Capacitação: Programação C# com ESP32 - 2024

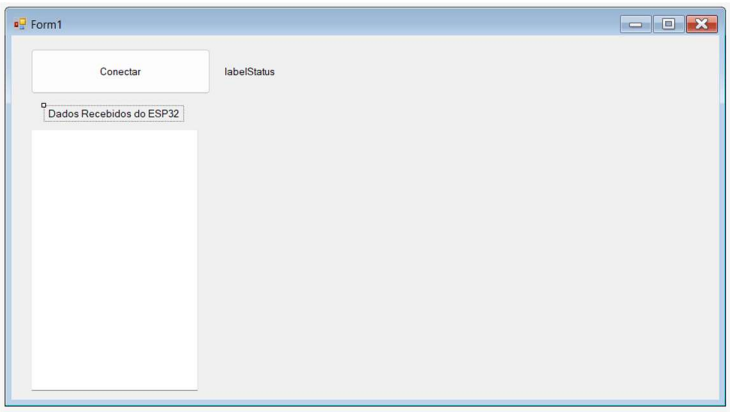
Aplicativo WFA com Banco de Dados

Autor: Prof. Jean Vieira

Passo a Passo:

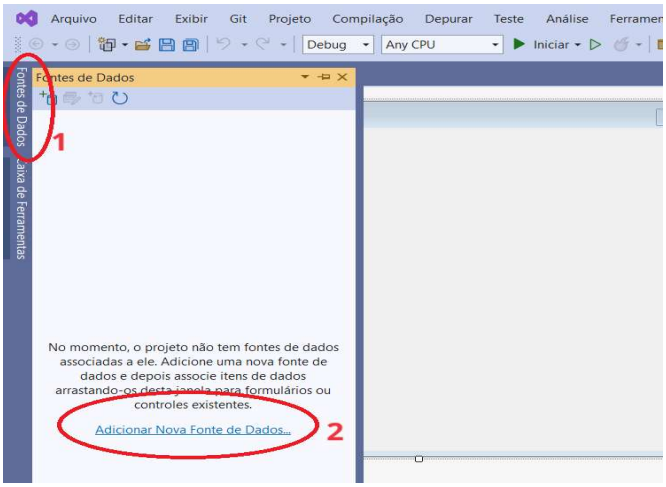
1. **Criar um novo Projeto Windows Forms no Visual Studio:**
    - Abra o Visual Studio e crie um novo projeto de Windows Forms.
2. **Adicionar Componentes ao Formulário:**
    - **Label**: Adicione um Label para exibir o ângulo do servomotor.
        - Altere a propriedade **Name** para **labelAnglulo**.
        - Altere a propriedade **Text** para Ângulo: 0°.
    - **Label**: Adicione um Label para exibir o Status da conexão
        - Altere a propriedade **Name** para **labelStatus**.
        - Altere a propriedade **Text** para **Desconectado**
    - **Button**: Adicione um botão para conectar ao ESP32.
        - Altere a propriedade **Text** para **Conectar**.
    - **TextBox**: Adicione um textBox para exibir as mensagens recebidas do ESP32.
        - Altere a propriedade **MultiLine** para **true**.

Teremos visualmente o formulário do projeto como na imagem a seguir:

