



PDM: Projeto para Dispositivos Móveis
Aula 12: Integrando Android e Arduino utilizando o Ethernet Shield

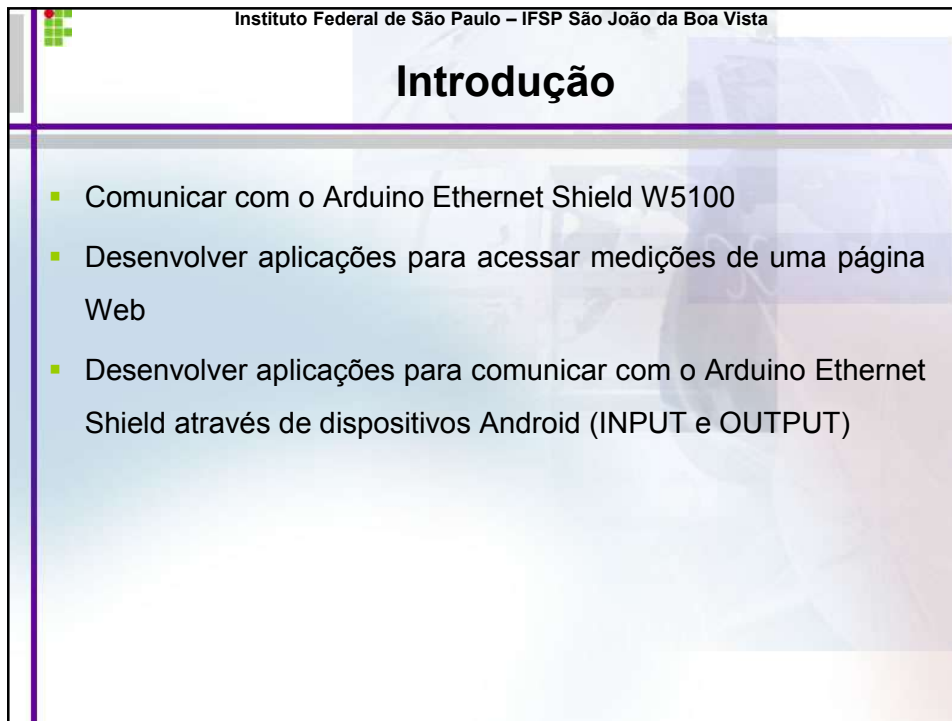
Breno Lisi Romano

<http://sites.google.com/site/blromano>

Instituto Federal de São Paulo – IFSP São João da Boa Vista
Especialização em Desenvolvimento para Dispositivos Móveis



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista



Instituto Federal de São Paulo – IFSP São João da Boa Vista

Introdução

- Comunicar com o Arduino Ethernet Shield W5100
- Desenvolver aplicações para acessar medições de uma página Web
- Desenvolver aplicações para comunicar com o Arduino Ethernet Shield através de dispositivos Android (INPUT e OUTPUT)

Arduino Ethernet Shield W5100 (1)

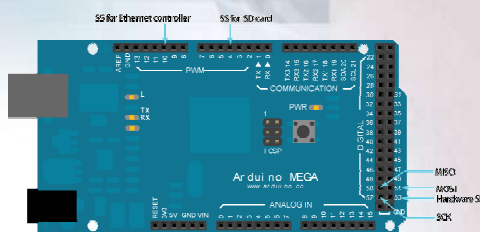
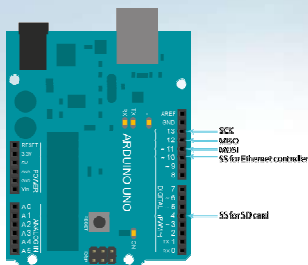
- O Arduino Ethernet Shield W5100 é um dispositivo para controlar sensores ou enviar informações remotamente, possibilitando o acesso às informações na sua rede local ou ainda pode ser conectado à internet e permitir o seu monitoramento de qualquer lugar
- Acoplando o Arduino Ethernet Shield W5100 ao seu Arduino, basta um cabo de rede para monitorar o estado de sensores, chaves e outros dispositivos à partir do browser do seu computador ou celular
- Este Shield fornece um endereço IP compatível com os protocolos TCP e UDP



LED	INDICAÇÃO
TX	Transmissão
RX	Recepção
COLL	Colisão
FULLD	Modo de conexão Full Duplex
100M	Conexão a 100 Mbits
LINK	Conexão estabelecida
PWR	Módulo Ligado

Arduino Ethernet Shield W5100 (2)

- Ele possui um slot Micro-SD que pode ser utilizado para armazenar arquivos para enviar pela rede. Deve-se utilizar biblioteca SD.h
- O Arduino se comunica com o W5100 e com o SD card utilizando o barramento SPI (Serial)

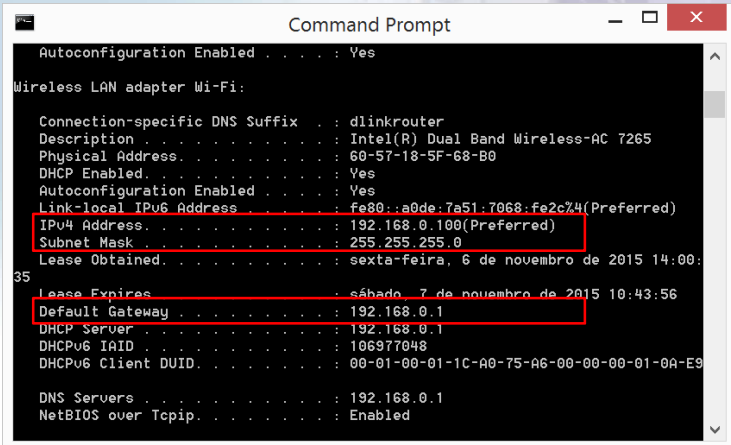


<https://www.arduino.cc/en/Reference/Ethernet>

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Configuração do Arduino Ethernet Shield W5100 (1)

