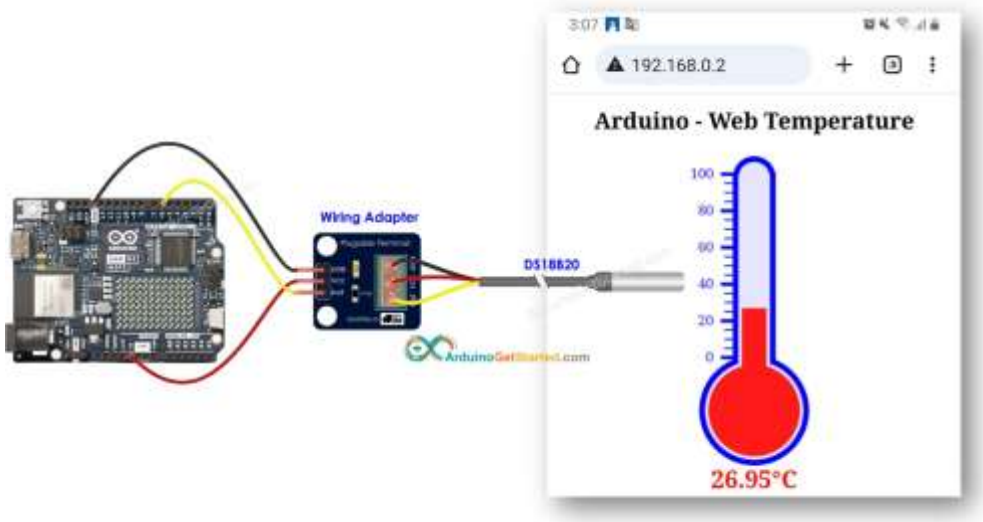
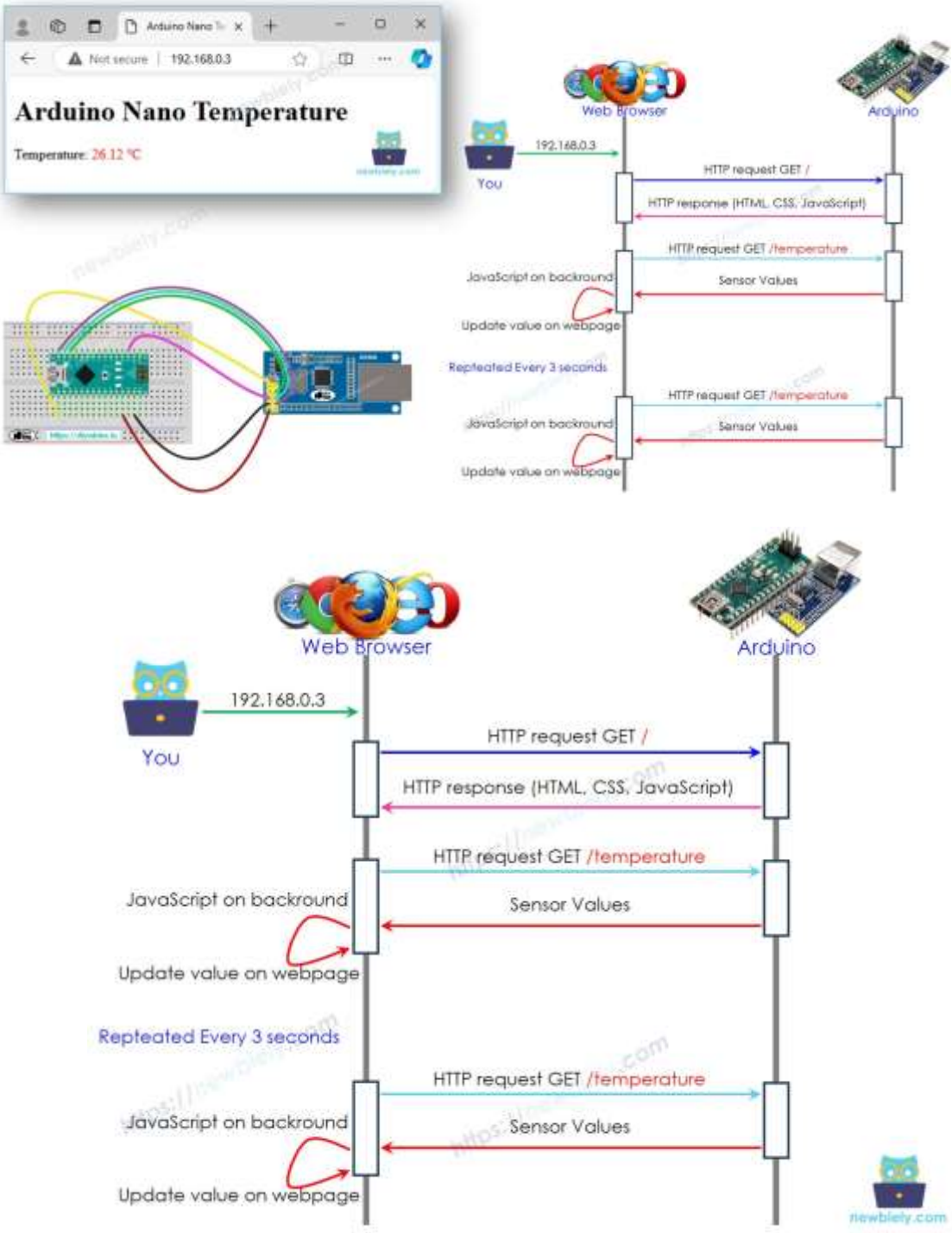
