gps *myGPS);//FUNCTION PROTOTYPE
void get_gps(gps *myGPS);
int confirmaAT();



int main(void)

{

   gps myGPS;



   WDTCTL = WDTPW + WDTHOLD;// PARAR WATCHDOG TIMER

   BCSCTL1 = CALBC1_8MHZ;// FREQUENCIA DA MSP430 A 8MHz
   DCOCTL = CALDCO_8MHZ;

   P1DIR &= ~BTN;
   P1REN |= BTN;
   P1OUT |= BTN;

   TACTL0= TASSEL_1 + ID_1 + MC_2 + TAIE;

   uart_init(); // CALL THE UART INIT FUNCTION WHICH IS AVAILAIBLE IN THE FILE

   //_bis_SR_register(GIE);
   _bis_SR_register(GIE+LPM1_bits);

while(1)
 {
  if((P1IN&BTN)==0){
   __delay_cycles(100000);
   get_gps(&myGPS);
    while(gsm(&myGPS)!=0);// CALL THE GSM FUNCTION
    }
 }
}



int gsm(gps *myGPS){
    uart_puts((char *)"AT"); // COMMAND FOR INITIALIZING GSM
    uart_putc(0x0A);//ENTER
    uart_putc(0x0D);//CARRIAGE RETURN

    if(confirmaAT()!=0)
     return 1;

 //    __delay_cycles(10000000);//DELAY...WAIT FOR OK FROM GSM
    uart_puts((char *)"AT+CMGF=1");//COMMUNICATION
    uart_putc(0x0A);
    uart_putc(0x0D);


    if(confirmaAT()!=0)
     return 1;
  //  __delay_cycles(10000000);//WAIT FOR OK

    uart_puts((char *)"AT+CMGS=\"61996356120\"");//SEND A MESSAGE TO PARTICULAR NUMBER
    uart_putc(0x0A);
    uart_putc(0x0D);

    uart_puts(myGPS->Date);
    uart_puts(myGPS->Lat);
    uart_puts(myGPS->Long);
    uart_puts(myGPS->Time);

    uart_putc(0x1A);//CTRL Z

    //AFTER HARDWARE CONFIGURATION THE MESSAGE WILL GET SEND

   //ATTACH THE UART FILES OR WRITE THE CODE FOR INIT AND SENDING MESSAGE IN THE SAME FILE...
    return 0;

}
void get_gps(gps *myGPS){
```

```
 char temp_sentence[RMC_SENT_LEN];
 char temp_char;
 for(;;){
       temp_char = uart_getc();  //Collect GPS Data
       if (temp_char == '$') {
          temp_char = uart_getc();
          if (temp_char == 'G') {
             temp_char = uart_getc();
             if (temp_char == 'P') {
                temp_char = uart_getc();
                if (temp_char == 'R') {
                   temp_char = uart_getc();
                   if (temp_char == 'M') {
                      temp_char = uart_getc();
                      if (temp_char == 'C') {
 //Code to save specific portions of GPS DATA Stream
                         uart_gets(temp_sentence,
RMC_SENT_LEN);
                         strncpy(myGPS->Time,
(temp_sentence), 11);
                         strncpy(myGPS->Lat,
(temp_sentence + 13), 10);
                         strncpy(myGPS->Long,
(temp_sentence + 25), 11);
                         strncpy(myGPS->Date,
(temp_sentence + 50), 7);
                         return;
                      }
                   }
                }
             }
          }
       }
    }
}

int confirmaAT(){
  int n;
  char status[11];
  status[0] = '\r';
  while(uart_getc()!='\r');
  for(n=1; n<=9; n++){
    status[n] = uart_getc();
    if(status[n]=='\n'){
     status[n+1]='\0';
     n=9;
    }
  }

  return strcmp(status, "\r\nOK\r\n");
}

interrupt(TIMER0_A1_VECTOR
manda_localicacao(void){
  static int cont = 0;
  gps myGPS;

  cont++;
  if(cont>=15){
// Para CLK 32768 dividido por 2 no timer, 15 contagens
será o tempo de 1 min
    get_gps(&myGPS);
    gsm(&myGPS);
    cont=0;
    }
}
```

Código GPS + LCD (C para MSP430):

