Instituto Federal de Educação, Ciência e Tecnologia do Ceará

Disciplina: Projeto de Sistemas Embarcados

Professor: Paulo Regis Carneiro de Araújo

Alunos: Carla Teixeira | Haisten Farias | José Araújo | Francikelbe Freire

TRABALHO PRÁTICO 01: ULTRASSOM

FORTALEZA/CE

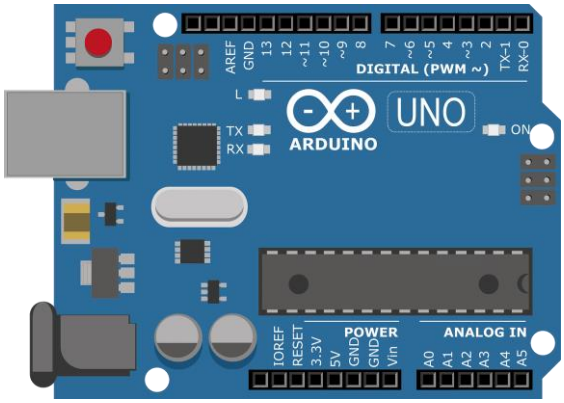
2023

INTRODUÇÃO

O objetivo geral desse trabalho é demonstrar na prática a execução dos conteúdos abordados em sala de aula durante o semestre, sendo um compilado de todos os componentes, prototipagem usando Arduino e os métodos da linguagem de programação em C.

Para as práticas que serão explanadas foram utilizados tais componentes:

● Arduino UNO: Arduino é uma plataforma programável de prototipagem eletrônica (para testes e projetos eletrônicos) de placa única e hardware livre (código aberto), que permite aos usuários criar objetos eletrônicos interativos e independentes. A Placa de microcontrolador contém 14 pinos digitais de entrada / saída, 6 entradas analógicas, conexão através de cabo USB, pode ser alimentada por um adaptador AC/DC. Em relação ao software, o Arduino usa o chamado Arduino Integrated Development Environment (IDE), onde há um editor de texto onde se pode escrever o código, sendo que a mesma está em funções de C e C ++.

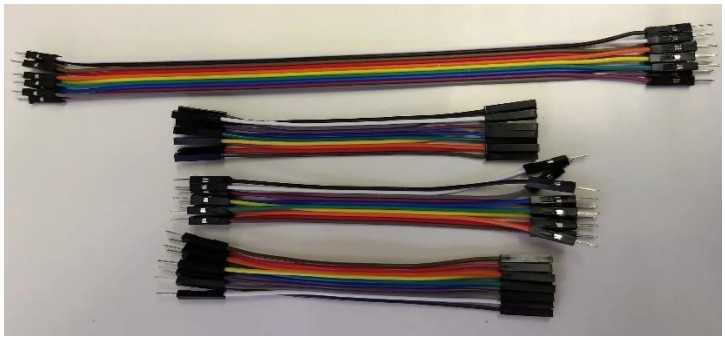


● Cabo USB: O cabo utilizado para transferir o programa do computador (IDE) para a placa do Arduino, assim como servir de alimentação para ele.

● Display LCD 16x2: Serve para facilitar a conexão de uma interface homem-máquina (IHM), facilitando projetos em que o usuário precisa de uma resposta visual, indicando o funcionamento de sua configuração realizada, exibindo caracteres, letras e números deforma nítida e clara. Conta com 16 colunas e 2 linhas, e pode ser de luz de fundo azul ou verde, com cores brancas das letras. Dos 16 pinos, 12 são utilizados para conexão básica. 11 pinos são de entrada/saída (I/O). O pino 1 e 2 servem para conexões de alimentação, os pinos 15 a 16 para o backlight e o pino 3 para o contraste. É ideal para programadores, profissionais da tecnologia, estudantes da área, projetistas ou simplesmente para pessoas que desejam praticar a programação.

