Dettagli di sviluppo

Software utilizzati:

- Componente Arduino: Arduino IDE
- Componente C#: Microsoft Visual Studio 2017
- Componenti HTML/Javascript/PHP: JetBrains PHPStorm

Indice del documento:

Componente C# di lettura e scrittura su server FTP

```
/*
 *          ITT G. Fauser
 *     OpenDay - 18/01/2020
 *
 *     Progetto in Arduino: Misurazioni ambientali
 *                          Componente C#
 */

using System;
using System.IO;
using System.IO.Ports;
using System.Net;
using System.Threading;
using System.Threading.Tasks;

namespace TemperaturaUmiditaFTP_HTML
{
    class Program
    {
        // Semaforo = gestione della mutua esclusione
        static SemaphoreSlim mutex = new SemaphoreSlim(1);
        static void Main(string[] args)
        {
            // Stringa costante, indicante il laboratorio di posizionamento di Arduino
            const string LABORATORIO = "LS";
            // Valore booleano che indica la volontà di voler eliminare le misurazioni effettuate per il laboratorio corrente
            const bool CANCELLAPRECEDENTI = false;
            // Classe SerialPort, specializzata nella gestione delle stringhe in input dalle porte seriali
            SerialPort portaSeriale = new SerialPort();
            // Baud Rate pari a 9600
            portaSeriale.BaudRate = 9600;
            // Nome corrispondente alla porta scelta
            portaSeriale.PortName = "COM7";
            portaSeriale.Open();
            // Delay tra le letture delle misurazioni (500 ms). Deve essere corrispondente a quello indicato nello script di Arduino
            // Per evitare problemi di bufferizzazione
            const int RITARDO = 500;
            // Valori di luminosità e umidità che indicano il range di valori entro i quali possono spaziare le misurazioni prima di essere scritte sul server
            const int DIFFERENZAUMIDITA = 5;
            const int DIFFERENZALUMINOSITA = 10;
            // Variabili di memorizzazione delle ultime misurazioni scritte
            int umiditaPrec = -1, luminositaPrec = -1;
            double temperaturaPrec = -1, indiceCalorePrec = -1;
            // Se si è scelto di cancellare le misurazioni precedenti...
            if (CANCELLAPRECEDENTI)
            {
                // ... si crea un file da zero
                using (FileStream FS = new FileStream("misure" + LABORATORIO + ".txt", FileMode.Create))
                    FS.Close();
            }
            while (true)
            {
                try
                {
                    Console.WriteLine("Valori letti in tempo reale: ");
                    Console.WriteLine();
                    /*
                     * Esempio di riga letta:
                     * Umidita': 62.00 % Temperatura: 25.30 ??C Indice di calore: 25.50 ??C Luminosita': 59 %
                     */
                    // Lettura dei valori e parsing dei valori letti in funzione della posizione nella stringa.
                    string lettura = portaSeriale.ReadLine().Replace("??", " °");
                    int umidita = Convert.ToInt32(Convert.ToDouble(lettura.Split(' ')[1].Substring(0, lettura.Split(' ')[1].Length - 2).Replace(".", ",")));
                    double temperatura = Convert.ToDouble(lettura.Split(' ')[4].Replace(".", ","));
                    double indiceCalore = Convert.ToDouble(lettura.Split(' ')[10].Replace(".", ","));
                    int luminosita = Convert.ToInt32(lettura.Split(' ')[14].Replace(".", ","));
                    // Stampa delle misurazioni in tabella.
                    Console.WriteLine("=========================================================");
                    Console.WriteLine(" UMIDITA'    TEMPERATURA    INDICE DI CALORE   LUMINOSITA'");
                    Console.WriteLine("=========================================================");
                    Console.WriteLine("{0,5} {1,15} {2,16} {3,14}", umidita, temperatura.ToString("#.00"), indiceCalore.ToString("#.00"), luminosita);
                    Console.WriteLine("=========================================================");
                    // Accedo ad una risorsa condivisa. Lo faccio utilizzando la mutua esclusione
                    mutex.Wait();
                    // Utilizzo delle classi FileStream e StreamWriter per la memorizzazione delle misurazioni su file di testo
                    using (FileStream FS = new FileStream("misure" + LABORATORIO + ".txt", FileMode.Append))
                    using (StreamWriter SW = new StreamWriter(FS))
                    {
                        // Se vi è una differenza (in valore assoluto), pari ad almeno il tetto prefissato...