```
#include "msp430g2553.h"
#include "uart.c"
#include <string.h>
#include "lcd.c"
#define RMC_SENT_LEN 57
#define BTN BIT3

typedef struct{
 char Time[12];
 char Lat[14];
 char Long[15];
 char Date[8];
} gps;

void get_gps(gps *myGPS);
void manda_local();
void confirma();

int main(void){
    WDTCTL = WDTPW + WDTHOLD;
    BCSCTL1 = CALBC1_8MHZ;
    DCOCTL = CALDCO_8MHZ;

    P1DIR &= ~BTN;
    P1REN |= BTN;
    P1OUT |= BTN;
    P1IES |= BTN;
    P1IE  |= BTN;

    InitLCD();
    CLR_DISPLAY;
    Send_String("GPS:");
    Send_String(" ok");
    confirma();
    manda_local();

    TA0CTL= TASSEL_2 + ID_3 + MC_2 + TAIE;
   __bis_SR_register(GIE+LPM0_bits);
```

```
  while(1);
}

void manda_local(){
 gps myGPS;
 int cont;

 get_gps(&myGPS);
 CLR_DISPLAY;
 for(cont=0;cont<6;cont++){
  Send_Data(myGPS.Time[cont]);
  if(((cont%2)!=0)&&(cont!=5))
   Send_Data(':');
 }
 confirma();
 CLR_DISPLAY;
 for(cont=0;cont<6;cont++){
  Send_Data(myGPS.Date[cont]);
  if(((cont%2)!=0)&&(cont!=5))
   Send_Data('/');
 }
 confirma();
 CLR_DISPLAY;
 Send_String(myGPS.Lat);
 confirma();
 CLR_DISPLAY;
 Send_String(myGPS.Long);
 confirma();
 P1IFG &=~BTN;

 return;
}

void get_gps(gps *myGPS){
 char temp_sentence[RMC_SENT_LEN];
 char temp_char;
 int sent=0;
 int cont;
 int gps_cont=0;

 uart_init();
 __enable_interrupt();
 for(;;){
      temp_char = uart_getc();
      if (temp_char == '$') {
         temp_char = uart_getc();
         if (temp_char == 'G') {
            temp_char = uart_getc();
            if (temp_char == 'P') {
                  temp_char = uart_getc();
                  if (temp_char == 'R') {
                     temp_char = uart_getc();
                     if (temp_char == 'M') {
                        temp_char = uart_getc();
                        if (temp_char == 'C') {
                           uart_gets(temp_sentence,
RMC_SENT_LEN);

for(cont=0;cont<RMC_SENT_LEN;cont++){
                              if(temp_sentence[cont]==',')
                               sent++;
                              switch(sent){
                              case 1:{
                               if(temp_sentence[cont]!=','){

*(myGPS->Time+gps_cont)=temp_sentence[cont];
                                gps_cont++;
                               }
                               else{
                                gps_cont=0;
                               }
                               break;
                              }
                              case 3:{
                               if(temp_sentence[cont]!=','){

*(myGPS->Lat+gps_cont)=temp_sentence[cont];
                                gps_cont++;
                               }
                               else{
                                *(myGPS->Time+gps_cont)='\0';
                                gps_cont=0;
                               }
                               break;
                              }
                              case 4:{
                               if(temp_sentence[cont]!=','){

*(myGPS->Lat+gps_cont)=temp_sentence[cont];
                                gps_cont++;
                               }
                               else{
                                *(myGPS->Lat+gps_cont)='-';
                                gps_cont++;
                               }
                               break;
                              }
                              case 5:{
```

```
                        if(temp_sentence[cont]!=','){

*(myGPS->Long+gps_cont)=temp_sentence[cont];
                         gps_cont++;
                        }
                        else{
                         *(myGPS->Lat+gps_cont)='\0';
                         gps_cont=0;
                        }
                        break;
                       }
                       case 6:{
                        if(temp_sentence[cont]!=','){