- O primeiro passo é configurar com um endereço IP válido da sua rede ao Ethernet Shield. No prompt, digite: `ipconfig /all`



```

Command Prompt
Autoconfiguration Enabled . . . . . : Yes
Wireless LAN adapter Wi-Fi:
Connection-specific DNS Suffix  . : dlinkrouter
Description . . . . . : Intel(R) Dual Band Wireless-AC 7265
Physical Address. . . . . : 60-57-18-5F-68-B0
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::a0de:7a51:7068:fe2c%4(Preferred)
IPv4 Address. . . . . : 192.168.0.100(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : sexta-feira, 6 de novembro de 2015 14:00:35
Lease Expires . . . . . : sábado, 7 de novembro de 2015 10:43:56
Default Gateway . . . . . : 192.168.0.1
DHCP Server . . . . . : 192.168.0.1
DHCPv6 IAID . . . . . : 106977048
DHCPv6 Client DUID. . . . . : 00-01-00-01-1C-A0-75-A6-00-00-00-01-0A-E9
DNS Servers . . . . . : 192.168.0.1
NetBIOS over Tcpip. . . . . : Enabled
  
```

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Configuração do Arduino Ethernet Shield W5100 (2)

- Esses três parâmetros são definidos logo no início do Sketch do Arduino e devem ser alterados de acordo com a configuração da rede:
 - IPAddress ip(192,168,0,188):** Troque por um endereço IP no mesmo formato daquele apresentado na janela de prompt de comando, mas o último número deve ser diferente. Exemplo: o IP do PC é 192.168.0.100, e no Sketch será utilizado o 192.168.0.188. Antes de usar qualquer endereço da rede, certifique-se que o mesmo ainda não está em uso por nenhum outro equipamento
 - IPAddress gateway(192,168,0,1):** Utilize o mesmo endereço do Gateway Padrão apresentado na janela de prompt de comando. Neste caso, 192.168.0.1
 - IPAddress subnet(255,255,255,0):** Utilize o mesmo endereço referente à máscara de sub-rede, apresentado na janela de prompt de comando : 255.255.255.0
- Nos comandos acima, utiliza-se virgula mesmo ao invés de ponto

Configuração do Arduino Ethernet Shield W5100 (3)

- Vamos testar o funcionamento da placa utilizando um programa que simplesmente configura o endereço IP:

```

1 // Programa : Ethernet Shield Wiznet W5100 - Define endereço IP
2
3 #include <SPI.h>
4 #include <Ethernet.h>
5
6 // A linha abaixo permite que voce defina o endereço
7 // físico (MAC ADDRESS) da placa de rede
8 byte mac[] = { 0xAB, 0xCD, 0x12, 0x34, 0xFF, 0xCA };
9
10 // Os valores abaixo definem o endereço IP, gateway e máscara.
11 // Configure de acordo com a sua rede.
12 IPAddress ip(192,168,0,101); //Define o endereço IP
13 IPAddress gateway(192,168,0,1); //Define o gateway
14 IPAddress subnet(255, 255, 255, 0); //Define a máscara de rede
15
16 void setup()
17 {
18   Ethernet.begin(mac, ip); //Inicializa a placa com os dados fornecidos
19 }
20
21 void loop() {}

```

Configuração do Arduino Ethernet Shield W5100 (4)

- Para verificar se funcionou a configuração inicial:

```

Command Prompt

C:\Users\blromano>ping 192.168.0.101

Pinging 192.168.0.101 with 32 bytes of data:
Reply from 192.168.0.101: bytes=32 time=4ms TTL=128
Reply from 192.168.0.101: bytes=32 time=2ms TTL=128
Reply from 192.168.0.101: bytes=32 time=2ms TTL=128
Reply from 192.168.0.101: bytes=32 time=3ms TTL=128

Ping statistics for 192.168.0.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 4ms, Average = 2ms

C:\Users\blromano>

```

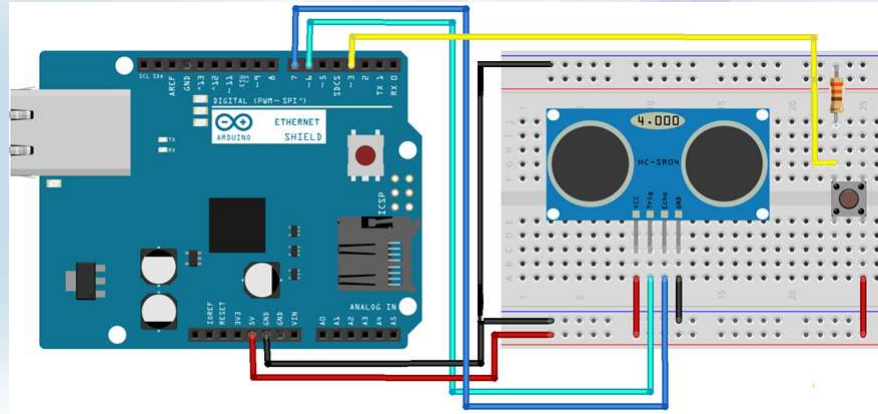
Problem Based Learnd - PBL 01

- **Problema:** Apresentar, em uma Página Web, o valor da distância em centímetros medida ao se utilizar o Sensor Ultrassônico HC-SR04 e se um botão mantém-se pressionado ou não, ao se utilizar o Arduino acoplado com o Ethernet Shield W5100

Componentes Utilizados

- Arduino UNO / Arduino MEGA
- Cabo USB
- Protoboard
- Arduino Ethernet Shield W5100
- 01 Sensor Ultrassônico SR-04
- 01 Push Button
- 01 Resistor de 330 ohms
- Jumpers

Construindo o Circuito no Protoboard



Código Fonte para Solução (1)

```

1 //Programa : Arduino Ethernet Shield W5100 e HC-SR04
2
3 #include <Ultrasonic.h>
4 #include <SPI.h>
5 #include <Ethernet.h>
6
7 //Define os parametros para o sensor ultrasonico HC-SR04
8 #define PINO_TRIGGER 6 //Porta ligada ao pino Trigger do sensor
9 #define PINO_ECHO 7 //Porta ligada ao pino Echo do sensor
10
11 //Inicializa o sensor ultrasonico
12 Ultrasonic ultrasonic(PINO_TRIGGER, PINO_ECHO);
13
14 //Definicoes de IP, mascara de rede e gateway
15 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
16 IPAddress ip(192,168,0,101); //Define o endereco IP
17 IPAddress gateway(192,168,0,1); //Define o gateway
18 IPAddress subnet(255, 255, 255, 0); //Define a máscara de rede
19
20 //Inicializa o servidor web na porta 80
21 EthernetServer server(80);
22
23 void setup()
24 {
25 //Inicializa a interface de rede
26 Ethernet.begin(mac, ip, gateway, subnet);
27 server.begin();
28 }