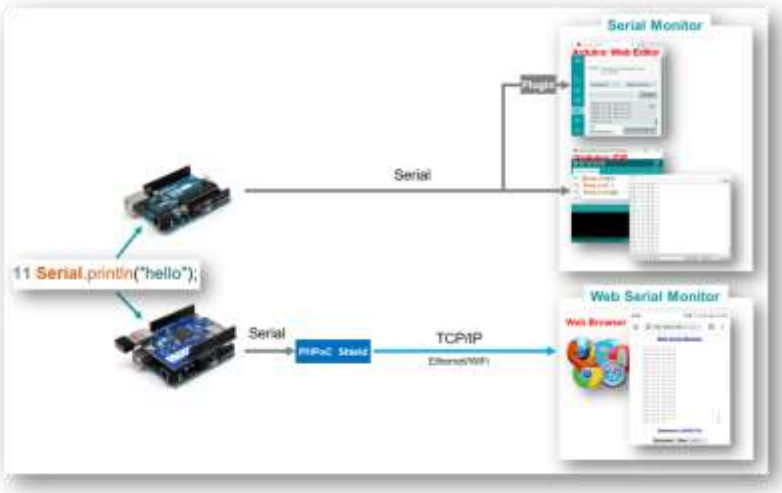
IoT_ Send Arduino Data to the Web (PHP MySQL D3.js)

