Módulo guiado para monitoramento de ambientes de potencial risco

Módulo para monitoramento guiado por Bluetooth

Vanessa Oliveira Nóbrega
Faculdade UnB Gama
Gama-DF, Brasil
vanessa.nobrega@outlook.com

Brenda Medeiros Santos
Faculdade UnB Gama
Gama-DF, Brasil
brenda.eng.unb@gmail.com

I. JUSTIFICATIVA

Monitorar ambientes com potenciais ameaças à segurança, tais como: áreas com riscos de desmoronamentos, explosões, princípios de incêndios, vazamentos de gás, entre outros. Tendo como intuito antecipar possíveis ameaças à integridade dos profissionais que irão acessar esses locais.

II. OBJETIVOS

Desenvolver um módulo que possa monitorar aspectos como níveis de luminosidade, temperatura, além de possibilitar melhor percepção do espaço ocupado, através de um sensor ultrassônico de distância. A mesma estará fixada a um suporte móvel que terá seu movimento controlado via bluetooth, por um aplicativo para Android.

III. HARDWARE

A. Lista de materiais:

- Módulo Bluetooth HC-05;
- MSP-EXP430G2553LP
- Kit Chassi Redondo Smart 2 Rodas Robótica
- Driver motor ponte H L298N com 2 canais
- Sensor de temperatura (LM35);
- Sensor de distância ultrassom (HC SR04)
- Sensor de luz (LDR 5mm);
- Regulador de tensão (7805);
- Bateria 9V;
- Resistor de 10KΩ.

B. Descrição do Hardware

Para que o módulo atinja seu objetivo de ser operado a uma distância segura, é preciso que ele disponha de autonomia para que possa ser operado remotamente. Para isso utilizou-se um módulo *bluetooth* para envio de comandos e recebimento de dados. Uma bateria de 9V com um regulador de tensão também foi necessário para alimentação dos sensores que necessitam de 5V, pois o MSP430 fornece uma saída de até 3,3V o que não é suficiente para alguns componentes.

Os três sensores foram conectados de forma a utilizar a pinagem correta do MSP430, utilizando-se o esquemático fornecido pela *Texas Instruments* como referência (Figura 3). O driver do motor ocupou os pinos com saídas digitais do MSP430 e sua alimentação precisou ser fornecida pela bateria, pois não opera abaixo de 5V. Na Figura 1 pode ser observado o esquemático da ligação de todos os componentes ao microcontrolador e à alimentação.

O sensor de luz LDR funciona como um resistor variável, cuja resistência varia com a luminosidade recebida, e para se verificar as medidas de luz é preciso montar um circuito divisor de tensão utilizando outro resistor, para isso foi utilizado o valor de resistência de 10KΩ. O LDR varia de 30KΩ sem iluminação, para até 100Ω com forte iluminação. O MSP430 receberá a tensão sobre o LDR na escala analógica, convertendo para digital e em seguida esse valor será situado em uma escala de luminosidade que será exibida ao usuário.

O sensor de distância utiliza ondas ultrassonoras que são atiradas (trigger) e recebidas (echo) convertendo o tempo que demorou para retornar em distância através da equação:

$$duração = \frac{tempo}{29,4}\ x\ \frac{1}{2}$$

O sensor de temperatura possui um termopar e a cada 10mV ele representa uma variação 1ºC. A expressão para converter o valor em temperatura é:

$$temperatura\_final = \frac{tensão}{4095}\ x\ 100$$

O chassi utiliza dois motores de passo controlados por um driver ponte H L298n que controla o acionamento dos motores. Ele é alimentado com uma tensão de 5V.