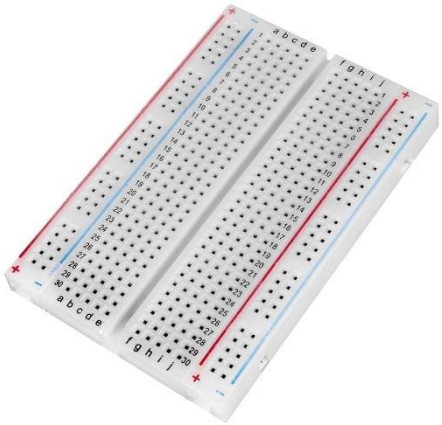
● Sensor de distância ultrassônico HC-SR04: ele é capaz de medir distâncias de 2cm a 4m com ótima precisão e baixo preço. Este módulo possui um circuito pronto com emissor e receptor acoplados e 4 pinos (VCC, Trigger, ECHO, GND) para medição. Para começar a medição é necessário alimentar o módulo e colocar o pino Trigger em nível alto por mais de 10 μs. Assim, o sensor emitirá uma onda sonora que, ao encontrar um obstáculo, será rebatida de volta em direção ao módulo. Durante o tempo de emissão e recebimento do sinal, o pino ECHO ficará em nível alto. Logo, o cálculo da distância pode ser feito de acordo com o tempo em que o pino ECHO permaneceu em nível alto após o pino Trigger ter sido colocado em nível alto. A fórmula: (Distância = [Tempo ECHO em nível alto * Velocidade do Som / 2).

● Jumpers: São fios removíveis utilizados para realizar as conexões entre Arduino, placa de ensaio, display LCD e o sensor HC-SR04, que determinam como o hardware deve ser configurado.



● Resistor: O resistor é um componente elétrico passivo que tem a função primária de limitar o fluxo da corrente elétrica em um circuito. Em nossa prática, foi utilizado um resistor de 1kΩ onde tem função primária de limitar o fluxo de corrente elétrica em um circuito.

● Placa de Ensaio ou Protoboard: é uma placa muito útil pois possibilita conectar diversos componentes, permitindo uma maior precisão na montagem dos circuitos. Em nossa prática, foi utilizada para conectar o display LCD, Arduino, sensor HC-SR04, resistor e jumpers.



METODOLOGIA

Para a realização de nossa prática do Ultrassom, foram utilizados dois softwares:

● Tinkercad: Utilizado para projetar um ambiente em protoboard, conseguindo fazer todas as ligações e conexões necessárias dos componentes, além de inserir o código de programação para testar e simular a efetividade da prática antes de gravar e inserir na placa do Arduino.

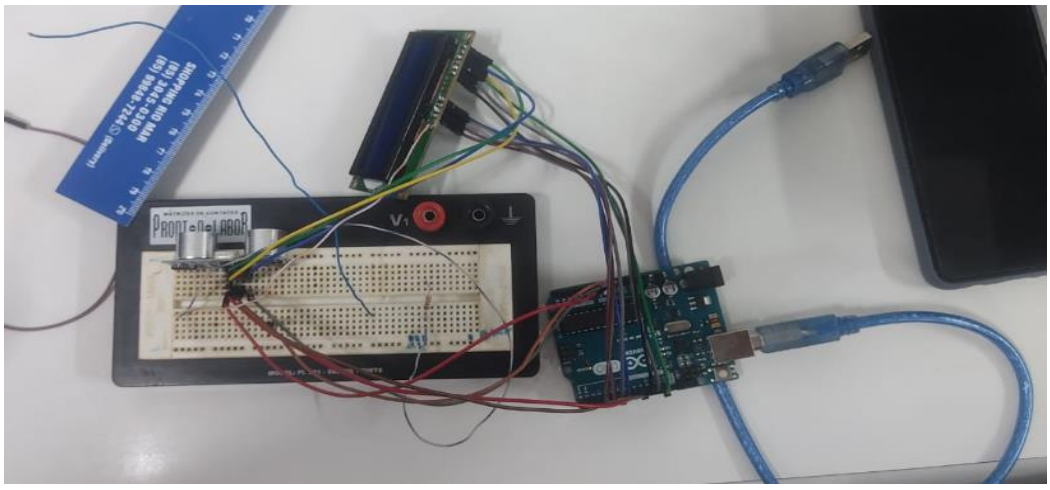
● Arduino IDE: Ambiente físico para aplicação do hardware e inserir o código de programação da aplicação para que a placa Arduíno Uno consiga escrever e obter os resultados esperados.

