Ponto de Controle 3

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 and other data as movement. This pet collar is waterproof with a durable battery. These data can be obtained by the owner anytime and anywhere.**

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

***Resumo* – Esse projeto consiste no desenvolvimento de uma coleira inteligente com rastreamento GPS e da obtenção de outros parâmetros como movimentação para monitorar o nível de atividade do animal. Essa coleira conta com um encapsulamento a prova d'água e com uma bateria durável. Esses dados poderão ser acessados pelo dono a qualquer momento em qualquer lugar.**

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

I. JUSTIFICATIVA

Quando um animal se perde, os únicos recursos que pode-se 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. Além disso, nela estaria embutido informações do animal e do seu dono.
Outros valores podem ser agregados a esse dispositivo, como, por exemplo, o monitoramento de atividade do animal, pois, muitas vezes, devido a alimentação indevida, o animal se torna sedentário diminuindo a sua expectativa e a sua qualidade de vida.

II. OBJETIVOS

- Transmissão e recepção de sinais de GPS
- Dados de movimentação
- Dados de temperatura do animal
- Dados do dono,em caso de perda
- Utilizar baixo consumo de energia

III. BENEFÍCIOS

Este projeto traz benefícios a sociedade na área dos animais domésticos, pois com o advento da tecnologia das coleiras inteligentes pode-se ter um controle mais sistematizado da localização de animais de estimação, e de outras informações relevantes a saúde dos animais, como suas temperaturas, e batimentos cardíacos.

Todas estas informações serão útil por exemplo se um cachorro eventualmente se perder, de modo que com a informação no GPS da localização do cachorro, permitirá que o controlador mande para o dono as informações da localização do cachorro. Também pela da temperatura do animal de se estimação, é possível perceber anomalias na saúde do animal.

IV. REVISÃO BIBLIOGRÁFICA

Ao buscar tecnologias relacionadas a coleira inteligente, encontramos várias tecnologias neste intuito, entre coleiras específicas para animais domésticos, e também coleiras usadas para o estudo da vida selvagem. Muitos estudos foram feitos com a finalidade de obter um dispositivo que localizasse o animais em estudo, para se obter informações sobre a movimentação de seus grupos, isto porque ao se conseguir a localização de um animal, se adquire a partir desta a localização de todo um bando.

O primeiro projeto a ser citado é um estudo para colocar coleiras com GPS para elefantes, esse projeto teve como motivação o problema em que os elefantes ao se deslocarem na natureza em procura de alimento, por vezes acabavam por chegar em plantações de populações locais de agricultores, e traziam prejuízos muito grandes a estes agricultores, de modo que para contornar este problema, os agricultores acabam por ter que ordenar pessoas para vigiarem estas plantações, para evitar o prejuízo causado pelos elefantes, que acabam por pisotear as plantações, e a comer grandes quantidades de alimentos das plantações. Então a resolução proposta para este problema foi a implementação de uma coleira com GPS, que marcava a localização dos elefantes, e permitia saber se os

elefantes estavam perto das plantações, para se poder tomar as devidas medidas, além disso a coleira conta com buzzers e flashes de luz, para que ao chegarem na área delimitada, o controlador possa ativar efeitos sonoros e luminosos para se poder encontrar os elefantes mesmo em horários noturnos, com pouca iluminação.(CAMBROM, 2014)

O segundo é um estudo para coleiras para animais selvagens para o estudo da vida selvagem, para compreender o comportamento dos animais, seus deslocamentos, e ter os mais variados tipos de dados, tudo isto com a proposta de ter um melhor gerenciamento de energia, com tecnologias que eram novas para a época. O projeto conta com sensores de temperatura, acelerômetro, GPS, tecnologia Zigbee para transmitir os dados e uma memória para armazenar estes dados. O gerenciamento de energia deste dispositivo torna possível o funcionamento deste por mais de 3 meses sem precisar de carregar o dispositivo.(SONG, 2011)

