Projeto Final de Microcontroladores - Smart Bike

Isabella Mika Taninaka e Tomás Abril

17 de dezembro de 2017

1 Introdução

Este projeto foi desenvolvido como projeto final para a disciplina de Microcontroladores na UTFPR sob orientação do professor Guilherme Peròn.

A especificação do projeto deveria incluir o uso de de um sensor, interface visual para o usuário, algum tipo de comunicação serial e a utilização da placa didática P51USB com o microcontrolador 8051. Além disso, também seria interessante um projeto que pudesse mostrar a utilidade dos conceitos desenvolvidos em aula em problemas reais. Portanto, o problema escolhido foi desenvolver um computador para bicicleta que mostre informações interessantes para o ciclista durante uma corrida.

1.1 Definição do Problema

Nomeado *Smart Bike*, este projeto se propõem a suprir algumas necessidades de um ciclista durante uma corrida, como a visualização da distância percorrida desde o início do percurso, o tempo que foi gasto, a velocidade instantânea atual.

Assim como um painel de um carro, o projeto visa reduzir ao máximo as distrações para o ciclista, mas o mantendo a par das informações que lhe interessam. Com isso em vista, o foco do *design* do sistema foram dois: manter o painel simples e a utilização direta, sem muitas opções.

1.2 Objetivos

- Detectar a rotação da roda da bicicleta;
- Calcular a distância percorrida desde o início da corrida;
- Calcular o tempo percorrido desde o início da corrida;
- Calcular a velocidade instantânea durante a corrida;
- Mostrar a distância percorrido desde o início da corrida;
- Mostrar o tempo percorrido desde o início da corrida;
- Mostrar a velocidade instantânea durante a corrida;
- Mostrar o total percorrido em quilômetros da última corrida, caso o usuário deseje;
- Mostrar o tempo total gasto da última corrida, caso o usuário deseje;
- Mostrar a velocidade de deslocamento média da última corrida, caso o usuário deseje.

2 Lista de Materiais

- 1 Resistor 10k
- 1 *Reed Switch*
- 3 Ímãs
- 1 Teclado Matricial 16 Teclas

- 1 *Display* LCD 16x2
- 1 Placa *Tiny RTC*
- 1 Bicicleta
- 1 Fonte de alimentação USB 5V
- 19 *Jumpers*
- 1 Placa P51USB

3 Solução

Smart Bike é um projeto que utiliza um *reed switch* para medir a velocidade de uma bicicleta, a distância percorrida e o tempo total de trajeto.

O modelo da bicicleta fornece alguns dados, como o tamanho da roda, de aro 26". Assim:

$$d = 26" = 0,6604m \tag{1}$$

Sabe-se que este é o diâmetro da roda. Portanto, o perímetro é dado por:

$$P = \pi * d \tag{2}$$

Aplicando (1) em (2):

$$P = 2.075m \tag{3}$$

Para o projeto, foi utilizado que $P = 2m$ o que ocasiona um erro de $3,75\%$ no calculo final, o que se considerou aceitável para o escopo do projeto.

O *reed switch* é fixado próximo a roda da bicicleta, enquanto um ímã é utilizado é fixado na roda. Este ímã ativa o *reed switch* a cada volta da bicicleta. Este sensor, então, é colocado na interrupção externa 0 do controlador, que guarda a quantidade de voltas desde o início da corrida.

A cada um segundo, é verificado quantas voltas foram dadas e multiplicado pelo valor do perímetro, gerando, assim, a distância total percorrida. Com essa base de tempo, é possível fazer o calculo da velocidade instantânea da bicicleta.

O calculo da velocidade é feito da seguinte forma:

1. Salva-se a distância percorrida até o segundo anterior numa variável chamada `distancia_ant`;
2. A nova distância percorrida é salva em uma variável chamada `distancia`;
3. É calculada a velocidade utilizando essas duas distâncias.