```


Código Fonte para Solução (2)

```

30 void loop() {
31   float cmMsec;
32   long microsec = ultrasonic.timing();
33
34   //Le e armazena as informacoes do sensor ultrasonico
35   cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);
36
37   //Aguarda conexao do browser
38   EthernetClient client = server.available();
39   if (client) {
40     // Um Request HTTP termina com uma linha em branco
41     boolean currentLineIsBlank = true;
42     while (client.connected()) {
43       if (client.available()) {
44         char c = client.read();
45
46         //Se recebeu uma nova linha e a linha eh branca, a HTTP Request eh finalizada
47         //Então pode enviar um reply
48         if (c == '\n' && currentLineIsBlank) {
49           // send a standard http response header
50           client.println("HTTP/1.1 200 OK");
51           client.println("Content-Type: text/html");
52           client.println("Connection: close");
53           client.println("Refresh: 2"); //Recarrega a pagina a cada 2seg
54           client.println();
55           client.println("<!DOCTYPE HTML>");
56           client.println("<html>");

```

As informações serão enviadas pelo Webserver da placa Ethernet à cada 2 segundos.

Código Fonte para Solução (3)

```

60 //Configura o texto e imprime o titulo no browser
61 client.print("<font color=#FF0000><b><u>");
62 client.print("Envio de informacoes pela rede utilizando Arduino");
63 client.print("</u></b></font>");
64 client.println("<br />");
65 client.println("<br />");
66
67 //Mostra o estado da porta digital 3
68 int porta_digital = digitalRead(3);
69 client.print("Porta Digital 3 : ");
70 client.print("<b>");
71 client.print(porta_digital);
72 client.println("</b>");
73 client.print(" (0 = Desligada, 1 = Ligada)");
74 client.println("<br />");
75
76 //Mostra as informacoes lidas pelo sensor ultrasonico
77 client.print("Sensor Ultrasonico : ");
78 client.print("<b>");
79 client.print(cmMsec);
80 client.print(" cm");
81 client.println("</b></html>");
82 break;
83 }

```

Também pode-se configurar os comandos HTML para formatação, como por exemplo `` para exibir o texto do título na cor vermelha, `` para negrito e `<u>` para sublinhado. Pode-se utilizar outros comandos HTML.

Código Fonte para Solução (4)

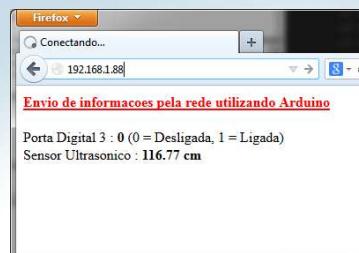
```

85     if (c == 'n') {
86         // Inicia uma nova linha
87         currentLineIsBlank = true;
88     }
89     else if (c != 'r') {
90         // Recebeu um caracter na nova linha
91         currentLineIsBlank = false;
92     }
93 }
94 }
95 // give the web browser time to receive the data
96 delay(1);
97
98 // close the connection:
99 client.stop();
100 }
101 }

```

Verificação e Simulação do Código Desenvolvido

- Embarcar o código-fonte na aplicação (Sem o Ethernet Shield estar acoplado no Arduino)
 - Conectar o Ethernet Shield W5100 no cabo de rede
 - Ligar o arduino na fonte externa
- Para testar o funcionamento, abra o browser no seu computador e digite na barra de endereços o IP configurado no programa



Navegador no Computador



Navegador no Celular

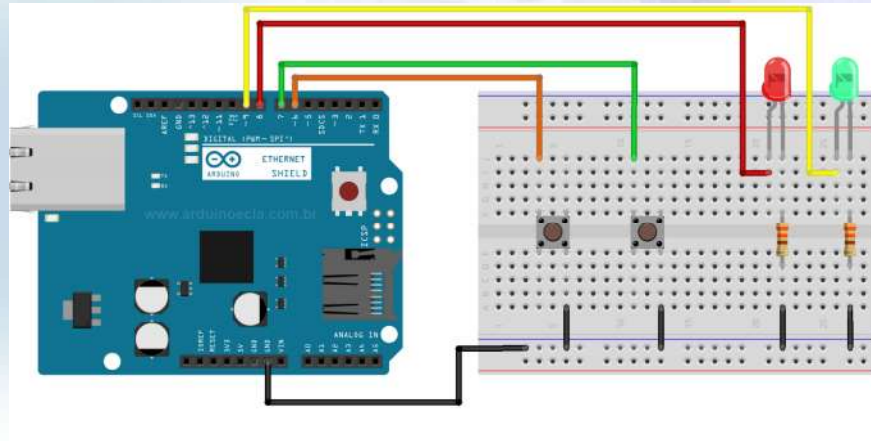
Problem Based Learnd - PBL 02

- **Problema:** Apresentar, em uma Página Web, se dois LEDs foram ou não acionados pelo usuário através de *push-buttons*, mantendo-se acesos por 5 segundos. Deve-se utilizar o Arduino acoplado com o Ethernet Shield W5100
 - Os LEDs não ficam acionados simultaneamente

Componentes Utilizados

- Arduino UNO / Arduino MEGA
- Cabo USB
- Protoboard
- Arduino Ethernet Shield W5100
- 02 LEDs
- 02 Push Button
- 04 Resistor de 330 ohms
- Jumpers

