Competências, Habilidades e Bases Tecnológicas da disciplina de Sistemas Embarcados.

III.3 SISTEMAS EMBARCADOS	
Função: Desenvolvimento de aplicações para sistemas embarcados **Classificação: Execução**	
Atribuições e Responsabilidades	
• Desenvolver sistemas embarcados.	
Valores e Atitudes	
• Fortalecer a persistência e o interesse na resolução de situações-problema. • Estimular a organização. • Incentivar a criatividade.	
Competências	**Habilidades**
1. Analisar modelos de sistemas embarcados.	1.1 Identificar as características de sistemas embarcados.
2. Desenvolver aplicações com microcontroladores.	2.1 Programar sistemas para microcontroladores. 2.2 Executar instruções para microcontroladores.
Orientações	
• Detalhamento das Bases Tecnológicas - Anexo I	
Bases Tecnológicas	
Introdução aos microcontroladores Princípios de elétrica e eletrônica Descrição da plataforma de desenvolvimento Escrita de programa para microcontroladores Conceitos de entrada e saída digital Utilização de controle de tempo Entrada e saída analógica Manipulação de memória física e lógica Controle de fluxo de programa Laços de repetição Programação modular Funções predefinidas Sensores, sons, interrupções e comunicação serial	

Carga horária (horas-aula)					
Teórica	00	**Prática em Laboratório***	60	**Total**	**60 Horas-aula**
Teórica (2,5)	00	**Prática em Laboratório* (2,5)**	50	**Total (2,5)**	**50 Horas-aula**

* Possibilidade de divisão de classes em turmas, conforme o item 4.8 do Plano de Curso.

* Todos os componentes curriculares preveem prática, expressa nas habilidades, relacionadas às competências. Para este componente curricular está prevista divisão de classes em turmas.

Para ter acesso às titulações dos Profissionais habilitados a ministrarem aulas neste componente curricular, consultar o site: http://www.cpscetec.com.br/crt/

1-) Comunicação de alunos com alunos e professores:

- Um e-mail para a sala é de grande valia para divulgação de material, notícias e etc.
- Criação de grupo nas redes sociais também é interessante.

2-) Uso de celulares:

Para o bom andamento das aulas, recomendo que utilizem os celulares em vibracall, para não atrapalhar o andamento da aula.

3-) Material das aulas:

A disciplina trabalha com NOTAS DE AULA que são disponibilizadas ao final de cada aula estará disponível no site: **www.marcoscosta.eti.br**, na página principal.

4-) Prazos de trabalhos e atividades:

Toda atividade solicitada terá uma data limite de entrega, de forma alguma tal data será postergada, ou seja, se não for entregue até a data limite a mesma receberá menção I, isso tanto para atividades entregues de forma impressa ou enviadas ao e-mail da disciplina. No caso de PTCC o calendário será seguido, e no dia marcado de determinada atividade só receberão menção os alunos **presentes**, caso não estejam a menção para aquela atividade será I.

5-) Qualidade do material de atividades:

- Impressas ou manuscritas:
  Muita atenção na qualidade do que será entregue, atividades sem grampear, faltando nome e número de componentes, rasgadas, amassadas, com rebarba de folha de caderno e etc. serão desconsiderados por mim.

- Digitais:
  Ao enviarem atividades para o e-mail da disciplina, SEMPRE no assunto deverá ter o nome da atividade que está sendo enviada, e no corpo do e-mail deverá ter o(s) nome(s) do(s) integrante(s) da atividade, sem estar desta forma a atividade será DESCONSIDERADA.

6-) Menções e critérios de avaliação:

Na ETEC os senhores serão avaliados por MENÇÃO, onde temos:
- MB - Muito Bom;
- B - Bom;
- R - Regular;
- I - Insatisfatório.

Cada trimestres teremos as seguintes formas de avaliação:
- 1 avaliação teórica;
- 1 avaliação prática (a partir do 2º trimestre, e as turmas do 2º módulo em diante);
- Seminários;
- Trabalhos teóricos e/ou práticos;
- Assiduidade;
- Outras que se fizerem necessário.

