Placa Arduíno MEGA 2560 – IDE – ambiente e linguagem de programação

A IDE (ambiente para desenvolvimento integrado) de desenvolvimento das aplicações para o Arduíno pode ser obtida em https://www.arduino.cc/en/software.

O ambiente da IDE está bem descrito em Arduino - Environment

Uma síntese da linguagem utilizada pode ser obtida em https://www.arduino.cc/reference/en/.

Nesta nota iremos tratar especificamente de tópicos específicos do ambiente e da linguagem da IDE. A nota prossegue com a apresentação do uso da pinagem apresentada na nota 1, aplicando as funções de entrada e saída apresentadas na nota 2.

Ambiente da IDE

A figura apresenta a tela principal da IDE.

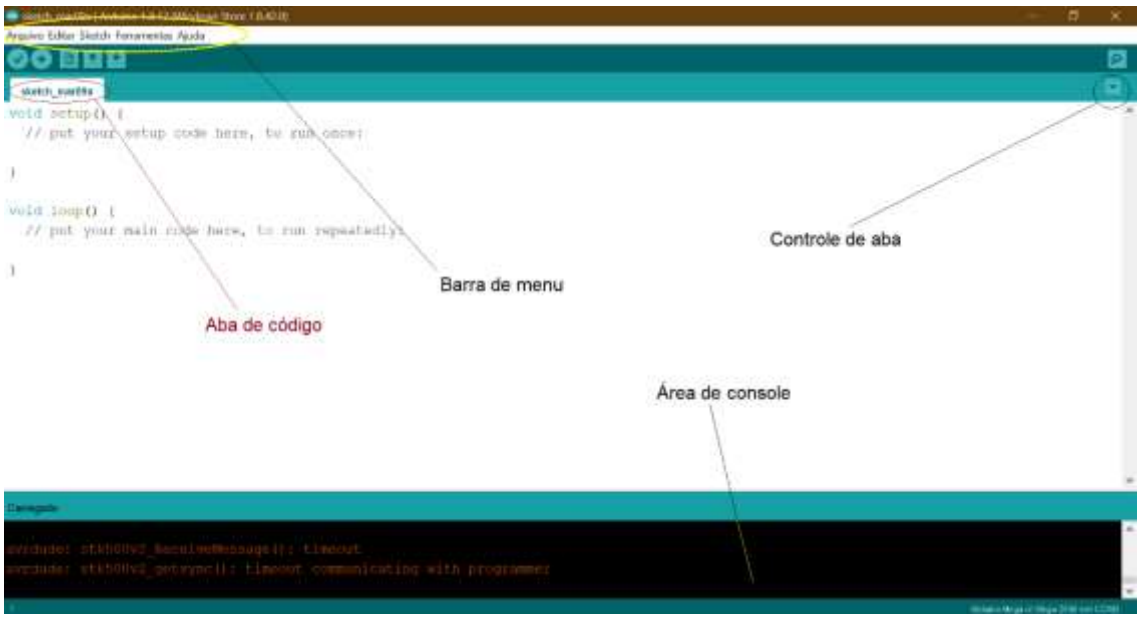

Figura 1 – Tela inicial da IDE do Arduíno

A barra de menu é constituída por: *Arquivo*, *Editar*, *Sketch*, *Ferramentas* e *Ajuda*. Vamos nos concentrar somente em alguns elementos da barra de menu.

Os elementos mais simples não estão discutidos. Não discutimos também os elementos que se referem ao programador. O programador é o código de inicialização da placa. Por enquanto, se desejar aprofundar os detalhes, pode começar consultando Arduino - Bootloader.

No menu Sketch, conforme apresenta a Figura 2, focaremos nos elementos detalhados a seguir.