O calculo é feito da seguinte forma:

$$v_{(m/s)} = \frac{distancia - distancia\_ant}{1s} \tag{4}$$

Então se obtêm a velocidade em m/s. Para transformá-la em km/h:

$$v_{(km/h)} = 3.6 * v_{(m/s)} \tag{5}$$

Estas operações são feitas dentro do ciclo de 1s que é gerado pelo RTC, usando sua saída de 1 Hz. Essa saída é colocada para fazer a ativação da interrupção externa 1. Dentro desde ciclo também é feita a atualização do *display* com as informações sobre o tempo gasto, a distância percorrida e a velocidade instantânea.

Após a corrida, é feita um sumário dessas informações e elas ficam disponíveis para serem visualizadas antes do início de uma nova corrida.

3.1 Diagrama Esquemático

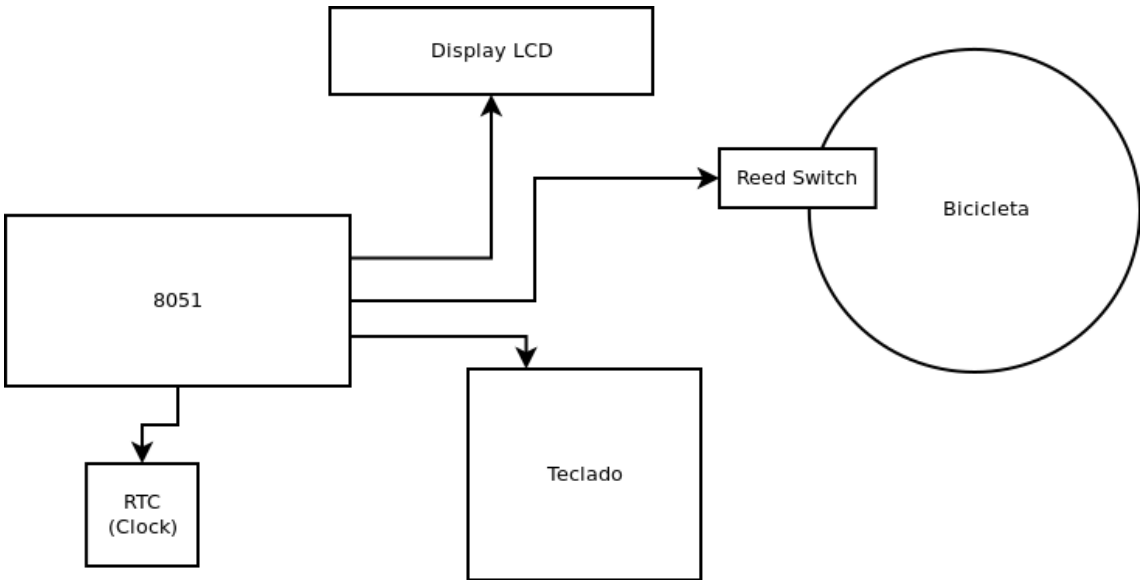

Figura 1: Diagrama Esquemático.