*(myGPS->Long+gps_cont)=temp_sentence[cont];
                         gps_cont++;
                        }
                        else{
                         *(myGPS->Long+gps_cont)='-';
                         gps_cont++;
                        }
                        break;
                       }
                       case 9:{
                        if(temp_sentence[cont]!=','){

*(myGPS->Date+gps_cont)=temp_sentence[cont];
                         gps_cont++;
                        }
                        else{
                         *(myGPS->Long+gps_cont)='\0';
                         gps_cont=0;
                        }
                        break;
                       }
                       case 10:{
                         *(myGPS->Date+gps_cont)='\0';
                        cont=RMC_SENT_LEN;
                       }
                       }
                      }
                      IE2 &= ~UCA0RXIE;
                      __disable_interrupt();
                      return;
                    }
                  }
                }
              }
            }

        }
    }
}

void confirma(){
  Atraso_us(50000);
  while((P1IN&BTN)==0);
  while((P1IN&BTN)!=0);
}

#pragma vector=TIMER0_A0_VECTOR
__interrupt void manda_local_t (void)
{
  static int cont = 0;

  cont++;
  if(cont>=1000){ // para clk 32768 dividido por 2 no
timer, 15 contagens será o tempo de 1 min
    TA0CTL= TACLR;
    TA0CTL=0;
    manda_local();
    cont=0;
  }
  TA0CTL= TASSEL_2 + ID_3 + MC_2 + TAIE;
}

#pragma vector=PORT1_VECTOR
__interrupt void manda_local_b(void){

    P1IFG &=~BTN;
    P1IE &= ~BTN;
    manda_local();
    P1IE |= BTN;
}
```

Código GPS + GSM (C para MSP430):

```
#include <msp430g2553.h>
#include "uart.h" // ATTACH THE UART FILE WITH
THE MAIN CODE
#include <string.h>
#define RMC_SENT_LEN 57
#define BTN BIT3
#define SEL BIT4

typedef struct{
  char Time[11];
  char Lat[10];
  char Long[11];
  char Date[7];
} gps;
```

```
int gsm(gps *myGPS);//FUNCTION PROTOTYPE
void get_gps(gps *myGPS);
int confirmaAT();
void manda_local();



int main(void){
   WDTCTL = WDTPW + WDTHOLD;
   BCSCTL1 = CALBC1_8MHZ;
   DCOCTL = CALDCO_8MHZ;

   P1DIR &= ~BTN;
   P1REN |= BTN;
   P1OUT |= BTN;
   P1IES |= BTN;
   P1IE  |= BTN;

   manda_local();

   TA0CTL= TASSEL_1 + ID_1 + MC_2 + TAIE;
  __bis_SR_register(GIE+LPM3_bits);
  while(1);
}

void manda_local(){
 gps myGPS;

 get_gps(&myGPS);
 gsm(&myGPS);

 return;
}

int gsm(gps *myGPS){
     uart_init();
     __enable_interrupt();

     P1OUT |= SEL;

     uart_puts((char *)"AT"); // COMMAND FOR
INITIALIZING GSM
     uart_putc(0x0A);//ENTER
     uart_putc(0x0D);//CARRIAGE RETURN

     if(confirmaAT()!=0)
       return 1;
 //     __delay_cycles(10000000);//DELAY...WAIT FOR
OK FROM GSM
     uart_puts((char
*)"AT+CMGF=1");//COMMUNICATION
     uart_putc(0x0A);
     uart_putc(0x0D);



     if(confirmaAT()!=0)
       return 1;
  //  __delay_cycles(10000000);//WAIT FOR OK

     uart_puts((char
*)"AT+CMGS=\"61996356120\"");//SEND A MESSAGE
TO PARTICULAR NUMBER
     uart_putc(0x0A);
     uart_putc(0x0D);

     uart_puts(myGPS->Date);
     uart_puts(myGPS->Lat);
     uart_puts(myGPS->Long);
     uart_puts(myGPS->Time);

     uart_putc(0x1A);//CTRL Z

     IE2 &= ~UCA0RXIE;
     __disable_interrupt();

