# RELATÓRIO FINAL
# GRUPO 2
# CARRINHO CONTROLADO POR MICROCONTROLADOR VIA BLUETOOTH

*Brunno Cardoso de Santana*

Programa de Engenharia Eletrônica
Faculdade Gama - Universidade de Brasília
Área Especial de Indústria Projeção A
Gama, 72.444-240
email: brunno-cardoso-santana@outlook.com

*Milena Andressa da Silva Santos*

Programa de Engenharia Eletrônica
Faculdade Gama - Universidade de Brasília
Área Especial de Indústria Projeção A
Gama, 72.444-240
email: milena.andressa_@outlook.com

***Resumo*—Sistemas controlados remotamente, são utilizados frequentemente em diversas aplicações profissionais. Então, foi desenvolvido um sistema controlado via Bluetooth, visando solucionar com mais praticidade situações cotidianas sem colocar em risco o operador. O projeto realizado, possui 3 blocos principais, que são, módulo Bluetooth (HC-05), microcontrolador (MSP430F5529) e driver para motor DC(L298n). Estes blocos, se comunicam entre si, podendo assim, realizar movimentos provindos de comandos de um aparelho celular. Os resultados esperados foram alcançados e foi possível projetar desenvolvimentos futuros para ampliar ainda mais a capacidade de utilização do sistema controlado remotamente.**

***Palavras chaves—bluetooth, MSP430, aplicativo, robô, celular.***

## 1. INTRODUÇÃO

Nos dias atuais, com o avanço da tecnologia, tornou-se possível a utilização de robôs controlados remotamente em diversos ambientes. Estes robôs são aplicados em diversas áreas, como por exemplo para observação e navegação no mar, nas indústrias, desarme de bombas, resgates. entre outros.

Robôs utilizados para desarme de bombas ou para resgates, são normalmente controlados por meio de radiofrequência, utilizando controle remoto e câmeras, possuindo capacidade de adentrar em locais confinados e tendo como principal objetivo, evitar possíveis acidentes envolvendo trabalhadores.

Hoje, o robô antibomba e o de resgate, são capazes de percorrer grande capacidade de alcance remotamente, inclusive em locais irregulares e são capazes de erguer peso através de um braço mecânico.

Visando estes robôs, foi desenvolvido um veículo controlado remotamente, entretanto não foi utilizado o controle remoto usual, mas sim um celular com sistema operacional Android, que por meio de um aplicativo, é possível controlar as funções básicas do veículo, como os movimentos, frente, trás, esquerda, direita e faróis (dianteiro e traseiro).

Para a base deste projeto, foram revisados 2 artigos. No primeiro artigo foi desenvolvido um veículo controlado remotamente através do acelerômetro do celular. É utilizado um aplicativo desenvolvido para o sistema operacional do smartphone, que é responsável por realizar a leitura do sensor acelerômetro do aparelho e, em função da posição relativa do mesmo em relação ao solo, enviar comandos através de comunicação Bluetooth para o módulo veicular. Neste módulo, os comandos são recebidos e entregues ao microprocessador embarcado.[1] O projeto é realizando utilizando o microcontrolador Arduino UNO, e foi colocado sensores de proximidade que permitem a verificação de obstáculos próximo do veículo.

O segundo artigo possui o mesmo objetivo, a construção de um carro controlado por dispositivos moveis utilizando o sistema operacional Android e uma placa Arduino uno para controle e processamento, que é uma placa de prototipagem de código aberto. A transmissão dos comandos foi realizada utilizando a tecnologia Bluetooth.[2] Com base nesse artigo, foi possível encontrar um aplicativo gratuito que suprisse as necessidades do projeto, no que se refere a comunicação entre usuário e veículo.

## 2. DESENVOLVIMENTO

Para tornar a comunicação do veículo possível, foram utilizados dados transmitidos por um celular através da comunicação Bluetooth. Esses dados são interpretados por um microcontrolador, onde os mesmos são enviados, de acordo com a lógica correspondente ao movimento para o

driver dos motores DC, fazendo com que os motores rotacionem de acordo com o desejo do usuário.

O esquemático do projeto pode ser visualizado no APÊNDICE C.

Nos blocos abaixo, é mostrado como será o fluxo de informações dos componentes físicos:


### 2.1. Materiais

Na construção do veículo foram utilizados os seguintes materiais abaixo:

2 x Motores DC
1 x Microcontrolador (MSP430F5529)
4 x Resistores de 220 Ohms
4 x LEDs
1 x Carregador portátil
1 x Driver para motor DC (L298N)
1 x Módulo Bluetooth (HC-05)
1 x Chassis de carro infantil

### 2.2. Microcontrolador

Foi utilizada a MSP430F5529, pois é um microcontrolador que possui muita versatilidade, aplicabilidade e funcionalidade. Possui um baixo consumo de energia, clock bastante flexível, múltiplos periféricos, capacidade de realizar múltiplas tarefas, possuindo um CPU de 16 bits e uma programação simples que pode ser feita tanto em C como em Assembly, além de poder mesclar as duas linguagens.

O microcontrolador MSP430F5529, possui baixo consumo, na faixa de 2,5 µA para o modo standby power e 404 µA em funcionamento normal. Possui memória não volátil de 128 KB, memória volátil de 10 KB e clock de até 25 MHz.[3] Perfeito para diversas aplicações pois sua tensão de operação é baixa, na ordem de 1,8 V a 3,6 V.