Dasar :





https://newbiely.com/tutorials/arduino-nano/arduino-nano-web-server



https://forum.phpoc.com/articles/tutorials/1253-arduino-web-serial-monitor

Sending data from an Arduino to the web typically involves connecting the Arduino to the internet and then using web protocols to transmit data to a server or web page. Here are common methods and steps:

1. Hardware Setup:

Ethernet Shield:

For wired connections, connect an Ethernet shield to your Arduino and an Ethernet cable to your router.

Wi-Fi Module (e.g., ESP8266, ESP32):

For wireless connections, integrate a Wi-Fi module with your Arduino. These modules can directly handle Wi-Fi connectivity and often have enough processing power to act as a web server themselves.

2. Arduino Code:

Include Libraries:

Depending on your chosen hardware, include the necessary libraries (e.g., `Ethernet.h`, `WiFi.h`, `ESP8266WiFi.h`).

Network Configuration:

Configure your network details (SSID, password for Wi-Fi; MAC address for Ethernet).

Data Acquisition:

Read data from your sensors or other inputs connected to the Arduino.

Web Request:

HTTP GET/POST: Construct an HTTP GET or POST request to send data to a server-side script (e.g., PHP, Python) or a web API. This script can then process the data and display it on a web page or store it in a database.

JSON/XML: Format your data into a structured format like JSON or XML for easier parsing on the server.

Example (sending temperature via POST to a PHP script):

```
#include <ESP8266WiFi.h>

const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";
const char* host = "your_server_address.com";
const char* path = "/your_script.php";

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
}

void loop() {
  float temperature = analogRead(A0) * (5.0 / 1023.0) * 100; // Example sensor reading

  WiFiClient client;
  if (!client.connect(host, 80)) {
    Serial.println("Connection failed");
    return;
  }

  String postData = "temperature=" + String(temperature);

  client.print("POST ");
  client.print(path);
  client.println(" HTTP/1.1");
  client.print("Host: ");
  client.println(host);
  client.println("Content-Type: application/x-www-form-urlencoded");
  client.print("Content-Length: ");
  client.println(postData.length());
  client.println();
  client.print(postData);

  while (client.connected()) {
    String line = client.readStringUntil('\n');
    if (line == "\r") {
      break;
    }
  }
  String response = client.readString();
  Serial.println(response);
  client.stop();

  delay(5000); // Send data every 5 seconds
}
```

3. Server-Side Processing (if applicable):

Receive Data: Create a server-side script (e.g., `your_script.php`) to receive the data sent by the Arduino (e.g., using `$_POST` in PHP).

Process Data: Parse the received data, perform any necessary calculations or formatting.

Store Data: Store the data in a database (e.g., MySQL, PostgreSQL).

Generate Web Page: Dynamically generate an HTML page to display the data, or update an existing page.

4. Web Page Display:

HTML/JavaScript:

Create an HTML page to display the data. You can use JavaScript to periodically fetch data from your server-side script or API and update the web page in real-time or near real-time.

WebSockets:
For real-time updates, consider using WebSockets to establish a persistent connection between the Arduino (or an intermediary server) and the web browser.

Simplified Approach (using ESP modules as web servers):

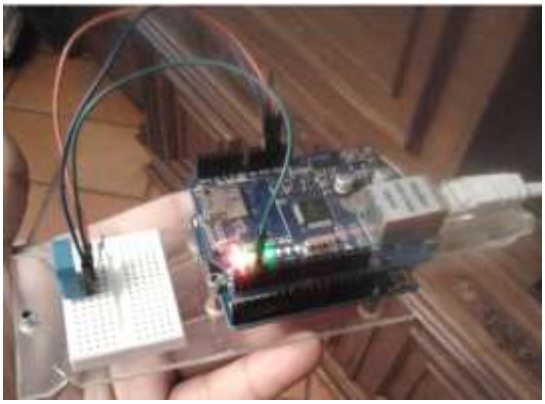
ESP8266/ESP32 as Web Server: These modules can directly host a simple web server. You can write Arduino code to serve an HTML page that displays sensor data directly from the ESP module when accessed via its IP address in a web browser. This eliminates the need for a separate server-side script and database for basic applications.

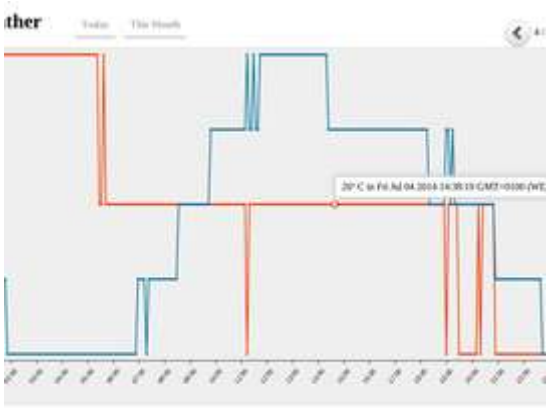
Arduino to read a sensor and send the values to the internet, to be stored in a Web Server and displayed

Arduino Web client Application: reads the sensor values and sends them to the webserver.