     //AFTER HARDWARE CONFIGURATION THE
MESSAGE WILL GET SEND

    //ATTACH THE UART FILES OR WRITE THE
CODE FOR INIT AND SENDING MESSAGE IN THE
SAME FILE...
    return 0;

}
void get_gps(gps *myGPS){
  char temp_sentence[RMC_SENT_LEN];
  char temp_char;
  int sent=0;
  int cont;
  int gps_cont=0;

  uart_init();
   __enable_interrupt();

      P1OUT &= ~SEL;
```

```
  for(;;){
      temp_char = uart_getc();  //Collect GPS Data
      if (temp_char == '$') {
         temp_char = uart_getc();
         if (temp_char == 'G') {
            temp_char = uart_getc();
            if (temp_char == 'P') {
               temp_char = uart_getc();
               if (temp_char == 'R') {
                  temp_char = uart_getc();
                  if (temp_char == 'M') {
                     temp_char = uart_getc();
                     if (temp_char == 'C') {
                        //Code to save specific portions of
GPS DATA Stream
                           uart_gets(temp_sentence,
RMC_SENT_LEN);

for(cont=0;cont<RMC_SENT_LEN;cont++){
                           if(temp_sentence[cont]==',')
                            sent++;
                           switch(sent){
                           case 1:{
                            if(temp_sentence[cont]!=','){

*(myGPS->Time+gps_cont)=temp_sentence[cont];
                             gps_cont++;
                            }
                            else{
                             gps_cont=0;
                            }
                            break;
                           }
                           case 3:{
                            if(temp_sentence[cont]!=','){

*(myGPS->Lat+gps_cont)=temp_sentence[cont];
                             gps_cont++;
                            }
                            else{
                             *(myGPS->Time+gps_cont)='\0';
                             gps_cont=0;
                            }
                            break;
                           }
                           case 4:{
                            if(temp_sentence[cont]!=','){

*(myGPS->Lat+gps_cont)=temp_sentence[cont];
                             gps_cont++;
                            }
                            else{
                             *(myGPS->Lat+gps_cont)='-';
                             gps_cont++;
                            }
                            break;
                           }
                           case 5:{
                            if(temp_sentence[cont]!=','){

*(myGPS->Long+gps_cont)=temp_sentence[cont];
                             gps_cont++;
                            }
                            else{
                             *(myGPS->Lat+gps_cont)='\0';
                             gps_cont=0;
                            }
                            break;
                           }
                           case 6:{
                            if(temp_sentence[cont]!=','){

*(myGPS->Long+gps_cont)=temp_sentence[cont];
                             gps_cont++;
                            }
                            else{
                             *(myGPS->Long+gps_cont)='-';
                             gps_cont++;
                            }
                            break;
                           }
                           case 9:{
                            if(temp_sentence[cont]!=','){

*(myGPS->Date+gps_cont)=temp_sentence[cont];
                             gps_cont++;
                            }
                            else{
                             *(myGPS->Long+gps_cont)='\0';
                             gps_cont=0;
                            }
                            break;
                           }
                           case 10:{
                             *(myGPS->Date+gps_cont)='\0';
                            cont=RMC_SENT_LEN;
                           }
```

```
                        }
                    }
                        IE2 &= ~UCA0RXIE;
                    __disable_interrupt();

                    return;
                }
            }
        }
    }
  }
}

int confirmaAT(){
  int n;
  char status[11];
  status[0] = '\r';
  while(uart_getc()!='\r');
  for(n=1; n<=9; n++){
    status[n] = uart_getc();
    if(status[n]=='\n'){
            status[n+1]='\0';
            n=9;
    }
  }

  return strcmp(status, "\r\nOK\r\n");
}


#pragma vector=TIMER0_A0_VECTOR
__interrupt void manda_local_t (void)
{
  static int cont = 0;

  cont++;
  if(cont>=900){ // para clk 32768 dividido por 2 no timer,
15 contagens será o tempo de 1 min
    TA0CTL= TACLR;
    TA0CTL=0;
    manda_local();
    cont=0;
  }
  TA0CTL= TASSEL_2 + ID_3 + MC_2 + TAIE;
}

