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 a segurança humana, 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, umidade, temperatura, ruído, presença de gases, além de possibilitar a visualização da área em que o módulo estiver, através de uma câmera. A mesma estará fixada a um suporte móvel que terá seu movimento controlado via bluetooth.

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)
- Decibelímetro (KY038);
- Sensor de luz (LDR 5mm);
- Regulador de tensão (7805);
- Bateria 9V;
- Resistor de 10KΩ.

B. Descriçao 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 então 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.

Os 4 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 que varia com a luminosidade recebida, e para se verificar as medidas de luz é preciso montar um circuito divisor de tensão utilizando outra resistência, para isso foi utilizado o valor de resistência de 10KΩ. O LDR varia de 30KΩ sem forte iluminação para até 100Ω com forte iluminação. O MSP430 receberá a tensão sobre o LDR na escala analógica.

O sensor de distância utiliza ondas ultrassonaras que são atiradas (trigger) e recebidas (echo) convertendo o tempo quedemorou 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:

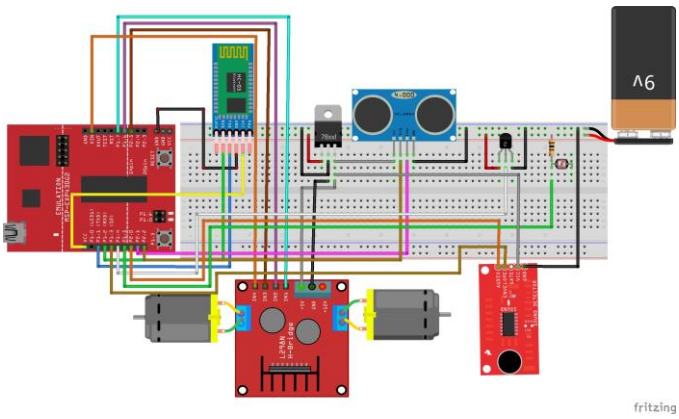

Fig. 1. Esquemático do circuito dos sensores e controle do motor do módulo de monitoramento.

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. Sua programação necessitou somente verificar se havia dados do bluetooth sendo recebidos e caso existisse seria armazenada na variável char motor.

O sensor de luz necessita somente de um pino AnalogRead() para receber os dados que são uma constante. A entrada analógica do MSP430 tem 12 bits, logo a leitura da tensão (que vai de 0 a 3,3V) é quantizada em 4096 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 recebidos)	***Nível***
200	1
400	2
600	3
800	4
else	5

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

V. 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 5 informações: temperatura, umidade, ruído, presença de gases e luminosidade, informações que são necessárias para detecção de incêndios, vazamento de água ou gases, além de desabamento ou presença de vítimas, que poderiam ser detectadas através do ruído gerado.

O módulo será consistido de um sensor LM32 para temperatura, para a umidade um sensor DHT11 (que também poderá ser usado para medições de temperatura), um decibelímetro para medição do ruído e um LDR para medições de luminosidade. 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

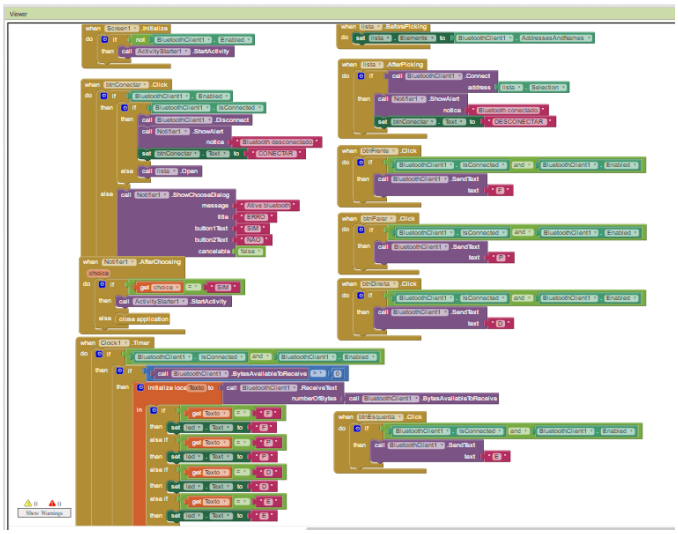

Fig. 4. Programação MIT App Inventor.

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

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

/***Definicao pinos dos sensores***/

/***SOM***/
int digital_som = P2_0;
int analog_som = P1_3; //analogRead()

float valor_Asom=0;
int valor_Dsom = 0;

/***DISTANCIA***/
int echo = P2_1;
int trigger = P2_2;

long duracao;
long distancia;

/***TEMPERATURA***/
int temperatura = P1_4; //analogRead()

float valor_temp;
float temp_final;

/***LUZ***/
int luz = P1_5;
int valor_luz = 0;

/***BLUETOOTH***/
int RX = P1_1; //receptor bluetooth
int TX = P1_2; //transmissor bluetooth

/***MOTORES***/
int inA = P1_7; //A do motor
int inB = P1_6; //B do motor
int inC = P2_5; //C do motor
int inD = P2_6; //D do motor



char motor;

void setup() {
  /*BLUETOOTH*/
  pinMode(RX, INPUT);
  pinMode(TX, OUTPUT);

  /*MOTORES*/
  pinMode(inA, OUTPUT);
```

```
  pinMode(inB, OUTPUT);
  pinMode(inC, OUTPUT);
  pinMode(inD, OUTPUT);

  /*SOM*/
  pinMode(digital_som, INPUT);
  pinMode(analog_som, INPUT);

  /*LUZ*/
  pinMode(luz, INPUT);

  /*DISTANCIA*/
  pinMode(echo, INPUT);
  pinMode(trigger, OUTPUT);

  /*TEMPERATURA*/
  pinMode(temperatura, INPUT);

  Serial.begin(9600);
}

void loop() {

  while(Serial.available() > 0) //Verifica se há alguma
informaçao enviada pelo aplicativo via Bluetooth
  {
    motor = Serial.read();
    Serial.println(motor); //Printar o valor de motor para
conferir funcionamento

    if(motor == 'F'){ //Andar para frente
    digitalWrite(inA, HIGH); /*Sempre em alto, pq não vamos
usar sentido anti-horario*/
    digitalWrite(inB, LOW);
    digitalWrite(inC, HIGH); /*Sempre em alto*/
    digitalWrite(inD, LOW);
    delay (2000);
  }

    else if(motor == 'P'){ //Parar motor
    digitalWrite(inA, HIGH); /*Sempre em alto*/
    digitalWrite(inB, HIGH);
    digitalWrite(inC, HIGH); /*Sempre em alto*/
    digitalWrite(inD, HIGH);
    delay (2000);
  }

  else if(motor == 'D'){  //Direita
    digitalWrite(inA, HIGH); /*Sempre em alto*/
    digitalWrite(inB, LOW);
    digitalWrite(inC, HIGH); /*Sempre em alto*/
    digitalWrite(inD, HIGH);
    delay (2000);

}
  else if(motor == 'E'){
    digitalWrite(inA, HIGH); /*Sempre em alto*/
    digitalWrite(inB, HIGH);
    digitalWrite(inC, HIGH); /*Sempre em alto*/
    digitalWrite(inD, LOW);
    delay (2000);

}
  }

  /*******SOM******/
    valor_Asom = analogRead(analog_som);
    valor_Dsom = digitalRead(digital_som);
    Serial.write("Nível ruído: ");
    Serial.write((valor_Asom)/76.5); //Mede ruído em dB
    Serial.write("dB");
    Serial.write(" Digital som: ");
    Serial.write(valor_Dsom);

    /*******DISTANCIA******/
    digitalWrite(trigger, LOW);
    digitalWrite(trigger, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigger, LOW);
    duracao = pulseIn(echo, HIGH); // Mede tempo do pulso
    distancia = (duracao/29.4)/2; //expressao para medir em cm
    Serial.write("Distância: ");
    Serial.write(distancia);
    Serial.write("cm");

  /*******TEMPERATURA******/
  valor_temp = analogRead(temperatura);
  temp_final =(valor_temp*500)/4095; //expressao para medir
em graus Celsius
  Serial.write("Temperatura: ");
  Serial.write(temp_final);
  Serial.write("ºC");

  /*******LUZ******/
  valor_luz = analogRead(luz);
  Serial.write("Luminosidade (1 a 5");
  if (valor_luz<200){
    Serial.write("Nível 1.");
  }
  else if (valor_luz<400){
    Serial.write("Nível 2.");
  }
  else if (valor_luz<600){
    Serial.write("Nível 3.");
  }
  else if (valor_luz<800){
    Serial.write("Nível 4.");
  }
  else{
    Serial.write("Nível 5.");
  }

  delay (10);
}
```