Construindo o Circuito no Protoboard



Código Fonte para Solução (1)

```

1 // Programa : Webserver com aviso de acionamento de botoes
2 // Baseado no programa exemplo Webserver, de David Mellis e Tom Igoe
3
4 #include <SPI.h>
5 #include <Ethernet.h>
6
7 // A linha abaixo permite que voce defina o endereço fisico (MAC ADDRESS)
8 // da placa de rede
9 byte mac[] = { 0xAB, 0xCD, 0x12, 0x34, 0xFF, 0xCA };
10
11 // Os valores abaixo definem o endereço IP, gateway e máscara.
12 // Configure de acordo com a sua rede.
13 IPAddress ip(192,168,0,188); //Define o endereço IP
14 IPAddress gateway(192,168,0,1); //Define o gateway
15 IPAddress subnet(255, 255, 255, 0); //Define a máscara de rede
16
17 // Inicializa a biblioteca da placa ethernet com as
18 // configurações de IP fornecidas
19 EthernetServer server(80);
20
21 int botao1 = 6; //Botao que aciona o led vermelho
22 int botao2 = 7; //Botao que aciona o led verde
23 int pinoled=8; //Pino ligado ao led vermelho
24 int pinoled2=9; //Pino ligado ao led verde
25 int leitura = 0; //Armazena o valor de leitura do botao1
26 int leitura2 = 0; //Armazena o valor de leitura do botao2

```

Código Fonte para Solução (2)

```

28 void setup()
29 {
30   pinMode(pinoled, OUTPUT); //Led
31   pinMode(pinoled2, OUTPUT); //Led
32   pinMode(botao1, INPUT);
33   digitalWrite(botao1, HIGH);
34   pinMode(botao2, INPUT);
35   digitalWrite(botao2, HIGH);
36   //Inicializa a conexão ethernet e o servidor web na porta 80
37   Ethernet.begin(mac, ip);
38   server.begin();
39   Serial.print("server is at ");
40   Serial.println(Ethernet.localIP());
41 }
42
43 void loop()
44 {
45   //Verifica o status do Botao1 e imprime mensagem no browser
46   leitura=digitalRead(botao1);
47   if (leitura == 0)
48   {
49     //Altera Estado de LED e Imprime
50     digitalWrite(pinoled, HIGH);
51     apresentados("Botao 1 acionado!", "Aguardando...");
52     delay(5000); //Mantem o led aceso por 5 segundos
53
54     //Imprime mensagem padrao, aguardando novo acionamento
55     apresentados("Aguardando...", "Aguardando...");
56     digitalWrite(pinoled, LOW);
57   }

```

Código Fonte para Solução (3)

```

59 //Verifica o status do Botao2 e imprime mensagem no browser
60 leitura2=digitalRead(botao2);
61 if (leitura2 == 0)
62 {
63   //Altera Estado de LED e Imprime
64   digitalWrite(pinoled2, HIGH);
65   apresentados("Aguardando...", "Botao 2 acionado!");
66   delay(5000); //Mantem o led aceso por 5 segundos
67
68   //Imprime mensagem padrao, aguardando novo acionamento
69   apresentados("Aguardando...", "Aguardando...");
70   digitalWrite(pinoled2, LOW);
71 }
72 }
73
74 // Rotina que recebe os valores de Mensagem e Mensagem2,
75 // imprimindo o resultado no browser
76 void apresentados(char msg[], char msg2[])
77 {
78   // listen for incoming clients
79   EthernetClient client = server.available();
80   if (client) {
81     Serial.println("new client");
82     // an http request ends with a blank line
83     boolean currentLineIsBlank = true;
84     while (client.connected()) {
85       if (client.available()) {
86         char c = client.read();
87         Serial.write(c);

```

Código Fonte para Solução (4)

```

95 // if you've gotten to the end of the line (received a newline
96 // character) and the line is blank, the http request has ended,
97 // so you can send a reply
98 if (c == '\n' && currentLineIsBlank) {
99 // send a standard http response header
100 client.println("HTTP/1.1 200 OK");
101 client.println("Content-Type: text/html");
102 // the connection will be closed after completion of
103 // the response
104 client.println("Connection: close");
105
106 // refresh the page automatically every 5 sec
107 client.println("Refresh: 0");
108 client.println();
109 client.println("<!DOCTYPE HTML>");
110 client.println("<html>");
111 // output the value of each analog input pin
112 client.print("Estado Botao 1 : ");
113 client.print(msg);
114 client.println("<br />");
115 client.print("Estado Botao 2 : ");
116 client.print(msg2);
117 client.println("<br />");
118 client.println("</html>");
119 break;
120 }

```

Código Fonte para Solução (5)

```

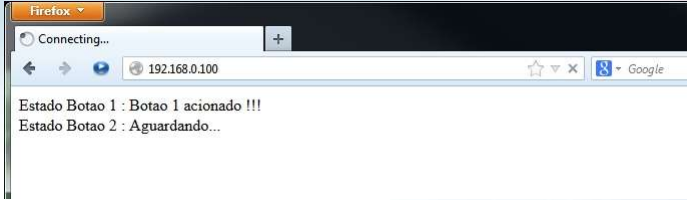
121 if (c == '\n') {
122 // you're starting a new line
123 currentLineIsBlank = true;
124 }
125 else if (c != '\r') {
126 // you've gotten a character on the current line
127 currentLineIsBlank = false;
128 }
129 }
130 }
131 // give the web browser time to receive the data
132 delay(1);
133 // close the connection:
134 client.stop();
135 Serial.println("client disconnected");
136 }
137 }

```

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Verificação e Simulação do Código Desenvolvido


- Embarcar o código-fonte na aplicação (Sem o Ethernet Shield estar acoplado no Arduino)
- Conectar o Ethernet Shield W5100 no cabo de rede
- Para testar o funcionamento, abra o browser no seu computador e digite na barra de endereços o IP configurado no programa