A MSP430F5529, irá realizar a interpretação dos comandos enviados através do aplicativo. Então, decidirá qual ação será realizada pelos componentes físicos do carrinho, como acender o farol dianteiro, ir para frente, ligar o buzzer, etc. No caso dos motores DC, ainda será enviada a informação ao drive l298n que fará a interpretação final, afim de realizar a rotação dos motores.

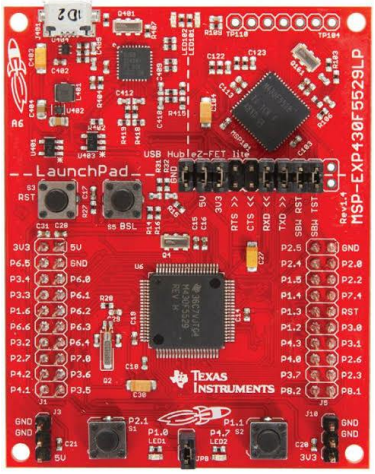



**Fig 1.** MSP430F5529LP – Texas Instruments

### 2.3. Driver para Motor

Com Driver L298n é possível controlar independentemente a velocidade e rotação de 2 motores DC. No projeto, foi implementado com o objetivo de controlar um motor para realizar os movimentos de FRENTE/RÉ e outro motor para realizar os movimentos ESQUERDA/DIREITA.

A lógica para os motores é definida de acordo com o datasheet do módulo, que analisa os dois valores nas portas IN1 e IN2 para o motor 1 e IN3 e IN4 para o motor 2. Assim, determinando se o movimento será frente, ré ou parada do motor.

| Inputs | | Function |
|---|---|---|
| $V_{en}$ = H | C = H ; D = L | Forward |
| | C = L ; D = H | Reverse |
| | C = D | Fast Motor Stop |
| $V_{en}$ = L | C = X ; D = X | Free Running Motor Stop |

L = Low H = High X = Don't care

**Fig 2.** Tabela de Funções (Datasheet), Módulo L298N. [4]

### 2.4. Aplicativo

O aplicativo utilizado para comandar os movimentos chama-se *Bluetooth RC Controller*, é possível fazer o download do mesmo no Play Store. É um aplicativo disponível para aparelhos com sistema Android.

O aplicativo oferece diversas funções, como movimentos direito/esquerda, frente/ré, acelerador controlado por PWM, sinaleira traseira, sinaleira frontal,

além de poder acessar o acelerômetro do celular para se movimentar através de variações de espaço.

Seu funcionamento, é basicamente enviando um caractere ASCII através da comunicação via Bluetooth, e então este caractere será compreendido pelo microcontrolador.

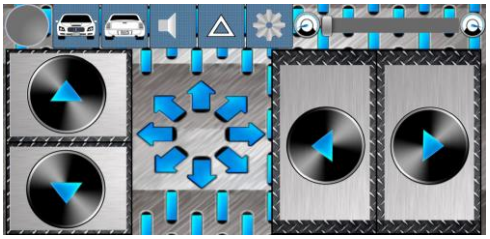


**Fig 3.** Aplicativo Bluetooth RC Controller.

### 2.5. Código

O código foi completamente desenvolvido no software Code Composer Studio.

O Code Composer Studio é um ambiente de desenvolvimento integrado (IDE) que suporta o portfólio de microcontroladores e processadores embarcados da TI. [2] Com ele é possível desenvolver códigos utilizando C/ C++ e Assembly, e ainda possibilita mesclar as duas linguagens.

O código foi realizado em vária partes, primeiro foi feito uma biblioteca onde foram definidas as funções de direção do motor, bem como as saídas dos motores. Onde IN1 e IN2 correspondem ao motor 1 e IN3 e IN4 ao motor 2.

```
10 #define IN1 BIT2
11 #define IN2 BIT3
12 #define IN3 BIT4
13 #define IN4 BIT5
14
15 void motor_init();
16 void motorFrente();
17 void motorRe();
18 void motorParar();
19 void motorDireita();
20 void motorEsquerda();
21 void DirecaoFrenteDireita();
22 void DirecaoFrenteEsquerda();
23 void DirecaoReDireita();
24 void DirecaoReEsquerda();
25 void velocidadeMotor();
```

No código abaixo (apenas uma parte do código total), foi implementado as lógicas de movimento dos motores, sendo que cada movimento possui sua lógica correspondente. Essas lógicas, serão interpretadas pelo driver L298N e o mesmo será responsável por acionar os motores.

```
1 #include <msp430.h>
2 #include <msp430f5529.h>
3 #include "motorDrivers.h"
4
5 void motor_init(){
6     P1DIR |= IN1 + IN2 + IN3 + IN4;
7     TA0CCR0 |= 255;
8     TA0CCTL1 |= OUTMOD_7;
9     TA0CTL |= TASSEL_2 + MC_1;
10 }
11
12 void motorRe(){
13     P1OUT |= IN1;
14     P1OUT &= ~IN2;
15     P1OUT &= ~IN3;
16     P1OUT &= ~IN4;
17 }
```

Abaixo se refere a main do projeto, nele são chamadas as funções dos motores e dos LEDs, sendo que a função acerca dos LEDs, são funções feitas em Assembly para atender ao critério imposto pelo professor.

Também, foi feito o código referente ao Bluetooth, onde temos o controle das portas Rx e Tx do microcontrolador. Para que o Tx e Rx possam ser usados em portas diferentes é necessário um controle por interrupção, onde o Tx é lido primeiro e o Rx carrega o caractere.