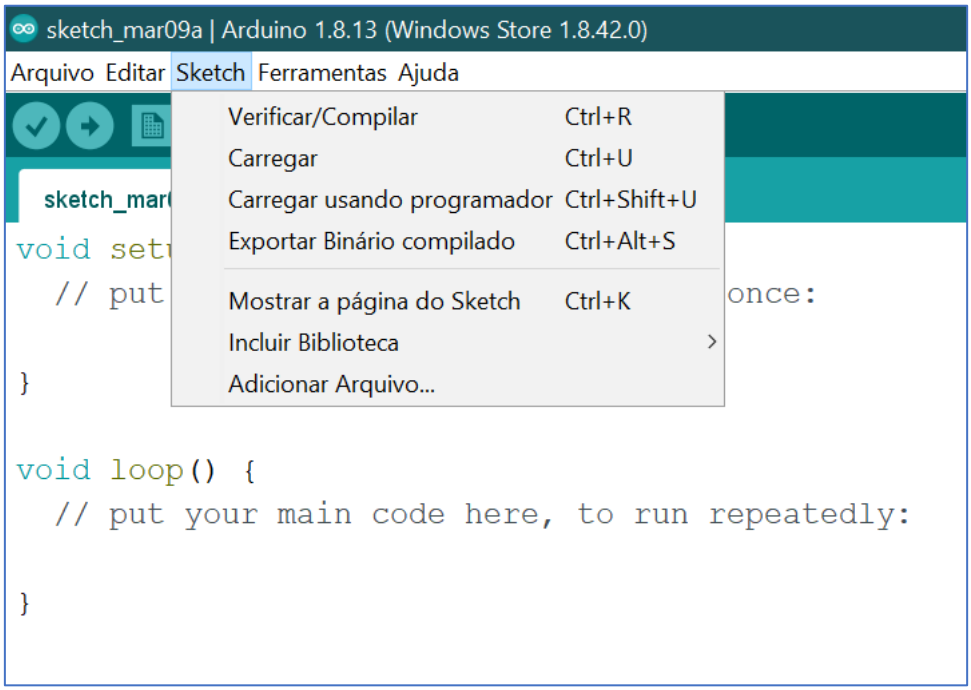

Figura 2 – Menu > Sketch

- *Sketch>Verificar/Compilar* – verifica erros no código, compilando-o; reporta na área do console o uso de memória e as variáveis;
- *Sketch>Carregar* – compila e carrega o arquivo binário na placa;
- *Sketch>Exportar binário compilado* – salva um arquivo .hex, que pode ser mantido ou enviado para a placa usando outras ferramentas;
- *Sketch>Incluir bibliotecas* – adiciona uma biblioteca ao seu sketch pela inserção do comando *#include* do pré-processador da linguagem; adicionalmente, a partir deste item de menu, pode-se acessar o *Gerenciador de Biblioteca* e importar novas bibliotecas de arquivos .zip, conforme ilustra a Figura 3;

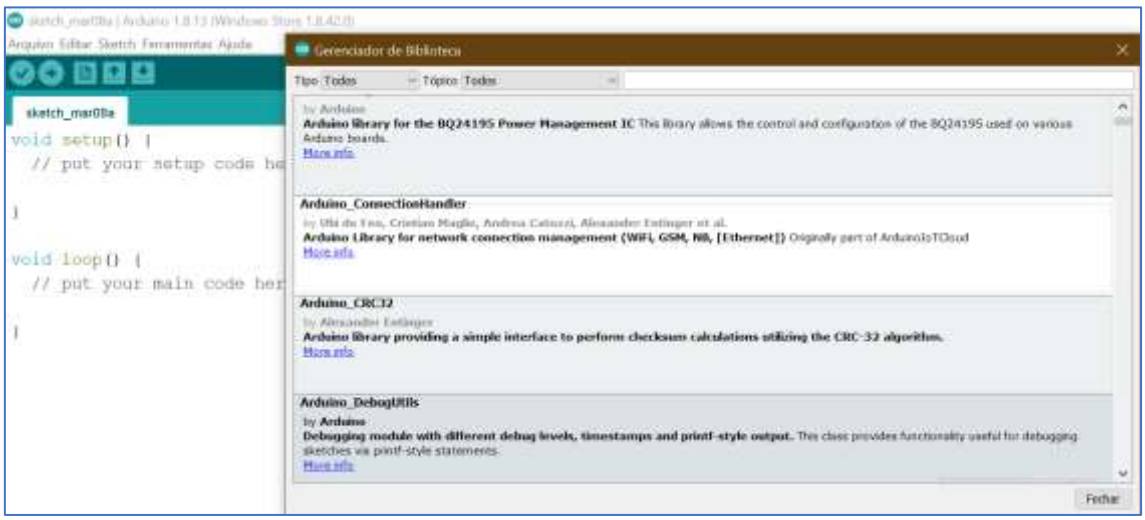

Figura 3 – Formulário de acesso ao Gerenciador de Biblioteca, pelo menu > Sketch > Incluir Biblioteca