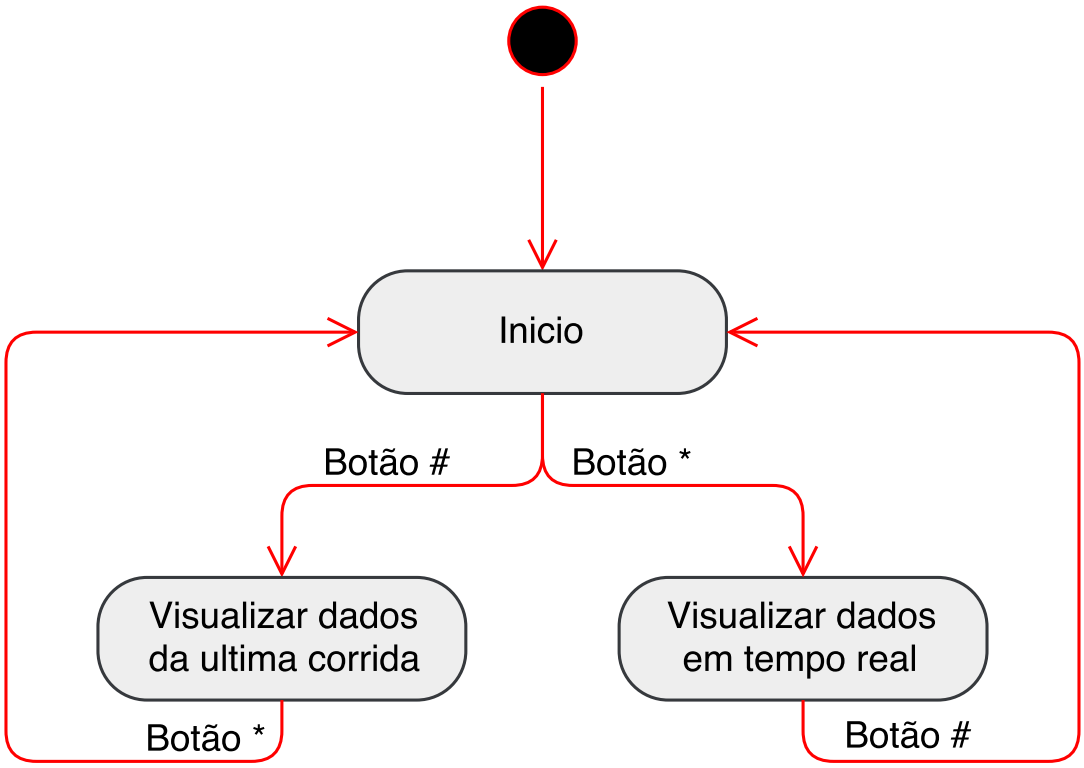

Figura 2: Diagrama de Estados.

4 Conclusão

É possível dizer que o projeto foi bem sucedido. Todos os itens dos objetivos foram cumpridos e a prova do conceito funciona.

Se fosse possível, a apresentação do sistema funcionando com uma bicicleta real seria muito mais interessante.

Alguns detalhes do código poderiam ser melhorados, como cálculos menos aproximados das variáveis utilizadas e um ajuste mais fino do sensor que provavelmente ficaria melhor com um circuito de *debouncing* feito no *hardware*. No entanto, este problema foi resolvido via *software* com um *trade-off*: não seria possível ultrapassar cerca de 64km/h na medição, mas não se considerou um problema muito grande, pois seria uma velocidade difícil de se alcançar.

Outro problema encontrado é o limite do tamanho de código no *software* da *Keil*. Não somente para a simulação, o programa em sua versão gratuita não gera os `.hex` se ele ultrapassar o limite. O código foi reduzido para atender esses parâmetros, como a remoção dos *timers* do sistema.

Há, também, problemas com as variáveis: ora o código não era compilado utilizando apenas variáveis locais, ora não era possível fazer certas declarações pois isso excederia a memória do microcontrolador. Devido a isso, alguns números foram truncados e variáveis do tipo *float* foram trocadas por *int*.

Mesmo com as limitações, foi possível apresentar um projeto interessante e com aplicações reais.

A Código Principal

```
#include "LCD.h"
#include "interrupt.h"
#include "rtc.h"
#include "keypad.h"
#include "delay.h"
#include <at89c5131.h>

#define PERIMETRO 2

bit inRun = 0;

unsigned char tecla = '0';
unsigned char a[7]; // valores do RTC em
unsigned char texto[16] = "                _"; //texto para imprimir
unsigned char dist[16] = "                "; //texto para imprimir
unsigned char dist_final[8] = "00,00km";

unsigned char seg = 0;
unsigned char seg_final = 0;
unsigned char min = 0;
unsigned char min_final = 0;
unsigned char hor = 0;
unsigned char hor_final = 0;

unsigned long int loops = 0;
unsigned long int distancia = 0;
unsigned long int distancia_ant = 0;

unsigned int i=0;
unsigned char resultado[2];


void intToChar(int num){
        int dezena;
        int unidade;

        dezena = num/10;
        unidade = num%10;

        resultado[0] = 0x30 + dezena;
        resultado[1] = 0x30 + unidade;
}

void cleartexto(){
        i =0;
        while(i<16){
                texto[i] = ' ';
                i++;
        }
}

int Dec_To_BCD(int dec) { return ((dec / 10 * 16) + (dec % 10)); }
```

```
int BCD_To_Dec(int val) { return ( (val/16*10) + (val%16) );    }

void readAllReg() {
        unsigned char j = 0;
  I2CStart();
  I2CSend(0xD0);
  I2CSend(0x00);
  I2CStop();
  I2CStart();
  I2CSend(0xD1);

  for (j = 0; j < 8; j++) {
    a[j] = I2CRead();
    if (j == 7)
      I2CNak();
    else
      I2CAck();
  }
  I2CStop();
}

void setTime(int Sec, int Min, int Hour, int Dow, int Dom, int Month,
             int Year) {
  I2CStart();
  I2CSend(0xD0);
  I2CSend(0x00);
  I2CSend(Dec_To_BCD(Sec) & 0x7f);
  I2CSend(Dec_To_BCD(Min) & 0x7f);
  I2CSend(Dec_To_BCD(Hour) & 0x3f);
  I2CSend(Dec_To_BCD(Dow) & 0x07);
  I2CSend(Dec_To_BCD(Dom) & 0x3f);
  I2CSend(Dec_To_BCD(Month) & 0x1f);
  I2CSend(Dec_To_BCD(Year) & 0xff);
  I2CStop();
}

void initRtcSqrt(){
        I2CStart();
        I2CSend(0xD0);  //slave address
        I2CSend(0x07);  //control register address
        I2CSend(0x10);  //control register 1
        I2CStop();

}

void writeTime(){
        seg = BCD_To_Dec(a[0]);
        min = BCD_To_Dec(a[1]);
        hor = BCD_To_Dec(a[2]);

        cleartexto();

        intToChar(hor);
        texto[0] = resultado[0];
        texto[1] = resultado[1];
        texto[2] = ':';

        intToChar(min);
        texto[3] = resultado[0];
        texto[4] = resultado[1];
        texto[5] = ':';

        intToChar(seg);
        texto[6] = resultado[0];
        texto[7] = resultado[1];


        LCDwrite(texto, 0x80);
}

void writeDistancia(){
        unsigned int km, m;
```

```
        distancia_ant = distancia;
        distancia = PERIMETRO*loops;

        km = distancia/1000;
        m = distancia % 1000;
        m = m/10;

        cleartexto();
        intToChar(km);
        dist[0] = resultado[0];
        dist[1] = resultado[1];
        dist[2] = ',';

        intToChar(m);
        dist[3] = resultado[0];
        dist[4] = resultado[1];

        dist[5] = 'k';
        dist[6] = 'm';

        LCDwrite(dist, 0xC0);

}

void velocidade_inst(){
        //imprime na tela em km/h
        //distancia em metros
        //distancia_ant
        unsigned long int diferenca = distancia - distancia_ant;
        unsigned long int vel_inst = diferenca*3.6;
        intToChar(vel_inst);
        LCDwrite(resultado, 0xC9);
        LCDwrite("km/h", 0xCC);
}

void clear() {
        // as operacoes fazem em sua maioria OR com os registradores especiais
        // entao antes do programa iniciar, essa funcao limpa todos os
            registradores
        // interrupcoes
        IE = 0x00;
        IP = 0x00;
        // timers
        TMOD = 0x00;
        TCON = 0x00;
}

void velocidade_final(){
//imprime na tela em km/h
//distancia em metros
unsigned int tempo_final = seg + min*60 + hor*60*60; //tempo percorrido em
    segundos
unsigned int vel_med = (distancia/tempo_final)*3.6;

        if(tempo_final == 0) vel_med = 0;
intToChar(vel_med);
LCDwrite(resultado, 0xC9);
LCDwrite("km/h", 0xCC);
}

// responsavel pela contagem do sensor
// tem prioridade maxima
void int_interrupt0(void) interrupt 0 {
        loops++;
        DELAY_ms(200);
        //colocar um delay menor que 240ms aqui!
}

void int_interrupt1(void) interrupt 2 {
        readAllReg();
        writeTime();
        writeDistancia();
```

```
        velocidade_inst();
}

void setup() {
  clear();
  LCD();
  initRtcSqrt();
}

void lastRun(){
        intToChar(hor_final);
        texto[0] = resultado[0];
        texto[1] = resultado[1];
        texto[2] = ':';

        intToChar(min_final);
        texto[3] = resultado[0];
        texto[4] = resultado[1];
        texto[5] = ':';

        intToChar(seg_final);
        texto[6] = resultado[0];
        texto[7] = resultado[1];

        LCDwrite(texto, 0x80);
        LCDwrite(dist_final, 0xC0);

        velocidade_final();
        while(tecla != '*')
                tecla = Read_Keypad();
}

void loop() {


        while(inRun == 0){
                LCDwrite("Nova Corrida?", 0x80);
                LCDwrite("________________", 0xC0);
                tecla = Read_Keypad();
                if(tecla == '#'){
                        lastRun();
                        tecla = '_';
                }
                else if (tecla == '*'){
                        inRun = 1;
                        //Sec Min Hour DayofWeek DayofMonth Month Year
                        setTime(0, 0, 0, 0, 0, 0, 0);
                        interrupt0();
                        interrupt1();
                        loops = 0;
                        distancia = 0;
                        }
                else inRun = 0;
                }

        tecla = '_';

        while(tecla != '#'){
                tecla = Read_Keypad();
        }
        EA = 0;
        inRun = 0;

        for(i = 0; i < 7; i++){
                dist_final[i] = dist[i];
        }
        readAllReg();
        seg_final = seg;
        min_final = min;
        hor_final = hor;

        LCDwrite("Fim Corrida!", 0x80);
        LCDwrite("* Nova, # Ultima", 0xC0);
```

```
}

void main() {
  setup();
  while (1)
    loop();
}
```