Instituto Federal de São Paulo – IFSP São João da Boa Vista

Problem Based Learnd - PBL 03

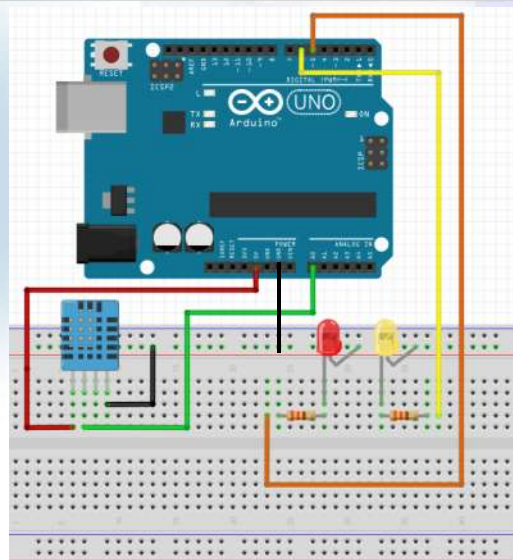
- Problema:** Apresentar, uma Página Web embarcada no Arduino+Ethernet Shield W5100, as seguintes informações:
 - Medições de Temperatura e Umidade Utilizando o DHT 11
 - Um Botão que ao ser acionado faz com que o LED Amarelo fique piscando (Efeito Blink)
 - Um Botão que ao ser acionado faz com que o LED Vermelho fique mudando seu brilho de claro para escuro (Efeito FADE)



Componentes Utilizados

- Arduino UNO / Arduino MEGA
- Cabo USB
- Protoboard
- Arduino Ethernet Shield W5100
- 02 LEDs (Amarelo e Vermelho)
- 01 DHT 11
- 02 Resistor de 330 ohms
- Jumpers

Construindo o Circuito no Protoboard



Código Fonte para Solução (1)

```

1 #define LED_Amarelo 5 //Define LED_Amarelo como 5
2 #define LED_Vermelho 6 //Define LED_Vermelho como 6
3
4 #include <SPI.h> //Inclui a biblioteca SPI.h
5 #include <Ethernet.h> //Inclui a biblioteca Ethernet.h
6 #include "DHT.h"
7
8 //Definicao das informacoes de temperatura
9 #define DHTPIN A0 // pino que estamos conectados
10 #define DHTTYPE DHT11 // DHT 11
11 DHT dht(DHTPIN, DHTTYPE);
12
13 // Configurações para o Ethernet Shield
14 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFF, 0xED};
15 IPAddress ip(192,168,0,100); //Define o endereço IP
16 IPAddress gateway(192,168,0,1); //Define o gateway
17 IPAddress subnet(255, 255, 255, 0); //Define a máscara de rede
18
19 EthernetServer server(80); //Inicializa a biblioteca EthernetServer com os valores de IP acima citados e configura a porta de acesso(80)
20
21 byte COD = B1000;
22
23 String A1_carga = "LED Amarelo Blink"; //Função do primeiro botão
24 String A2_carga = "LED Vermelho Fade"; //Função do segundo botão
25
26 boolean A1_estado=false; //Variável para armazenar o estado do primeiro botão
27 boolean A2_estado=false; //Variável para armazenar o estado do segundo botão
28
29 long anteriorMillis = 0; //Variável para auxiliar no timer do BLINK do LED Amarelo
30 long intervalo = 1000; //Variável para armazenar a frequência do BLINK do LED Amarelo

```

Código Fonte para Solução (2)

```

31 int brilhoFade = 0; // Variável para armazenar o valor do brilho do FADE do LED Vermelho
32 int degrauFade = 5; // Variável para armazenar o valor do degrau do FADE do LED Vermelho
33
34 void setup()
35 {
36   pinMode(LED_Amarelo,OUTPUT); //Define o pino 5 como saída
37   pinMode(LED_Vermelho,OUTPUT); //Define o pino 6 como saída
38   dht.begin(); //inicializar o sensor de temperatura e umidade
39   Ethernet.begin(mac, ip, gateway, subnet);
40   server.begin();
41 }
42
43 void loop()
44 {
45   acionamentos(); //Vai para a função que executa o acionamento dos botões
46   float temperatura = dht.readTemperature();
47   float umidade = dht.readHumidity();
48
49   EthernetClient client = server.available(); // Verifica se tem alguém conectado
50
51   if (client)
52   {
53     boolean currentLineIsBlank = true; // A requisição HTTP termina com uma linha em branco indica o fim da linha
54     String valPage;
55   }
56 }

```

Código Fonte para Solução (3)

```

58 while (client.connected())
59 {
60     if (client.available())
61     {
62         char c = client.read(); //Variável para armazenar os caracteres que forem recebidos
63         valPag.concat(c); // Pega os valor após o IP do navegador ex: 192.168.0.108/0001
64
65         //Compara o que foi recebido
66         if(valPag.endsWith("0001")) //Se o que for pego após o IP for igual a 0001
67         {
68             COD = COD ^ B0001; //Executa a lógica XOR entre a variável atual de COD e o valor B0001
69             AI_estado = !AI_estado; //Inverte o estado do primeiro acionamento
70         }
71
72         else if(valPag.endsWith("0010")) //Senão se o que for pego após o IP for igual a 0010
73         {
74             COD = COD ^ B0010; //Executa a lógica XOR entre a variável atual de COD e o valor B0010
75             AD_estado = !AD_estado; //Inverte o estado do segundo acionamento
76         }
77
78         //=====
79
80         if (c == '\n' && currentLineIsBlank)
81         {
82             //Inicia página HTML
83             client.println("HTTP/1.1 200 OK");
84             client.println("Content-Type: text/html");
85             client.println();
86             client.print("<HTML> ");
87             client.println("<BR><center><B>CONTROLAR LEDS e MEDIR TEMPERATURA</B></center>");

```

Código Fonte para Solução (4)

```

88 //Display da Temperatura e Unidade
89 if(COD==B1000)
90 {
91     client.print("<BR>");
92
93     client.print("<center>Temperatura do Sistema: <font size=7> <font color=#ff6600> ");
94     client.print(String(temperatura));
95     client.print(" °C. </font></font></center>");
96
97     client.print("<BR><BR>");
98     client.print("<center>Unidade do Sistema: <font size=7> <font color=#ff6600> ");
99     client.print(String(unidade));
100     client.print(" & </font></font></center>");
101 }
102 //=====
103
104 client.print("<BR><BR>");
105
106 //Primeiro BOTAO
107 client.print("<center><button onclick='\"window.location.href=http://192.168.0.108/0001'\">0</button> >Codigo: 0001 > ");
108 if(AI_estado)
109 {
110     client.print("<BR><span style='color: #00ff00;'>");
111     client.print(AI_carga);
112     client.print("<BR>");
113     client.print("</span></center>");
114 }
115 else
116 {
117     client.print("<BR><span style='color: #ff0000;'>");
118     client.print(AI_carga);