1. INTRODUÇÃO

De acordo com as aulas do 1º módulo, iremos fazer uma revisão de alguns conceitos básicos aprendidos e daremos sequência em novos conteúdos, a ideia desse módulo é trabalharmos a Internet das Coisas – IoT.

2. REVISÃO

De acordo com as aulas do 1º módulo, iremos fazer uma revisão de alguns conceitos abordados no inicio de nossas aulas.

2.1 ESTRUTURAS

São duas funções principais que deve ter todo programa em Arduino.

A função setup() é chamada quando um programa começa a rodar. Use esta função para inicializar as sua variáveis, os modos dos pinos, declarar o uso de livrarias, etc. Esta função será executada apenas uma vez após a placa Arduino ser ligada ou ressetada.

```
setup(){

}
```

Após criar uma função setup() que declara os valores iniciais, a função loop() faz exatamente o que seu nome sugere, entra em looping (executa sempre o mesmo bloco de código), permitindo ao seu programa fazer mudanças e responder. Use esta função para controlar ativamente a placa Arduino.

```
loop(){

}
```

2.2 FUNÇÃO pinMode()

Configura o pino especificado para que se comporte ou como uma entrada (input) ou uma saída (output).

Sintaxe:

```
pinMode(pin, mode)
pinMode(9, OUTPUT); // determina o pino digital 9 como uma saída.
```

2.3 FUNÇÃO digitalWrite()

Escreve um valor HIGH (ligar) ou um LOW (desligar) em um pino digital.

Sintaxe:

```
digitalWrite(pin, valor)
digitalWrite(9,HIGH) // o LED na porta 9 acenderá
```

2.4 FUNÇÃO delay()

É um temporizador, onde o valor passado(em milisegundos) será o tempo em que o Arduino não irá executar atividades até que este tempo passes.

Sintaxe:

```
delay(valor em milisegundos)
delay(1000); // neste caso o Aduino ficará 1 segundo sem executar atividades.
```

2.5 Função digitalRead()

Lê o valor de um pino digital especificado, HIGH ou LOW.

Sintaxe:

```
digitalRead(pin)
buttonState = digitalRead(9); // Leitura do estado de um botão no pino 9.
```

2.6 Função tone()

A função é utilizada para trabalharmos com buzzer e possui a função tone() que tem 2 sintaxes: tone(pino, frequência) e tone(pino, frequência, duração), onde pino referencia qual é o pino que irá gerar a frequência (ligado ao positivo do buzzer), a frequência é definida em hertz e a duração (opcional) é em milissegundos. Caso opte pela sintaxe sem duração é necessário usar a função noTone(pino) para parar a frequência enviada pelo pino definido

Exemplo:

tone(pino, frequencia, duração)

onde a **frequencia** do tom é setada em hertz, e a **duração**, em milisegundos.

2.7 Condicional If else (Se e Senão)

If, que é usado juntamente com um operador de comparação, verifica quando uma condição é satisfeita, como por exemplo um input acima de um determinado valor. O formato para uma verificação **if** é:

```
if (algumaVariavel > 50) {
        // faça alguma coisa
} else{
        // faça outra coisa
}
```

O programa checa se algumaVariavel (colocar acentos em nomes de variáveis não é uma boa ideia) é maior que 50. Se for, o programa realiza uma ação específica. Colocado de outra maneira, se a sentença que está dentro dos parêntesis é verdadeira o código que está dentro das chaves roda; caso contrário o programa salta este bloco de código, e vai para o **else,** que é a negação do IF, realizando outra tarefa.

A sentença que está sendo verificada necessita o uso de pelo menos um dos operadores de comparação:

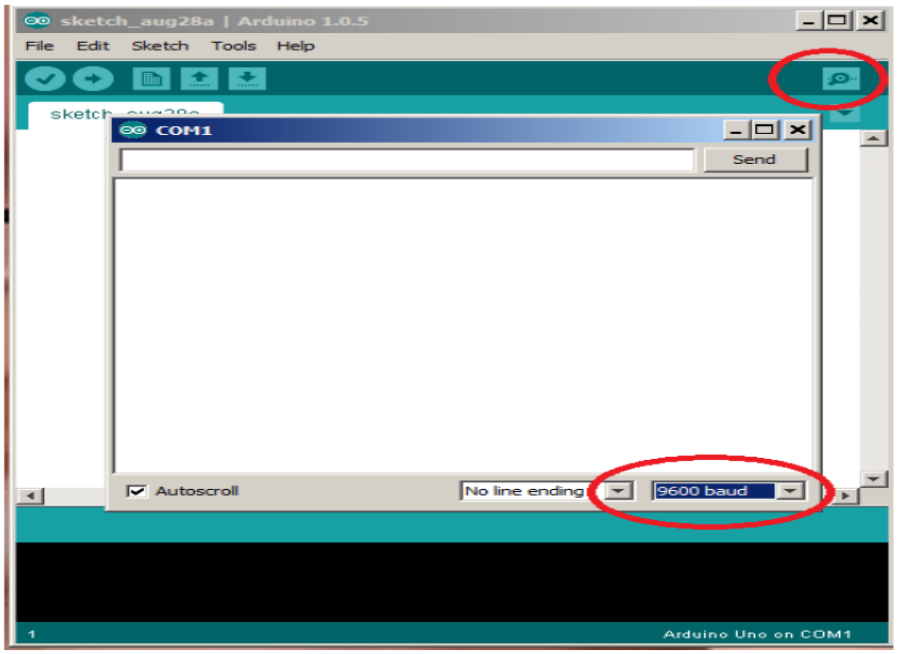
x == y (x é igual a y)
x != y (x é não igual a y)
x < y (x é menor que y)
x > y (x é maior que y)
x <= y (x é menor ou igual a y)
x >= y (x é maior ou igual a y)

2.8 Serial.print

Exibe dados seriais sendo enviados da placa Arduino para o computador. Para enviar dados para a placa, digite o texto e clique no botão "enviar" ou pressione enter.

A comunicação entre a placa Arduino e seu computador pode acontecer em várias velocidades padrão pré-definidas. Para que isso ocorra é importante que seja definida a mesma velocidade tanto na Sketch quanto no Monitor Serial.

Na Sketch esta escolha é feita através da função Serial.begin. E no Monitor Serial através do menu drop down do canto inferior direito.



3. Revisão na prática

3.1. Medindo distância com sensor ultrassônico

Neste projeto, iremos fazer um sensor que mede a distância de um determinado objeto e veremos o resultado na Serial Monitor.

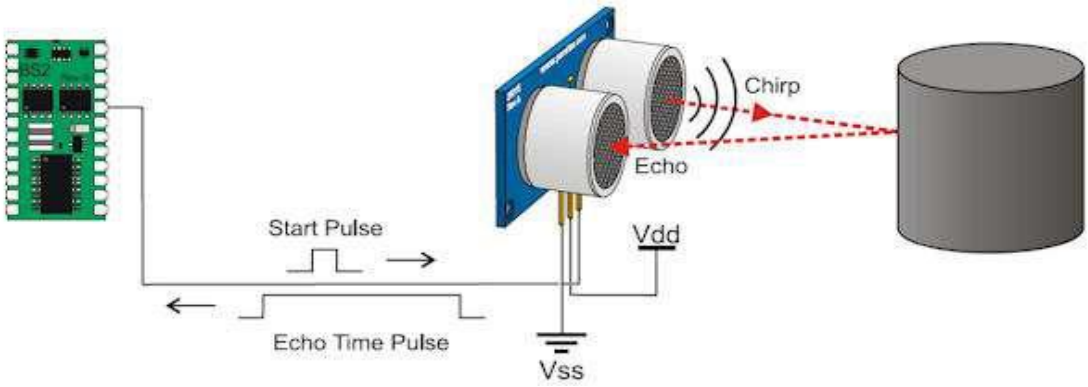
A comunicação seria no computador é vista em uma tela à parte, que pode ser acessada pelo atalho Ctrl+Shift+M (Serial monitor) na IDE do Arduino.