Outro fator importante é o USCIA0 para receber a informação de interrupção do dispositivo Bluetooth, conectado via UART. Ele define o nível conforme a solicitação e define a flag para a tarefa.

O UCA0RXBUF contém o último caractere provindo do registrador.

O UCA0CTL1 configura o clock ou frequência do PWM. Isso é feito através dos comandos UCSWRST e UCSSEL_2.

O UCA0BR1 é referente a taxa de transmissão (byte alto). O UCA0BR0 é referente a taxa de transmissão (byte baixo).

UCA0MCTL é referente ao controle de modulação do registrador.

```
26
27      //CONTROLE BLUETOOTH
28      P3SEL |= BIT3+BIT4;
29      UCA0CTL1 |= UCSWRST;
30      UCA0CTL1 |= UCSSEL_2;
31      UCA0BR0 = 109;
32      UCA0BR1 = 0;
33      UCA0MCTL |= UCBRS_1 + UCBRF_0;
34      UCA0CTL1 &= ~UCSWRST;
```

```
81      //CONTROLE UTILIZANDO INTERRUPÇÃO
82      UCA0IE |= UCRXIE;
83
84     __bis_SR_register(LPM0_bits + GIE);
85     __no_operation();
86  }
87
88  // Echo back RXed character, confirm
89
90   #pragma vector=USCI_A0_VECTOR
91  __interrupt void USCI_A0_ISR(void)
92  {
93      UCA0IFG &= ~UCRXIFG;
94      if(UCA0RXBUF == 'F'){ //Frente
95          motor_init();
96          motorFrente();
```

Temos a mescla entre C e Assembly, que foi feito através de funções externas, sendo que na main, apenas foi chamado a função externa criada, que no caso são funções para acender e desligar os LEDs.

Segue um trecho deste código abaixo:

```
File:     LedFrente.asm
*******************************************************

    .cdecls C,NOLIST,"msp430.h"       ; Processor specific definitions


;-------------------------------------------------------
; Acende os dois led's da frente que correspomdem aos farois
;-------------------------------------------------------
    .global   LedFrenteAcende            ; Declare symbol to be exported
    .sect ".text"                        ; Code is relocatable
LedFrenteAcende:    .asmfunc
     mov.b #00000110b, &P6OUT       ;Acende os farói

 .if ($defined(__MSP430_HAS_MSP430XV2_CPU__) | $defined
(__MSP430_HAS_MSP430X_CPU__))
     reta
 .else
     ret
 .endif
     .endasmfunc
```

Por último temos as funções para o buzzer, que quando acionado irá tocar uma música. Para esta execução foi feito uma função beep que define o tempo de execução da nota e seta/reseta o pino de saída.

Segue um trecho desse código abaixo:

```
void beep(unsigned int note, unsigned int duration)
{
    int i;
    long delay = (long)(10000/note);  //SEMIPERIODO
    long time = (long)((duration*100)/(delay*2));
    for (i=0;i<time;i++)
    {
        P8OUT |= BIT2;      //PINO P1.2 SETADO
        delay_us(delay);    //DELAY DO SEMIPERIODO
        P8OUT &= ~BIT2;     //RESETA O ESTADO DO PIN
        delay_us(delay);    //DELAY DO SEMIPERIODO
    }
```

Outra parte importante foi a função que define como será a música, essa função basicamente decide quanto tempo terá cada nota.

```
}
void play(){

    beep(a, 500);
        delay_ms(100);
        beep(a, 500);
        delay_ms(100);
        beep(a, 500);
        delay_ms(100);
        beep(f, 350);
```

O código seguiu o fluxograma que está no APÊNDICE B, e o código completo pode ser visualizado no APÊNDICE A.

## 3. RESULTADOS

O veículo consegue atender com muita eficiência todos os comandos propostos pelo o projeto, sua comunicação com o aplicativo não apresenta nenhuma irregularidade, através dele é possível mover o veículo em todas as direções, incluindo a opção de realizar movimentos através do acelerômetro.

Tentou-se implementar o controle de velocidade de acordo com os caracteres enviados pela parte de aceleração do aplicativo, porém encontrou-se muita dificuldade na implementação do PWM, pois quando eram realizados os testes a função de controle de velocidade fazia com que o veículo não respondesse a nenhum comando. Acredita-se que o clock do bluetooth poderia estar afetando a aplicação do TIMER_A. Então assumisse-se que uma possível solução seja a implementação de múltiplos clock's no projeto.

Os testes foram realizados com o carrinho fixo e suas rodas suspensas, de modo que fosse possível observar todos os seus movimentos. Em seguida, foram realizados com o carrinho em uma superfície com mais atrito. Desta forma, foi possível observar suas características e limitações de movimentos. Como por exemplo, travessias em tapetes, bloqueios de movimento, etc.

Também foi otimizado o espaço na carcaça do carrinho, para ser mais viável a movimentação sem maiores dificuldades. Nas figuras 4 e 5 mostram como foi utilizado o espaço a fim de colocar todos os componentes sem que atrapalhasse o seu desempenho.

