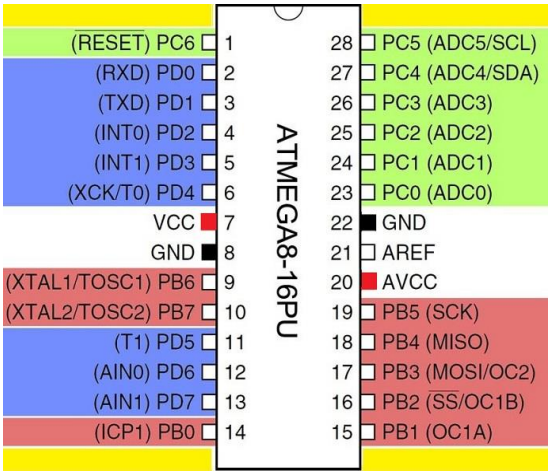
Experimento 1 - Entradas e saídas digitais
Fundamentação

Características de software

As entradas e saídas digitais dos microcontroladores da família AVR são realizadas através de registradores. O Atmega8 possui três portas de entrada e saída digital, as portas B, C e D.



Os registradores responsáveis pelo controle destas portas possuem a seguinte estrutura.

Registrador	Descrição
DDRx	**Usado para configurar a referida porta como entrada ou saída.**
PORTx	**Usado para enviar informações para a porta.**
PINx	**Usado para ler informações da porta.**

O x representa a porta b, C ou B.

Cada uma das portas é composta por 8 bits, e consequentemente por 8 entradas e/ou saídas. Assim, cada registrador é responsável por controlar 8 entradas e/ou saídas.

O registrador **DDRx (Data Direction Register)** determina se um pino é utilizado como entrada ou saída. Se o valor do bit no registrador é 0, o pino associado é considerado como entrada. Porém, se o valor do bit no registrador é 1, o pino associado é considerado como saída. Veja alguns exemplos.

```
DDRB = 0xFF;  // Configura toda a porta B (PORTB) como saída. (notação hexadecimal)
DDRC = 0b00000000; // Configura toda a porta C (PORTC) como entrada. (notação binária)
DDRD = 0b01001001; // Configura os pinos 0, 3 e 6 (PD0, PD3 e PD6) da porta D como saída, o restante como entrada.
DDRD = (1<<PD0) | (1<<PD3) | (1<<PD6); // Configura os mesmos pinos 0, 3 e 6 (PD0, PD3 e PD6) da porta D como saída, o restante como entrada.
```

O registrador **PORTx** determina se um pino de saída deve ficar em nível lógico alto (1) ou baixo (0). Se o valor do bit no registrador é 0, o pino associado permanece em nível 0. Porém, se o valor do bit no registrador é 1, o pino associado permanece em nível 1. Veja alguns exemplos.

```
PORTB = 0xFF;  // Todos os pinos da porta B recebem nível 1.
```

PORTC = 0b00000000; // Todos os pinos da porta C recebem nível 0.
PORTD = 0b00001000; // O pino PD3 recebe nível 1, os demais recebem nível 0.
PORTD = (1<<PD0) | (1<<PD3) | (1<<PD6); // Os pinos PD0, PD3 e PD6 recebem nível 1, os demais recebem nível 0.

Para alterar um único pino em uma porta sem alterar o valor dos outros sete pinos é necessário realizar uma operação lógica sobre o valor da porta. Vela os exemplos a seguir.

PORTD |= (1<<PD0); Liga o pino 0 da porta D (PD0)
PORTC &=~(1<<PC3); Desliga o pino 3 da porta C (PC3)

O registrador **PINx** permite verificar se os pinos de entrada estão em nível lógico 0 ou 1. Como se trata de um registrador de 8 bits, 8 pinos são acessados ao mesmo tempo. Para se verificar o estado de um pino individualmente é necessário realizar operações lógicas com o valor do registrador.

estado = ((PIND & (1<<PD4)) != 0); // retorna falso se PD4 for 0 e verdadeiro se PD4 for verdadeiro

Este tipo de operação pode ser usado no programa, em uma tomada de decisão, com o comando **if** por exemplo. Veja a seguir.

```
if((PINC& (1<<PC1)) != 0) // se o pino PC1 for 1...
{
        PORTD = (1<<PD0); // liga o pino PD0
}
```

Para que o compilador interprete corretamente o nome dos registradores é necessário incluir no programa a biblioteca “io.h”. outra biblioteca importante é a biblioteca “delay.h” que permite, entre outras coisas, a utilização da função “_delay_ms(x)”, que para o programa por x milisegundos. Para incluir estas bibliotecas a sintaxe é:

```
#include <avr/io.h>
#include <util/delay.h>
```