                        if (Math.Abs(umidita - umiditaPrec) >= DIFFERENZAUMIDITA || Math.Abs(luminosita - luminositaPrec) >= DIFFERENZALUMINOSITA)
                        {
                            // ... si procede con la scrittura su file
                            SW.Write(String.Format("{0};{1};{2};" + luminosita, umidita, temperatura.ToString("#.00"),
indiceCalore.ToString("#.00")).Replace(",", "."));
                            SW.WriteLine();
                            SW.Close();
                            umiditaPrec = umidita;
                            temperaturaPrec = temperatura;
                            indiceCalorePrec = indiceCalore;
                            luminositaPrec = luminosita;
                            Console.WriteLine();
                            Console.WriteLine("Rilevata nuova misura valida. Avvio aggiornamento contatori su server...");
                            // Delego ad un nuovo thread generato all'istante il compito di gestire la connessione TCP/FTP, per non creare problemi di
tempistiche di lettura
                            Task.Factory.StartNew(() =>
                            {
                                // Accedo al file solo se non è in uso da parte di altri thread, come il Main Thread
                                mutex.Wait();
                                scriviFTP1(LABORATORIO);
                                mutex.Release();
                            });
                        }
                        else
                        {
                            Console.WriteLine();
                            Console.WriteLine("Valori poco dissimili dai precedenti. Inizio nuova rilettura...");
                        }
```

```
                }
                // Rilascio il semaforo: la risorsa condivisa (file) non è più in uso
                mutex.Release();
                Thread.Sleep(RITARDO);
                Console.Clear();
            }
            catch (Exception)
            {
                Console.WriteLine("Valore invalido. Scartato.");
            }
        }
    }
    // Routine di scrittura su Server FTP
    static void scriviFTP1(string LAB)
    {
        try
        {
            // Creazione di un'istanza di classe WebClient (che sa gestire le connessioni TCP con protocollo FTP)
            using (var client = new WebClient())
            {
                // Credenziali di accesso al server
                client.Credentials = new NetworkCredential("nome-utente", "password");
                // Si suppone la root directory di FTP sia anche la cartella di upload del file
                client.UploadFile("ftp://URLSERVER:PORTA/misure" + LAB + ".txt", WebRequestMethods.Ftp.UploadFile, "misure" + LAB + ".txt");
            }
        }
        catch (Exception e)
        {
            Console.WriteLine("Errore FTP: " + e);
        }
    }
  }
}
```

Pagina Web HTML/Javascript/PHP (“misurazioni.php”)

```
<?php
// Richiesta di inserimento nella richiesta (GET o POST) di uno o due Laboratori validi per la visualizzazione delle misurazioni
$LABORATORIO = "LS";
if (empty($_GET["lab"]) == false) {
    $labUtente = $_GET["lab"];
    // Validazione della richiesta
    if ($labUtente == "LS" || $labUtente == "LM" || $labUtente == "LI" || $labUtente == "LE" || $labUtente == "MB" || $labUtente == "N3") {
        $LABORATORIO = $labUtente;
    } else {
        echo "<p>Laboratorio indicato non valido. Riprova.</p>";
        return;
    }
}
try {
    if (file_exists("misure" . $LABORATORIO . ".txt")) {
        $misureFile = fopen("misure" . $LABORATORIO . ".txt", "r");
        $misure = "";
        while (!feof($misureFile)) {
            $linea = fgets($misureFile);
            if (empty($linea) == false)
                $misure .= $linea;
        }
    } else {
        echo "<p>Laboratorio non abilitato o misure inesistenti. Riprova.</p>";
        return;
    }
} catch (Exception $e) {

}
?>
<!-- Sostituzione con la variabile "LABORATORIO" nelle zone in cui viene indicato -->
<!-- Laboratorio <?php echo $LABORATORIO ?> -->
<!DOCTYPE html>
<html lang="it">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="Cache-Control" content="no-cache, no-store, must-revalidate"/>
