Sistema de Segurança

João Pedro Moreira da Silva 11/0150511
Universidade de Brasília – Campus Gama
UnB
Gama, Brasil
jptekc@gmail.com

Rafael Mendes Felix 14/0159665
Universidade de Brasília – Campus Gama
UnB
Gama, Brasil
rafaelmendesf@hotmail.com

***Resumo* — Este trabalho visa apresentar um projeto de um sistema de segurança que é microcontrolado por um MSP430.**

Keywords* — *segurança, eletrônica, infravermelho, PIR, ultrassom.

I. Introdução

Segurança doméstica contra furtos, roubos e assaltos é uma preocupação recorrente de todo mundo. Conforme vamos adquirindo produtos de alto valor e guardando em nossas residências, passamos a nos preocupar com a segurança desses produtos quando deixamos nossas casas. Por isso, ter um sistema de segurança que monitora as nossas casas enquanto estamos ausentes nos deixa um pouco mais confortáveis e seguros quanto a possíveis furtos.

II. Justificativa

Escolhemos este projeto por vermos a necessidade de proteger nossos bens devido ao grande crescimento de furtos aos lares em nossas cidades e a possibilidade de ser implementado de uma forma custe bem menos quando comparado a grandes empresas fornecem sistemas de segurança sofisticados.

III. Objetivos

Visamos implementar um sistema de segurança doméstico que irá monitorar a residência em quanto os residentes estiverem ausentes. Caso haja alguma violação, o sistema irá avisar aos proprietários e as autoridades.

IV. Requisitos

Para executar este projeto iremos precisar do seguinte material:

- Microcontrolador MSP430 LaunchPad da Texas Instruments;
- Sensor de infravermelho (PIR – *passive infrared*);
- Teclado numérico do tipo matricial 4x3;
- Buzzer;
- Display 16x2.

O MSP430 será o grande responsável por controlar e agregar todos os sensores do circuito. Os sensores PIR serão instalados nas portas e janelas para detectar se houve alguma perturbação nesses ambientes. O teclado numérico será necessário para digitar a senha de bloqueio/desbloqueio do sistema. O sensor de ultrassom irá verificar se houve alguma perturbação dentro da casa. Já o buzzer irá ser responsável pelo aviso sonoro.

Haverá também um botão que será utilizado como “botão do pânico” ou alarme silencioso.

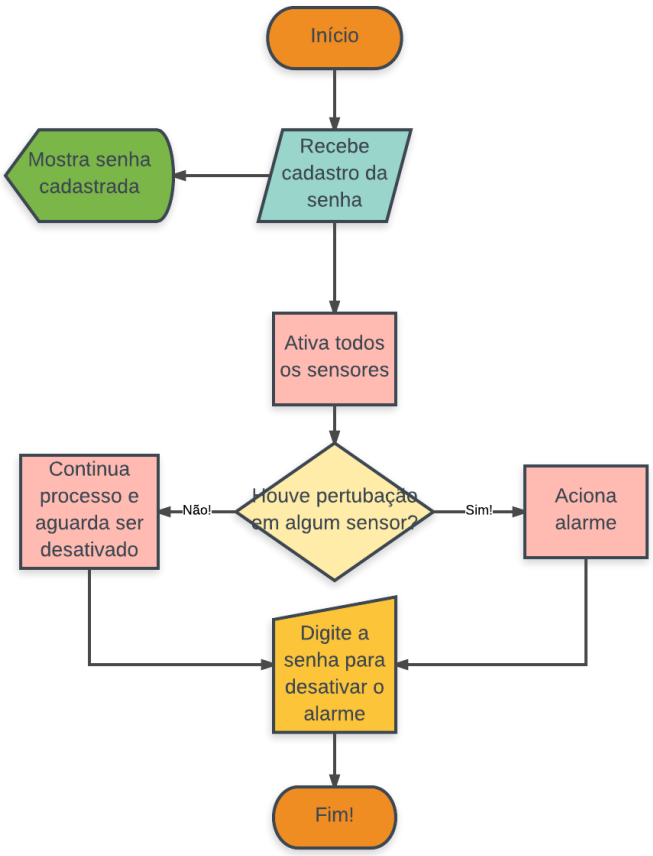

Figura 1 – Diagrama de blocos do projeto.

V. Descrição de Software

O sistema começa com a utilização do teclado para fazer o cadastro de uma nova senha. Esta senha será utilizada para ativar o alarme e também para desativá-lo. Então toda vez que

o sistema for iniciado será necessário digitar a senha e para desativá-lo também. Caso a senha digitada na hora de desativar o sistema esteja errada, através do display mostrará uma mensagem dizendo que a senha está incorreta e ele ficará nesse loop até que a senha correta seja digitada. A primeira parte do código do teclado mapeia através de uma variável char todas tecladas do teclado matricial. Os pinos da porta 1 (de 0 a 3) da MSP controlam as linhas do teclado enquanto os pinos da porta 2 (de 0 a 2) alimentam as colunas do teclado. O final do código mostra o que irá acontecer quando cada botão será pressionado.