A seguir temos um exemplo de programa que utiliza os conceitos apresentados.

```
#include <avr/io.h>
#define F_CPU 1000000 //define a frequência de clock
#include <util/delay.h>

int main(void)
{
    DDRC = (1<<PC0); // Configura o pino PC0 como saída
    while (1)
    {
        if((PINC&(1<<PC1)) != 0) // verifica o estado do pino PC1
        {
            PORTC|=(1<<PC0); // liga PC0
            _delay_ms(500); //aguarda meio segundo
            PORTC&=~(1<<PC0); //desliga PC0
            _delay_ms(500); //aguarda meio segundo
        }
    }
}
```

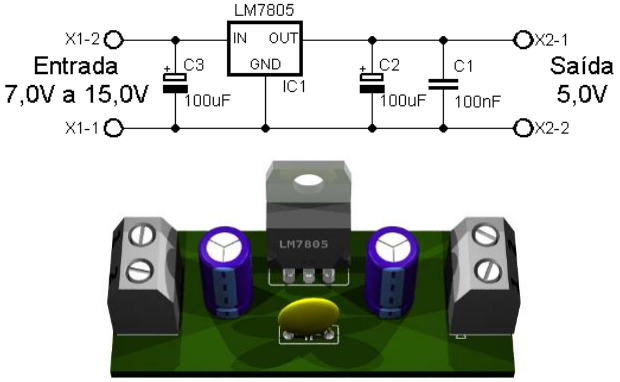
Características de hardware

A fonte de alimentação do microcontrolador

Para que um microcontrolador possa operar corretamente ele deve ser conectado a uma fonte de alimentação adequada. Esta fonte de alimentação deve ter uma tensão apropriada ao microcontrolador e aos periféricos utilizados. A família AVR aceita tensões de alimentação de 2,8 a 5,0V, porém na grande maioria das aplicações a tensão utilizada é de 5,0V. Isso quando a alimentação do circuito não é feita através de baterias, que exige circuitos apropriados.

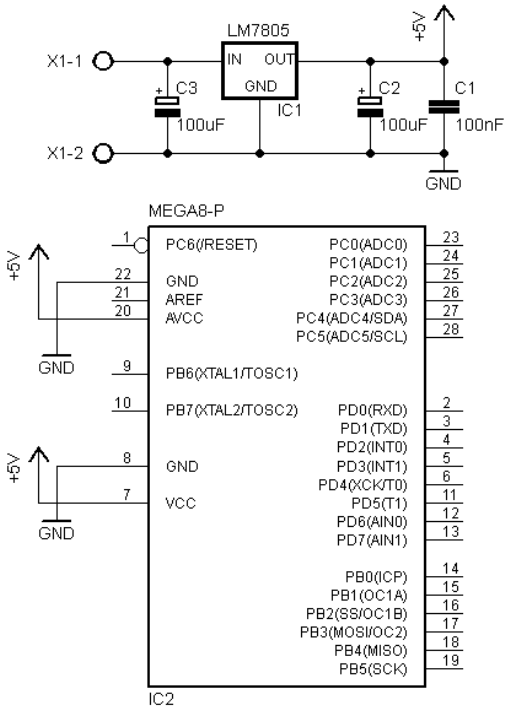
Na grande maioria das aplicações de microcontroladores a tensão de operação é de 5,0V, e para que esta tensão permaneça estável e livre de ruídos é apropriado que se utilize um circuito regulador de tensão protegido por filtros capacitivos.

A figura a seguir apresenta um exemplo de circuito para esta função, onde o regulador de tensão é o LM7805.



A conexão do microcontrolador a fonte

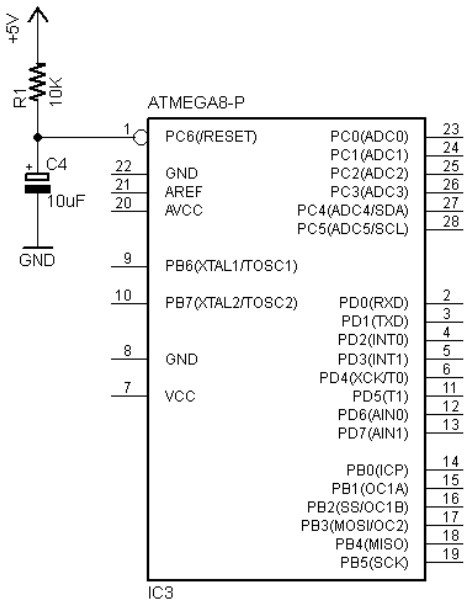
Os microcontroladores da família AVR possuem duas entradas de alimentação distintas, uma para os circuitos digitais e outra para o circuito analógico, do conversor analógico digital. A figura a seguir apresenta a conexão dos pinos 7 e 8 do microcontrolador a fonte, correspondendo a alimentação dos circuitos digitais, e dos pinos 20 e 22 correspondentes a parte analógica, do conversor A/D.



