Projeto: Varal Retrátil Automatizado com Sensor de Chuva

https://github.com/Brunomaiadesenv/ChuvaZero.git

Curso Técnico em Desenvolvimento de Sistemas
Disciplina: IoT

Aluno: Bruno, Nicolas, Vinicius e Otávio
Professor: Fred Aguiar

Sumário

1. Introdução

Este projeto consiste no desenvolvimento de um sistema automatizado para recolhimento de roupas em um varal retrátil, utilizando Arduino, um sensor de chuva e um motor controlado eletronicamente. O objetivo é proteger as roupas estendidas em caso de chuva, além de permitir ao usuário acionar manualmente o motor via aplicação em C# no computador.

2. Objetivos

- Criar um protótipo funcional de varal automatizado.
- Integrar sensor de chuva ao Arduino para acionamento automático.
- Permitir controle manual do sistema através de um aplicativo em C#.
- Demonstrar a aplicação prática de conceitos de IoT e automação residencial.

3. Materiais Utilizados

• 01 Arduino UNO (ou compatível)
• 01 Protoboard
• 01 Sensor de chuva (FC-37 ou YL-83)
• 01 Módulo Relé 5V
• 01 Motor DC 5–12V (ou Servo Motor)
• 01 Fonte de alimentação externa 9V–12V
• 02 LEDs (vermelho e verde)
• 02 Resistores 220Ω
• Cabos jumper macho-macho e macho-fêmea
• Arduino IDE
• Visual Studio ou Rider (C#)

4. Esquema de Ligações

1. Sensor de chuva → VCC (5V), GND, DO → pino D2
2. LED verde → pino D8 com resistor 220Ω
3. LED vermelho → pino D9 com resistor 220Ω
4. Módulo Relé IN → pino D7
5. Motor alimentado por fonte externa via relé

5. Funcionamento do Sistema

Modo automático: Quando o sensor detecta chuva, o Arduino aciona o relé, o motor recolhe o varal e o LED vermelho acende.
Modo manual: O usuário pode controlar o motor através da aplicação em C#, enviando comandos pela porta serial.

6. Código Arduino

```
/*
  ==  VARAL INTELIGENTE v1.1 (Baseado em Tempo com Monitoramento)  ==
  ATENÇÃO: Versão simplificada SEM sensores de fim de curso.
  O motor funciona por um tempo fixo. É crucial calibrar a variável TEMPO_MOVIMENTO.
*/

// --- PINOS ---
// Pinos de controle do Motor L28N
#define PINO_IN1 5
#define PINO_IN2 6

// Pino do Sensor de Chuva (Analógico)
#define PINO_ANALOGICO_CHUVA A5

// --- PARÂMETROS DE CONTROLE ---
// Limite analógico para considerar chuva (valores menores = chuva)
#define LIMITE_CHUVA 600

// TEMPO (em milissegundos) que o motor ficará ligado para mover o varal.
// !! VOCÊ PRECISA CALIBRAR ESTE VALOR !!
#define TEMPO_MOVIMENTO 1300 // Valor inicial de exemplo: 5 segundos

// --- VARIÁVEIS DE ESTADO ---
// 'true' se o varal estiver recolhido, 'false' se estiver estendido.
bool varalEstaRecolhido = false;
```

```
// --- FUNÇÃO SETUP ---
void setup() {
  Serial.begin(9600);
  Serial.println("Inicializando Varal Inteligente (versao baseada em tempo)...");

  // Configura os pinos do motor como saída
  pinMode(PINO_IN1, OUTPUT);
  pinMode(PINO_IN2, OUTPUT);

  // Garante que o motor comece parado
  digitalWrite(PINO_IN1, LOW);
  digitalWrite(PINO_IN2, LOW);

  Serial.println("Sistema pronto. Assumindo que o varal comeca estendido.");
}

// --- FUNÇÃO LOOP ---
void loop() {
  // Lê o sensor e imprime no monitor serial
  int valorChuva = analogRead(PINO_ANALOGICO_CHUVA);
  Serial.print("Leitura do Sensor de Chuva: ");
  Serial.print(valorChuva);

  bool estaChovendo = (valorChuva < LIMITE_CHUVA);
```

```
// Se está chovendo E o varal não está recolhido...
if (estaChovendo && !varalEstaRecolhido) {
  Serial.println(" -> Chuva detectada! Recolhendo o varal...");

  // Aciona o motor para recolher
  digitalWrite(PINO_IN1, LOW);
  digitalWrite(PINO_IN2, HIGH);
  delay(TEMPO_MOVIMENTO); // Espera o tempo definido
  digitalWrite(PINO_IN2, LOW); // Para o motor

  varalEstaRecolhido = true; // Atualiza o estado
  Serial.println("Movimento de recolhimento finalizado.");
}
// Se NÃO está chovendo E o varal está recolhido...
else if (!estaChovendo && varalEstaRecolhido) {
  Serial.println(" -> Tempo bom! Estendendo o varal...");

  // Aciona o motor para estender
  digitalWrite(PINO_IN1, HIGH);
  digitalWrite(PINO_IN2, LOW);
  delay(TEMPO_MOVIMENTO); // Espera o tempo definido
  digitalWrite(PINO_IN1, LOW); // Para o motor

  varalEstaRecolhido = false; // Atualiza o estado
  Serial.println("Movimento de extensao finalizado.");
} else {
```

```
    // Imprime apenas o status se nada acontecer
    if (varalEstaRecolhido) {
      Serial.println(" -> Status: Varal recolhido, aguardando tempo bom.");
    } else {
      Serial.println(" -> Status: Varal estendido, monitorando chuva.");
    }
  }

  // Espera um pouco antes de verificar novamente
  delay(1000);
}
```

