Monitor de Umidade do Solo

```
Umidade do Solo: 3778

Tempo desde a última vez ligada a bomba: 328 Segundos.

Condição atual do Solo: Solo Encharcado!

{"umidade":3792, "solo":"Solo Encharcado!", "segundos":142}
```

Trocando arquivos.

Os servidores estão comunicando entre si. O servidor Cliete esta requisitando com sucesso as informações gravadas no servidor do ESP32, Informações estás sendo.

- valor menino pelo sensor
- Função de verificação do estado do solo
- Seguntos deis da ultima vez que a bomba d'agua foi ligada.

Código

```
#include <Wieserver.h>

#include <WebServer.h>

// definir senha e nome do wifi que o ESP32 vai se conectar.

const char* ssid = "********";

const char* password = "*********";

WebServer server(80); // cria um servidor na porta 80

// conectar ao Wi-Fi
WiFi.begin(ssid, password);
```

//diz que está conectando no serial e enquanto não estiver em situação de conectado, ele escreverá pontos até conectar.

```
Serial.print("Conectando-se ao WiFi...");
  while (WiFi.status() != WL CONNECTED) {
     delay(1000);
     Serial.print(".");
  }
//quand estiver conectado, pula uma linha, e escreve o IP do Servidor criado
pelo ESP32 que será usado depois
  Serial.println("\nConectado ao WiFi!");
  Serial.print("Endereço IP: ");
  Serial.println(WiFi.localIP());
// configura um servidor com o nome ip/dados para envio de informações.
  server.on("/dados", []() {
//configuração de entrega de informações para o servidor
     valor = analogRead(Sensor); // Atualiza o valor do sensor
     String umidade = String(valor);
     String solo = Solo(Sensor);
//cria uma string escrita da maneira correta em que um arquivo .JASON é
escrito.
     String resultado = "{\"umidade\":" + umidade + ", \"solo\":\"" + solo + "\",
\"segundos\":" + String(segundos) + "}";
     // utiliza o protocolo de permissão de troca de informações entre site
chamado Cross-Origin Resource Sharing (CORS)
// ele permite que o site html possa puxar informações do site criado pelo esp,
Como foi colocado um "*" no parametro, ele permitira que qualquer um possa
puxar as informações do ESP32.
     server.sendHeader("Access-Control-Allow-Origin", "*");
```

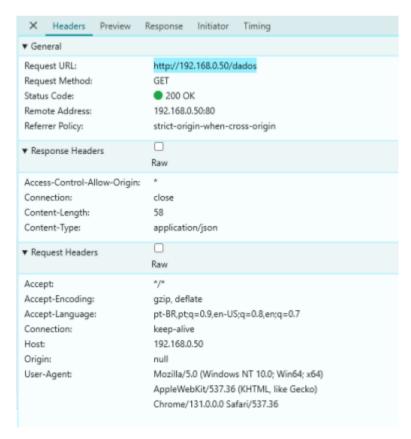
/*aqui ele enviará para o servidor com um código de confirmação 200, que para o HTML significa OK. Caso de erro de retorno do sinal ele retornara um 404, oque não é um ok.

tambem envia um arquivo chamado "application/json" com as informações de resultado para o servidor, tal arquivo será no futuro puxado pelo HTML. */

```
server.send(200, "application/json", resultado);
});

// inicia o servidor
server.begin();
}
```

Cliente Server Atualizando.



O Servidor Cliente esta por meio das configurações da imagem, requisitando arquivos do Servidor ESP32 e retornando um Sucesso sempre que a requisição deu certo.

Name	Status	Туре	Initiator	Size	Time
(i) dados	200	fetch	site HTML JSON JAVA SCI	181 B	277 ms
(i) dados	200	fetch	site HTML JSON JAVA SCI	181 B	288 ms
(i) dados	200	fetch	site HTML JSON JAVA SCI	181 B	295 ms
(;) dados	200	fetch	site HTML JSON JAVA SCI	181 B	299 ms
(i) dados	200	fetch	site HTML JSON JAVA SCI	181 B	311 ms
(i) dados	200	fetch	site HTML JSON JAVA SCI	181 B	308 ms
(i) dados	200	fetch	site HTML JSON JAVA SCI	181 B	218 ms
(;) dados	200	fetch	site HTML JSON JAVA SCI	181 B	328 ms
(i) dados	200	fetch	site HTML JSON JAVA SCI	181 B	335 ms
(i) dados	200	fetch	site HTML JSON JAVA SCI	181 B	343 ms

O servidor está requerendo as informações do sensor, do estado do solo e Segundos com sucesso sem dar nenhum erro de volta.