PHP/MySQL Application: handles the POST requests that are sent to the server and serves the pages to clients who connect

Data Visualization: The PHP application will use the Javascript Framework D3.js to display the values stored in the DB with graphics. It will allow to navigate to the past days to observe the readings





HARDWARE

- ✓ Arduino Uno
- ✓ Ethernet Shield (eBay clone)
- ✓ DHT 11 sensor
- ✓ breadboard
- ✓ 10k Ohm resistor
- ✓ USB cable
- ✓ Ethernet cable
- ✓ wires
- ✓ piece of acrylic
- ✓ PCB spacers

Software

- You need access to a web server (can be from a free hosting company) with capability to run PHP applications and also to create databases. (possibly cPanel with phpMyAdmin)

Step 1: ***Arduino Web Client + DHT11 Sensor***

```
#include <DHT.h>
#include <Ethernet.h>
#include <SPI.h>

byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x01 }; // RESERVED MAC ADDRESS
EthernetClient client;

#define DHTPIN 2 // SENSOR PIN
#define DHTTYPE DHT11 // SENSOR TYPE - THE ADAFRUIT LIBRARY OFFERS SUPPORT FOR MORE MODELS
DHT dht(DHTPIN, DHTTYPE);

long previousMillis = 0;
unsigned long currentMillis = 0;
long interval = 250000; // READING INTERVAL

int t = 0;  // TEMPERATURE VAR
int h = 0;  // HUMIDITY VAR
String data;

void setup() {
    Serial.begin(115200);

    if (Ethernet.begin(mac) == 0) {
        Serial.println("Failed to configure Ethernet using DHCP");
    }

    dht.begin();
    delay(10000); // GIVE THE SENSOR SOME TIME TO START

    h = (int) dht.readHumidity();
    t = (int) dht.readTemperature();

    data = "";
}

void loop(){

    currentMillis = millis();
    if(currentMillis - previousMillis > interval) { // READ ONLY ONCE PER INTERVAL
        previousMillis = currentMillis;
        h = (int) dht.readHumidity();
        t = (int) dht.readTemperature();
    }

    data = "temp1=" + t + "&hum1=" + h;

    if (client.connect("www.*****.*************.com",80)) { // REPLACE WITH YOUR SERVER ADDRESS
        client.println("POST /add.php HTTP/1.1");
        client.println("Host: *****.*************.com"); // SERVER ADDRESS HERE TOO
        client.println("Content-Type: application/x-www-form-urlencoded");
        client.print("Content-Length: ");
        client.println(data.length());
        client.println();
        client.print(data);
    }

    if (client.connected()) {
        client.stop();  // DISCONNECT FROM THE SERVER
    }

    delay(300000); // WAIT FIVE MINUTES BEFORE SENDING AGAIN
}
```

Step 2: PHP / MySQL Application

Temperature / Moisture Sensor reading

Timestamp	Temperature	Moisture
2025-18-03 12:10:15	27	55
2025-18-03 13:00:00	26	54
2025-18-03 14:00:00	25	53
2025-18-03 15:00:00	27	52
2025-18-03 16:00:00	25	54
2025-18-03 17:00:00	27	55
.....	...	...

In this second part i will explain briefly the PHP application and the database. The database is used obviously to store the sensor readings, so that they can be accessed later. It's a very simple DB, with just one table with 3 columns. It stores the time stamp and the corresponding temperature and humidity values.

```
CREATE TABLE tempLog (
                timeStamp TIMESTAMP NOT NULL PRIMARY KEY,
                temperature int(11) NOT NULL,
                humidity int(11) NOT NULL,
        );
```

The PHP application consists of 3 files:

- **connect.php**: this file is loaded every time we need to access to the database. It's loaded in the beginning of the almost each file. It contains a function that returns a new connection to be used by the PHP to execute query's to the DB. You need to store the **DB configs (hostname, database, user, password)** in this file.

- **add.php**: when the Arduino sends POST requests to the server, is to to this page. The PHP receives the values sent in the request and executes an insertion query with those values.

