Federal de Santa Catarina – UFSC
Centro Tecnológico – CTC
Departamento de Automação e Sistemas – DAS
Disciplina DAS5307 – Sistemas de Automação Discreta

# Controle de trajetória de ônibus elétrico

Integrantes:
Guilherme dos Santos (18202008)
Francisco Leonardo de Pinho (20205661)

Florianópolis, 22 de dezembro de 2022.

## 1. Introdução

Este relatório visa demonstrar a implementação técnica de controle em um microcontrolador ATMega88PA montado em um sistema dinâmico para aplicação do controle de um modelo de ônibus elétrico simulado em kit Arduino.

A ideia do projeto é fazer com que um ônibus consiga sair da sua parada inicial, chegar na próxima parada, e por fim, chegar ao fim com o melhor tempo possível sem que a bateria acabe no processo.

A dificuldade do projeto está em encontrar um balanceamento entre a velocidade e o gasto energético da bateria do ônibus, devemos encontrar um meio termo de maneira a maximizar a eficiência e reduzir o tempo gasto no processo, além do tempo que ele passa carregando em cada estação.

# 2. Dispositivos e Hardware

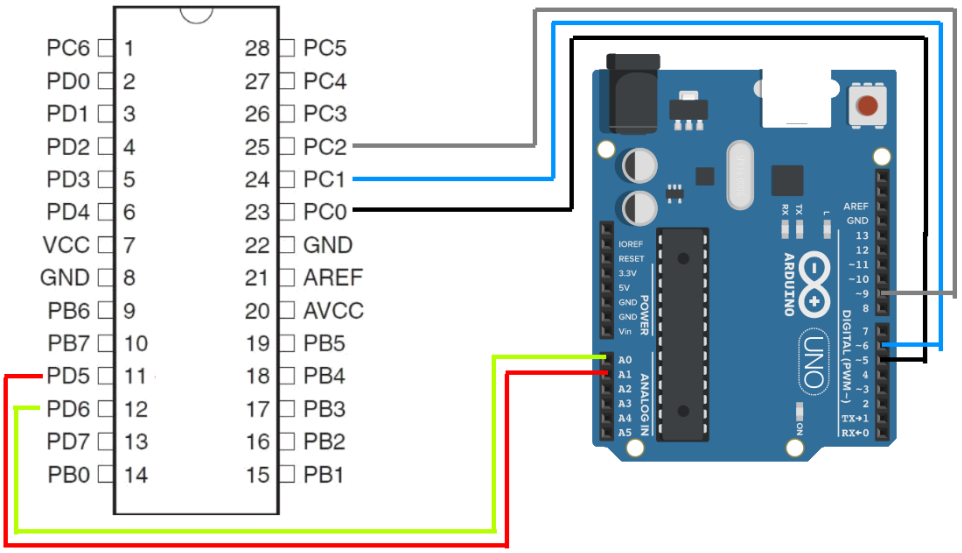
## 2.1. Modelos e especificações

Para o desenvolvimento do projeto foi utilizado alguns materiais que foram fornecidos pelo laboratório, dos quais são:

- Circuito integrado do microcontrolador (ATMEGA88PA-AU)
- Placa de contatos (protoboard);
- Gravador USBasp;
- Conectores, resistores, capacitores;
- Kit Arduino para simulação do sistema a controlar.

## 2.2. Configurações

A definição dos pinos do Arduino e do Microcontrolador ATMega88PA podem ser visualizadas na seguinte imagem