```

Código Fonte para Solução (5)

```

121     client.print (" - OFF");
122     client.print ("</span></B></center>");
123 }
124 //=====
125 //Segundo BOPAO
126 client.print ("<center><button onclick='\"window.location.href='\"http://192.168.0.188/0010'\">\>0</button> >Codigo: 0010 > ");
127 if (A2_estado)
128 {
129     client.print ("<B><span style='color: #00ff00;'>");
130     client.print (A2_carga);
131     client.print (" - ON");
132     client.print ("</span></B></center>");
133 }
134 else
135 {
136     client.print ("<B><span style='color: #ff0000;'>");
137     client.print (A2_carga);
138     client.print (" - OFF");
139     client.print ("</span></B></center>");
140 }
141 }
142 //=====
143 client.print ("<B><B>");
144 //COD ATUAL
145 client.print ("<center>COD ATUAL: <B><font size=?> ");
146 if (COD>=81000)
147 {
148     client.print (COD, BIN);
149     client.print ("</center></font></B>");
150 }
151 }
152

```

Código Fonte para Solução (6)

```

153     else
154     {
155         client.print (COD, HEX);
156         client.print ("<h</center></font></B>");
157     }
158 //=====
159 client.print (" <meta http-equiv='\"refresh\" content='\"5; url=http://192.168.0.188/'\"> ");
160 client.println("</HTML>");
161 break;
162 } //Fecha if (c == '\n' && currentLineIsBlank)
163 } //Fecha if (client.available())
164 } //Fecha While (client.connected())
165 delay(3); // Espera um tempo para o navegador receber os dados
166 client.stop(); // Fecha a conexão
167 } //Fecha if(client)
168 } //Fecha loop
169

```

Código Fonte para Solução (7)

```

179 void acionamentos()
180 { //Abre função acionamento()
181
182 //LED Amarelo - BLINK
183 if(A1_estado) //Se o botão da página estiver em estado ON
184 {
185     unsigned long atualMillis = millis(); //Armazena o tempo decorrido desde que o programa começou a ser rodado
186
187     if(atualMillis - anteriorMillis > intervalo) //Se o tempo decorrido for maior que 1000 milissegundos
188     {
189         anteriorMillis = atualMillis; //Armazena o valor atual de Millis em anteriorMillis
190         digitalWrite(LED_Amarelo, !digitalRead(LED_Amarelo)); //Inverte o estado do LED
191     }
192 }
193 else
194     digitalWrite(LED_Amarelo, LOW); //Se não, quando apaga o LED Amarelo
195
196 //LED Vermelho - FADE
197 if(A2_estado)
198 {
199     analogWrite(LED_Vermelho, brilhoFade); //Altera o valor de PWM no pino 6
200     brilhoFade = brilhoFade + degrauFade; //Soma na variável brilhoFade o valor do degrau
201
202     if (brilhoFade == 0 || brilhoFade == 255)
203         degrauFade = -degrauFade; //Se o brilho Fade chega no máximo ou no mínimo...
204                                     //Inverte o sentido de intensidade do Fade
205     delay(15); //Aguarda 15 milissegundos
206 }
207 else digitalWrite(LED_Vermelho, LOW);
208 }

```

Verificação e Simulação do Código Desenvolvido

- Embarcar o código-fonte na aplicação (Sem o Ethernet Shield estar acoplado no Arduino)
- Conectar o Ethernet Shield W5100 no cabo de rede
- Para testar o funcionamento, abra o browser no seu computador e digite na barra de endereços o IP configurado no programa

CONTROLAR LEDS e MEDIR TEMPERATURA

Temperatura do Sistema: **28.00 °C**

Unidade do Sistema: **66.00 %**

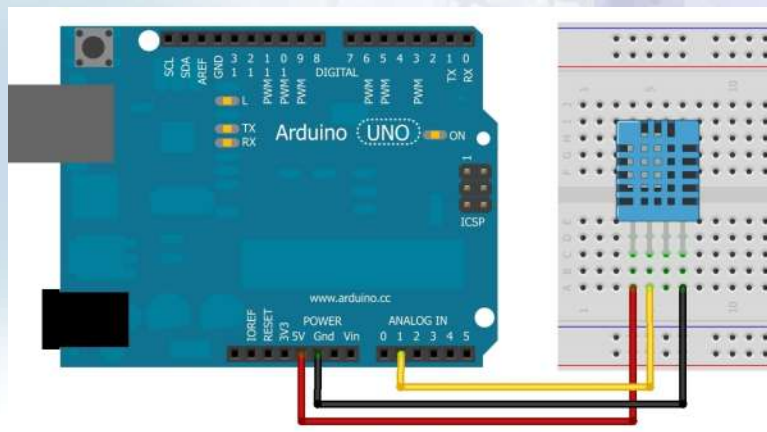
LED Amarelo Blink - OFF
LED Vermelho Fade - OFF

COD ATUAL: **1000b**

Problem Based Learnd - PBL 04

- **Problema:** Desenvolver (Simplificado) uma Aplicação que integre o Android e Arduino acoplado com o Ethernet Shield W5100, para mostrar as medições de Temperatura e Umidade no Celular, utilizando-se do Sensor DHT11, através de requisição por método GET