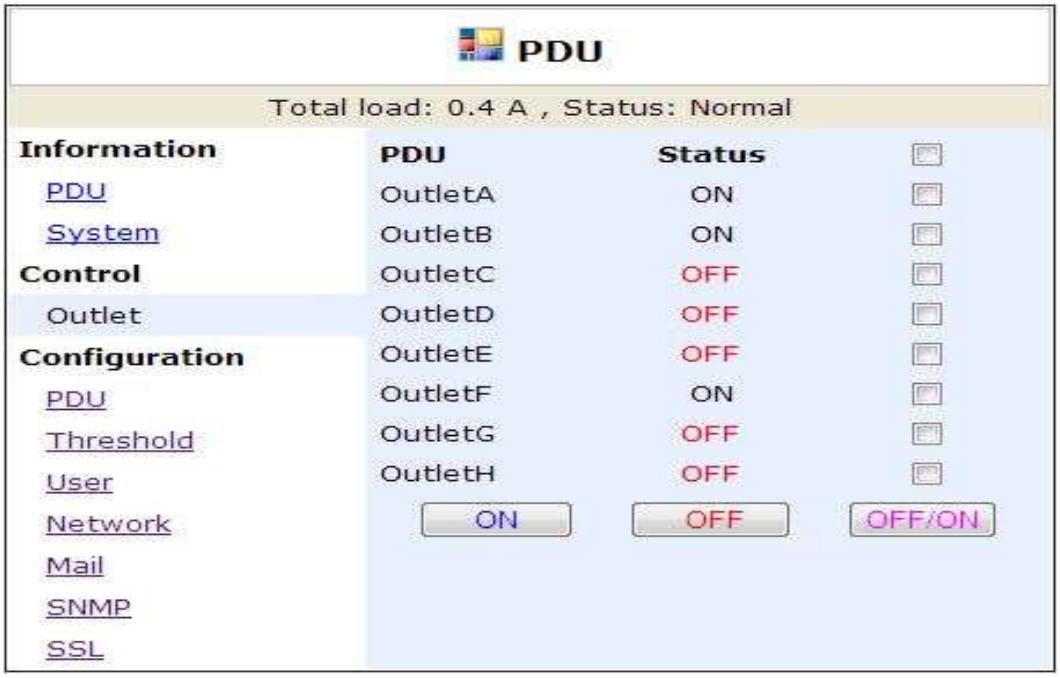
Sometimes you need to change the permissions of this file (should be 644), because it might be protected to allow only executions from the localhost.

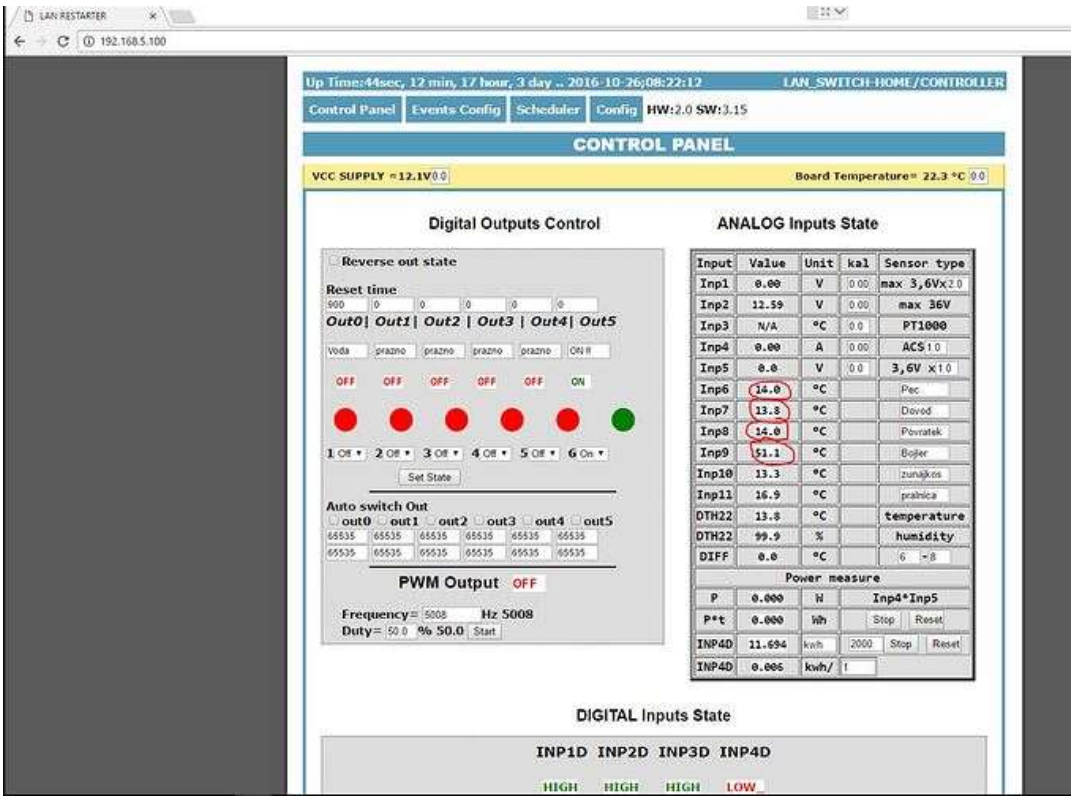
- **index.php**: this is the website landing page. It displays the values that are stored in the database. Right now, it will display all the values in a single HTML table, just to show that works.