A alimentação do conversor A/D é separada da alimentação do resto do chip para que se possa construir filtros contra ruído específicos para o conversor A/D, isso é necessário quando se deseja grande precisão nas leituras analógicas.

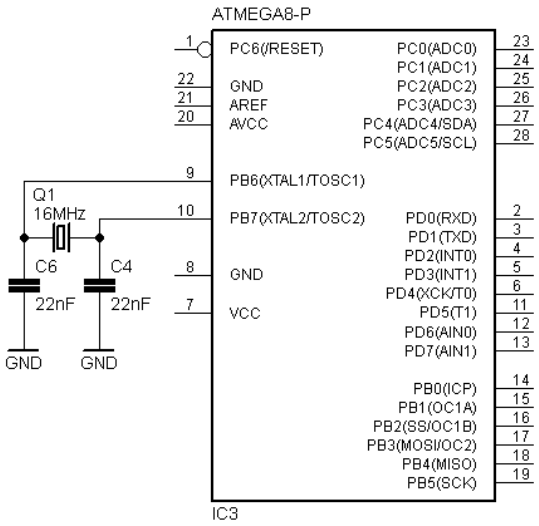
O circuito de “reset”

A maioria dos circuitos que utilizam algum tipo de processador ou microcontrolador necessitam de um sinal de reset, utilizado para reiniciar o sistema. Os microcontroladores da família AVR não são diferentes. O microcontrolador ATMega8 tem como função principal do pino 1 o reset. Normalmente não é necessário adicionar um botão de reset ao nosso projeto, porém o pino de reset deve ser conectado a um resistor e a um capacitor conforme a figura ao lado.



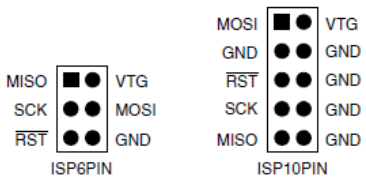
O circuito de “clock”

Todos os circuitos microprocessados, inclusive os microcontroladores necessitam de um circuito de relógio, ou circuito de clock. O microcontrolador ATmega8 possui um circuito interno de clock, que pode funcionar de duas formas. A primeira é através de um oscilador RC interno (padrão de fábrica), o que simplifica o circuito, porém não oferece precisão. A segunda é através de um cristal externo, o que encarece e complica o circuito, porém oferece grande precisão. A figura a seguir mostra como conectar um cristal de 16 Mhz ao microcontrolador.

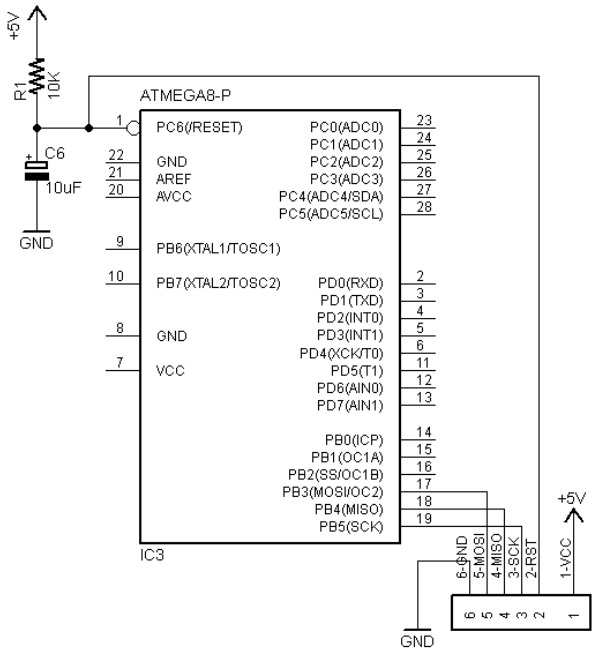


Conector de programação

Os microcontroladores necessitam ser programados para que possam operar corretamente. Isso pode ser feito fora da placa de circuito ou na própria placa onde o microcontrolador vai operar.
Para conectar o programador ao microcontrolador é normalmente utilizado um conector de programação. A figura ao lado mostra dois conectores ISP, sugeridos pelo fabricante e utilizados para esse fim.