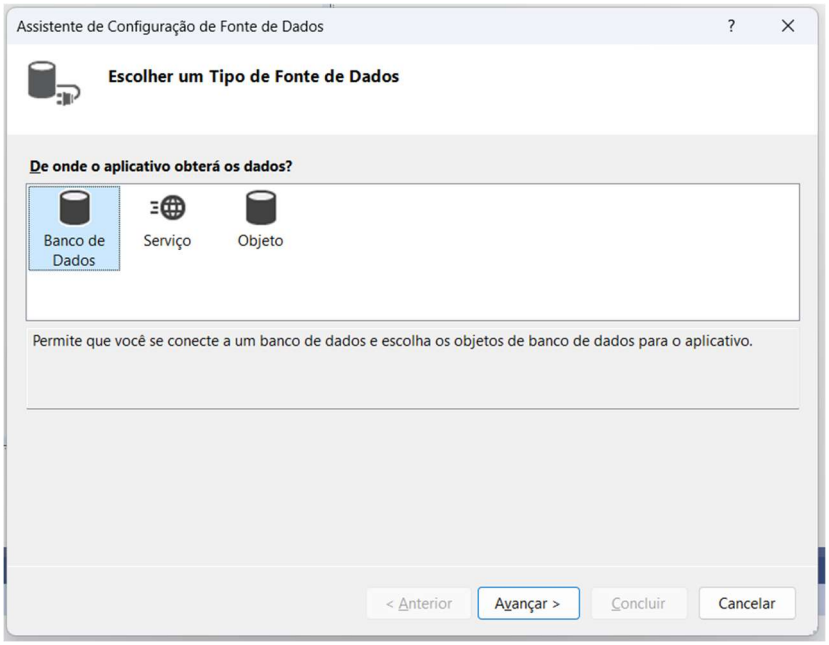


3. **Adicionar a conexão com o Banco de Dados seguindo os passos:**

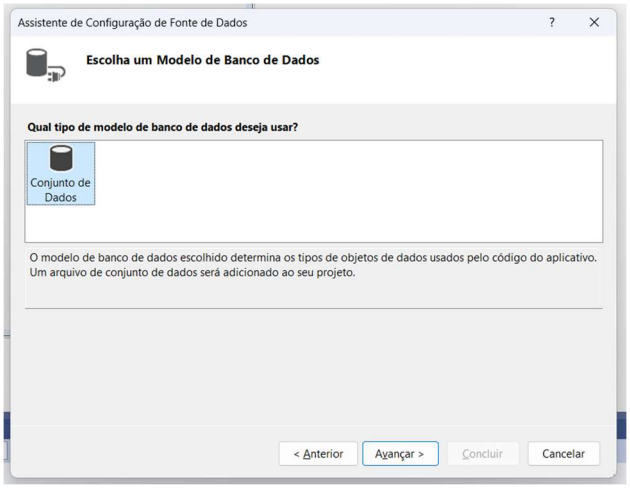
- Clicar na aba: **Fonte de Dados** e em seguida: **Adicionar Nova Fonte de Dados**, como na imagem a seguir:



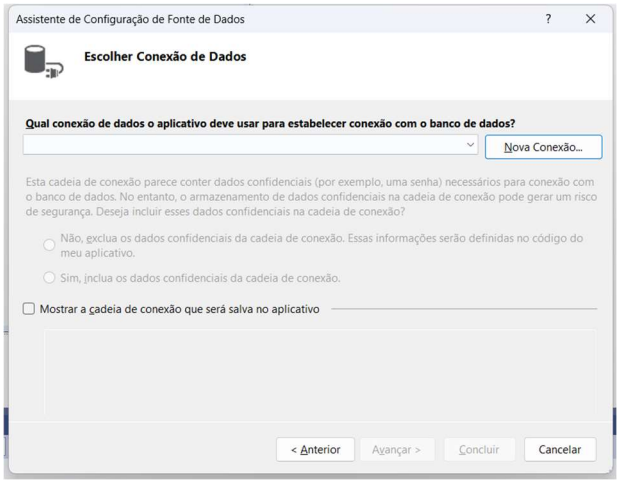
Escolha o Banco de Dados como na imagem:



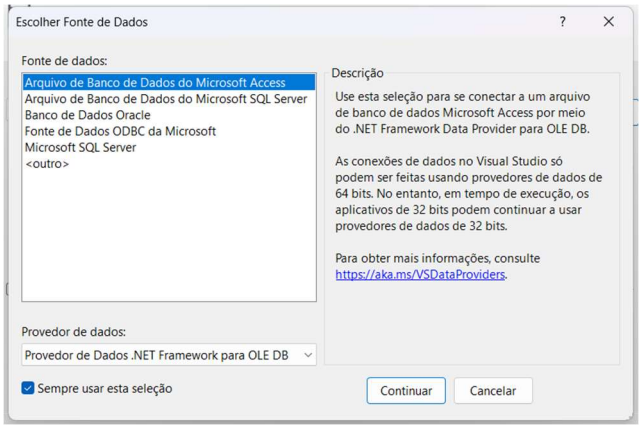
Escolha **Conjunto de Dados**:



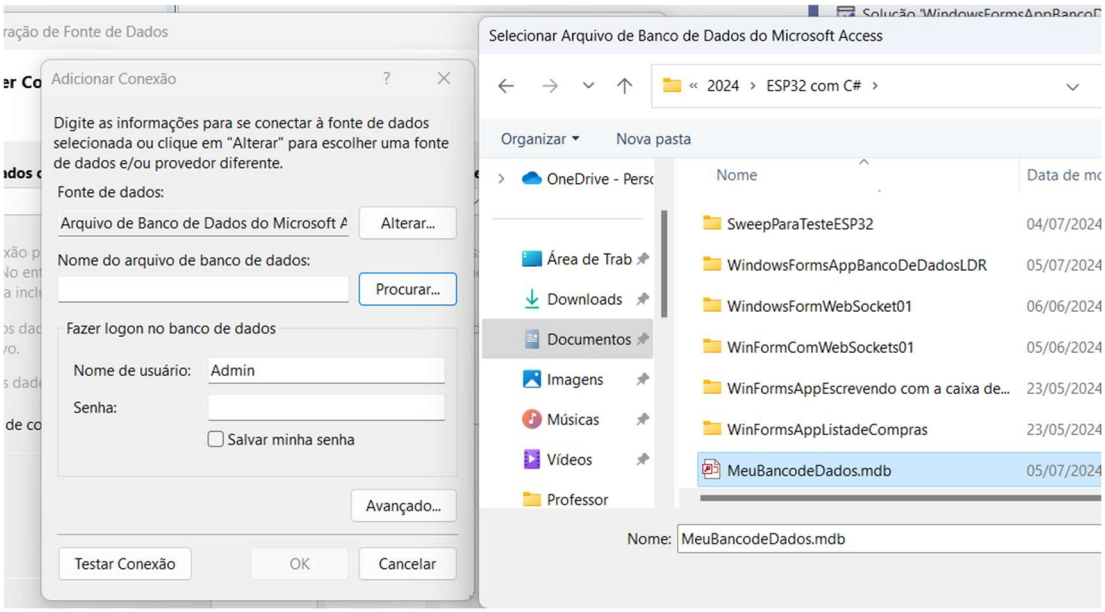
Clicar em **Nova Conexão**:



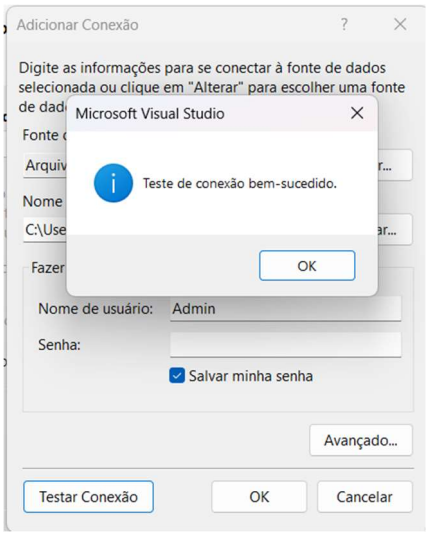
Escolha o **Microsoft Access**:



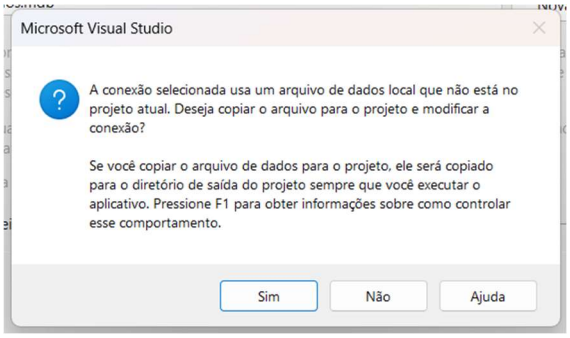
Escolha o Arquivo do Access, clicando em **Procurar**, localize onde foi criado o Banco de Dados, exemplo:



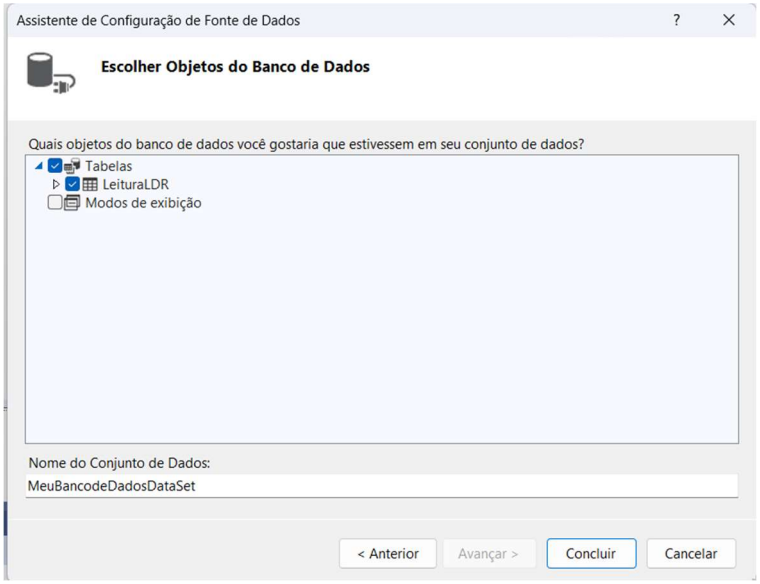
Dê um clique (Check) na opção **Salvar Senha e Testar Conexão**, o resultado deve ser este:



Neste aviso da imagem a seguir, clique em **SIM**, e será feita uma cópia do seu banco de dados, dentro do projeto do Visual Studio, o banco de dados já estará dentro do projeto, o que facilita muito a manipulação do banco de dados.



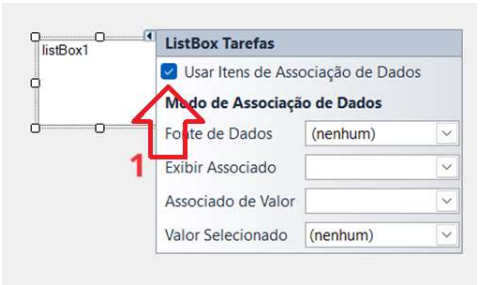
Marque as Tabelas desejadas do seu banco de dados como na imagem:



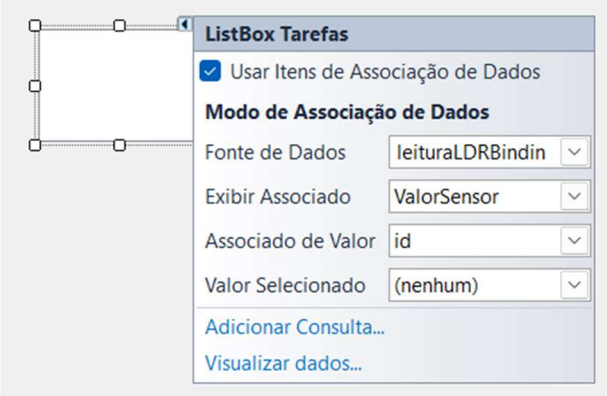
O banco de dados já está adicionado ao seu projeto Windows Forms com simplicidade, vamos a exibição dos dados.