Construindo o Circuito no Protoboard



Instituto Federal de São Paulo – IFSP São João da Boa Vista

Código Fonte para Solução Embarcada - Arduino (1)

```

1 #include "DHT.h"
2 #include <SPI.h>
3 #include <Ethernet.h>
4
5 byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
6 byte ip[] = { 192, 168, 0, 188 };
7 byte gateway[] = { 192, 168, 0, 1 };
8 byte subnet[] = { 255, 255, 255, 0 };
9 EthernetServer server(80); //CASO OCORRA PROBLEMAS COM A PORTA 80, UTILIZE OUTRA (EX:8082,8089)
10
11 #define DHTPIN A1
12 #define DHTTYPE DHT11 // DHT 11
13
14 // Conecte pino 1 do sensor (esquerda) ao +5V
15 // Conecte pino 2 do sensor ao pino de dados definido em seu Arduino
16 // Conecte pino 4 do sensor ao GND
17 // Conecte o resistor de 10K entre pin 2 (dados)
18 // e ao pino 1 (VCC) do sensor
19 DHT dht(DHTPIN, DHTTYPE);
20
21 String readString = String(30);
22
23 String umidade;
24 String temperatura;
25
26 void setup()
27 {
28   Ethernet.begin(mac, ip, gateway, subnet); // INICIALIZA A CONEXÃO ETHERNET
29   Serial.begin(9600); //TAXA DE COMUNICAÇÃO DA PORTA SERIAL
30   dht.begin();
31 }

```

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Código Fonte para Solução Embarcada - Arduino (2)

```

1 void loop()
2 {
3   EthernetClient client = server.available();
4   if (client) { // SE EXISTE CLIENTE
5     while (client.connected()) { //ENQUANTO EXISTIR CLIENTE CONECTADO
6       if (client.available()) {
7         char c = client.read();
8         if (readString.length() < 100)
9         {
10           readString += c;
11         }
12         Serial.print(c);
13       }
14       if (c == '\n') { // SE ENCONTRAR "\n" É O FINAL DO CABEÇALHO DA REQUISIÇÃO HTTP
15         if (readString.indexOf("?") < 0) //SE ENCONTRAR O CARACTER "?"
16         {
17           //SE NÃO
18           if(readString.indexOf("L=1") > 0) { //SE ENCONTRAR O PARÂMETRO "L=1"
19             umidade = String(dht.readHumidity());
20             temperatura = ","+String(dht.readTemperature());
21           }
22           client.println("HTTP/1.1 200 OK"); // ESCRIVE PARA O CLIENTE A VERSÃO DO HTTP
23           client.println("Content-Type: text/html"); // ESCRIVE PARA O CLIENTE O TIPO DE CONTEÚDO(texto/html)
24           client.println();
25           client.println(umidade); // RETORNA PARA O CLIENTE O VALOR DA UNIDADE
26           client.println(temperatura); // RETORNA PARA O CLIENTE O VALOR DA TEMPERATURA
27           readString="";
28           client.stop(); // FINALIZA A REQUISIÇÃO HTTP
29         }
30       }
31     }
32   }
33 }

```


Instituto Federal de São Paulo – IFSP São João da Boa Vista


Embarcando o Código Desenvolvido no Arduino

- Embarcar o código-fonte na aplicação (Sem o Ethernet Shield estar acoplado no Arduino)

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Desenvolvimento da Aplicação Android (1)

Application Name	Aula12-LerTemperaturaUmidadeDHT11EthernetShield
Package Name	pdm.android.aula12_ler_temperatura_umidade_dht_ethernet
Activity Name	UmidTempActivity



Desenvolvimento: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.umidade_temperatura"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Aula12 - Umidade e Temperatura Ethernet W5100"
        android:theme="@style/AppTheme" >

        <activity
            android:name="com.example.umidade_temperatura.UmidTempActivity"
            android:label="Aula12 - Umidade e Temperatura Ethernet W5100"
            android:screenOrientation="portrait">

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Desenvolvimento da Activity: ConnectHttpClient.java (1)

```
public class ConnectHttpClient {
    public static final int HTTP_TIMEOUT = 2 * 1000;
    private static HttpClient httpClient;

    private static HttpClient getHttpClient() {
        if (httpClient == null) {
            httpClient = new DefaultHttpClient();
            final HttpParams httpParams = httpClient.getParams();
            HttpConnectionParams.setConnectionTimeout(httpParams, HTTP_TIMEOUT);
            HttpConnectionParams.setSoTimeout(httpParams, HTTP_TIMEOUT);
            ConnManagerParams.setTimeout(httpParams, HTTP_TIMEOUT);
        }
        return httpClient;
    }
}
```

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Desenvolvimento da Activity: ConnectHttpClient.java (2)

```

public static String executaHttpGet(String url) throws Exception {
    BufferedReader bufferedReader = null;
    try {
        HttpClient client = getHttpClient();
        HttpGet httpGet = new HttpGet(url);
        httpGet.setURI(new URI(url));
        HttpResponse httpResponse = client.execute(httpGet);
        bufferedReader = new BufferedReader(new InputStreamReader(httpResponse.getEntity().getContent()));
        StringBuffer stringBuffer = new StringBuffer("");
        String line = "";
        String LS = System.getProperty("line.separator"); // \s
        while ((line = bufferedReader.readLine()) != null) {
            stringBuffer.append(line + LS);
        }
        bufferedReader.close();

        String resultado = stringBuffer.toString();
        return resultado;
    } finally {
        if (bufferedReader != null) {
            try {
                bufferedReader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

```

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Desenvolvimento da Activity: UmidTempActivity.java (1)

```

public class UmidTempActivity extends ActionBarActivity implements OnClickListener {

    TextView tvUmidade, tvTemperatura;
    EditText et_Ip;
    Button btConectar;
    String l, hostIp = "192.168.0.188";
    Handler mHandler;
    long lastPress;

    // METODO QUE CRIA A PRIMEIRA TELA DA APLICACAO
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        telaIp();
    }

    public void telaIp(){
        setContentView(R.layout.tela_ip);
        et_Ip = (EditText) findViewById(R.id.et_Ip);

        btConectar = (Button) findViewById(R.id.btConectar);
        btConectar.setOnClickListener(this);

        if(btConectar.isPressed()){
            onClick(btConectar);
        }
    }

    public void telaPrincipal(){
        setContentView(R.layout.activity_umid_temp);

        mHandler = new Handler();
        mHandler.post(mUpdate);

        tvUmidade = (TextView) findViewById(R.id.tvUmidade);
        tvTemperatura = (TextView) findViewById(R.id.tvTemperatura);
    }
}