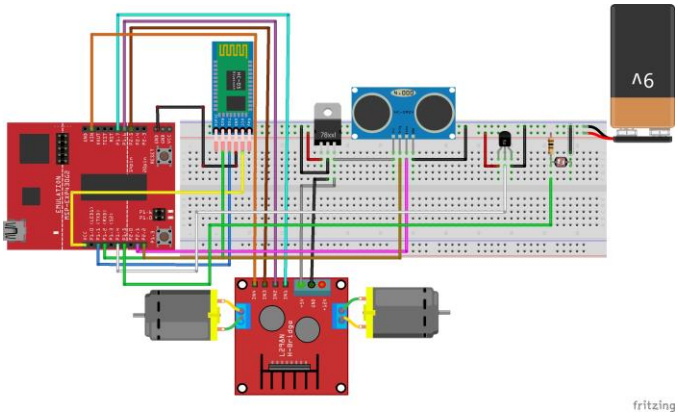
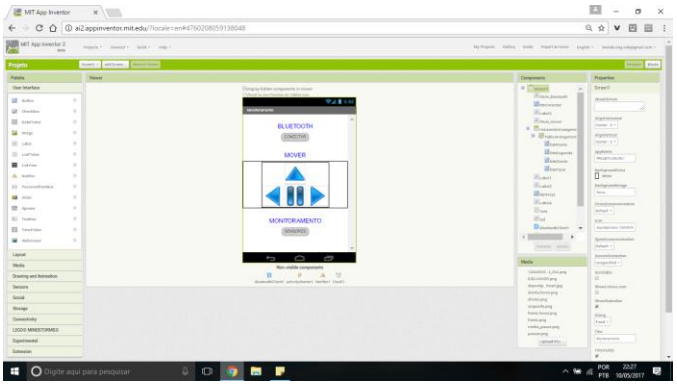


Fig. 1. Esquemático do circuito dos sensores e controle do motor do módulo de monitoramento.

IV. SOFTWARE

Para o controle wireless do módulo, através de um dispositivo *Bluetooth*, foi criado por meio do software online *MIT App Inventor* um aplicativo Android para celular que controla os movimentos do chassi e visualiza as informações dos sensores (Figura 2).



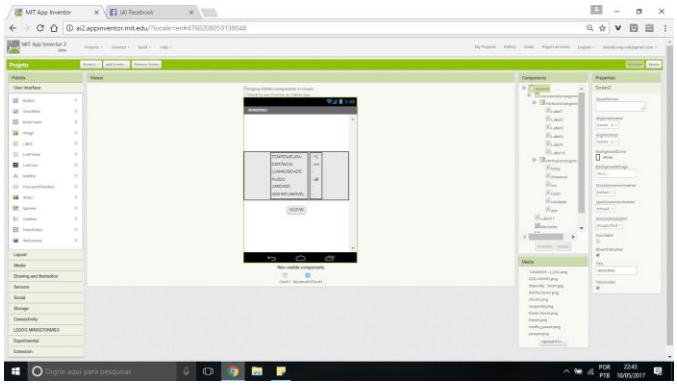


Fig. 2. Aplicativo para controle e recebimento de dados criado pela plataforma *MIT App Inventor*.

O módulo bluetooth precisou dos pinos TX (transmissor) e RX (receptor) da launchpad para a comunicação UART. Os pinos usados no MSP430 são P1.1 e P1.2 em sua configuração alternativa, selecionada a partir de P1SEL. Sua programação necessitou verificar se havia dados do bluetooth sendo recebidos e caso existisse seria comparado com os caracteres definidos para cada movimento do módulo (direita, esquerda, frente ou parar), além disso, o mesmo módulo será utilizado para o envio de dados adquiridos com os sensores.

O sensor de luz necessita somente de um pino analógico, além da alimentação, pois trata-se de um LDR. A entrada analógica do MSP430 é lida através do ADC10MEM, que tem 10 bits, logo a leitura da tensão (que vai de 0 a 3,3V) é quantizada em 1024 bits. Para saber qual o nível de luminosidade foi realizada uma escala da seguinte forma:

TABLE I. ESCALA DE LUMINOSIDADE DO SENSOR LDR

Escala de luminosidade	
Tensão (bits ocupados na conversão AD)	***Nível***
300	1
500	2
700	3
900	4
else	5

O sensor de distância utilizou os pinos digitais, para o echo e para o trigger. Para conversão do tempo percebido em distância acrescentamos a equação ao código e o seu resultado indica a distância do módulo ao obstáculo em frente, em centímetros.