B Biblioteca de Interrupção

```
#include "interrupt.h"

void interrupt0(){
        EA = 1;
        EX0 = 1;
        IT0 = 1;
}

void interrupt1(){
        EA = 1;
        EX1 = 1;
        IT1 = 1;
}
```

C Biblioteca de Leitura do Teclado

```
#include "keypad.h"

void Delay_key(int a){
  int j;
  int i;
  for(i=0;i<a;i++)
  {
    for(j=0;j<100;j++)
    {
    }
  }
}


unsigned char Read_Keypad()
{
  C1=1;
  C2=1;
  C3=1;
  C4=1;
  R1=0;
  R2=1;
  R3=1;
  R4=1;
  if(C1==0){
          Delay_key(100);
          while(C1==0);
          return 'D';
        }
  if(C2==0){Delay_key(100);while(C2==0);return 'C';}
  if(C3==0){Delay_key(100);while(C3==0);return 'B';}
  if(C4==0){Delay_key(100);while(C4==0);return 'A';}
  R1=1;
  R2=0;
  R3=1;
  R4=1;
  if(C1==0){Delay_key(100);while(C1==0);return '#';}
  if(C2==0){Delay_key(100);while(C2==0);return '9';}
  if(C3==0){Delay_key(100);while(C3==0);return '6';}
  if(C4==0){Delay_key(100);while(C4==0);return '3';}
  R1=1;
  R2=1;
  R3=0;
  R4=1;
```

```
  if(C1==0){Delay_key(100);while(C1==0);return '0';}
  if(C2==0){Delay_key(100);while(C2==0);return '8';}
  if(C3==0){Delay_key(100);while(C3==0);return '5';}
  if(C4==0){Delay_key(100);while(C4==0);return '2';}
  R1=1;
  R2=1;
  R3=1;
  R4=0;
  if(C1==0){Delay_key(100);while(C1==0);return '*';}
  if(C2==0){Delay_key(100);while(C2==0);return '7';}
  if(C3==0){Delay_key(100);while(C3==0);return '4';}
  if(C4==0){Delay_key(100);while(C4==0);return '1';}
  return 0;
}
```