So, this concludes the first part of this Instructable. Feel free to ask questions about anything related, i'm glad to help.

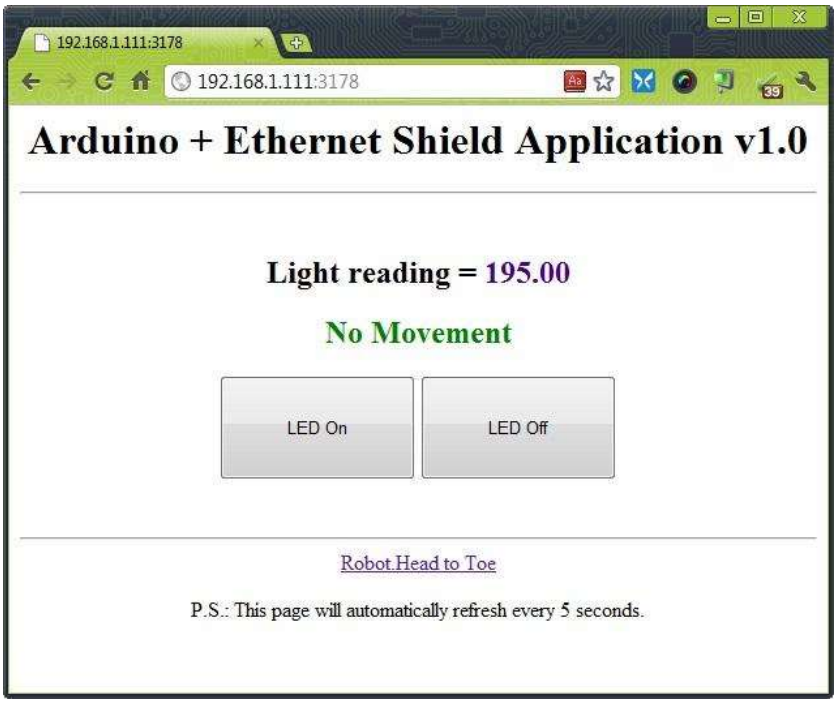
RESOURCES

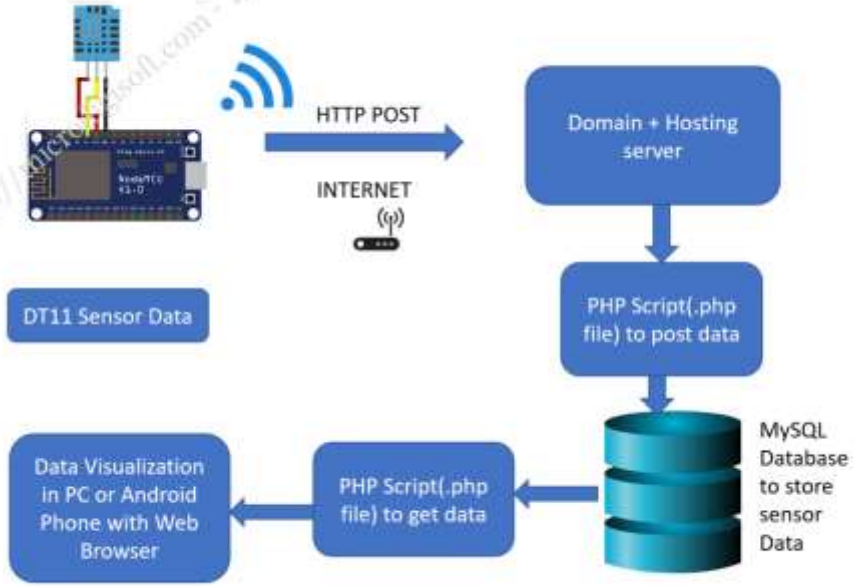
- Request Maker: This online tool is very useful to test the PHP application. You can simulate the POST requests that will be made by the Arduino and check if everything is working well.
- DHT11 sensor library from Adafruit
- Arduino Web Client
- Arduino IDE
- Arduino Ethernet


PDU
Total load: 0.4 A , Status: Normal
Information
PDU
System
Control
Outlet
Configuration
PDU
Threshold
User
Network
Mail
SNMP
SSL
PDU
Status
OutletA ON
OutletB ON
OutletC OFF
OutletD OFF
OutletE OFF
OutletF ON
OutletG OFF
OutletH OFF
ON
OFF
OFF/ON


LAN RESTARTER
192.168.5.100
Up Time:44sec, 12 min, 17 hour, 3 day .. 2016-10-26;08:22:12
LAN_SWITCH-HOME/CONTROLLER
Control Panel
Events Config
Scheduler
Config
HW:2.0 SW:3.15
CONTROL PANEL
VCC SUPPLY =12.1V
Board Temperature= 22.3 °C
Digital Outputs Control
Reverse out state
Reset time
Out0| Out1| Out2 | Out3 | Out4| Out5
Set State
Auto switch Out
PWM Output OFF
Frequency= 5008 Hz 5008
Duty= % 50.0 Start
ANALOG Inputs State
Input Value Unit kal Sensor type