O sensor de temperatura (LM35) funciona através de um pino analógico, o valor em tensão medido será convertido para digital, em seguida, o valor será convertido para graus Celsius, sabendo que para cada 10mV ele irá corresponder a 1ºC.

O motor utilizou a seguinte lógica para sua programação:

TABLE II. NÍVEIS LÓGICOS PARA CONTROLE DO MOTOR

ACIONAMENTO MOTORES				
DIREÇÃO	***NÍVEL***			
	MOTORA	***MOTORB***	***MOTORC***	***MOTORD***
FRENTE	HIGH	LOW	HIGH	LOW
ESQUERDA	HIGH	LOW	HIGH	HIGH
DIREITA	HIGH	HIGH	HIGH	LOW
PARAR	HIGH	HIGH	HIGH	HIGH

Observou-se que as entradas MOTORA e MOTORC sempre permaneceriam em nível alto, sendo assim, eles deixaram de ser saídas e foram conectados diretamente na alimentação. Com isso, foi estabelecido um condicional em sua programação, que definia o movimento dos motores a partir da variação das saídas nos pinos de acordo com o caractere recebido no módulo.

Na tabela 3 é possível verificar quais pinos serão utilizados no MSP430 e quais funções de cada pino, sendo assim, apenas a porta 1 será ocupada, todos os pinos da porta 2 estarão em saídas com nível baixo, para que haja menor consumo de energia e para que os pinos não fiquem em aberto.

TABLE III. PINOS UTILIZADOS NO MSP430

PINOS	***FUNÇÃO***
P1.0	MOTORB
P1.1	RX BLUETOOTH
P1.2	TX BLUETOOTH
P1.3	MOTORD
P1.4	ECHO (DISTÂNCIA)
P1.5	ECHO (DISTÂNCIA)
P1.6	LUZ (ADC10MEM)
P1.7	TEMPERATURA (ADC10MEM)

V. DISCUSSÃO

Nessa etapa do projeto várias dificuldades foram encontradas, tornando extremamente necessária a busca por referências bibliográficas que facilitassem a compreensão da comunicação UART. A primeira dificuldade encontrada esteve na taxa de transmissão de dados, uma vez que a configuração para cada taxa é diferente.

Outra dificuldade esteve no uso de dados analógicos digitalizados, que são os advindos dos sensores. Esses dados precisam ser digitalizados da melhor forma possível para que não sejam muito divergentes dos valores originais de distância (em cm), temperatura (em ºC) e luminosidade (em lúmens ou conforme Tabela 1).

Anteriormente, seriam utilizados outros sensores para captar informações adicionais como umidade, níveis de ruído ou presença de gases. Porém, afim de simplificar o projeto inicial decidiu-se apenas pelos principais: distância, temperatura e luminosidade. Sendo assim outros sensores poderão ser adicionados ao projeto, bastando utilizar outros pinos que façam a leitura de valores analógicos.

O código ainda está incompleto, uma vez que houve uma enorme dificuldade em receber os valores analógicos dos sensores. A conversão através do ADC10MEM será feita para os dois sensores que usarão pinos analógicos P1.6 e P1.7.

Outra informação é que os valores analógicos serão lidos duas vezes por segundo, uma vez que as variações de temperatura e luminosidade não são muito bruscas, geralmente. E esse foi o critério utilizado para selecionar a taxa de amostragem dos nossos valores analógicos.

VI. RESULTADOS

Os fios dos motores apresentaram problema na soldagem, sendo assim o movimento realizado não está conforme o esperado. A parte do código para os valores analógicos ainda está em fase de testes individuais, para então serem aplicados ao código final, apresentado em anexo.

O módulo bluetooth está funcionando corretamente, tanto como receptor como enquanto transmissor, recebendo uma string de exemplo a cada vez que um caractere é recebido pelo RX. Observe que por se tratar de interrupções esse será o modo de funcionamento do módulo: a cada valor esperado recebido, a mensagem desejada será enviada pelo transmissor, que é quando a interrupção é habilitada.

As strings enviadas, que são apenas texto, serão concatenadas com os valores lidos dos sensores que terão que ser convertidos de inteiros para valores da tabela ASCII. A caixa de texto do código será responsável pela exibição dessas strings com os valores convertidos.