CÓDIGO FONTE - ULTRASSOM

```
#include <HCSR04.h>
#include <LiquidCrystal.h>

#define Trigger 9
#define Echo    8

UltraSonicDistanceSensor distanceSensor(Trigger, Echo);
LiquidCrystal lcd(12, 11, 10, 7, 6, 5, 4);

int distancia = 0;

void setup()
{
  pinMode(7, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(4, OUTPUT);
  lcd.begin(16,2);
}

void loop() {
  lcd.home();
  distancia = distanceSensor.measureDistanceCm();
  lcd.print (distancia);
  delay(1000);
}
/*
  lcd.setCursor(0,2);
  lcd.print(100);
```

```
delay(2000);
lcd.clear();
lcd.print("A disciplina");
delay(2000);
lcd.clear();
delay(1000);
lcd.print("Sistemas");
lcd.setCursor(0,2);
lcd.print("Embarcados");
delay(2000);
lcd.clear();
delay(1000);
lcd.print("esta sendo");
delay(2000);
lcd.clear();
delay(1000);
lcd.print("realizada");
delay(2000);
lcd.clear();
*/
```

ESQUEMÁTICO



EXPLICAÇÃO DO PROJETO

O projeto foi devidamente testado no Tinkercad (teste do LCD) e a sua devida codificação antes de ser inserido no Arduino. A configuração inicial do LCD e do sensor foi realizada baseada na definição das pinagens indicadas na imagem acima. A partir disso incrementamos o código para o nosso caso de estudo (ultrassom) e ele foi compilado. Para que fosse feita a medição utilizamos o sensor e a leitura que estava sendo feita no LCD, e para medidas de comprovação usamos uma régua para verificar se a métrica da distância em cm estava coerente. Após a leitura e o período definido, o LCD dá um clear e a tela se limpa.

CONCLUSÃO

O projeto do Sensor de Distanciamento Ultrassônico HC-SR04 com Arduíno UNO apresentou resultados que foram equivalentes à proposta na atividade, fazendo a leitura e a escrita no LCD corretamente e com métricas reais medidas pelo sensor. Observamos que por mais que seja uma aplicação simples e um sensor básico ele consegue medir boas distâncias e pode ser usado em diversas aplicações no mercado.

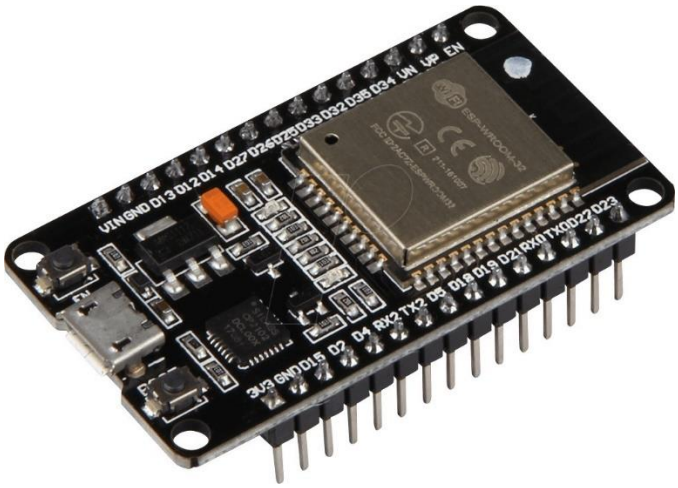
TRABALHO PRÁTICO 02: IoT

INTRODUÇÃO

A ESP32 é uma placa de desenvolvimento programável via Wi-Fi e Bluetooth, que é integrada com a plataforma de programação das placas Arduino, que possui todos os recursos que é preciso para criar projetos, principalmente voltados para projetos de Internet de Todas as Coisas (IoT). Possui 34 pinos GPIO, dos quais 22 são pinos digitais e 12 são pinos analógicos, com hardware aberto que é baseada em um processador dual-core de 32 bits e possui 520 KB de memória flash.