O sensor PIR irá informar para o sistema se houve alguma perturbação no sistema. Para este tipo de sensor a primeira coisa que devemos fazer é a calibração do mesmo. Nas primeiras linhas do código o sensor emite um pequeno impulso para saber a área a ser varrida durante o tempo que o sistema esteja ativo. O restante do código consiste em mapear os pinos do PIR para os da MSP.

VI. RESULTADOS

Infelizmente não foi possível averiguar muita coisa do projeto, pois passamos por dificuldades de compilação e do software que compilava os códigos.

VII. CONCLUSÕES

Nesta etapa percebemos que estamos bastante atrasados no cronograma do projeto. Cada código deve ser testado novamente para averiguarmos sua funcionalidade. Os códigos apresentados são funcionais mas devem ser adaptados às necessidades do projeto.

Houve bastante dificuldade em lidar com os compiladores, então pouco se evoluiu no desenvolvimento do projeto.

Cada módulo do projeto utiliza muitos pinos de saída da msp, então devemos saber otimizar cada saída que temos para que com dois controladores seja suficiente desenvolver o projeto por inteiro.

Já foi percebido uma dificuldade em ligar os módulos para operarem em conjunto, então para as próximas etapas devemos trabalhar de forma mais eficiente visto que o desempenho do projeto está abaixo do esperado.

REFERÊNCIAS

[1] G. Eason, B. Noble, and I.N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (*references*)

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[3] I.S. Jacobs and C.P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

APÊNDICE

Códigos utilizados no projeto.

- Código do teclado matricial 4x3:

Início

```
#include <msp430g2553.h>
#include <legacymsp430.h>
#include <stdint.h>

char keymap_char[12] = {'1', '2', '3',
                        '4', '5', '6',
                        '7', '8', '9',
                        '*', '0', '#'};

uint32_t keymap_hex[12] = {0x0, 0x1, 0x2,
                           0x4, 0x5, 0x6,
                           0x8, 0x9, 0xA,
                           0xC, 0xD, 0xE};

void led_output(uint8_t bits);

int main(void)
{
  // Disable watch dog timer
  WDTCTL = WDTPW+WDTHOLD;

  // Set all pins as inputs
  // pins 0, 1, 2, 3 of port 1 read values from the rows of the numpads
  P1DIR = 0;
  P2DIR = 0;

  // Pins 0, 1, 2 of port 2 feed into the columns of the numpad
  // pins 7, 6 of port 1 and pins 5 and 4 of port 2 feed the 4 LEDs
  P1DIR |= (BIT0 | BIT1 | BIT2 | BIT3 | BIT7 | BIT6);
  P2DIR |= (BIT4 | BIT5);

  // Set initial outputs values to 0
  P1OUT = 0;
  P2OUT = 0;
```

```
  uint32_t io_pins[4] = {BIT0, BIT1, BIT2, BIT3};

  while(1) {
    for(uint8_t row = 0; row < 4; row++)
    {
      P1OUT |= io_pins[row];
      for(uint8_t col = 0; col < 4; col++)
      {
        if(P2IN & io_pins[col])
        {
          led_output(keymap_hex[row*4+col]);
        }
      }
      P1OUT &= ~(BIT0 | BIT1 | BIT2 | BIT3);
    }
  }
}

void led_output(uint8_t bits)
{
  (bits & BIT0) ? (P1OUT |= BIT7) : (P1OUT &= ~BIT7);
  (bits & BIT1) ? (P1OUT |= BIT6) : (P1OUT &= ~BIT6);
  (bits & BIT2) ? (P2OUT |= BIT5) : (P2OUT &= ~BIT5);
  (bits & BIT3) ? (P2OUT |= BIT4) : (P2OUT &= ~BIT4);
}
```

Fim

- Código do sensor PIR:

Início

```
int calibrationTime = 30;

//the time when the sensor outputs a low impulse
long unsigned int lowIn;

//the amount of milliseconds the sensor has to be low
//before we assume all motion has stopped
long unsigned int pause = 5000;

boolean lockLow = true;
boolean takeLowTime;

int pirPin = A5;    //the digital pin connected to the PIR sensor's
output
int ledPin = 13;
int spkPin = 12;
int relPin = 11;

////////////////////////////
//SETUP
void setup(){
//  Serial.begin(9600);
pinMode(pirPin, INPUT);
pinMode(ledPin, OUTPUT);
pinMode(spkPin, OUTPUT);
pinMode(relPin, OUTPUT);
digitalWrite(pirPin, LOW);

//give the sensor some time to calibrate
//  Serial.print("calibrating sensor ");
for(int i = 0; i < calibrationTime; i++){
//      Serial.print(".");
delay(1000);
}
//    Serial.println(" done");
//    Serial.println("SENSOR ACTIVE");
delay(50);
}

////////////////////////////
//LOOP
void loop(){

if(digitalRead(pirPin) == HIGH){
digitalWrite(ledPin, HIGH);    //the led visualizes the sensors
output pin state
digitalWrite(spkPin, HIGH);
digitalWrite(relPin, LOW);
if(lockLow){
//makes sure we wait for a transition to LOW before any further
output is made:
lockLow = false;
//         Serial.println("---");
//         Serial.print("motion detected at ");
//         Serial.print(millis()/1000);
//         Serial.println(" sec");
delay(50);
}
takeLowTime = true;
}

if(digitalRead(pirPin) == LOW){
digitalWrite(ledPin, LOW);   //the led visualizes the sensors
output pin state
digitalWrite(spkPin, LOW);
digitalWrite(relPin, HIGH);
if(takeLowTime){
lowIn = millis();          //save the time of the transition from high
to LOW
takeLowTime = false;       //make sure this is only done at the
start of a LOW phase
}
//if the sensor is low for more than the given pause,
//we assume that no more motion is going to happen
if(!lockLow && millis() - lowIn > pause){
//makes sure this block of code is only executed again after
//a new motion sequence has been detected
lockLow = true;
//          Serial.print("motion ended at ");      //output
```

```
//        Serial.print((millis() – pause)/1000);
//        Serial.println(” sec”);
delay(50);
}
}
}
```

Fim

- Código do Display LCD:

Início

```
#include <msp430g2553.h>
//#include <legacymsp430.h>
#define LCDOUT P1OUT
#define LCDDIR P1DIR
#define D4 BIT0
#define D5 BIT1
#define D6 BIT2
#define D7 BIT3
#define RS BIT4
#define EN BIT5
#define TODOS (D4 + D5 + D6 + D7 + RS + EN)
void Atraso(volatile unsigned int micro_secs)
{
      TACTL &= ~(MC0 + MC1);
      TACTL |= TACLR;
      TACCR0 = micro_secs-1;
      TACTL = MC_1 + ID_0 + TASSEL_2 + TAIE;
      _BIS_SR(LPM0_bits + GIE);
 }
//interrupt(TIMER0_A1_VECTOR) TA0_ISR(void)
#pragma vector=TIMER0_A1_VECTOR
__interrupt void TA0_ISR(void)
{
      TACTL &= ~(MC0+MC1+TAIFG+TAIE);
      LPM0_EXIT;
 }

 void SendNibble(volatile unsigned char nibble,
                        volatile char tipo,
                        volatile unsigned int micro_secs)
 {
      LCDOUT = nibble & 0xF;
      if(tipo==1) LCDOUT |= RS;
      LCDOUT |= EN;
      LCDOUT &= ~EN;
      Atraso(micro_secs);
 }

void SendByte(volatile unsigned char byte2send,
                        volatile char tipo,
                        volatile unsigned int micro_secs)
 {
      SendNibble(byte2send >> 4,  tipo, micro_secs);
      SendNibble(byte2send & 0xF, tipo, micro_secs);
 }

void ClearDisplay(void)
{
      SendByte(0x01, 0, 2000);
}

void initLCD(void)
{
      SendNibble(0x2, 0, 50); // Informar o LCD que o
                        // MSP430 trabalhará com nibbles
      SendByte(0x20, 0, 50);  // Trabalhar com nibbles,
                        // escrever somente uma linha,
                        // usar caracteres 5x7
      SendByte(0x10, 0, 50);  // Cursor se movimenta
                        // para a esquerda
      SendByte(0x0F, 0, 50);  // Display ligado, cursor
                        // piscando e com underline
      SendByte(0x06, 0, 50);  // Display parado
      ClearDisplay();
}

void initMSP(void)
{
      WDTCTL = WDTPW + WDTHOLD;
      BCSCTL1 = CALBC1_1MHZ;
      DCOCTL  = CALDCO_1MHZ;
      LCDOUT = 0;
      LCDDIR = TODOS;
}

void main(void)
{
      volatile int i, j;
      initMSP();
      initLCD();

      for(;;)
      {
            for(i=0; i<10; i++)
            {
                  SendByte(i+'0', 1, 50);
                  for(j=0; j<10; j++) Atraso(50000);
            }
            SendByte(' ', 1, 50);
            SendByte('B', 1, 50);
              SendByte('O', 1, 50);
              SendByte('O', 1, 50);
              SendByte('M', 1, 50);
              SendByte('!', 1, 50);
              for(j=0; j<10; j++) Atraso(50000);
              ClearDisplay();
        }
    }
```