VII. BENEFÍCIOS

Existem situações em que é necessário inspecionar algum ambiente, ou monitorá-lo, e é indesejável que isso seja feito por pessoas, como em situações de risco, altas temperaturas, acidentes químicos, ambientes instáveis, entre outros.

Nesse sentido, o módulo guiado permite algumas informações prévias, o que proporciona maior segurança aos funcionários, possibilitando maior planejamento ao acessar o local. Diante disso, ideia inicial da dupla é desenvolver um módulo que permita a análise de três informações: temperatura, distância para obstáculos e luminosidade.

O módulo será consistido de um sensor LM35 para temperatura, um LDR para medições de luminosidade, um sensor ultrassônico para distância HC-SR04. Apesar da proposta ser monitorar ambientes inóspitos, o mesmo módulo pode ser usado em fábricas ou indústrias, em ambientes em que há pouca movimentação ou acesso, tubulações e encanamentos, para que qualquer modificação do seu estado normal possa indicar algum problema e assim, permitir uma ação rápida afim de corrigir isto.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] C. P. Alvaristo, G. C. Santos, M. F. Rodrigues, P. G. Dallepiane, T. M. Faistel. Protótipo de robô elétrico com controle remoto para medições de gases inflamáveis. Unijuí. 8º Congresso Brasileiro de Metrologia, Bento Gonçalves/RS, 2015.

[2] T. O. Loupo, M. Torres, F. M. Millian, P. E. Ambrósio. Bluetooth Embedded System for Room-Safe Temperature Monitoring. IEEE Latin America Transactions, Vol. 9, no. 6, October 2011.

[3] Home Automation With HomeGenie. Disponível em: <http://www.instructables.com/id/Home-Automation-with-HomeGenie/> Acessado em: 02 de Abril de 2017.

[4] MSP430G2231 Standalone Environment Temperature Automatic Control System (Any Fan). Disponível em:<https://www.instructables.com/id/MSP430G2231-Standalone-environment-temperature-aut/> Acessado em: 02 de Abril de 2017.

[5] W. Souza, A. Daques, G. Tedesco, W. Akira. Carrinho controlado por Celular Android. Trabalho de conclusão do curso técnico em telecomunicações. São Paulo, 2013.

[6] Launchpad - Comunicación Serial Con Matlab. Disponível em:<http://www.instructables.com/id/Launchpad-Comunicaci%C3%B3n-Serial-con-Matlab/> Acessado em: 02 de Abril de 2017.