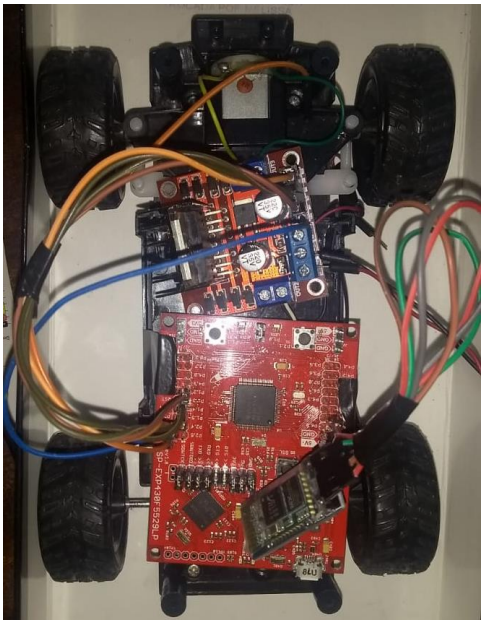


**Fig 4.** Carrinho com seus componentes.

**Fig 5.** Montagem completa do veículo.

## 4. CONCLUSÕES

Neste projeto foi possível aprender e evoluir no uso do microcontrolador, bem como possibilitar soluções para possíveis situações na vida profissional.

No quesito solução/problema, o sistema apresentou todos os resultados esperados e possibilita integrações futuras de novos periféricos, como câmeras, sensores, garras robóticas, etc. Visando por exemplo, operações de resgate ou situações de alto risco para o ser humano.

Também, os integrantes foram capazes de desenvolver habilidades em programação, focando em aplicações com microcontrolador. Detectando problemas e solucionando os mesmos com mais eficiência.

## 5. REFERÊNCIAS


[1] SOUZA. Rafael A. Veículo controlado remotamente através do sensor acelerômetro de celular com sistema operacional Android. Tese (Graduação em Engenharia Eletrônica) – Universidade Federal do Rio de Janeiro. 2016.

[2] MOURA. Gabriel N., RALL. Ricardo. Carro robô controlado por dispositivos moveis via tecnologia bluetooth. Artigo (Graduação em Análise e Desenvolvimento de Sistemas) –Fatec Botucatu, São Paulo. 2019.

[3] TEXAS INSTRUMENTS. Descrição e Parâmetros do MSP430F5529LP. Disponível em < http://www.ti.com/product/MSP430F5529>. Acesso em 8 de dezembro de 2019.

[4] DUAL FULL-BRIDGE DRIVER - L298. Disponível em: <https://pdf1.alldatasheet.com/datasheetpdf/view/22440/STMICROELECTRONICS/L298N.html>. Acesso em: 20 de outubro de 2019.

[5] ÜNSALAN, C., GÜRHAN, H. D., Programmable Microcontrollers with Applications – MSP430 Launchpad with CCS and Grace, McGraw-Hill, 2014.

[6] DAVIES, J., MSP430 Microcontroller Basics, Elsevier, 2008.