4. Criar a interface de dados:

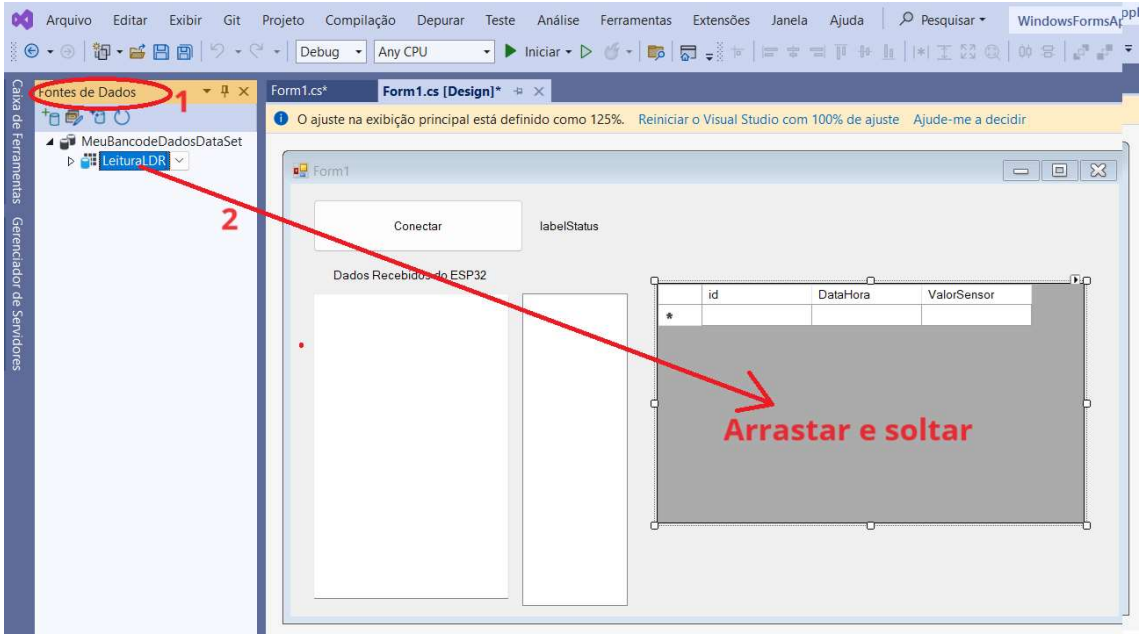
ListBox: Adicione um ListBox ao Formulário e siga os próximos passos:



Escolha a fonte de dados criada no projeto:



Vamos adicionar também um **GridView**, seguindo a imagem a seguir:



5. **Codificando o C# no Projeto:**

Criar a variável para o WebSockets, como o exemplo:

```
public partial class Form1 : Form
 {
        ClientWebSocket webSocket; //adicione esta linha
```

Com duplo clique no botão Conectar (button1) e inserir o código:

```
private async void button1_Click(object sender, EventArgs e)
 {
     webSocket = new ClientWebSocket(); // Inicializa uma nova instância de
ClientWebSocket
     Uri serverUri = new Uri("ws://192.168.33.11:81"); // Define o URI do
servidor WebSocket, substitua pelo IP do seu ESP32

     try
     {
         await webSocket.ConnectAsync(serverUri, CancellationToken.None); //
Tenta conectar ao servidor WebSocket de forma assíncrona
         labelStatus.Invoke((MethodInvoker)(() => labelStatus.Text =
"Conectado")); // Atualiza o lblStatus para mostrar "Conectado" na UI
         await ReceiveMessages(); // Chama o método para receber mensagens do
servidor WebSocket
     }
     catch (Exception ex) // Bloco de captura para exceções
     {
         labelStatus.Invoke((MethodInvoker)(() => labelStatus.Text =
"Desconectado")); // Atualiza o lblStatus para mostrar "Desconectado" na UI
         MessageBox.Show("Erro ao conectar: " + ex.Message); // Exibe uma
mensagem de erro ao usuário
     }
}
```

Adicione agora as funções para receber os dados e gravar os dados no Banco de Dados :

```
        private async Task ReceiveMessages() // Método assíncrono para receber
mensagens do WebSocket
        {
            var buffer = new byte[1024]; // Cria um buffer de bytes para
armazenar os dados recebidos
            while (webSocket.State == WebSocketState.Open) // Loop enquanto a
conexão WebSocket estiver aberta
            {
```

```
                try
                {
                    var result = await webSocket.ReceiveAsync(new
ArraySegment<byte>(buffer), CancellationToken.None); // Recebe uma mensagem do
WebSocket de forma assíncrona
                    var message = Encoding.UTF8.GetString(buffer, 0,
result.Count); // Converte os bytes recebidos em uma string usando UTF-8
                    textBox1.Invoke((MethodInvoker)(() =>
textBox1.AppendText($"Received: {message}\n\r"))); // Atualiza textBoxRecebidas
na UI para mostrar a mensagem recebida
                    int numeroLDR;
                    // Tenta converter a string para um número inteiro
                    bool ehNumeroInteiro = int.TryParse(message.ToString(), out
numeroLDR);
                    //se for número grava no Banco de dados
                    if (ehNumeroInteiro)
                    {
                      Grava_Dados_Recebidos_do_LDR(message.ToString());
                    }

                }
                catch (Exception ex) // Bloco de captura para exceções (erros)
                {
                    labelStatus.Invoke((MethodInvoker)(() => labelStatus.Text =
"Desconectado")); // Atualiza o lblStatus para mostrar "Desconectado" na UI
                    MessageBox.Show("Erro ao receber mensagem: " + ex.Message);
// Exibe uma mensagem de erro ao usuário
                }
            }
        }

        void Grava_Dados_Recebidos_do_LDR(string valorLDR)
        {
            if (valorLDR != "")
            {
                //cria o comando INSERT para adicionar uma nova linha no Banco
de dados
                leituraLDRTableAdapter.Adapter.InsertCommand.CommandText =
"INSERT INTO LeituraLDR (DataHOra, ValorSensor) VALUES ('" + DateTime.Now + "'," 
+ valorLDR + ")";
```

```
                //conectar com o banco

                leituraLDRTableAdapter.Connection.Open();

                //executar o comando INSERT

                int deuCerto =
leituraLDRTableAdapter.Adapter.InsertCommand.ExecuteNonQuery();

                //fechar o conexão

                leituraLDRTableAdapter.Connection.Close();

                //testar se inseriu

                if (deuCerto > 0)

                {

                    labelDadosRecebidos.Text = "Recebeu dados do ESP32, com
valor: " + valorLDR + ", em: " + DateTime.Now;

                    // TODO: esta linha de código carrega dados na tabela
'meuBancodeDadosDataSet.LeituraLDR'. Você pode movê-la ou removê-la conforme
necessário.


this.leituraLDRTableAdapter.Fill(this.meuBancodeDadosDataSet.LeituraLDR);

                }

            }

        }
```

A fim de alimentar este software do Windows Forms junto ao Banco de Dados, devemos realizar a montagem do hardware com o circuito e software (sketch) Arduino para o ESP32.

Neste exemplo, iremos configurar um ESP32 para ler valores de um sensor LDR (*Light Dependent Resistor*) e enviar essas leituras para um aplicativo Windows Forms via WebSockets a cada 20 segundos. Adicionalmente, o ESP32 irá piscar um LED duas vezes rapidamente após cada envio.

Requisitos

- ESP32
- LDR (Sensor de Luminosidade)
- Resistor adequado para o LDR (1K ohms)
- LED
- Resistor adequado para o LED (100 ohms)
- Fios de conexão
- Projeto WFA configurado para receber mensagens via WebSocket

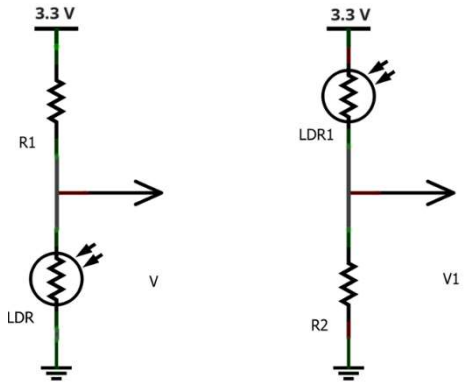
Vamos ligar o LDR a um pino do ESP32, os pinos mais comuns são:

- GPIO 32 (ADC1_4)
- GPIO 33 (ADC1_5)
- GPIO 34 (ADC1_6) - Apenas entrada
- GPIO 35 (ADC1_7) - Apenas entrada
- GPIO 36 (ADC1_0 / VP) - Apenas entrada
- GPIO 39 (ADC1_3 / VN) - Apenas entrada
- GPIO 25 (ADC2_8)
- GPIO 26 (ADC2_9)
- GPIO 27 (ADC2_7)
- GPIO 14 (ADC2_6)
- GPIO 12 (ADC2_5)
- GPIO 13 (ADC2_4), etc.

Sensor LDR

O LDR é um tipo de resistor cujo valor da resistência entre os terminais varia em função da quantidade de luz que chega até ele. É quase igual o potenciômetro, porém ao invés de rodar a haste você aplica menos ou mais luminosidade.