ANEXOS

Fig. 3. Pinagem MSP430

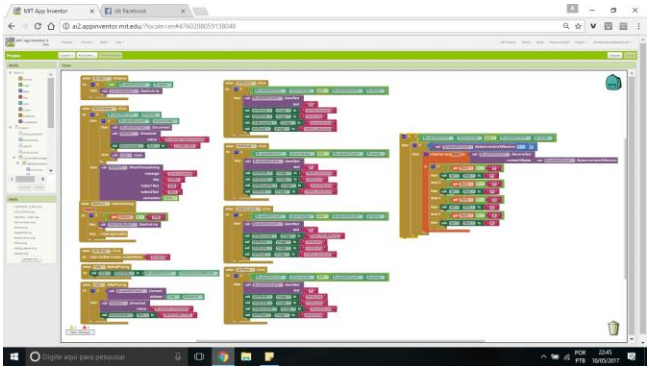


Fig. 4. Diagrama de blocos da programação do aplicativo através do MIT App Inventor.

```
/*PROJETO MICROCONTROLADORES***/
/*Brenda Medeiros e Vanessa Oliveira**/

/***Definicao pinos dos sensores***/
#include "msp430g2553.h"

#define MOTORB BIT0
#define RXD BIT1
#define TXD BIT2
#define MOTORD BIT3
#define ECHO BIT4
#define TRIGGER BIT5
#define LUZ BIT6
#define TEMPERATURA BIT7

//Todos os pinos da porta 1 estão ocupados, porta 2 não será usada

unsigned int i; //Contador
unsigned int ResultadoTemp = 0; // para ADC10MEM TEMPERATURA
unsigned int ResultadoLuz= 0; //para ADC10MEM LUZ

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD; // Para o Watchdog Timer
  DCOCTL = 0; // Select lowest DCOx and MODx settings
  BCSCTL1 = CALBC1_1MHZ; // DCO = 1MHz
  DCOCTL = CALDCO_1MHZ;
  P2DIR |= 0xFF; // Todos os pino P2.X declarados como saídas
  P2OUT &= 0x00; // Todos os pinos P2.X em baixo (diminuir consumo de energia)
  P1SEL |= RXD + TXD ; // modo UART P1.1 = RXD, P1.2=TXD
  P1SEL2 |= RXD + TXD ; // modo UART P1.1 = RXD, P1.2=TXD
  P1DIR |= (MOTORB + MOTORD); //Saídas nos motores
  P1OUT &= 0x00; //Todas as saídas dos motores em nível baixo inicialmente
  UCA0CTL0 = 0; //Configurando para comunicação UART 9600
  UCA0CTL1 = UCSSEL_2;
  UCA0BR0 = 6;
  UCA0BR1 = 0;
  UCA0MCTL = UCBRF_8 + UCOS16;
  UCA0CTL1 &= ~UCSWRST; //**Initialize USCI state machine**
  UC0IE |= UCA0RXIE; // Enable USCI_A0 RX interrupt
  __bis_SR_register(CPUOFF + GIE); //Modo de baixo consumo e Interrupção para recebimento
  while (1) //loop infinito
  { }
}

/* CÓDIGO PARA ENVIO DE STRINGS VIA BLUETOOTH (valores dos sensores analogicos)

#pragma vector= USCIAB0TX_VECTOR
__interrupt void USCI0TX_ISR(void)
{
    UCA0TXBUF = string[i++]; // proximo caractere
   if (i == sizeof string - 1) // é ultima posição da string?
     UC0IE &= ~UCA0TXIE; // Desabilita USCI_A0 TX interrupt
}
*/

#pragma vector= USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
  if (UCA0RXBUF == 'F') // 'F' received?
  {
          P1OUT &= 0x00; //Todas as saídas dos motores em nível baixo inicialmente
    //LIGA MOTOR FRENTE(A - alto, B- baixo, C - alto, D - baixo)
     i = 0;
```

```
      UC0IE |= UCA0TXIE;  // Habilita USCI_A0 TX interrupt
     UCA0TXBUF  = string[i++]; // proximo caractere
   }

   if (UCA0RXBUF  == 'P') // 'P' received?
   {
            P1OUT  &= 0x00;  //Todas as saídas dos motores em
nível baixo inicialmente
     //PARA MOTOR(A  - alto, B- alto, C - alto, D - alto)
      P1OUT |= (MOTORB  + MOTORD);
      i = 0;
      UC0IE |= UCA0TXIE;  // Habilita USCI_A0 TX interrupt
     UCA0TXBUF  = string[i++]; // proximo caractere
   }

   if (UCA0RXBUF  == 'D') // 'D' received?
   {
            P1OUT  &= 0x00;  //Todas as saídas dos motores em
nível baixo inicialmente
     //LIGA  MOTOR DIREITA  (A - alto, B- baixo,  C - alto, D
- alto)
      P1OUT |= (MOTORD);
      i = 0;
      UC0IE |= UCA0TXIE;  // Habilita USCI_A0 TX interrupt
     UCA0TXBUF  = string[i++]; // proximo caractere
   }

   if (UCA0RXBUF  == 'E') // 'E' received?
   {
            P1OUT  &= 0x00;  //Todas as saídas dos motores em
nível baixo inicialmente
     //LIGA  MOTOR ESQUERDA  (A - alto, B- alto, C -
baixo,  D - alto)
      P1OUT |= (MOTORB);
      i = 0;
      UC0IE |= UCA0TXIE;  // Habilita USCI_A0 TX interrupt
     UCA0TXBUF  = string[i++]; // proximo caractere
            }
}


int Temperatura(void){  //esboço
ResultadoTemp = ADC10MEM;

}

int Luz(void){  //esboço
ResultadoLuz = ADC10MEM;
}
```