Inp1 0.00 V max 3,6Vx2.0
Inp2 12.59 V max 36V
Inp3 N/A °C PT1000
Inp4 0.00 A ACS10
Inp5 0.0 V 3,6V x10
Inp6 14.0 °C Pec
Inp7 13.8 °C Dovod
Inp8 14.0 °C Povratek
Inp9 51.1 °C Bojler
Inp10 13.3 °C zunajkos
Inp11 16.9 °C pralnica
DTH22 13.8 °C temperature
DTH22 99.9 % humidity
DIFF 0.0 °C
Power measure
P 0.000 W Inp4*Inp5
P*t 0.000 Wh Stop Reset
INP4D 11.694 kwh Stop Reset
INP4D 0.006 kwh/
DIGITAL Inputs State
INP1D INP2D INP3D INP4D
HIGH HIGH HIGH LOW


File Edit View History Bookmarks Tools Help
Webserver as pin controller
172.18.67.93
Meistbesucht
My AliExpress : Mana...
Alle Boards - Die Über...
AliExpress.
Webserver as pin controller
Buttons turn pins on or off
2 On Off
3 On Off
4 On Off
5 On Off
6 On Off active
7 On Off
8 On Off
9 On Off


192.168.1.111:3178
192.168.1.111:3178
Arduino + Ethernet Shield Application v1.0
Light reading = 195.00
No Movement
LED On
LED Off
Robot Head to Toe
P.S.: This page will automatically refresh every 5 seconds.


HTTP POST
INTERNET
Domain + Hosting server
PHP Script(.php file) to post data
MySQL Database to store sensor Data
PHP Script(.php file) to get data
Data Visualization in PC or Android Phone with Web Browser
DT11 Sensor Data