Possui alguns benefícios, como:

- Baixo custo com alto processamento;
- Conectividade sem fio por conta da integração de Wi-fi e Bluetooth;
- Suporte a interfaces de áudio e vídeo
- Possui uma maior flexibilidade quando se trata da aceitação de vários tipos de linguagem de programação;
- Sistema Operacional em tempo real, que permite a execução de várias tarefas simultaneamente e com maior eficiência.



COMPONENTES

● Microcontrolador ESP32: Tipo ESP32 DOIT DEVKIT e ESP32 NODEMCU. O NodeMCU ESP32 é uma placa de desenvolvimento que utiliza o microcontrolador ESP32 e possui uma interface WiFi e Bluetooth integrada. Com estrutura de 30 pinos de entrada/saída, dos quais 18 possuem funções especiais, incluindo suporte a PWM, I2C, SPI, UART, entre outros.

● Cabo USB: O cabo utilizado para transferir o programa do computador (IDE) para a placa do Arduino.

● Roteador Wireless: Roteador TP-LINK utilizado para configurar uma rede wireless local (rede IFCE-COMUTACAO) na qual o microcontrolador ESP32 foi conectado para prover o serviço para os demais equipamentos conectados na rede.

METODOLOGIA

Para a prática de IoT foi utilizado o Arduíno IDE para estudo, modificação e para escrever o código da aplicação e transferir para a placa ESP32 utilizada, e conectada ao computador.

CÓDIGO FONTE

Disponível em: https:\\dontpad.com\pse022023

```
/*
 WiFi Web Server LED Blink

 A simple web server that lets you blink an LED via the web.
 This sketch will print the IP address of your WiFi Shield (once connected)
 to the Serial monitor. From there, you can open that address in a web browser
 to turn on and off the LED on pin 5.

 If the IP address of your shield is yourAddress:
 http://yourAddress/H turns the LED on
 http://yourAddress/L turns it off

 This example is written for a network using WPA encryption. For
 WEP or WPA, change the Wifi.begin() call accordingly.

 Circuit:
 * WiFi shield attached
 * LED attached to pin 5

 created for arduino 25 Nov 2012
 by Tom Igoe

ported for sparkfun esp32
31.01.2017 by Jan Hendrik Berlin
```

```
*/

#include <WiFi.h>

const char* ssid     = "IFCE-COMUTACAO";
const char* password = "comut@wifi";

WiFiServer server(80);

void setup()
{
   Serial.begin(115200);
   pinMode(2, OUTPUT);      // set the LED pin mode

   delay(10);

   // We start by connecting to a WiFi network

   Serial.println();
   Serial.println();
   Serial.print("Connecting to ");
   Serial.println(ssid);

   WiFi.begin(ssid, password);

   while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
   }
```

```
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  server.begin();

}

int value = 0;

void loop(){
  WiFiClient client = server.available();   // listen for incoming clients

  if (client) {                             // if you get a client,
    Serial.println("New Client.");          // print a message out the serial port
    String currentLine = "";                // make a String to hold incoming data from the client
    while (client.connected()) {            // loop while the client's connected
      if (client.available()) {             // if there's bytes to read from the client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        if (c == '\n') {                    // if the byte is a newline character

          // if the current line is blank, you got two newline characters in a row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming, then a blank line:
            // Display the HTML web page
            // client.println("<!DOCTYPE html><html>");
```

```
            // client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
            // CSS to style the on/off buttons
            // Feel free to change the background-color and font-size attributes to fit your preferences
            //client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");
            //client.println("text-decoration: none; font-size: 50px; margin: 2px; cursor: pointer;}");

            // Web Page Heading
            client.println("  <body><h1>ESP32 Web Server</h1>");
            client.println();

            // the content of the HTTP response follows the header:
            client.print("Click <a href=\"/H\">here</a> to turn the LED on pin 2 on.<br>");
            client.print("Click <a href=\"/L\">here</a> to turn the LED on pin 2 off.<br>");

            client.print("</body>");

            // The HTTP response ends with another blank line:
            client.println();
            // break out of the while loop:
            break;
          } else {    // if you got a newline, then clear currentLine:
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything else but a carriage return character,
          currentLine += c;      // add it to the end of the currentLine
        }

        // Check to see if the client request was "GET /H" or "GET /L":
```

```
            if (currentLine.endsWith("GET /H")) {
              digitalWrite(2, HIGH);              // GET /H turns the LED on
            }
            if (currentLine.endsWith("GET /L")) {
              digitalWrite(2, LOW);               // GET /L turns the LED off
            }
          }
        }
        // close the connection:
        client.stop();
        Serial.println("Client Disconnected.");
    }
}
```