- *Sketch>Adicionar Arquivo* – adiciona um arquivo fonte ao *sketch*, que é carregado como uma nova aba e pode ser apagado usando o triângulo invertido na parte superior direita da janela.

No menu Ferramentas, conforme apresenta a Figura 4, focaremos nos elementos detalhados a seguir.

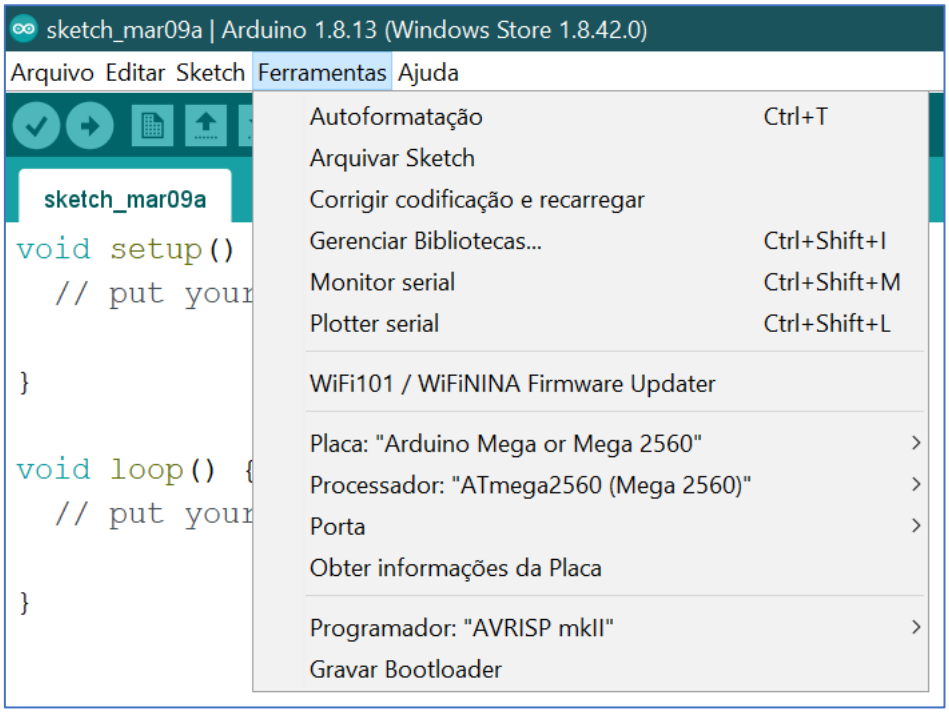

Figura 4 – Menu Ferramentas

- *Ferramentas>Corrigir codificação e recarregar* – corrige possíveis discrepâncias entre o mapa de codificação de caracteres do editor e mapas de caracteres de outros sistemas;
- *Ferramentas>Monitor serial* – abre uma janela de monitor serial e inicia a troca de dados com qualquer placa conectada na porta selecionada; normalmente reinicializa a placa se ela suportar a reinicialização pela porta serial – é necessário que a placa esteja alimentada e conectada;
- *Ferramentas>Plotter serial* – abre uma janela que mostra graficamente informações de comunicação serial das portas analógicas e digitais – é necessário que a placa esteja alimentada e conectada.

Os *sketchs* são disponibilizados em Sketchbook, uma pasta padrão de uso da IDE, que podem ser acessados em *Arquivo>Sketchbook* e pode ser configurada em *Arquivo>Preferências*.

Antes de carregar o *sketch* é necessário habilitar a porta serial correta. No Windows pode ser COM1, COM2 ou COM4, ... Para encontrar a porta adequada consultar a aplicação Gerenciador de Dispositivos do Windows.