        <meta http-equiv="Pragma" content="no-cache"/>
        <meta http-equiv="Expires" content="0"/>
        <title>Laboratorio <?php echo $LABORATORIO ?> - Progetto in Arduino - ITT G. Fauser - OpenDay 18 gennaio
            2020</title>
        <link rel="apple-touch-icon" sizes="180x180" href="apple-touch-icon.png">
        <link rel="icon" type="image/png" sizes="32x32" href="favicon-32x32.png">
        <link rel="icon" type="image/png" sizes="16x16" href="favicon-16x16.png">
        <link rel="manifest" href="site.webmanifest">
        <link rel="mask-icon" href="safari-pinned-tab.svg" color="#5bbad5">
        <meta name="msapplication-TileColor" content="#ffc40d">
        <meta name="theme-color" content="#ffffff">
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
        <style type="text/css">
            /* Classi di definizione dello stile della pagina */
            .table {
                width: auto !important;
                background: #FFFFFF;
            }

            table.tabella th {
                background-color: #5289D6;
                color: white;
            }

            body {
                background: white;
            }
        </style>
        <script type="text/javascript">
            // Funzione non più adottata: utilizzata per ricaricare la pagina "manualmente" ogni 10 secondi.
            /* setTimeout(function () {
                location.reload();
            }, 10000); */

            function avviaAggiornamento() {
                var placeholderMisure = document.getElementById("placeholder").innerText.replace("<br/>", "");
                // Se PHP non è riuscito a leggere il file, se ne occupa Javascript
                if (placeholderMisure.length == 0) {
                    document.getElementById("sottotitolo").innerHTML = "Aggiornamento in corso...";
                    const URL = "misure<?=$LABORATORIO?>.txt";
                    var filehttp = new XMLHttpRequest();
                    // Richiesta "GET" del file
                    filehttp.open("GET", URL, true);
                    filehttp.send();
                    filehttp.onreadystatechange = function () {
                        // Se ricevo Status Code 200
                        if (this.readyState == 4 && this.status == 200) {
                            Main(tornaArrayMisure(filehttp.responseText));
                            document.getElementById("sottotitolo").innerHTML = "";
                        }
                    };
                } else {
                    // Altrimenti, passo direttamente alla funzione "Main" i parametri fondamentali per iniziare
                    Main(tornaArrayMisure(placeholderMisure));
                    document.getElementById("sottotitolo").innerHTML = "";
                }
            }

            // Funzione che restituisce l'array completo delle misurazioni (stringhe), divise per "new line"
            function tornaArrayMisure(mis) {
                return mis.split('\n');
            }

            // Funzione che restituisce la singola riga, effettuando lo split del file CSV
            function tornaSingolaRiga(mis, pos) {
                return mis[pos].split(';');
            }

            // Funzione che restituisce l'ultima riga, effettuando lo split del file CSV
            function tornaUltimaMisura(arrayMis, maxCiclo) {
                return tornaSingolaRiga(arrayMis, maxCiclo - 1);
            }

            function Main(arrayMisure) {
                var tabellaStr = "";
                var ultimaMisura;
                // Generazione della tabella nella variabile "tabellaStr"
                tabellaStr += '<table class="table table-bordered table-sm shadow tabella">';
                tabellaStr += '<thead><tr><th></th><th scope="col">  Umidit&agrave;  </th><th
scope="col">  Temperatura  </th><th scope="col">  Indice di calore  </th><th
scope="col">  Luminosit&agrave;  </th></tr></thead><tbody>';
```

```
                for (var i = arrayMisure.length - 2; i >= 0; i--) {
                    var riga = tornaSingolaRiga(arrayMisure, i);
                    tabellaStr += '<tr class="text-center">';
                    tabellaStr += "<th> " + (arrayMisure.length - 2 - i + 1) + " </th>";