Os circuitos abaixo mostram as possibilidades de interligação de LDR:

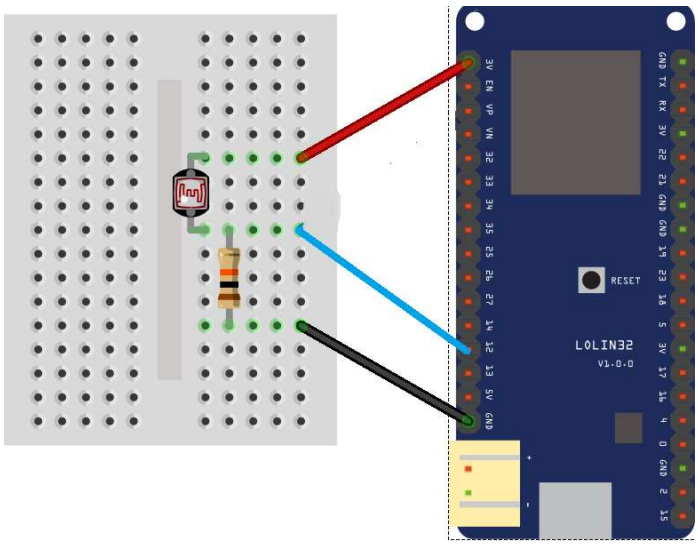


No circuito da esquerda, como a tensão é proporcional ao valor da resistência, então com o aumento da intensidade de luz, menor será a tensão aplicada à entrada analógica, diminui com a presença da luz.

Já no circuito da direita, a tensão que será entregue à porta analógica aumentará com a presença da luz, como a resistência do LDR diminui com a luz, a tensão no resistor R2 que será entregue à porta analógica, aumenta com a presença da luz.

O valor lido pela instrução: **analogRead(número do pino);** retorna um valor analógico convertido para digital de 12 bits, 2^{12}, que é um intervalor entre 0 e 4095. Que equivale de 0 a 3.3V.

Neste exemplo utilizamos o **pino 12 (GPIO 12) do ESP32**, porém este número pode ser modificado.