**APÊNDICE A**

**Funções do veículo:**

```
#include <msp430.h>
#include <msp430f5529.h>
#include "motorDrivers.h"

void motor_init(){
P1DIR |= IN1 + IN2 + IN3 + IN4;          //(IN1 E IN2) MOTOR TRASEIRO //(IN3 E IN4) MOTOR DIANTEIRO
   //P2DIR |= ENABLE;
  // P2OUT |= BIT0;
   TA0CCR0 |= 255;
   TA0CCTL1 |= OUTMOD_6;
   TA0CTL |= TASSEL_2 + MC_1;
}

void motorRe(){
   P1OUT |= IN1;
   P1OUT &= ~IN2;
   P1OUT &= ~IN3;
   P1OUT &= ~IN4;
}

void motorFrente(){
   P1OUT |= IN2;
   P1OUT &= ~IN1;
   P1OUT &= ~IN3;
   P1OUT &= ~IN4;
}

void motorParar(){
   P1OUT &= ~IN1;
   P1OUT &= ~IN2;
   P1OUT &= ~IN3;
   P1OUT &= ~IN4;
}

void motorDireita(){
   P1OUT &= ~IN2;
   P1OUT &= ~IN1;
   P1OUT |= IN3;
   P1OUT &= ~IN4;
}

void motorEsquerda(){
   P1OUT &= ~IN2;
   P1OUT &= ~IN1;
   P1OUT |= IN4;
   P1OUT &= ~IN3;
}

void DirecaoFrenteDireita(){
   P1OUT |= IN2;
   P1OUT &= ~IN1;
   P1OUT |= IN3;
   P1OUT &= ~IN4;
}

void DirecaoFrenteEsquerda(){
   P1OUT |= IN2;
   P1OUT &= ~IN1;
   P1OUT |= IN4;
   P1OUT &= ~IN3;
}

void DirecaoReDireita(){
   P1OUT |= IN1;
   P1OUT &= ~IN2;
   P1OUT |= IN3;
   P1OUT &= ~IN4;
}

void DirecaoReEsquerda(){
   P1OUT |= IN1;
   P1OUT &= ~IN2;
   P1OUT |= IN4;
   P1OUT &= ~IN3;
}

/*void velocidadeMotor(int speed){

   TA0CCR1 |= speed * 255;
   P1SEL |= BIT0;


}*/
void delay_ms(unsigned int ms )
{
   unsigned int i;
   for (i = 0; i<= ms; i++)
      __delay_cycles(500); //SUSPENDE A EXECUCAO POR 500MS
}

void delay_us(unsigned int us )
{
   unsigned int i;
   for (i = 0; i<= us/2; i++)
      __delay_cycles(1);
}
void beep(unsigned int note, unsigned int duration)
{
   int i;
   long delay = (long)(10000/note);  //SEMIPERIODO DE CADA NOTA.
   long time = (long)((duration*100)/(delay*2)); //TEMPO TOTAL DE EXECUCAO DA NOTA
```

```
   for (i=0;i<time;i++)
   {
      P8OUT |= BIT2;     //PINO P8.2 SETADO
      delay_us(delay);   //DELAY DO SEMIPERIODO
      P8OUT &= ~BIT2;     //RESETA O ESTADO DO
PINO
      delay_us(delay);   //DELAY DO SEMIPERIODO
   }
   delay_ms(20); //DELAY MINIMO PARA SEPARAR
UMA NOTA DA OUTRA
}
void play(){

   beep(a, 500);
      delay_ms(100);
      beep(a, 500);
      delay_ms(100);
      beep(a, 500);
      delay_ms(100);
      beep(f, 350);
      delay_ms(100);
      beep(cH, 150);
      delay_ms(100);
      beep(a, 500);
      delay_ms(100);
      beep(f, 350);
      delay_ms(100);
      beep(cH, 150);
      delay_ms(100);
      beep(a, 650);
      delay_ms(100);

      delay_ms(350);
      //INVERTALO ENTRE PRIMEIRA E SEGUNDA
PARTE

      beep(eH, 500);
      delay_ms(100);
      beep(eH, 500);
      delay_ms(100);
      beep(eH, 500);
      delay_ms(100);
      beep(fH, 350);
      delay_ms(100);
      beep(cH, 150);
      delay_ms(100);
      beep(gS, 500);
      delay_ms(100);
      beep(f, 350);
      delay_ms(100);
      beep(cH, 150);
      delay_ms(100);
      beep(a, 650);

      delay_ms(350);
      beep(aH, 500);
      delay_ms(100);
      beep(a, 300);
      delay_ms(100);
      beep(a, 150);
      delay_ms(100);
      beep(aH, 400);
      delay_ms(100);
      beep(gSH, 200);
      delay_ms(100);
      beep(gH, 200);
      delay_ms(100);
      beep(fSH, 125);
      delay_ms(100);
      beep(fH, 125);
      delay_ms(100);
      beep(fSH, 250);

      delay_ms(250);

      beep(aS, 250);
      delay_ms(100);
      beep(dSH, 400);
      delay_ms(100);
      beep(dH, 200);
      delay_ms(100);
      beep(cSH, 200);
      delay_ms(100);
      beep(cH, 125);
      delay_ms(100);
      beep(b, 125);
      delay_ms(100);
      beep(cH, 250);

      delay_ms(250);

      beep(f, 125);
      delay_ms(100);
      beep(gS, 500);
      delay_ms(100);
      beep(f, 375);
      delay_ms(100);
      beep(a, 125);
      delay_ms(100);
      beep(cH, 500);
      delay_ms(100);
      beep(a, 375);
      delay_ms(100);
      beep(cH, 125);
      delay_ms(100);
      beep(eH, 650);

      beep(aH, 500);
      delay_ms(100);
```

```
    beep(a, 300);
    delay_ms(100);
    beep(a, 150);
    delay_ms(100);
    beep(aH, 400);
    delay_ms(100);
    beep(gSH, 200);
    delay_ms(100);
    beep(gH, 200);
    delay_ms(100);
    beep(fSH, 125);
    delay_ms(100);
    beep(fH, 125);
    delay_ms(100);
    beep(fSH, 250);

    delay_ms(250);

    beep(aS, 250);
    delay_ms(100);
    beep(dSH, 400);
    delay_ms(100);
    beep(dH, 200);
    delay_ms(100);
    beep(cSH, 200);
    delay_ms(100);
    beep(cH, 125);
    delay_ms(100);
    beep(b, 125);
    delay_ms(100);
    beep(cH, 250);

    delay_ms(250);

    beep(f, 250);
    delay_ms(100);
    beep(gS, 500);
    delay_ms(100);
    beep(f, 375);
    delay_ms(50);
    beep(cH, 125);
    delay_ms(100);
    beep(a, 500);
    delay_ms(100);
    beep(f, 375);
    delay_ms(100);
    beep(cH, 125);
    delay_ms(100);
    beep(a, 650);
}
```

**Biblioteca para as funções:**

```
//#ifndef MOTORDRIVERS_H_
//#define MOTORDRIVERS_H_
#define IN1 BIT2
#define IN2 BIT3
#define IN3 BIT4
#define IN4 BIT5
//#define ENABLE BIT2

#define c 261
#define d 294
#define e 329
#define f 349
#define g 391
#define gS 415
#define a 440
#define aS 455
#define b 466
#define cH 523
#define cSH 554
#define dH 587
#define dSH 622
#define eH 659
#define fH 698
#define fSH 740
#define gH 784
#define gSH 830
#define aH 880

void motor_init();
void motorFrente();
void motorRe();
void motorParar();
void motorDireita();
void motorEsquerda();
void DirecaoFrenteDireita();
void DirecaoFrenteEsquerda();
void DirecaoReDireita();
void DirecaoReEsquerda();
void velocidadeMotor();
void play();
```

**Programa principal:**

```
#include <msp430f5529.h>
#include <msp430.h>
#include <stdint.h>
#include "motorDrivers.h"

extern void LedFrenteAcende(void);
extern void LedFrenteApaga(void);
extern void LedTraseiraAcende(void);
extern void LedTraseiraApaga(void);

int main(void)
{
```

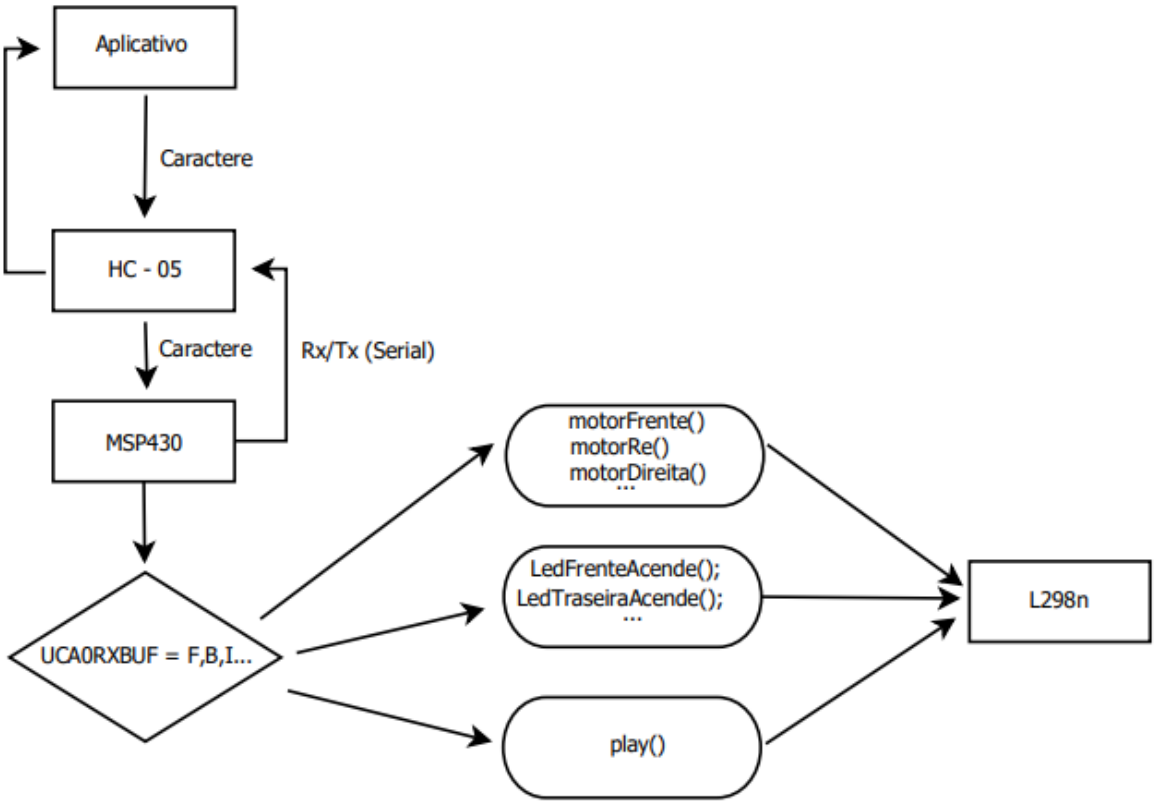
```
    WDTCTL = WDTPW | WDTHOLD; //stop watchdog timer

    //LEDS
    P6DIR |= BIT1+BIT2;                 //LED's Frente
    P2DIR |= BIT4+BIT5;                 //LED's Traseira
    P8DIR |= BIT2;

    //CONTROLE BLUETOOTH
    P3SEL |= BIT3+BIT4;                 // P3.3,3.4 = USCI_A1 TXD/RXD
    UCA0CTL1 |= UCSWRST;                // **Estado da máquina em reset**
    UCA0CTL1 |= UCSSEL_2;               // SMCLK
    UCA0BR0 = 109;                      // 1MHz 115200 (see User's Guide)
    UCA0BR1 = 0;                        // 1MHz 115200
    UCA0MCTL |= UCBRS_1 + UCBRF_0;      // Modulation UCBRSx=1, UCBRFx=0
    UCA0CTL1 &= ~UCSWRST;               // **Inicializa estado USCI da máquina**


  //CONTROLE UTILIZANDO INTERRUPÇÃO
  UCA0IE |= UCRXIE;                     // Habilita interrupção USCI_A1 RX

  __bis_SR_register(LPM0_bits + GIE);   // Entre LPM0, interrupção habilitada
  __no_operation();                     // Para debugger

}

 // Echo back RXed character, confirm TX buffer is ready first

 #pragma vector=USCI_A0_VECTOR
__interrupt void USCI_A0_ISR(void)
{
   UCA0IFG &= ~UCRXIFG;
   if(UCA0RXBUF == 'F'){ //Frente
      motor_init();
      motorFrente();

   }else if(UCA0RXBUF == 'B'){ // Ré
      motor_init();
      motorRe();

   }else if(UCA0RXBUF == 'I'){//FrenteDireita
      motor_init();
      DirecaoFrenteDireita();

   }else if(UCA0RXBUF == 'S'){ //Parar
      motor_init();
      motorParar();

   }else if(UCA0RXBUF == 'G'){ //FrenteEsquerda
      motor_init();
      DirecaoFrenteEsquerda();

   }else if(UCA0RXBUF == 'L'){ //Esquerda
      motor_init();
      motorEsquerda();

   }else if(UCA0RXBUF == 'R'){ //Direita
      motor_init();
      motorDireita();

   }else if(UCA0RXBUF == 'H'){//ReEsquerda
      motor_init();
      DirecaoReEsquerda();

   }else if(UCA0RXBUF == 'J'){//ReDireita
      motor_init();
      DirecaoReDireita();

   }else if(UCA0RXBUF == 'W'){
      LedFrenteAcende();

   }else if(UCA0RXBUF == 'w'){
      LedFrenteApaga();

   }else if(UCA0RXBUF == 'U'){
      LedTraseiraAcende();

   }else if(UCA0RXBUF == 'u'){
      LedTraseiraApaga();

   }else if(UCA0RXBUF == 'V'){

        play();
   }/*else if(UCA0RXBUF == 'X'){
      motor_init();
      velocidadeMotor(0.10);

 }*/

 }
```