7. Código C#

FRONT

```
<Window x:Class="VaralInteligenteApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:VaralInteligenteApp"
        mc:Ignorable="d"
        Title="Painel de Controle - Varal Inteligente v1.1" Height="500" Width="800"
        FontFamily="Segoe UI" WindowStartupLocation="CenterScreen">
    <Grid Margin="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
```

```
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>

    <Border Grid.Row="0" BorderBrush="LightGray" BorderThickness="1" Padding="10" CornerRadius="5">
        <StackPanel Orientation="Horizontal">
            <Label Content="Porta Serial:" VerticalAlignment="Center"/>
            <ComboBox x:Name="CboSerialPorts" Width="120" Margin="5,0" VerticalAlignment="Center"/>
            <Button x:Name="BtnConnect" Content="Conectar" Width="100" Margin="5,0" Click="BtnConnect_Click"/>
            <Button x:Name="BtnDisconnect" Content="Desconectar" Width="100" Margin="5,0" IsEnabled="False" Click="BtnDisconnect_Click"/>
            <TextBlock Text="Status:" VerticalAlignment="Center" Margin="15,0,5,0" FontWeight="Bold"/>
            <TextBlock x:Name="TxtConnectionStatus" Text="Desconectado" VerticalAlignment="Center" Foreground="Red" FontWeight="Bold"/>
        </StackPanel>
    </Border>

    <Grid Grid.Row="1" Margin="0,10,0,0">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>

        <GroupBox Grid.Column="0" Header="Status do Varal" Margin="0,0,5,0" FontWeight="Bold">
```

```
            <StackPanel Margin="10">
              <TextBlock FontSize="18" Margin="0,5">
                <Run Text="Estado Atual:"/>
                <Run x:Name="TxtVaralStatus" Text="---" FontWeight="Bold"/>
              </TextBlock>
              <TextBlock FontSize="18" Margin="0,5">
                <Run Text="Condição:"/>
                <Run x:Name="TxtClimaStatus" Text="---" FontWeight="Bold"/>
              </TextBlock>
            </StackPanel>
          </GroupBox>

          <GroupBox Grid.Column="1" Header="Leitura do Sensor de Chuva" Margin="5,0,0,0"
FontWeight="Bold">
            <Viewbox Margin="10">
              <Grid>
                <ProgressBar x:Name="SensorProgressBar" Minimum="0" Maximum="1023"
Value="0" Width="200" Height="20" />
                <TextBlock x:Name="TxtSensorValue" Text="0" HorizontalAlignment="Center"
VerticalAlignment="Center" Foreground="White" FontWeight="Bold"/>
              </Grid>
            </Viewbox>
          </GroupBox>

        </Grid>

        <GroupBox Grid.Row="2" Header="Log de Eventos do Arduino" Margin="0,10,0,0"
FontWeight="Bold">
```

```
        <TextBox x:Name="TxtLog" IsReadOnly="True" VerticalScrollBarVisibility="Auto"
TextWrapping="Wrap" FontFamily="Consolas"/>
    </GroupBox>

  </Grid>
</Window>
```