Exemplo de ligação do LDR ao ESP32 no pino 12

Após a ligação é necessário escrever o software e transferir para o ESP32 utilizando a IDE do Arduino, utilizando o código a seguir:

```
#include <WiFi.h> // Inclui a biblioteca WiFi para conectar o ESP32 à rede WiFi
#include <WebSocketsServer.h> // Inclui a biblioteca WebSocketsServer para criar um servidor WebSocket

const char* ssid = "Seu Wifi"; // Define o SSID (nome) da rede WiFi
const char* senhaWiFi = "Sua senha do Wifi"; // Define a senha da rede WiFi
const int pinoLED = 2; // Define o pino do LED embarcado, geralmente o pino 2
const int pinoLDR = 12; // Define o pino analógico para o LDR (use um pino analógico adequado no seu ESP32)

WebSocketsServer webSocket = WebSocketsServer(81); // Cria uma instância do servidor WebSocket na porta 81
```

```
unsigned long ultimoTempoEnvio = 0; // Armazena o último tempo em que a leitura foi enviada
const unsigned long intervaloEnvio = 20000; // Intervalo de 20 segundos que pode ser alterado

void setup() {
  Serial.begin(115200); // Inicializa a comunicação serial a 115200 bps para debug
  pinMode(pinoLED, OUTPUT); // Configura o pino do LED como saída
  digitalWrite(pinoLED, LOW); // Inicializa o LED como desligado

  WiFi.begin(ssid, senhaWiFi); // Conecta-se à rede WiFi usando o SSID e a senha fornecidos
  while (WiFi.status() != WL_CONNECTED) { // Loop até que a conexão WiFi seja estabelecida
    delay(1000); // Espera 1 segundo
    Serial.println("Conectando no WiFi..."); // Imprime mensagem de conexão ao WiFi
  }
  Serial.println("Conectou no WiFi"); // Imprime mensagem de conexão bem-sucedida ao WiFi

  // Exibe o endereço IP do ESP32
  Serial.print("IP do ESP32 na Rede Conectada: "); // Imprime rótulo para o endereço IP
  Serial.println(WiFi.localIP()); // Imprime o endereço IP local do ESP32

  webSocket.begin(); // Inicia o servidor WebSocket
  webSocket.onEvent(webSocketEvent); // Define a função de callback para eventos do WebSocket
}

void loop() {
  webSocket.loop(); // Mantém o servidor WebSocket em execução, verificando por novas conexões e mensagens

  unsigned long tempoAtual = millis();
  if (tempoAtual - ultimoTempoEnvio >= intervaloEnvio) {
    ultimoTempoEnvio = tempoAtual;
```

```
    //*************************************************************************
*****************************************

    // ----------------------------------     Atenção escolha uma das duas opções
abaixo      ------------------------------------


    //Linha que sorteia o valor, usada para teste sem ligar o LDR a placa do ESP32
    int leituraLDR = random(0, 4095);//sorteia um valor qualquer entre 0 e 4095


    //Linha 49 - descomente esta linha abaixo para realizar a leitura do LDR ligado a
placa do ESP32 e comente a linha acima para remover o sorteio
    //int leituraLDR = analogRead(pinoLDR); // Lê o valor do LDR
    Serial.print("LDR Valor: "); // Imprime o valor do LDR no Serial Monitor
    Serial.println(leituraLDR);
    String leituraString = String(leituraLDR); // Cria uma variável String com a
leitura do LDR
    webSocket.broadcastTXT(leituraString); // Envia o valor do LDR para todos os
clientes conectados via WebSocket
    // Piscada dupla rápida do LED
    piscarLED();
  }
}
void piscarLED() {
  for (int i = 0; i < 2; i++) {
    digitalWrite(pinoLED, HIGH);
    delay(100); // Atraso de 100 milissegundos
    digitalWrite(pinoLED, LOW);
    delay(100); // Atraso de 100 milissegundos
  }
}


// Função de callback para eventos do WebSocket
void webSocketEvent(uint8_t numero_msg, WStype_t type, uint8_t * conteudo_recebido,
size_t tamanho_conteudo) {
  switch(type) { // Verifica o tipo de evento
    case WStype_DISCONNECTED: // Caso a conexão seja desconectada
      Serial.printf("[%u] Desconectou!\n", numero_msg); // Imprime no serial que o
cliente numero_msg foi desconectado
      break;
```

```
    case WStype_CONNECTED: // Caso uma nova conexão seja estabelecida
      Serial.printf("[%u] Conectou!\n", numero_msg); // Imprime no serial que o
cliente numero_msg foi conectado
      webSocket.sendTXT(numero_msg, "Conectou"); // Envia mensagem de confirmação de
conexão ao cliente
      break;
    case WStype_TEXT: // Caso uma mensagem de texto seja recebida
      Serial.printf("[%u] Recebeu pelo Socket o texto: %s\n", numero_msg,
conteudo_recebido); // Imprime no serial a mensagem recebida do cliente numero_msg
      break;
  }
}
```

Atenção: Leia atentamente as linhas 40 a 50 do código anterior.

Testando a Comunicação do ESP32 com o Windows Forms

Inicializando o ESP32:

- Conecte o ESP32 ao seu computador e abra o Monitor Serial na IDE do Arduino.
- Carregue o código no ESP32 e verifique se ele se conecta à rede WiFi e começa a enviar dados do LDR a cada 20 segundos.

Executando o Aplicativo Windows Forms:

- Execute o aplicativo Windows Forms no seu PC.
- Clique no botão para conectar ao WebSocket do ESP32.
- Verifique se os valores do LDR estão sendo recebidos e exibidos no TextBox ou Label.
- Observe o Monitor Serial na IDE do Arduino e seu sincronismo com o Programa C#

Os dados devem ser recebidos no Windows Forms e exibidos na tela do software assim que gravados no Banco de Dados.

<u>Atenção:</u> Comportamento de Teste do Windows Forms Application com Microsoft Access

Ao desenvolver e testar um Windows Forms Application (WFA) que utiliza um banco de dados Microsoft Access, é importante entender como o Visual Studio gerencia os arquivos do banco de dados. Um comportamento comum é que o banco de dados parece perder dados entre as

execuções do aplicativo, mas na verdade não perde, são apenas dados de teste. Isso ocorre porque cada vez que você executa o aplicativo, o Visual Studio copia o arquivo do banco de dados da pasta do projeto para a pasta de execução temporária de saída.

Então toda vez que o WFA é executado ele zera o Banco de dados para um novo teste. Quando o sistema for colocado em produção real, isso não vai mais ocorrer.