O terceiro projeto é o wildCENSE, que tem como objetivo também o estudo dos animais em seu habitat, com diferencial de além de capturar as posições dos animais pelo GPS, este dispositivo tem a funcionalidade de capturar informações climáticas também. O projeto conta com 5 parâmetros a serem monitorados, que são a posição(pelo GPS), a temperatura, umidade, orientação da cabeça do animal, e luz ambiente, de modo que estas informações são monitoradas em cada coleira nos animais e é enviada por meio de uma conexão a internet para uma base de dados. Também foi adicionado ao projeto um gerenciamento de energia solar para prolongar a vida da bateria.(JAIN, 2008)

O quarto é o Doggy Pal Collar, direcionado para ser utilizado em cachorros. Este é um projeto em forma de coleira que tem por objetivo monitorar cachorros domésticos, fornecendo informações importantes ao dono, e promovendo o bem-estar dos animais de estimação, ajudando no acompanhamento da saúde do animal, e de sua localização. Este dispositivo monitora os batimentos cardíacos do cão, a sua temperatura, localização por GPS e posição pelo acelerômetro. O dispositivo é controlado por um microcontrolador, o TM4C123GH6PM, que permite o gerenciamento de todos os dados que são coletados pelos sensores, e então estes dados são mandados via wireless ao dono por um módulo de comunicação wireless.(BRYON, 2016)

Com estes projetos, pode-se concluir que é viável tecnologicamente a implementação do projeto em proposta neste relatório, de modo que há bases em bibliografia pregressa para se construir um novo projeto em cima dos conhecimentos já alcançados na área.

Além disso, 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.

V. 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.

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 para MPU(energia):

```
#include<Wire.h>
```

```
//Endereco I2C do MPU6050
const int MPU=0x68;
//Variaveis para armazenar valores dos sensores
int AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
void setup()
{
  Serial.begin(9600);
//  //Inicializa o LCD
//  lcd.begin(20, 4);
//  Wire.begin();
//  Wire.beginTransmission(MPU);
//  Wire.write(0x6B);

  //Inicializa o MPU-6050
  Wire.write(0);
  Wire.endTransmission(true);
}
void loop()
{
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);  // starting with register 0x3B
(ACCEL_XOUT_H)
  Wire.endTransmission(false);
  //Solicita os dados do sensor
  Wire.requestFrom(MPU,14,true);
  //Armazena o valor dos sensores nas variaveis
correspondentes
  AcX=Wire.read()<<8|Wire.read();  //0x3B
(ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read();  //0x3D
(ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read();  //0x3F
(ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp=Wire.read()<<8|Wire.read();  //0x41 (TEMP_OUT_H)
& 0x42 (TEMP_OUT_L)
  GyX=Wire.read()<<8|Wire.read();  //0x43
(GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read();  //0x45
(GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read();  //0x47
(GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)

  //Envia valor X do acelerometro para a serial
  Serial.print("AcX = "); Serial.print(AcX);

  //Envia valor Y do acelerometro para a serial
  Serial.print(" | AcY = "); Serial.print(AcY);

  //Envia valor Z do acelerometro para a serial
  Serial.print(" | AcZ = "); Serial.print(AcZ);

  //Envia valor da temperatura para a serial
  //Calcula a temperatura em graus Celsius
  Serial.print(" | Tmp = "); Serial.print(Tmp/340.00+36.53);

  //Envia valor X do giroscopio para a serial
  Serial.print(" | GyX = "); Serial.print(GyX);

  //Envia valor Y do giroscopio para a serial
  Serial.print(" | GyY = "); Serial.print(GyY);


  //Envia valor Z do giroscopio para a serial
  Serial.print(" | GyZ = "); Serial.println(GyZ);

  //Aguarda 300 ms e reinicia o processo
  delay(300);
}
```

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;

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




int main(void)

{

        gps myGPS;



   WDTCTL = WDTPW + WDTHOLD;// STOP
WATCHDOG TIMER

   BCSCTL1 = CALBC1_8MHZ;// MAKE THE
FREQUNCY AS PER THE LAUNCHPAD 8MHZ
   DCOCTL = CALDCO_8MHZ;

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

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

   __enable_interrupt();// ENABLE INTERRUPT

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




void gsm(gps *myGPS)

{




     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=\"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;

}
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;
                      }
                   }
                }
             }
          }
       }
    }
}
```