#pragma vector=PORT1_VECTOR
__interrupt void manda_local_b(void){

    P1IFG &=~BTN;
    P1IE &= ~BTN;
    manda_local();
    P1IE |= BTN;
}
```

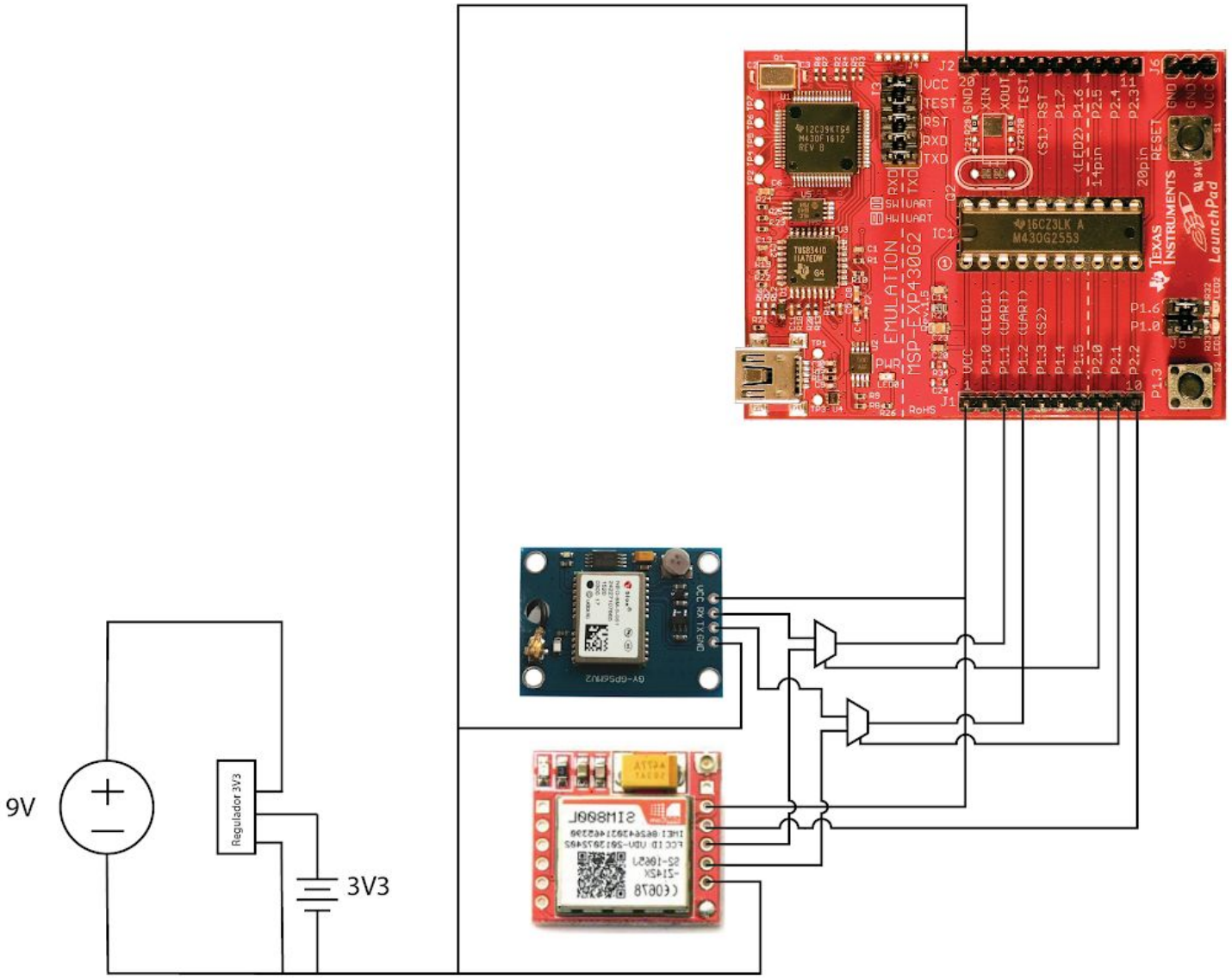


Figura 1: Esquemático Teórico

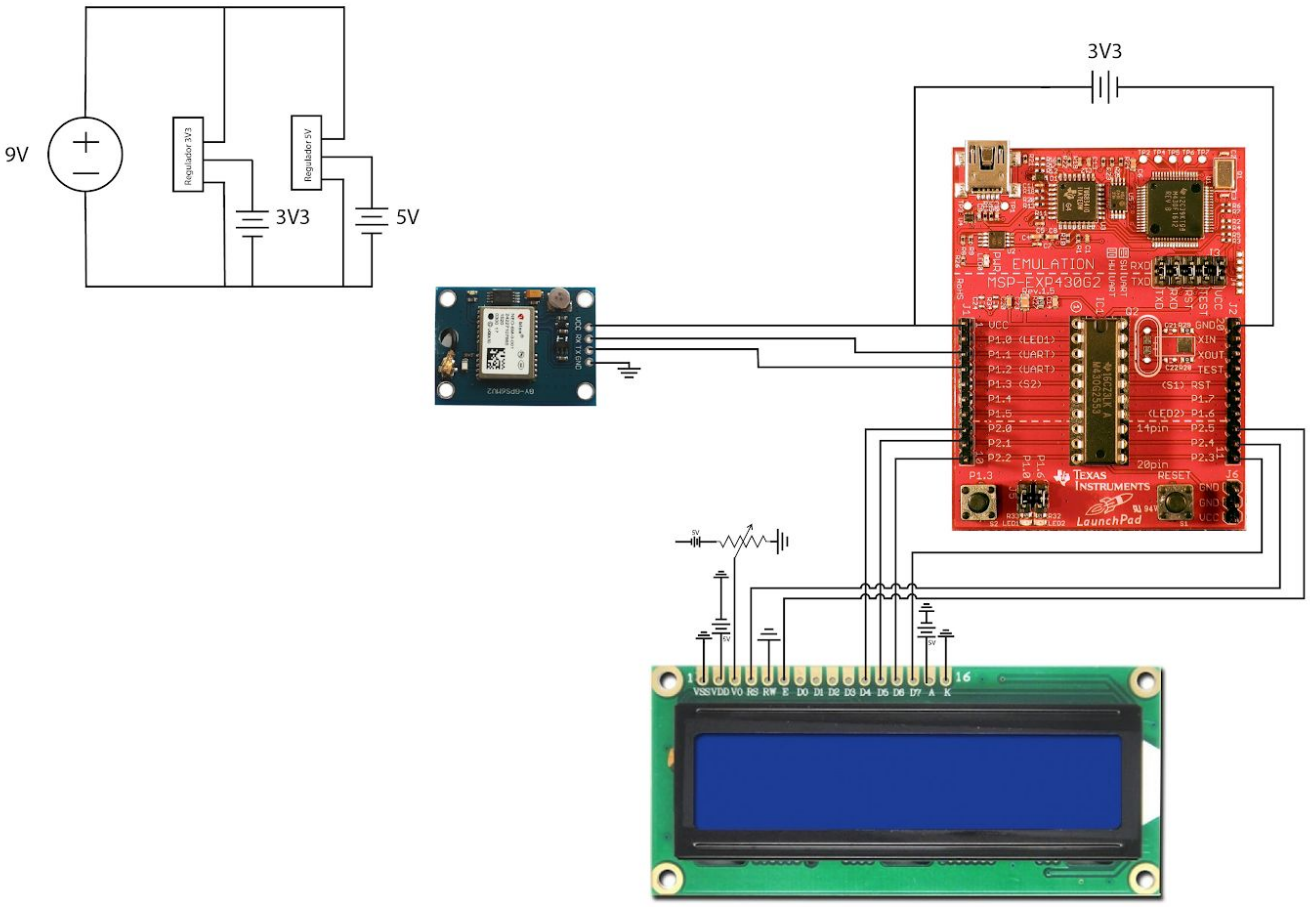


Figura 2: Esquemático Experimental