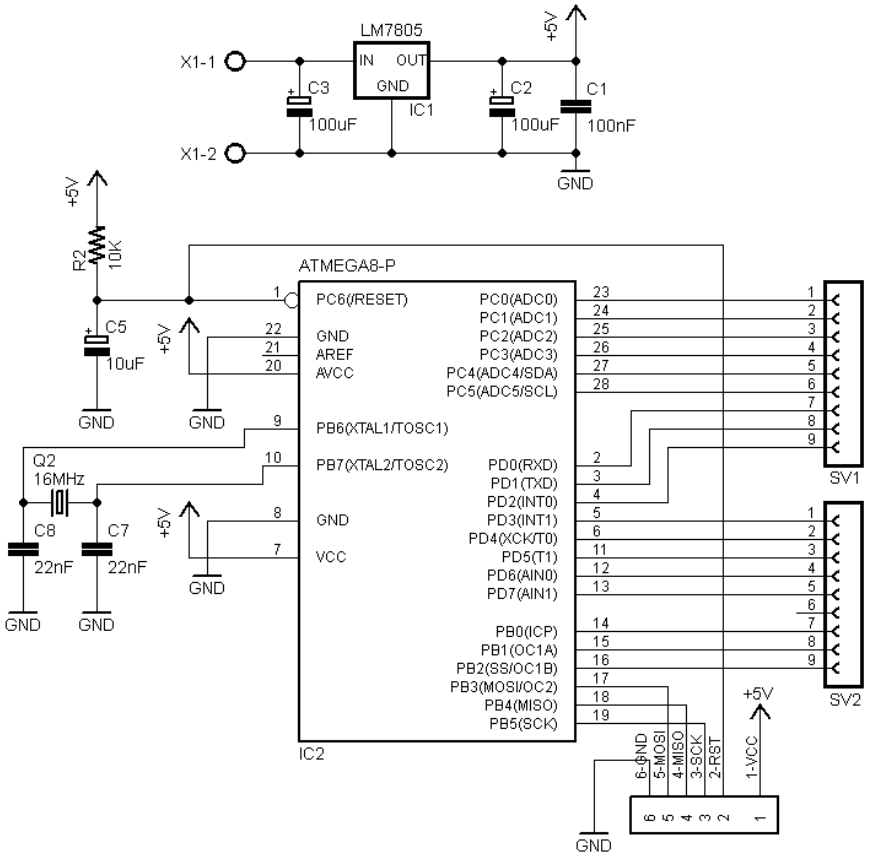


Estes padrões de conector não são apropriados para ser utilizados no protoboard, assim para nossos experimentos usaremos um padrão de conexão alternativo, que é apresentado na figura a seguir.

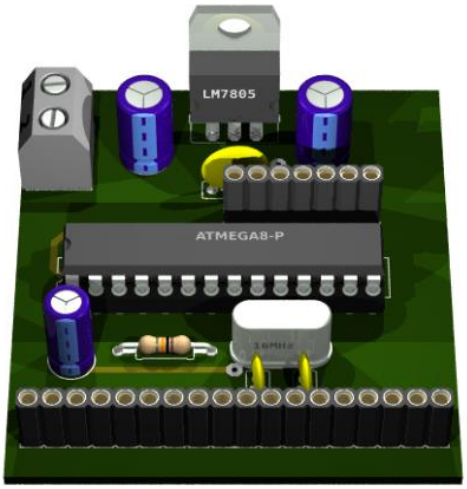


Um circuito completo

Unindo todos os circuitos necessários para o funcionamento de um microcontrolador e adicionando conectores aos pinos restantes temos o circuito a seguir.

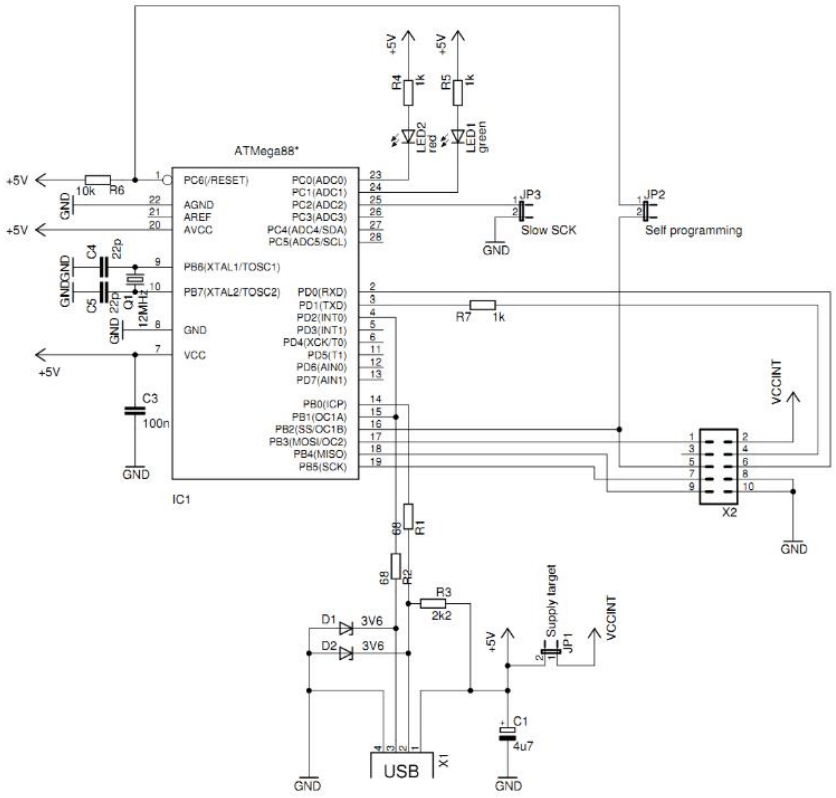


A imagem final de uma placa com este circuito pode ser visualizada na figura a seguir.



O gravador

Para realizar a gravação do programa no microcontrolador, ou seja, transferir o programa do computador para a memória interna do microcontrolador é utilizado um gravador. Existem vários modelos de gravadores, mas em nossas aulas utilizaremos um chamado USBASP, que pode ser facilmente adquirido ou construído. A figuras a seguir mostram um gravador USBASP comercial e o diagrama de circuito deste tipo de gravador.



Este tipo de gravador se conecta ao computador através da USB e ao microcontrolador através do conector de programação.

Experimento 1
Entradas e saídas digitais

Introdução

Este primeiro experimento tem o objetivo de exercitar a programação e a implementação de sistemas microcontrolados que utilizam entradas e saídas digitais simples. Serão realizadas duas montagens, uma com saídas digitais simples e uma com saídas digitais conectadas a relês.

Parte 1

Realizar a montagem de um circuito e sua programação para controlador de ponte rolante. Um modelo de ponte rolante é apresentado na figura a seguir.

Deseja-se construir um circuito microcontrolado para controlar o movimento de uma ponte rolante. Os movimentos são controlados através de dois botões normalmente abertos. Um deles serve para movimentar o carro da ponte para a esquerda e o outro para a direita. Como medida de segurança, nas extremidades da ponte, devem ser instalados sensores de fim de curso, também normalmente abertos.

A metodologia de projeto consiste em, a partir do problema, identificar a lógica de operação, os sinais de entrada e saída, e então construir o circuito e programa apropriado.

Para o caso da ponte rolante, têm-se:

Sinais de entrada:

- E - Botão que determina movimento para a esquerda;
- D - Botão que determina movimento para a direita;
- Se - Sensor de fim de curso da extremidade esquerda;
- Sd - Sensor de fim de curso da extremidade direita.

Sinais de saída:

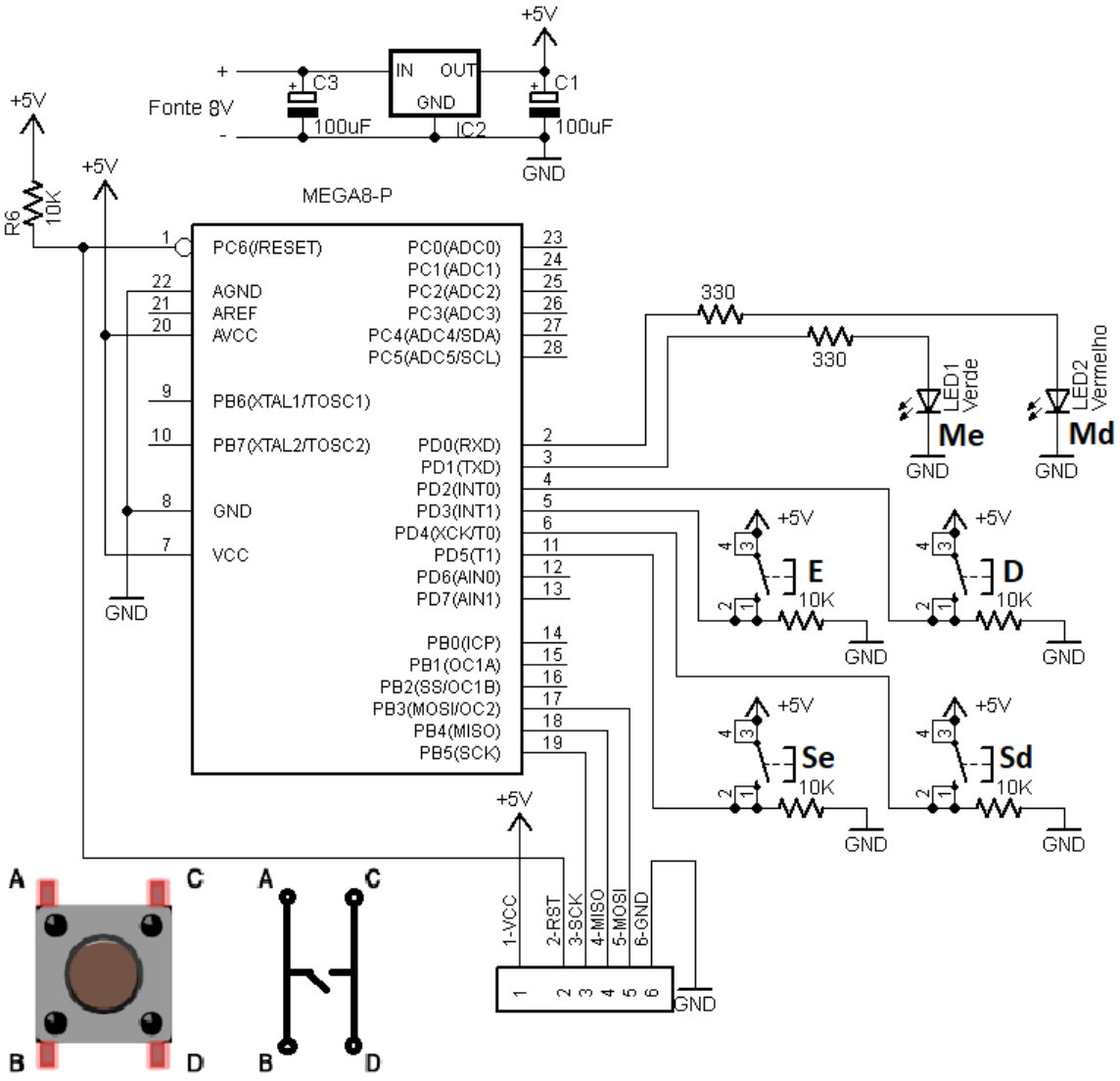
- Me - motor ligado para a esquerda;
- Md - motor ligado para a direita.

Lógica de funcionamento desejada:

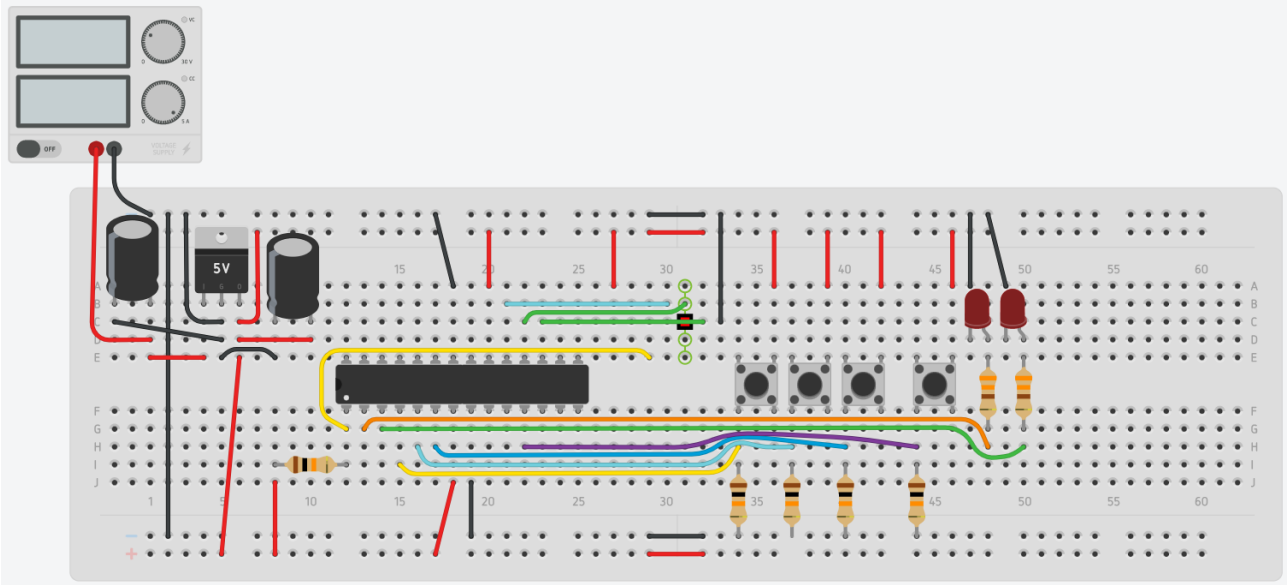
E=1: botão do movimento para a esquerda acionado;
E=0: botão do movimento para a esquerda não acionado;
D=1: botão do movimento para a direita acionado;
D=0: botão do movimento para a direita não acionado;
Se=1: sensor de fim de curso da esquerda acionado;
Se=0: sensor de fim de curso da esquerda não acionado;
Sd=1: sensor de fim de curso da direita acionado;
Sd=0: sensor de fim de curso da direita não acionado;
Me=1: motor ligado, tracionando para a esquerda;
Me=0: motor desligado;
Md=1: motor ligado, tracionando para a direita;
Md=0 motor desligado.

Metodologia

A partir do comportamento desejado para o sistema deve ser implementado um programa para o microcontrolador. Para a verificação do funcionamento do programa deve ser montado um circuito completo com o microcontrolador e os circuitos auxiliares necessários. Para as entradas serão utilizados botões e para as saídas serão utilizados LEDs. A figura a seguir apresenta o circuito proposto.