BACK-END

```
using System;
using System.IO.Ports;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;

namespace VaralInteligenteApp
{
  public partial class MainWindow : Window
  {
    private SerialPort _serialPort;
    private const int LIMITE_CHUVA = 600; // O mesmo limite do código Arduino

    public MainWindow()
    {
      InitializeComponent();
      LoadSerialPorts();
      Closed += MainWindow_Closed; // Garante que a porta será fechada ao sair
```

```
    }

    private void LoadSerialPorts()
    {
        string[] ports = SerialPort.GetPortNames();
        CboSerialPorts.ItemsSource = ports;
        if (ports.Length > 0)
        {
            CboSerialPorts.SelectedIndex = 0;
        }
    }

    private void BtnConnect_Click(object sender, RoutedEventArgs e)
    {
        if (CboSerialPorts.SelectedItem == null)
        {
            MessageBox.Show("Nenhuma porta serial selecionada.", "Erro",
MessageBoxButton.OK, MessageBoxImage.Error);
            return;
        }

        string selectedPort = CboSerialPorts.SelectedItem.ToString();
        _serialPort = new SerialPort(selectedPort, 9600); // Baud rate = 9600

        try
        {
```

```
            _serialPort.DataReceived += SerialPort_DataReceived;
            _serialPort.Open();

            // Atualiza a UI
            BtnConnect.IsEnabled = false;
            BtnDisconnect.IsEnabled = true;
            CboSerialPorts.IsEnabled = false;
            TxtConnectionStatus.Text = "Conectado";
            TxtConnectionStatus.Foreground = Brushes.Green;
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Erro ao conectar: {ex.Message}", "Erro de Conexão",
MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    private void SerialPort_DataReceived(object sender, SerialDataReceivedEventArgs e)
    {
        try
        {
            string line = _serialPort.ReadLine();
            Dispatcher.Invoke(() =>
            {
                // Adiciona a linha completa ao log
                TxtLog.AppendText(line + Environment.NewLine);
```

```
            TxtLog.ScrollToEnd();

            // Processa a linha para atualizar a UI
            ProcessArduinoData(line);
        });
    }
    catch (Exception)
    {
        // Ignorar erros que podem ocorrer ao fechar a porta
    }
}

private void ProcessArduinoData(string data)
{
    // Tenta extrair o valor do sensor
    if (data.Contains("Leitura do Sensor de Chuva: "))
    {
        string valueStr = data.Substring(data.IndexOf(':') + 1).Trim();
        valueStr = valueStr.Split(' ')[0]; // Pega apenas o número

        if (int.TryParse(valueStr, out int sensorValue))
        {
            TxtSensorValue.Text = sensorValue.ToString();
            SensorProgressBar.Value = sensorValue;

            // Atualiza o status do clima baseado no valor
```

```
            if (sensorValue < LIMITE_CHUVA)
            {
                TxtClimaStatus.Text = "Chovendo";
                TxtClimaStatus.Foreground = Brushes.DodgerBlue;
            }
            else
            {
                TxtClimaStatus.Text = "Tempo Bom";
                TxtClimaStatus.Foreground = Brushes.Orange;
            }
        }
    }

    // Atualiza o status do varal
    if (data.Contains("Status: Varal recolhido"))
    {
        TxtVaralStatus.Text = "Recolhido";
        TxtVaralStatus.Foreground = Brushes.Crimson;
    }
    else if (data.Contains("Status: Varal estendido"))
    {
        TxtVaralStatus.Text = "Estendido";
        TxtVaralStatus.Foreground = Brushes.ForestGreen;
    }
}
```

```
private void BtnDisconnect_Click(object sender, RoutedEventArgs e)
{
    Disconnect();
}

private void Disconnect()
{
    if (_serialPort != null && _serialPort.IsOpen)
    {
        _serialPort.DataReceived -= SerialPort_DataReceived;
        _serialPort.Close();
        _serialPort.Dispose();
    }

    // Atualiza a UI
    BtnConnect.IsEnabled = true;
    BtnDisconnect.IsEnabled = false;
    CboSerialPorts.IsEnabled = true;
    TxtConnectionStatus.Text = "Desconectado";
    TxtConnectionStatus.Foreground = Brushes.Red;

    // Reseta os valores
    TxtVaralStatus.Text = "---";
    TxtClimaStatus.Text = "---";
    TxtSensorValue.Text = "0";
    SensorProgressBar.Value = 0;
```

```
        }


        private void MainWindow_Closed(object sender, EventArgs e)
        {
            Disconnect(); // Garante que a porta seja fechada
        }
    }
}
```

8. Resultados Esperados

- Varal se recolhe automaticamente quando chove.
- Controle manual via C# funcionando corretamente.
- LEDs indicam estados do sistema.
- Integração hardware + software demonstrada com sucesso.

9. Conclusão

O projeto demonstrou como sensores, atuadores e software podem ser integrados para criar soluções inovadoras em automação residencial. O varal retrátil inteligente é uma solução prática que exemplifica o uso de IoT e integração entre Arduino e C#.