D Biblioteca de comunicação com RTC

```
#include "rtc.h"
//#include "delay.h"
unsigned int temp =0;


void DELAY_RTC(long int k) {
        temp=0;
        while(temp<0xFF){temp++;k=temp;}
}


void I2CStart() {
  SCL = 0;
  SDA = 1;
  DELAY_RTC(1);
  SCL = 1;
  DELAY_RTC(1);
  SDA = 0;
  DELAY_RTC(1);
  SCL = 0;
}

void I2CStop() {
  SCL = 0;
  DELAY_RTC(1);
  SDA = 0;
  DELAY_RTC(1);
  SCL = 1;
  DELAY_RTC(1);
  SDA = 1;
}

void I2CAck() {
  SDA = 0;
  DELAY_RTC(1);
  SCL = 1;
  DELAY_RTC(1);
  SCL = 0;
  SDA = 1;
}

void I2CNak() {
  SDA = 1;
  DELAY_RTC(1);
  SCL = 1;
  DELAY_RTC(1);
  SCL = 0;
  SDA = 1;
}

void I2CSend(unsigned char Data) {
  unsigned char i;
  for (i = 0; i < 8; i++) {

    SDA = Data & 0x80;
    SCL = 1;
```

```
    SCL = 0;
    Data <<= 1;
  }
  I2CAck();
}

unsigned char I2CRead() {
  unsigned char i, Data = 0;
  for (i = 0; i < 8; i++) {
    SCL = 1;
    Data |= SDA;
    if (i < 7)
      Data <<= 1;
    SCL = 0;
  }
  return Data;
}
```

E Biblioteca de LCD

```
#include "LCD.h"
#include <string.h>

void LCD(){
        LCDinst(0x30);
        LCDinst(0x30);
        LCDinst(0x30);

        LCDinst(0x38);

        LCDinst(0x0C);

        LCDinst(0x02);

        LCDinst(0x01);
}

void LCDbusy(){
        LCD_RS = 0;
        LCD_RW = 1;
        LCD_BUSY = 1;

        while(LCD_BUSY){
                LCD_EN = 1;
                if(!LCD_BUSY)
                        break;
                LCD_EN = 0;
        }
        LCD_RW = 0;
}

void LCDinst(unsigned char inst){
        LCDbusy();
        LCD_RS = 0;
        LCD_EN = 1;

        LCD_DATA = inst;
        LCD_EN = 0;
}

void LCDposi(unsigned char posi){
        LCDbusy();
        LCD_RS = 0;
        LCD_RW = 0;
        LCD_EN = 1;

        LCD_DATA = posi;
        LCD_EN = 0;
}

void LCDdata(unsigned char c){
        LCDbusy();
        LCD_RS = 1;
```

```
        LCD_RW = 0;
        LCD_EN = 1;

        LCD_DATA = c;
        LCD_EN = 0;
}

void LCDwrite(unsigned char text[], unsigned char posi){
        int i;
        LCDposi(posi);

        for(i = 0; i < strlen(text); i++){
                if(text[i] != NULL) LCDdata(text[i]);
                else break;
        }
}
```

F Biblioteca de Delay

```
#include"delay.h"

void DELAY_us(uint16_t us_count)
 {
    while(us_count!=0)
      {
         us_count--;
       }
 }

void DELAY_ms(uint16_t ms_count)
 {
     while(ms_count!=0)
      {
         DELAY_us(1);      //DELAY_us is called to generate 1ms delay
          ms_count--;
      }

 }

 #if (ENABLE_DELAY_SEC == 1)
void DELAY_sec(uint16_t sec_count)
 {
     while(sec_count!=0)
      {
         DELAY_ms(1000);     //DELAY_ms is called to generate 1sec delay
         sec_count--;
        }
 }
 #endif
```