Um exemplo de montagem deste circuito em um protoboard é apresentado na figura a seguir.

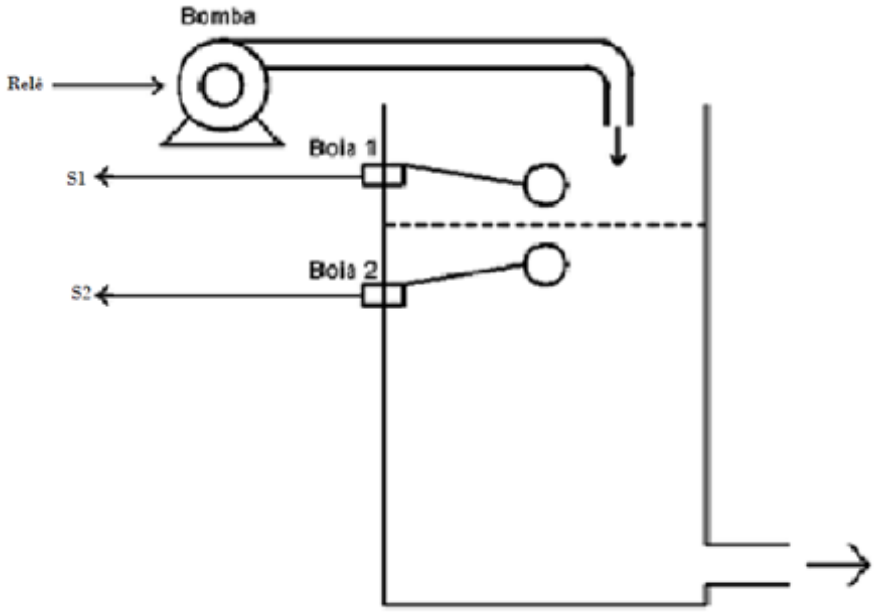


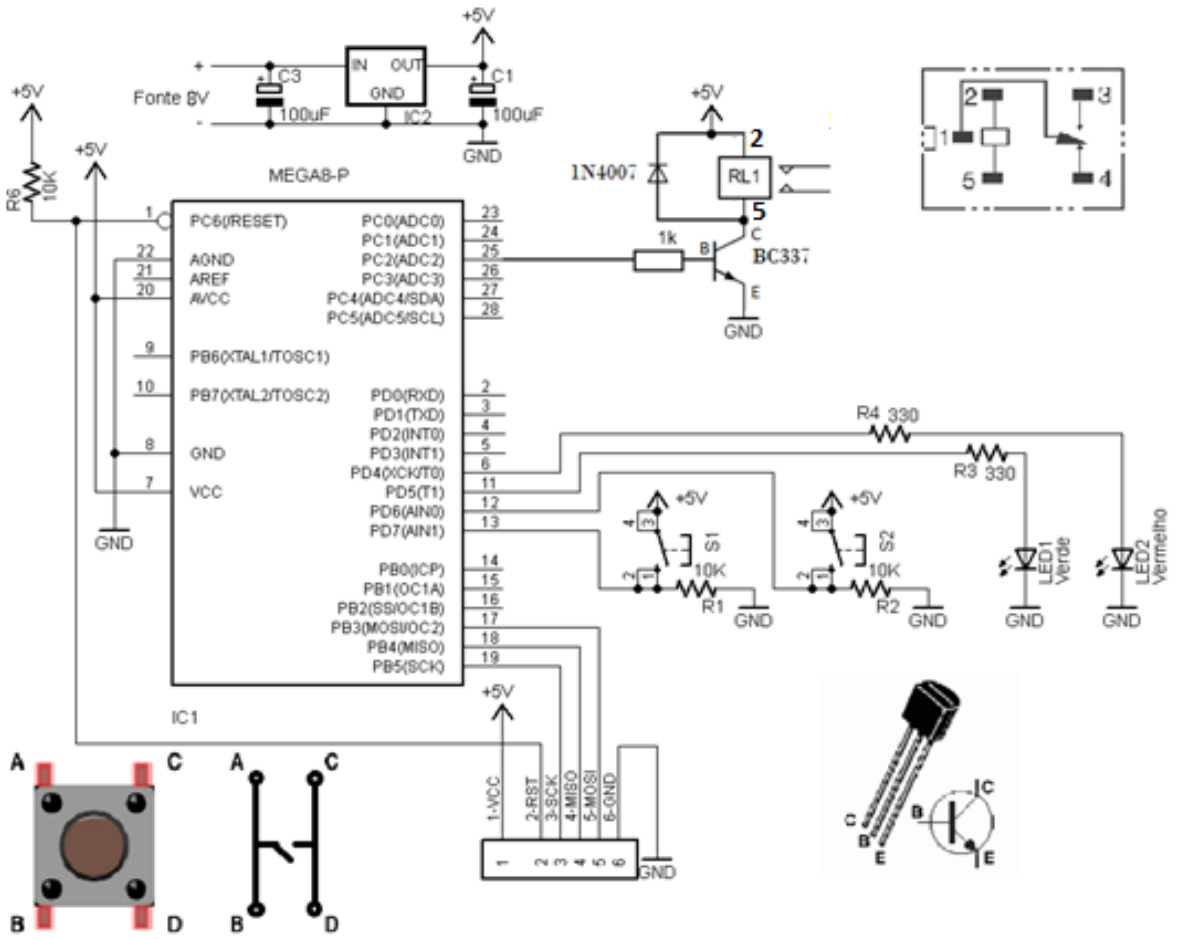
Para facilitar o desenvolvimento do sistema e seus testes o mesmo pode ser montado em um simulador de circuitos. Após a montagem e teste do sistema deve ser elaborado um relatório conforma instruções a seguir.