Uma vez selecionada a porta serial e a placa carregar o sketch desejado. A placa irá reiniciar automaticamente e fazer o 'upload'. O botão LED de TX irá piscar.

Após realizada a conexão da placa na porta adequada automaticamente o programador é executado (código de inicialização da placa). É possível ver o LED de TX piscando. A IDE apresentará uma mensagem que o sketch foi carregado ou que houve um erro.

Para ilustrar os pontos específicos da linguagem será utilizado um circuito de conexão com um sensor ultrassônico, cuja pinagem está apresentada na Figura 6.

A Tabela 1 sintetiza as funções da pinagem do sensor, apresentando sua função e descrição.

Tabela 1 – Pinagem do LCD 1602

Função	**Descrição**
VCC	Alimentação DC – 5V (típico)
Trig In	10 μseg, pulso +5V
Echo Out	Pulso +5V com largura (duração) proporcional à distância do objeto detectado
GND	Terra

O sensor ultrassônico opera detectando objetos entre distâncias de 2 cm a 4 m. Inclui o transmissor ultrassônico, receptor e circuito de controle. A operação básica está apresentada na Figura 5.

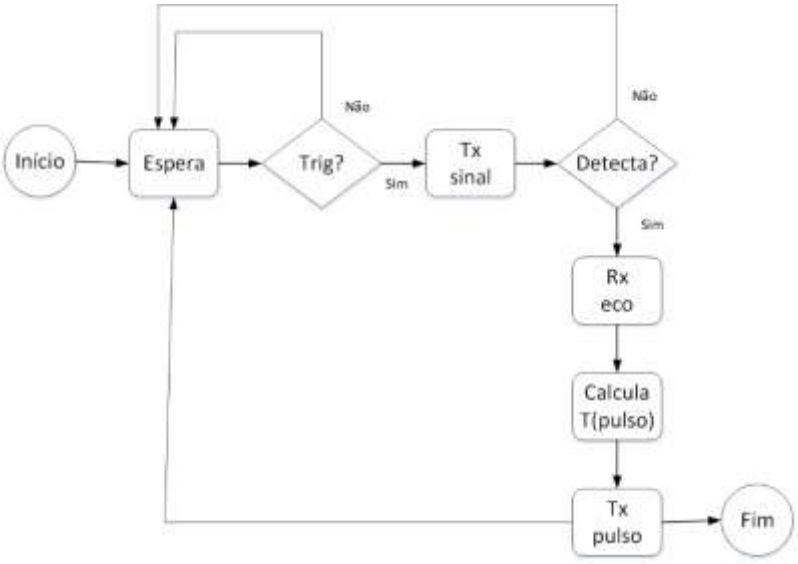

Figura 5 – Ciclo de operação do sensor ultrassônico

O sensor permanece em espera, alimentado, até receber no pino de *Trig* um pulso com duração mínima de 10 μseg. Ao receber o sinal, emite um *burst* de 8 pulsos de 40 kHz. Se não houver um objeto no raio de ação da operação do

sensor, permanece no estado de espera por um novo pulso de *Trig*. Se houver um objeto o sensor receberá um eco do sinal enviado. Com base na distância do objeto detectado, o controlador do sensor calcula a largura do pulso a ser enviado pelo pino Echo e envia um pulso de +5V com a duração calculada.

A Figura 6 apresenta o sensor ultrassônico com a pinagem e a disposição dos caracteres.

Figura 6 – Sensor ultrassônico com apresentação da pinagem

A Figura 7 apresenta o esquema das ligações utilizadas.

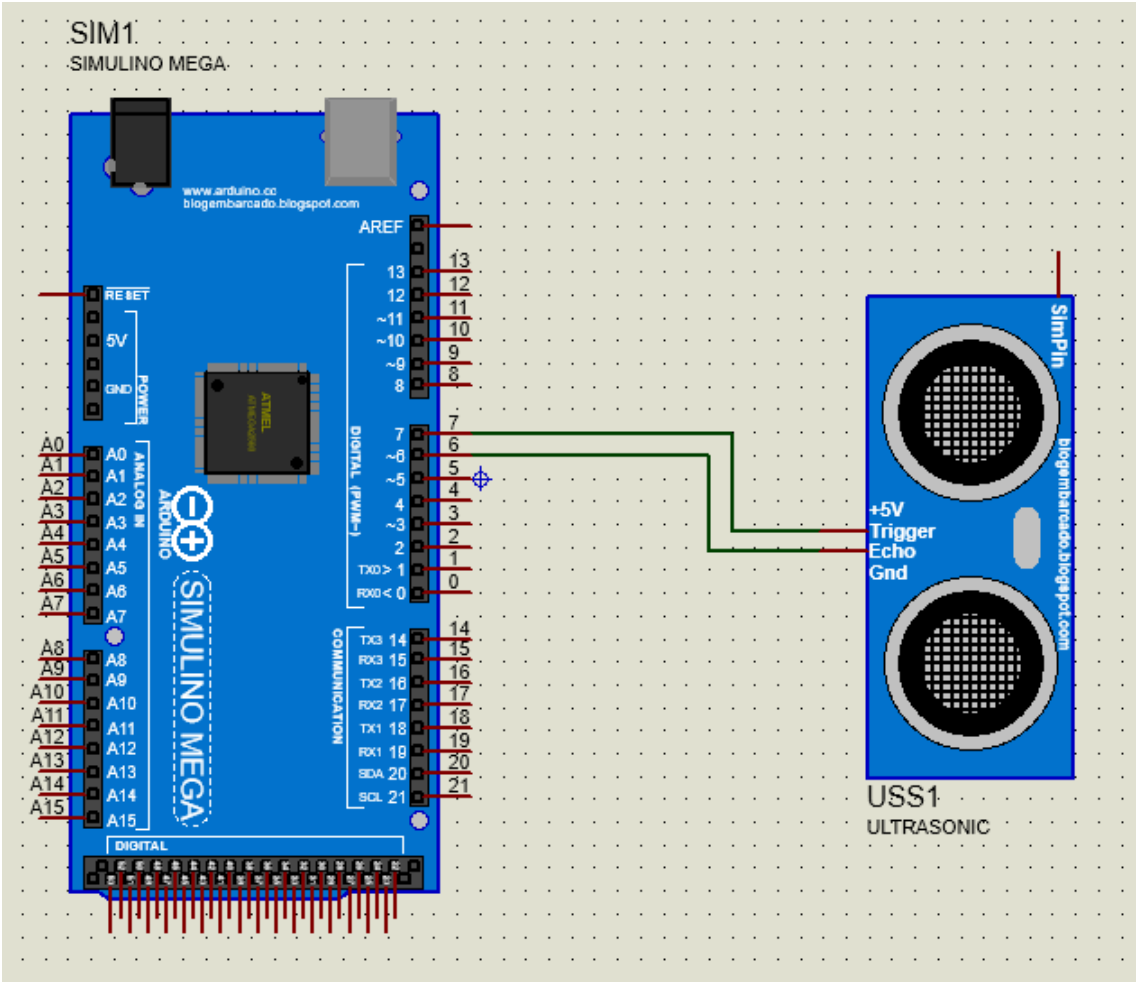

Figura 7 – Esquema elétrico das ligações da placa do Arduíno MEGA2560 com o sensor HC-SR04

O esquema foi elaborado com o software Proteus, versão 8.5. Não estão apresentadas as ligações para simulação do sensor, razão pela qual o pino SimPin não está conectado. O esquema no Proteus também não apresenta a alimentação e o referencial de terra para o sensor. Observe que estão sendo usados os pinos 6 e 7 do Arduíno, respectivamente para conexão com os pinos de *Echo* e *Trig* do sensor.

A Figura 8 apresenta o código utilizado para a montagem do modelo no *protoboard*. O código opera com o espaço de memória SRAM interna do microcontrolador Atmel 2560.

```
//Este código captura e armazena as distâncias detectadas por um sensor ultrassônico

// definição das variáveis globais
int pinEcho=6; // pino do Arduíno ligado ao pino Echo do sensor
int pinTrig=7; // pino do Arduíno ligado ao pino Trig do sensor
float *vetor; // ponteiro que irá armazenar os 1000 primeiros dados das distâncias
int *p_cont;  // ponteiro do contador

const int VEL_SOM=340; // definição da constante em metros por seg

void disparaTrig(){
  digitalWrite(pinTrig,HIGH);
  delayMicroseconds(10);
  digitalWrite(pinTrig,LOW);
}

void setup() {
  pinMode(pinEcho,INPUT);  // define os pinos do Arduíno
  pinMode(pinTrig,OUTPUT);
  vetor=0x200; // inicializa o endereço em 0x200 - endereço inicial da SRAM do MC2560
  p_cont=0x1394; // endereço para o contador a fim de não haver conflito com vetor

  // inicializa o pino de Trig e a porta serial
  digitalWrite(pinTrig,LOW);
  Serial.begin(9600);

}

void loop() {

  if (*p_cont<=1000){
    disparaTrig(); // dispara sinal de Trig
    *vetor=pulseIn(pinEcho,HIGH)*0.000001; // determina o tempo em seg
    *vetor=*vetor*VEL_SOM*100/2; // determina a distancia em cm
    Serial.println("Distancia em centimetros: "); // apresenta o resultado
    Serial.println(*vetor);
  }
  *p_cont++;
  vetor++; //incrementa o ponteiro

  delay(2000); //mantém ciclo de 2 seg de trabalho do sensor

}
```

Figura 8 – Sketch para uso do Arduíno MEGA 2560 com o LCD 1602

O código permite a coleta das informações do pulso de eco do sensor até o limite de 1001 observações.

A Figura 9 apresenta o mapeamento de memória do MC2560.

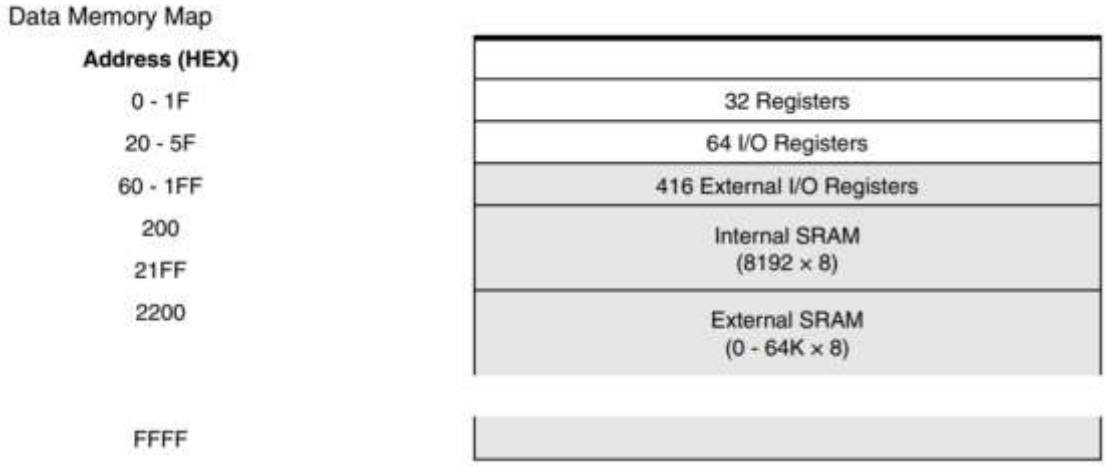

Figura 9 – Mapeamento de endereço MC2560

Algumas observações:

- Foi utilizada uma variável global do tipo ponteiro de inteiros (*p_*cont*), cujo valor inicial está acima do limite a ser usado pelo vetor de medidas – arbitrariamente utilizou-se 0x1394.
- Criou-se uma variável do tipo ponteiro de *float* (**vetor*) para armazenar os valores calculados em posições de memória adjacentes.
- Os dados do vetor são armazenados a partir do endereço 0x200, primeiro endereço da memória interna SRAM do microcontrolador 2560.
- É bom salientar que cada linha de endereço possui 1 byte, logo, como cada tipo *float* ocupa 4 bytes, serão armazenados 4x1001 bytes, ou seja, 4004 bytes.
- A faixa ocupada pelo vetor de medidas compreenderá 0x200 a 0x11A3.