3.1.1 O que vou aprender?

- O que é um sensor ultrassônico;
- Biblioteca Ultrasonic.h

3.1.2 O que é um sensor ultrassônico?

O sensor é usado para medir distâncias. A distância do sensor de ultrassom comum de medição é de cerca de 2 cm a 3-5m. O módulo sensor funciona assim:



O sensor ultrassônico é composto de um emissor e um receptor de ondas sonoras. Podemos compará-los a um alto-falante e um microfone trabalhando em conjunto. Entretanto, ambos trabalham com ondas de altíssima frequência, na faixa dos 40.000 Hz (ou 40KHz). Isto é muito, muito acima do que os nossos ouvidos são capazes de perceber. O ouvido humano consegue, normalmente, perceber ondas na entre 20 e 20.000 Hz e por isto o sinal emitido pelo sensor ultrassônico passa despercebido por nós.

O sinal emitido, ao colidir com qualquer obstáculo, é refletido de volta na direção do sensor. Durante todo o processo, o aparelho está com uma espécie de "cronômetro" de alta precisão funcionando. Assim, podemos saber quanto tempo o sinal levou desde a sua emissão até o seu retorno. Como a velocidade do som no ar é conhecida, é possível, de posse do tempo que o sinal levou para ir até o obstáculo e voltar, calcular a distância entre o sensor e o obstáculo. Para isto vamos considerar a velocidade do som no ar (340 m/s) na seguinte equação:

$$d = (V * t) /2$$

Onde:

d = Distância entre o sensor e o obstáculo (é o que queremos descobrir).

V = Velocidade do som no ar (340 m/s).

t = Tempo necessário para o sinal ir do sensor até o obstáculo e voltar (é o que o nosso módulo sensor ultrassom mede).

A divisão por dois existe pois o tempo medido pelo sensor é na realidade o tempo para ir e voltar, ou seja, duas vezes a distância que queremos descobrir.

O funcionamento do sensor ultrassônico trabalha com dois pinos que são utilizados para alimentar o sensor, um deles é utilizado para disparar o sinal ultrassônico e o outro para medir o tempo que ele leva para retornar ao sensor. Existem alguns sensores ultrassônicos, que possuem apenas três pinos e utilizam um único pino para disparar o pulso e medir o tempo de resposta. Se seu sensor tiver apenas três pinos as conexões e o código do Arduino devem ser devidamente ajustados.

VCC: Alimentação do módulo com +5 V.

Trig: Gatilho para disparar o pulso ultrassônico. Para disparar coloque o pino é HIGH por pelo menos 10us.

Echo: Gera um pulso com a duração do tempo necessário para o eco do pulso ser recebido pelo sensor.

Gnd: Terra.

Vale lembrar que existem limitações para o funcionamento do sensor ultrassônico. Primeiro você tem que levar em consideração que tipo de obstáculo está querendo detectar. Se o obstáculo for muito pequeno pode ser que ele não gere um sinal de retorno suficiente para ser

percebido pelo sensor. Se o obstáculo não estiver posicionado bem a frente do sensor você pode ter medidas imprecisas ou até mesmo não acusar a presença do mesmo. E por fim, a faixa de distância que o sensor trabalha fica entre 2cm e 3-5m e isto pode variar de sensor para sensor.

3.1.3 Biblioteca Ultrasonic.h

Neste exemplo não precisaremos calcular nada, a biblioteca Ultrasonic.h através do método Ranging passando o parâmetro “cm” de centímetros nos retorna o valor correspondente a distância.

Muito provavelmente não teremos esta bibiloteca em nosso computador para utilização, pois ela não existe nativamente instalada quando fazemos o download do Arduino. Na área do 3ºDSN no site www.marcoscosta.eti.br existe um Link para o download desta biblioteca.

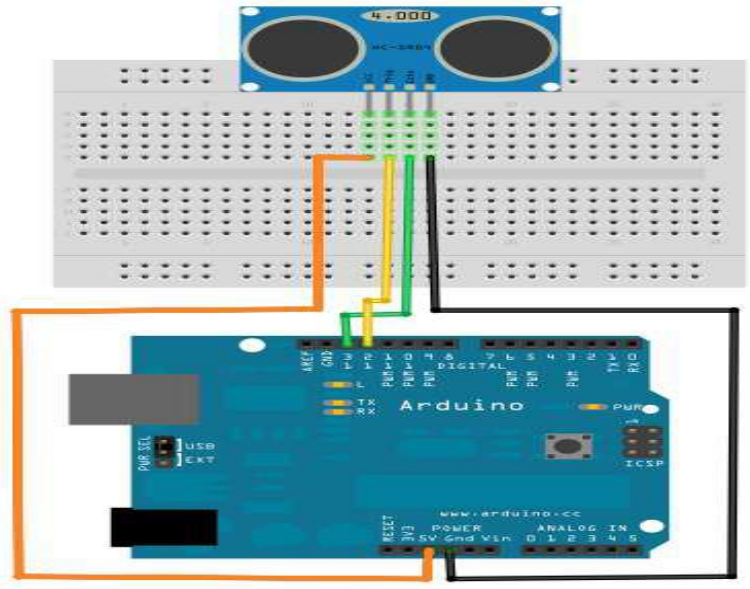
Ao fazer o download no computador em que queremos trabalhar, precisamos importar o arquivo em formato compactado (.ZIP), proceder dentro da IDE do Arduino conforme fizemos no item anterior deste material.

3.1.4 Material necessário

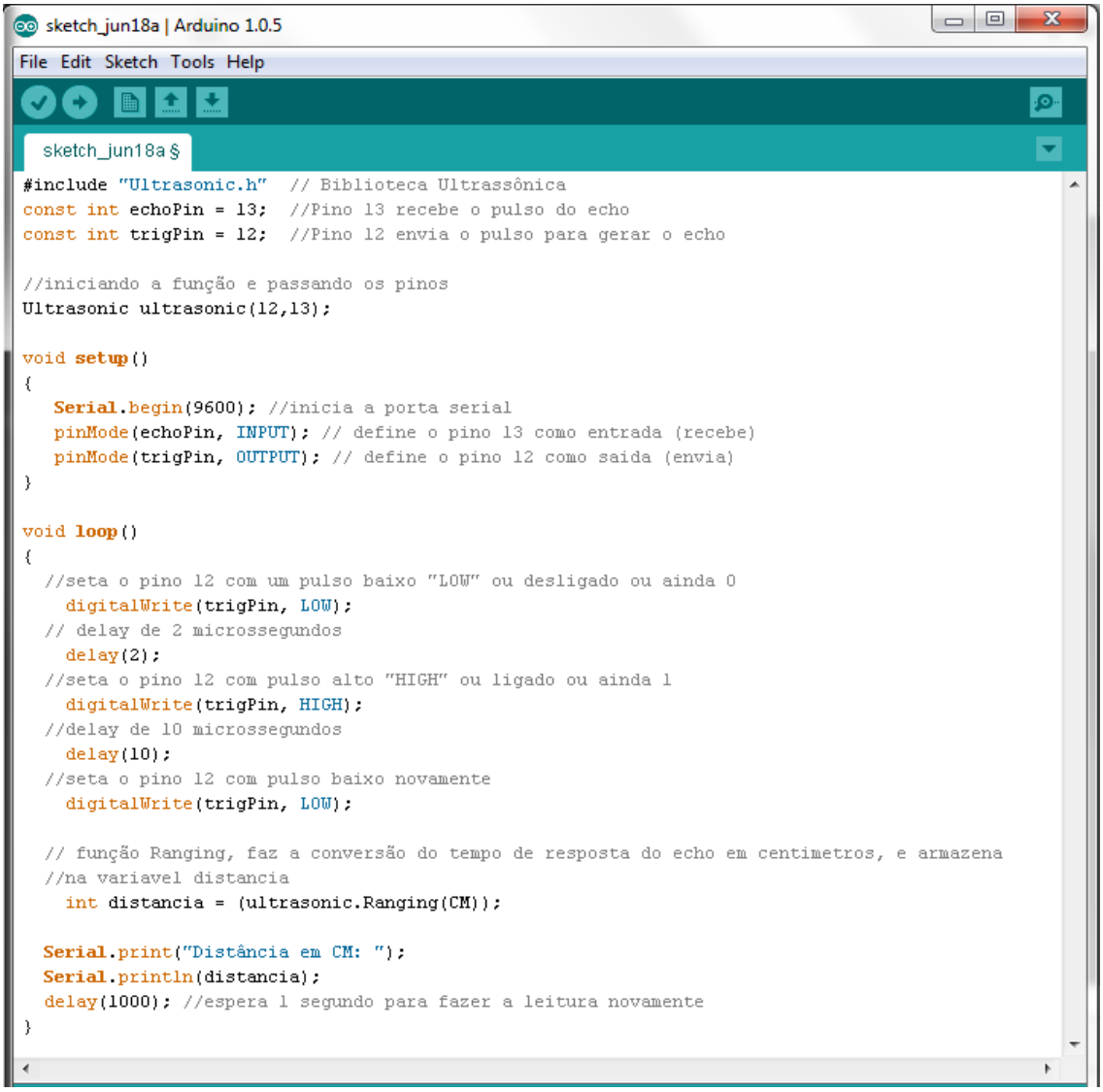
1 Arduino Uno **1 sensor ultrassônico** **1 Cabo USB AB**

Jumpers **1 Protoboard**

3.1.5 Diagrama



3.1.6 Código Fonte




```
#include "Ultrasonic.h"  // Biblioteca Ultrassônica
const int echoPin = 13;  //Pino 13 recebe o pulso do echo
const int trigPin = 12;  //Pino 12 envia o pulso para gerar o echo

//iniciando a função e passando os pinos
Ultrasonic ultrasonic(12,13);

void setup()
{
   Serial.begin(9600); //inicia a porta serial
   pinMode(echoPin, INPUT); // define o pino 13 como entrada (recebe)
   pinMode(trigPin, OUTPUT); // define o pino 12 como saida (envia)
}

void loop()
{
  //seta o pino 12 com um pulso baixo "LOW" ou desligado ou ainda 0
    digitalWrite(trigPin, LOW);
  // delay de 2 microssegundos
    delay(2);
  //seta o pino 12 com pulso alto "HIGH" ou ligado ou ainda 1
    digitalWrite(trigPin, HIGH);
  //delay de 10 microssegundos
    delay(10);
  //seta o pino 12 com pulso baixo novamente
    digitalWrite(trigPin, LOW);

  // função Ranging, faz a conversão do tempo de resposta do echo em centimetros, e armazena
  //na variavel distancia
    int distancia = (ultrasonic.Ranging(CM));

  Serial.print("Distância em CM: ");
  Serial.println(distancia);
  delay(1000); //espera 1 segundo para fazer a leitura novamente
}
```

3.1.7 Exercício

Utilizando como base esta atividade prática, incremente esse projeto, acrescente LED(s) e Buzzer ao mesmo, e simulem um sensor de estacionamento, e pensem na seguinte situação, quanto menor for a distância, emitam sons e irradie(m) luze(s) no(s) LED(s) com mais frequência, utilizem a criatividade de vocês. Essa é para ser realizada em tempo de aula e ao final, me chamem para avaliação, consideraremos como atividade para menção, o prazo para atividade é 16/08/2019.