ESQUEMÁTICO

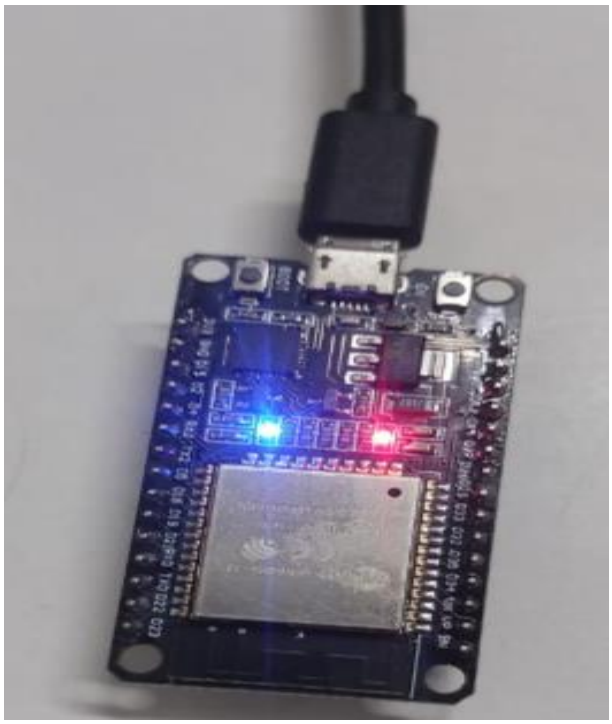
EXPLICAÇÃO DO PROJETO

Foi feito o estudo do código fonte, após isso foi configurado no código as definições de rede do roteador que queríamos usar para teste. Foram definidos: a porta de acesso do cabo USB, a taxa de transferência da serial (115200), e a ESP32 DOIT. Após isso o código foi compilado e começamos a colher as informações, onde pudemos obter o IP da rede (10.103.2.109), IP esse que foi usado para os demais testes da prática.

```
const char* password = "comut@wifi";

WiFiServer server(80);

void setup()
{
    Serial.begin(115200);
    pinMode(2, OUTPUT);      // set the LED pin mode

    delay(10);

    // We start by connecting to a WiFi network

    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
```

```
COM58
22:26:04.675 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv
22:26:04.675 -> mode:DIO, clock div:1
22:26:04.675 -> load:0x3fff0030,len:1184
22:26:04.675 -> load:0x40078000,len:13260
22:26:04.675 -> load:0x40080400,len:3028
22:26:04.675 -> entry 0x400805e4
22:26:04.955 ->
22:26:04.955 ->
22:26:04.955 -> Connecting to IFCE-COMUTACAO
22:26:05.751 -> .........
22:26:10.215 -> WiFi connected.
22:26:10.215 -> IP address:
22:26:10.215 -> 10.103.2.109
Auto-rolagem  Show timestamp  Nova linha  115200 velocidade
```

Após isso, foi feito acesso web para testar a funcionalidade da aplicação em acionar o LED quando acessado um respectivo site usando o projeto.



CONCLUSÃO

Com a realização dessa prática pudemos testar e visualizar na prática a configuração e uso da aplicação usando o ESP32 e sua desenvoltura em aplicações IoT. O microcontrolador de fato possui uma alta performance, o que é um ponto muito bom levando em consideração seu baixo custo de adesão e de energia. A prática foi bem-sucedida e podemos perceber que usando funcionalidade simples e uma simples rede, conseguimos fazer a aplicação funcionar bem em um servidor através do protocolo HTTP. Obtivemos resultado, visto que o LED respondeu bem, acendendo e apagando sua luz quando acionado via web.