```

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Desenvolvimento da Activity: UmidTempActivity.java (2)

```

public void telaPrincipal() {
    setContentView(R.layout.activity_umid_temp);

    mHandler = new Handler();
    mHandler.post(mUpdate);

    tvUmidade = (TextView) findViewById(R.id.tvUmidade);
    tvTemperatura = (TextView) findViewById(R.id.tvTemperatura);
}

// METODO QUE EXECUTA A ATUALIZACAO DO TEXTVIEW COM AS INFORMACOES RECEBIDAS DO ARDUINO
private Runnable mUpdate = () -> {
    arduinoStatus("http://" + hostIp + "/" + "1"); // CHAMA O METODO "arduinoStatus" E PASSA O PARAMETRO ENTRE "PARENTESSES"
    mHandler.postDelayed(this, 2000); // TEMPO DE INTERVALO PARA ATUALIZAR NOVAMENTE A INFORMACAO (2 SEGUNDOS)
};

// METODO "arduinoStatus"
public void arduinoStatus(String urlArduino) {
    String urlHost = urlArduino;
    String respostaRetornada = null; // CRIA UMA STRING CHAMADA "respostaRetornada" QUE POSSUI VALOR NULO

    try {
        respostaRetornada = ConnectHttpClient.executaHttpGet(urlHost); // STRING "respostaRetornada" RECEBE RESPOSTA RETORNADA PELO
        String resposta = respostaRetornada.toString(); // STRING "resposta"
        resposta = resposta.replaceAll("\\s+", "");

        String[] b = resposta.split(","); // O VETOR "String[] b" RECEBE O VALOR DE "resposta.split(",")"

        tvUmidade.setText(b[0] + "%"); // O TEXTVIEW RECEBE O VALOR RETORNADO PELO ARDUINO NA POSICAO 1 (umidade) DO VETOR
        tvTemperatura.setText(b[1] + "°C"); // O TEXTVIEW RECEBE O VALOR RETORNADO PELO ARDUINO NA POSICAO 2 (temperatura) DO VETOR
    } catch (Exception erro) {
    }
}

```

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Desenvolvimento da Activity: UmidTempActivity.java (3)

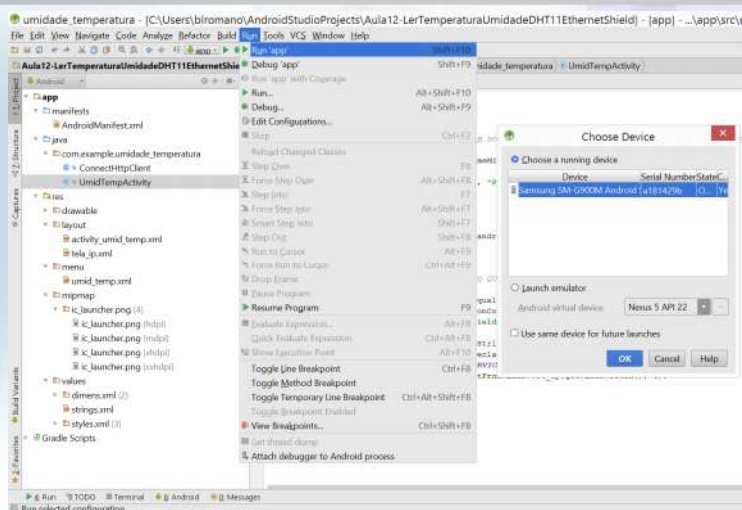
```

// METODO QUE VERIFICA O BOTAO DE VOLTAR DO DISPOSITIVO ANDROID E ENCERRA A APLICACAO SE PRESSIONADO 2 VEZES SEGUIDAS
public void onBackPressed() {
    long currentTime = System.currentTimeMillis();
    if (currentTime - lastPress > 5000) {
        Toast.makeText(getApplicationContext(), "Pressione novamente para sair.", Toast.LENGTH_LONG).show();
        lastPress = currentTime;
    } else {
        super.onBackPressed();
        android.os.Process.killProcess(android.os.Process.myPid());
    }
}

@Override
public void onClick(View bt) { // METODO QUE GERENCIA OS CLICK'S NOS BOTÕES
    if (bt == btConectar) {
        if (et_Ip.getText().toString().equals("")) {
            Toast.makeText(getApplicationContext(),
                "Digite o IP do Ethernet Shield!", Toast.LENGTH_SHORT).show();
        } else {
            hostIp = et_Ip.getText().toString();
            InputMethodManager escondeTeclado = (InputMethodManager) getSystemService(
                Context.INPUT_METHOD_SERVICE);
            escondeTeclado.hideSoftInputFromWindow(et_Ip.getWindowToken(), 0);
            telaPrincipal();
        }
    }
}
}

```

Embarcar o Código do Android no Celular



Após Embarcar o Código, verificar a integração entre o Android e o Arduino

Miniprojeto 22

- Desenvolver uma Aplicação que integre o Android e Arduino acoplado com o Ethernet Shield W5100 para realizar o controle de 03 LEDs distintos que podem ficar acesos ou apagados dependendo da escolha do usuário.
 - A comunicação entre o Android e o Arduino deve ser desenvolvida através de requisição por método GET
 - Na aplicação deve ser mostrada o Status Aceso/Apagado de cada um dos LEDs



Miniprojeto 23

- Desenvolver uma Aplicação que integre por Bluetooth o Android e Arduino utilizando botões para solicitar a Leitura das seguintes medições: Temperatura (Celsius e Fahrenheit), Umidade, Luminosidade (Baixo, Médio e Alto), Distância (Centímetros e Polegadas). As medições solicitadas devem ser apresentadas em um WebServer desenvolvido com o Ethernet Shield W5100
 - A página Web é apenas visualização das medições
 - O aplicativo Android é apenas o controlador das medições

Miniprojeto 24

- Desenvolver uma Aplicação que integre o Android e Arduino acoplado com o Ethernet Shield W5100 para realizar o controle de 03 LEDs por comandos de Voz utilizando a biblioteca do Google
 - Observação: É o mesmo circuito do Miniprojeto 22

PDM: Projeto para Dispositivos Móveis

Aula 12: Integrando Android e Arduino utilizando o Ethernet Shield

Breno Lisi Romano

Obrigado!

Instituto Federal de São Paulo – IFSP São João da Boa Vista
Especialização em Desenvolvimento para Dispositivos Móveis



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista