Ponto de Controle 4 – Grupo 02

Carrinho controlado por microcontrolador via Bluetooth

Brunno Cardoso de Santana

Milena Andressa da Silva Santos

Engenharia Eletrônica

Faculdade Gama – Universidade de Brasília

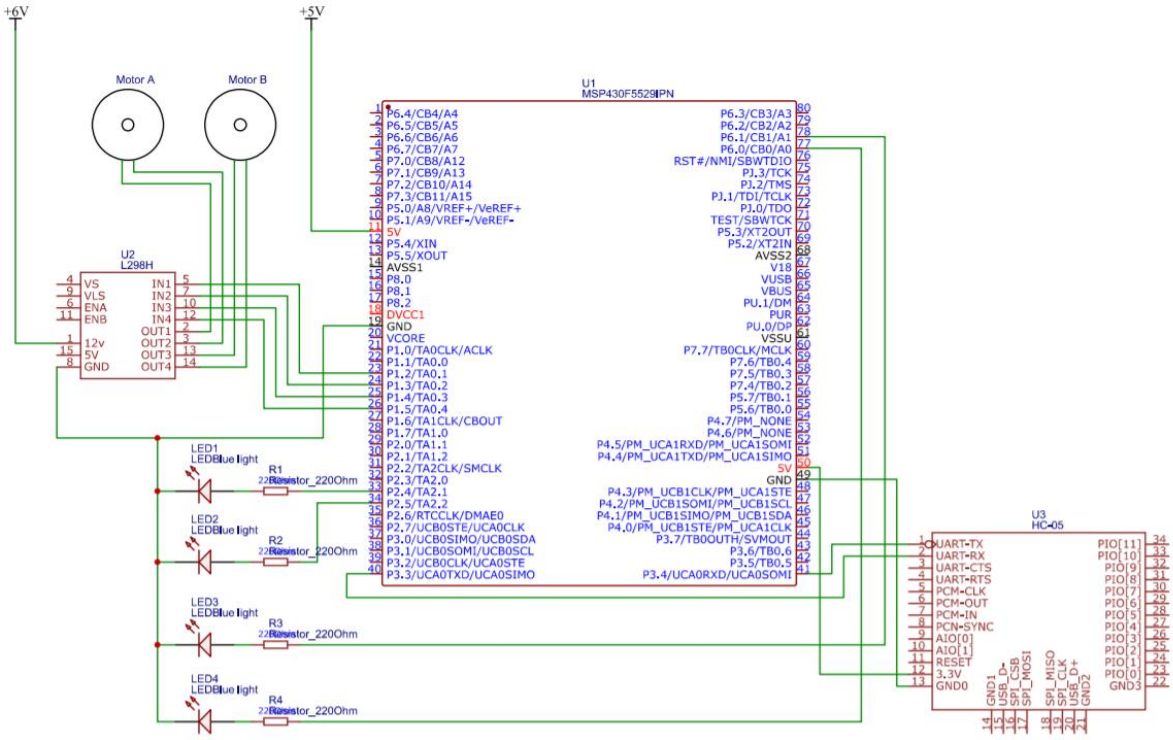
O projeto inicial era desenvolver um robô seguidor de linha, porém, foi sugerido pelo professor a realização de um carrinho controlado remotamente. Então, a ideia do novo projeto é que a partir de comandos enviados por um aplicativo no celular o carrinho possa ser controlado em qualquer direção.

O carrinho possui dois motores DC, sendo um responsável pelos movimentos (FRENTE & RÉ) e o outro pelos movimentos (ESQUERDA & DIREITA). Também serão utilizados um microcontrolador (MSP430F5529), driver para motor DC (L298n) e um módulo bluetooth (HC - 05).

O microcontrolador será responsável por interpretar os comandos recebidos pelo módulo bluetooth e logo após, o microcontrolador enviará a lógica correspondente ao comando recebido para o driver, que interpretará essa lógica e rotacionará um dos motores ou ambos.

O aplicativo usado para enviar os comandos ao módulo bluetooth será o BLUETOOTH RC CONTROLLER, que possui os camandos de direções, ascender LED's e direções com acelerômetro do celular.

A montagem do circuito está definida no esquemático abaixo:



O primeiro desafio foi correlacionar a lógica do driver com o código desenvolvido, pois, a programação deve fazer o controle dos pinos da MSP430 de forma que ocorra uma variação de tensão nos mesmos (low & high). Então, com a ajuda do datasheet foi possível desenvolver essa ligação entre microcontrolador e driver dc. O segundo desafio foi integrar o bluetooth ao código, pois, foi necessário utilizar a comunicação UART e interrupções para controle dos terminais Rx/Tx do módulo bluetooth.

Código

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.

```
/*
 * motorDrivers.h
*/
#define IN1 BIT2
#define IN2 BIT3
#define IN3 BIT4
#define IN4 BIT5
```

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

```
#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
  // TA0CCR0 |= 255;
   //TA0CCTL1 |= OUTMOD_7;
   //TA0CTL |= TASSEL_2 + MC_1;
}

void motorRe(){
   P1OUT |= IN1;
   P1OUT &= ~IN2;
```

```
    P1OUT &= ~IN3;
    P1OUT &= ~IN4;
}
```

O código abaixo se refere a main do projeto, nele são chamados as funções do motor e dos leds, sendo que a função a cerca dos leds, são funções feitas em assembly para atender ao critério imposto pelo professor.

Também, na main foi feito o código referente ao bluetooth, onde temos o controle das porta Rx e Tx do microcontrolador. Para que o Tx & 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.

```
#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

    //CONTROLE BLUETOOTH
    P3SEL |= BIT3+BIT4;                    // P3.3,3.4 = USCI_A1 TXD/RXD
    UCA0CTL1 |= UCSWRST;                   // **Coloca a máquina em estado de  reset**
    UCA0CTL1 |= UCSSEL_2;                  // SMCLK
    UCA0BR0 = 109;                         // 1MHz 115200
    UCA0BR1 = 0;                           // 1MHz 115200
    UCA0MCTL |= UCBRS_1 + UCBRF_0;         // Modulação UCBRSx=1, UCBRFx=0
    UCA0CTL1 &= ~UCSWRST;                  // **Inicia a máquina de estado USCI**

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

  __bis_SR_register(LPM0_bits + GIE);      // Entre LPM0, interrupção habilitada
  __no_operation();                        // Para debugger
}
        //Espera o Tx ser lido primeiro
 #pragma vector=USCI_A0_VECTOR
__interrupt void USCI_A0_ISR(void)
 {
   UCA0IFG &= ~UCRXIFG;
```

```
    if(UCA0RXBUF == 'F'){ //Frente
        motor_init();
        motorFrente();
```

Por último 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 LED's. Segue um trecho deste código abaixo:

```
;-------------------------------------------------------------------
; Acende os dois led's da frente que correspodentes 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
```