Parte 2

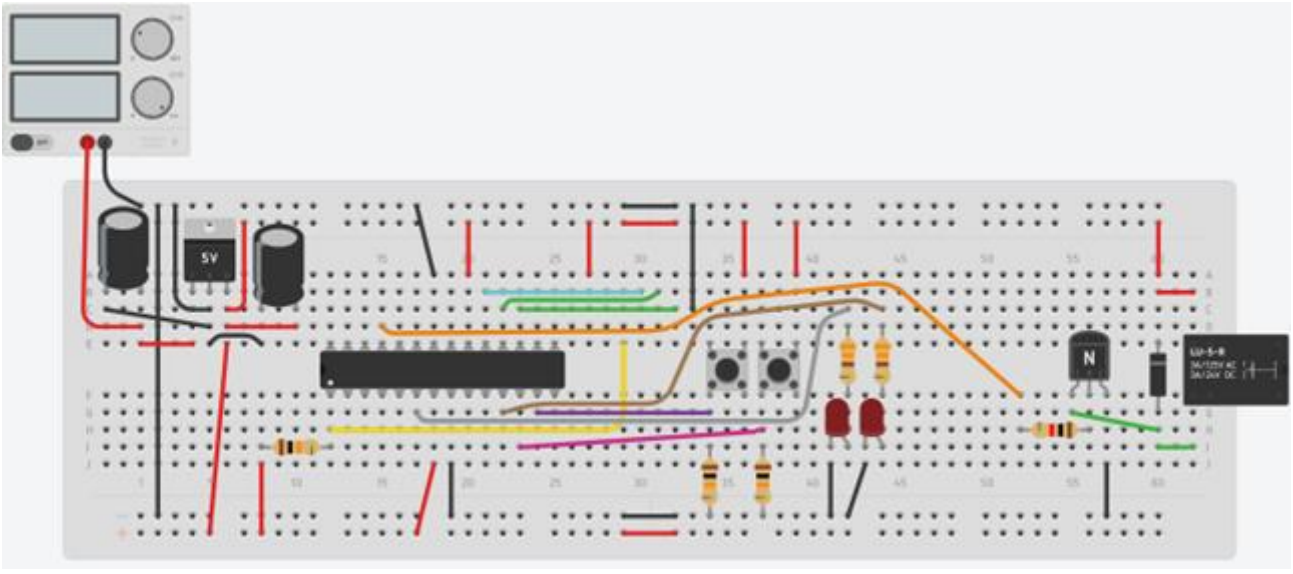
A Figura a seguir apresenta uma planta de controle de nível. A bomba é ligada ao microcontrolador através de um relê, as boias são simuladas pelos botões S1 e S2.

O funcionamento é simples, o microcontrolador monitora as duas entradas das boias, se o nível estiver acima da boia 1 a bomba é desligada e se o nível estiver abaixo da boia 2 a bomba é ligada. Ambas as boias enviam sinal lógico 1 quando o nível estiver acima delas e a bomba liga em nível lógico 1. O LED 1 deve indicar bomba ligada e o LED 2 deve indicar bomba desligada. Monte o circuito e faça um programa para controlar esta planta.





Um exemplo de montagem deste circuito em um protoboard é apresentado na figura a seguir.



Relatório

Após a realização dos experimentos deve ser elaborado um relatório seguindo o modelo disponibilizado. Este relatório deve ser submetido via sistema SIGAA para avaliação, em formato PDF, até a data estipulada em aula.

O modelo do relatório pode ser encontrado em: https://professor.luzerna.ifc.edu.br/ricardo-kerschbaumer/microcontroladores-experimental/

Serão avaliados os seguintes itens no relatório:

- Introdução
- Objetivo
- Fundamentação teórica
- Desenvolvimento
- Componentes utilizados
- Circuito eletrônico
- Código fonte do programa
- Resultados e discussões
- Conclusão