A ligação realizada pelo circuito foi montada dessa forma

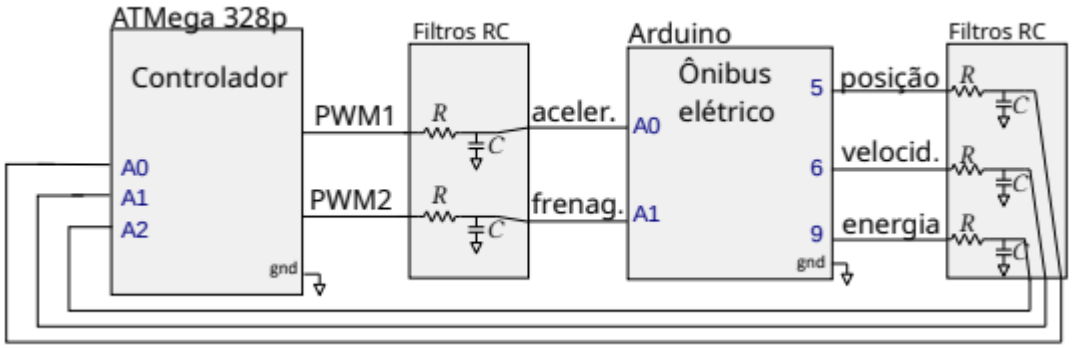



Diagrama do sistema de controle automático da simulação do ônibus elétrico

Suas conexões foram realizadas dessa forma:

Sinais de saída (medição do estado do ônibus),:

- Pino 5: representa a posição do ônibus, que varia de 0 a 1.000 m.
- Pino 6: indica o valor da velocidade, que varia de 0 a 72 km/h.
- Pino 9: fornece o estado da bateria, a qual pode ser recarregada nos pontos de parada.
- Pino A0: aceleração propriamente dita, definindo valores positivos ou zero.

Configuração da programação em C:

```
//  Inclui bibliotecas necessarias
#include <util/delay.h>
#include <avr/io.h>
#include <avr/interrupt.h>


//  Define Clock da CPU, _delay_ms
#define F_CPU 1000000UL

//  Posicao dos pinos no ATmega88PA
#define pinposition 0b0000
#define pinspeed 0b0001
#define pinenergy 0b0010

// Posicao dos pontos de recarga da bateria
#define BUS_STOP_1 250
#define BUS_STOP_2 450
#define BUS_STOP_3 700


//  Variavel bateria carregando
int carregar = 0;

//  Conta o numero de medias do ADC
int contagemdamedia = 0;
```

# 3. Controle do Microcontrolador

## 3.1. Posição (Modelo cinemático)

O modelo do ônibus descreve o comportamento cinemático do veículo, relacionando acelerações instantâneas com a velocidade e a posição ao longo do tempo. Supõe-se que efeitos dinâmicos (atritos, inclinação do trajeto, potência dos motores) sejam tratados pelo sistema mecânico de tal forma que valores de aceleração comandados pelo controlador sejam sempre aplicados ao veículo. Desta forma, o modelo do ônibus é dado por:

$$v(t) = \int_0^t a(\sigma)d\sigma,$$
$$s(t) = \int_0^t v(\sigma)d\sigma.$$

Portanto, a simulação consiste da integração iterativa da aceleração instantânea a(t) para cálculo da velocidade v(t), e da velocidade para cálculo da posição s(t). A aceleração é calculada por:

$$a(t) = K_a P_a(t) - K_b P_b(t),$$

com Pa(t) e Pb(t) representando os sinais de aceleração e de frenagem e Ka e Kb constantes. Considera-se, ainda, que a velocidade máxima do ônibus é de 72 km/h, ou 20 m/s.

### 3.2. Modelo de consumo energético

Para cálculo do consumo instantâneo de energia do ônibus, usa-se formula ad hoc dada por

:

$$C_e(t) = -K_{v1}v(t) - K_{v2}v(t)^2 - K_{acel}a(t), \qquad \text{se } a(t) \geq 0$$
$$C_e(t) = K_{fren}a(t), \qquad \text{se } a(t) < 0.$$

Note que, neste modelo, não há penalização para uso simultâneo de acelerador e freio, limitando o realismo da representação. A energia estocada é calculada como a integral do consumo instantâneo.

### 3.3. Atribuição de parâmetros

Para a integração numérica do modelo, usa-se o passo de simulação de 10 ms. A integração é feita com a fórmula de Euler. As constantes de aceleração foram definidas com base nos valores aplicados nas entradas de aceleração e de frenagem do Arduino, isto é, valores de 0 a 1023. Optou-se por definir a aceleração de 1 m/s2 para valor máximo, que é de -4 m/s2 como mínimo. Assim, tem-se que

$$K_a = 0.001$$
$$K_b = 0.004.$$

Os parâmetros relacionados ao consumo de energia são dados por:

$$K_{v1} = 0.1$$
$$K_{v2} = 0.02$$
$$K_{acel} = 5$$
$$K_{fren} = 1.$$

## 4. Código do Projeto

```
void setup(){
//  Define Pinos Digitais de saida(1) e de entrada(0)
    DDRD = 0b01100000;
//  Define os Pinos Analogicos conectados
    DDRC = 0b00000011;

//  Limpa os registradores
    TCCR0A = 0;
    TCCR0B = 0;

//  Preescaler clock do ADC em 128, 10bits de resolução
    ADCSRA |= (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);
//  Voltagem referencial do ADC
    ADMUX  |= (1<<REFS0) ;

//  Ativa as interrupcoes
    sei();
}


int analogRead(int pintoread){
//  zera o MUX que escolhe o pino a ser lido
    ADMUX = ADMUX & 0xF0;

//  escreve no MUX qual é o pino que vai ler
    ADMUX  |= pintoread;

//  delay para manter o controlador sincronizado
    _delay_ms(1);

//  Ativa o ADC, inicia a medicao
    ADCSRA |= (1<<ADEN)  | (1<<ADSC);


//  espera a medicao ter sido feita e o resultado estar pronto
    while (bit_is_set(ADCSRA, ADSC));

//  guarda o resultado da medição na variavel
    ADC_10bit_Result = ADC;

//  retorna o resultado da medicao
    return ADC_10bit_Result;
}
```

```
void analogWritePD5(int valuetowrite){
//  Ativa troca de estado D5 e configura o modo PWM
    TCCR0A |= (1<<COM0B1) | (1<<WGM01) | (1<<WGM00);

//  Prescaler 64
    TCCR0B |=  (1<<CS01) | (1<<CS00);

//  Troca o valor do PWM no registrador de comparacao, troca o pino D5
    OCR0B = valuetowrite;
}


int main(void)
{
    setup();
    while (1)
    {
//      Delay de 70ms, 30ms sao de processamento das medicoes
        _delay_ms(70);

        // zera as variavéis para rescrever a média novamente
        bus_position = 0;
        bus_speed = 0;
        bus_energy = 0;

//      Estabiliza o sinal tirando a media
        while (contagemdamedia < 10) {
            contagemdamedia++;

//      Posicao 0m até 1000m
            bus_position = bus_position + 0.1*(analogRead(pinposition) - 40)*1.042;

//      velocidade variando de 0 até 20 metros por segundo
            bus_speed = bus_speed + 0.1*(analogRead(pinspeed) - 182)*0.0250;

//      energia vaiando de 0 % até 100 %
            bus_energy = bus_energy + 0.1*(analogRead(pinenergy) - 200)*0.125;
        }

        contagemdamedia = 0;

        if (bus_position < 242) {
            PORTD &= ~(1 << PD6);
            carregar++;
            if (bus_speed < 6.1) {
                analogWritePD5(255);
```

```
                } else if (bus_speed < 7.1) {
                analogWritePD5(32);
                } else {
                analogWritePD5(0);
            }
            } else if (bus_position > 265 && bus_position < 442) {
            PORTD &= ~(1 << PD6);
            carregar++;
            if (bus_speed < 7.3) {
                analogWritePD5(255);
                } else if (bus_speed < 8.3) {
                analogWritePD5(32);
                } else {
                analogWritePD5(0);
            }
            } else if (bus_position > 465 && bus_position < 692) {
            PORTD &= ~(1 << PD6);
            carregar++;
            if (bus_speed < 6) {
                analogWritePD5(255);
                } else if (bus_speed < 7) {
                analogWritePD5(32);
                } else {
                analogWritePD5(0);
            }
            } else if (bus_position > 720 && bus_position < 992) {
            PORTD &= ~(1 << PD6);
            carregar++;
            if (bus_speed < 4.8) {
                analogWritePD5(255);
                } else if (bus_speed < 5.8) {
                analogWritePD5(32);
                } else {
                analogWritePD5(0);
            }
            } else if (carregar >= 50) {
            analogWritePD5(0);
            PORTD |= (1 << PD6);
            if (bus_energy > 97) {
                carregar = 0;
            }
            } else {
            PORTD &= ~(1 << PD6);
            if (bus_speed < 4) {
                analogWritePD5(255);
                } else {
                analogWritePD5(0);
            }
        }
    }

    return(0);
}
```