**Código em Assembly**

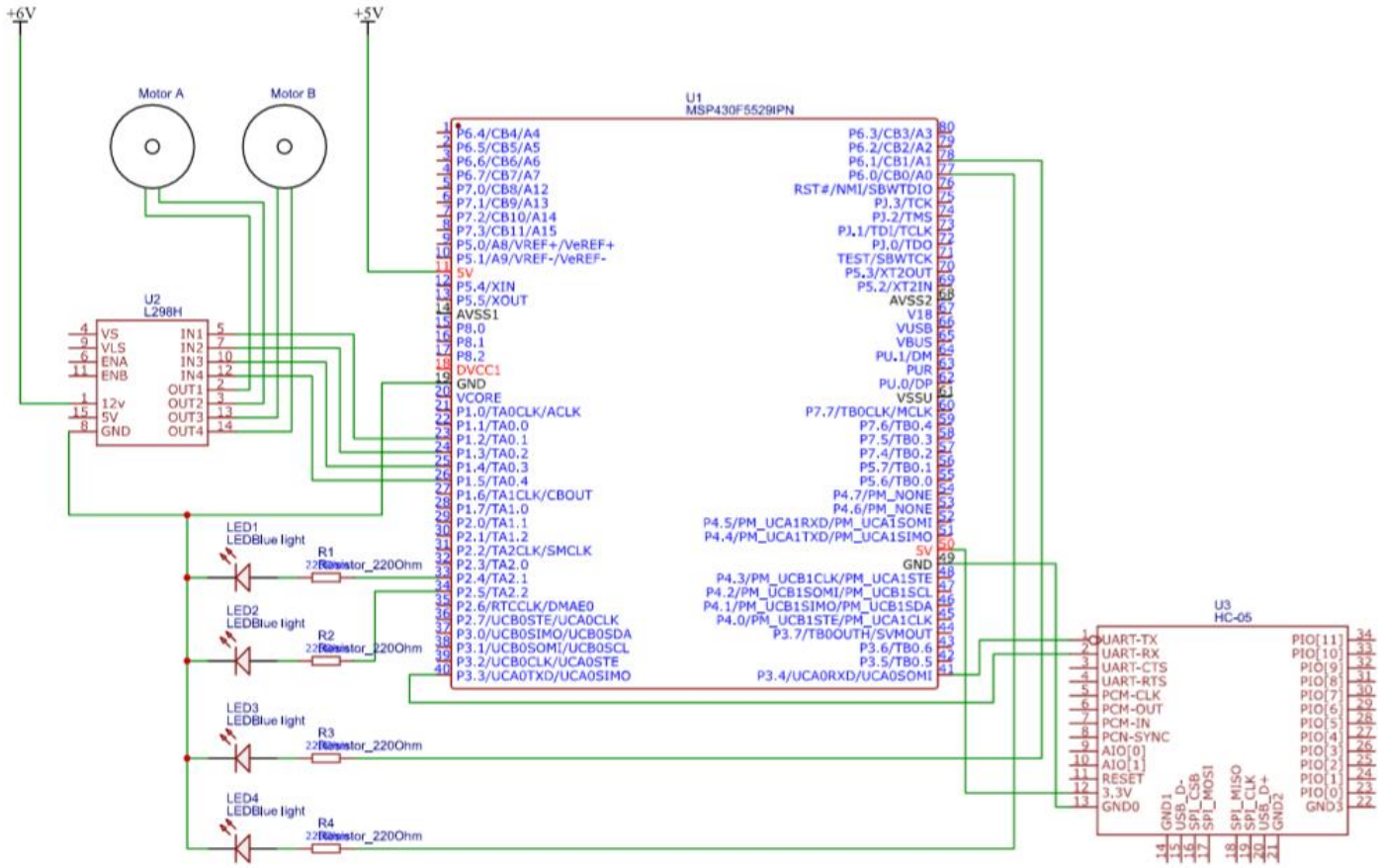
```
; File:     LedFrente.asm
;
*********************************************
```

```
    .cdecls C,NOLIST,"msp430.h"       ; Processor specific definitions


;=====================================
; Acende os dois led's da frente que correspondem aos farois
;=====================================
.global LedFrenteAcende       ; Declare symbol to be exported
    .sect ".text"             ; Code is relocatable
LedFrenteAcende:  .asmfunc
      mov.b #00000110b, &P6OUT      ;Acende os faróis

  .if ($defined(__MSP430_HAS_MSP430XV2_CPU__) | $defined(__MSP430_HAS_MSP430X_CPU__))
      reta
  .else
      ret
  .endif
       .endasmfunc
;=====================================
; Apaga os dois led's da frente que correspondem aos farois
;=====================================
.global LedFrenteApaga        ; Declare symbol to be exported
      .sect ".text"           ; Code is relocatable
LedFrenteApaga:  .asmfunc
      mov.b #00000000b, &P6OUT

  .if ($defined(__MSP430_HAS_MSP430XV2_CPU__) | $defined(__MSP430_HAS_MSP430X_CPU__))
      reta
  .else
      ret
  .endif
      .endasmfunc

;=====================================
; Acende os dois led's da traseira que correspondem as lanternas
;=====================================
.global LedTraseiraAcende     ; Declare symbol to be exported
      .sect ".text"           ; Code is relocatable
LedTraseiraAcende:  .asmfunc
      mov.b #00110000b, &P2OUT

  .if ($defined(__MSP430_HAS_MSP430XV2_CPU__) | $defined(__MSP430_HAS_MSP430X_CPU__))
      reta
  .else
      ret
  .endif
      .endasmfunc

;=====================================
; Apaga os dois led's da traseira que correspondem as lanternas
;=====================================
.global LedTraseiraApaga      ; Declare symbol to be exported
      .sect ".text"           ; Code is relocatable
LedTraseiraApaga:  .asmfunc
      mov.b #00000000b, &P2OUT

  .if ($defined(__MSP430_HAS_MSP430XV2_CPU__) | $defined(__MSP430_HAS_MSP430X_CPU__))
      reta
  .else
      ret
  .endif
      .endasmfunc
      .end
```

**APÊNDICE B**


Aplicativo
Caractere
HC - 05
Caractere
Rx/Tx (Serial)
MSP430
UCA0RXBUF = F,B,I...
motorFrente()
motorRe()
motorDireita()
...
LedFrenteAcende();
LedTraseiraAcende();
...
play()
L298n

**APÊNDICE C**


+6V
+5V
Motor A
Motor B
U1
MSP430F5529IPN
P6.4/CB4/A4
P6.5/CB5/A5
P6.6/CB6/A6
P6.7/CB7/A7
P7.0/CB8/A12
P7.1/CB9/A13
P7.2/CB10/A14
P7.3/CB11/A15
P5.0/A8/VREF+/VeREF+
P5.1/A9/VREF-/VeREF-
5V
P5.4/XIN
P5.5/XOUT
AVSS1
P8.0
P8.1
P8.2
DVCC1
GND
VCORE
P1.0/TA0CLK/ACLK
P1.1/TA0.0
P1.2/TA0.1
P1.3/TA0.2
P1.4/TA0.3
P1.5/TA0.4
P1.6/TA1CLK/CBOUT
P1.7/TA1.0
P2.0/TA1.1
P2.1/TA1.2
P2.2/TA2CLK/SMCLK
P2.3/TA2.0
P2.4/TA2.1
P2.5/TA2.2
P2.6/RTCCLK/DMAE0
P2.7/UCB0STE/UCA0CLK
P3.0/UCB0SIMO/UCB0SDA
P3.1/UCB0SOMI/UCB0SCL
P3.2/UCB0CLK/UCA0STE
P3.3/UCA0TXD/UCA0SIMO
P6.3/CB3/A3
P6.2/CB2/A2
P6.1/CB1/A1
P6.0/CB0/A0
RST#/NMI/SBWTDIO
PJ.3/TCK
PJ.2/TMS
PJ.1/TDI/TCLK
PJ.0/TDO
TEST/SBWTCK
P5.3/XT2OUT
P5.2/XT2IN
AVSS2
V18
VUSB
VBUS
PU.1/DM
PUR
PU.0/DP
VSSU
P7.7/TB0CLK/MCLK
P7.6/TB0.4
P7.5/TB0.3
P7.4/TB0.2
P5.7/TB0.1
P5.6/TB0.0
P4.7/PM_NONE
P4.6/PM_NONE
P4.5/PM_UCA1RXD/PM_UCA1SOMI
P4.4/PM_UCA1TXD/PM_UCA1SIMO
5V
GND
P4.3/PM_UCB1CLK/PM_UCA1STE
P4.2/PM_UCB1SOMI/PM_UCB1SCL
P4.1/PM_UCB1SIMO/PM_UCB1SDA
P4.0/PM_UCB1STE/PM_UCA1CLK
P3.7/TB0OUTH/SVMOUT
P3.6/TB0.6
P3.5/TB0.5
P3.4/UCA0RXD/UCA0SOMI
U2
L298H
VS
VLS
ENA
ENB
12v
5V
GND
IN1
IN2
IN3
IN4
OUT1
OUT2
OUT3
OUT4
LED1
LEDBlue light
R1
Resistor_220Ohm
LED2
LEDBlue light
R2
Resistor_220Ohm
LED3
LEDBlue light
R3
Resistor_220Ohm
LED4
LEDBlue light
R4
Resistor_220Ohm
U3
HC-05
UART-TX
UART-RX
UART-CTS
UART-RTS
PCM-CLK
PCM-OUT
PCM-IN
PCN-SYNC
AIO[0]
AIO[1]
RESET
3.3V
GND0
PIO[11]
PIO[10]
PIO[9]
PIO[8]
PIO[7]
PIO[6]
PIO[5]
PIO[4]
PIO[3]
PIO[2]
PIO[1]
PIO[0]
GND3
GND1
USB_D-
SPI_CSB
SPI_MOSI
SPI_MISO
SPI_CLK
USB_D+
GND2