                    tabellaStr += "<td>" + riga[0] + " &percnt;</td>";
                    tabellaStr += "<td>" + riga[1] + " &deg;C</td>";
                    tabellaStr += "<td>" + riga[2] + " &deg;C</td>";
                    tabellaStr += "<td>" + riga[3] + " &percnt;</td>";
                    tabellaStr += "</tr>";
                }
                tabellaStr += "</tbody></table>";
                // Scrittura nella pagina della tabella
                document.getElementById("tabella").innerHTML = tabellaStr;
                // Analisi dell'ultima Umidità e Luminosità
                ultimaMisura = tornaUltimaMisura(arrayMisure, arrayMisure.length - 1);
                var ultimaLuminosita = parseInt(ultimaMisura.slice(3, 4));
                ultimaMisura = parseInt(ultimaMisura.slice(0, 1));
                // In base al Range della misurazione, si adotta colore, frase e immagine adeguata
                if (ultimaMisura >= 40 && ultimaMisura <= 65) {
                    document.getElementById("immagine").innerHTML = '<img src="good1.png" height="400" width="400">';
                    document.getElementById("condizione").innerHTML = '<p> Umidit&#224; ideale! </p>';
                    document.getElementById("condizione").style.color = "#339966";
                } else if (ultimaMisura > 65 && ultimaMisura <= 70) {
                    document.getElementById("immagine").innerHTML = '<img src="good2.png" height="400" width="400">';
                    document.getElementById("condizione").innerHTML = '<p> L&#8217;umidit&#224; si sta alzando! </p>';
                    document.getElementById("condizione").style.color = "#FF8855";
                } else if (ultimaMisura > 70 && ultimaMisura <= 80) {
                    document.getElementById("immagine").innerHTML = '<img src="good3.png" height="400" width="400">';
                    document.getElementById("condizione").innerHTML = '<p> Sta iniziando a fare caldo! </p>';
                    document.getElementById("condizione").style.color = "#FF8855";
                } else if (ultimaMisura > 80) {
                    document.getElementById("immagine").innerHTML = '<img src="umido.png" height="400" width="400">';
                    document.getElementById("condizione").innerHTML = '<p> Troppo Umido! </p>';
                    document.getElementById("condizione").style.color = "#FF7070";
                } else if (ultimaMisura < 40) {
                    document.getElementById("immagine").innerHTML = '<img src="secco.png" height="400" width="400">';
                    document.getElementById("condizione").innerHTML = '<p> Troppo Secco! </p>';
                    document.getElementById("condizione").style.color = "#5289D6";
                }
                // Scrittura dell'ultima misurazione
                document.getElementById("ultimaUmidita").innerHTML = "<p> L'ultimo valore di umidit&#224; misurato &#232; pari al " + ultimaMisura + " %.</p>";
                document.getElementById("ultimaLuminosita").innerHTML = "<p> L'ultimo valore di lumonosit&#224; misurato &#232; pari al " + ultimaLuminosita +
" %.</p>";
                // Variazione del background del documento in funzione della luminosità
                if (ultimaLuminosita >= 55) {
                    document.body.style.backgroundColor = "white";
                    document.getElementById("scritte").style.color = "#000000";
                    document.getElementById("ultimaUmidita").style.color = "#000000";
                    document.getElementById("ultimaLuminosita").style.color = "#000000";
                } else if (ultimaLuminosita >= 30) {
                    document.body.style.background = "#A7A69D";
                    document.getElementById("scritte").style.color = "#FFFFFF";
                    document.getElementById("ultimaUmidita").style.color = "#FFFFFF";
                    document.getElementById("ultimaLuminosita").style.color = "#FFFFFF";
                } else {
                    document.body.style.backgroundColor = "#6A6C6E";
                    document.getElementById("scritte").style.color = "#FFFFFF";
                    document.getElementById("ultimaUmidita").style.color = "#FFFFFF";
                    document.getElementById("ultimaLuminosita").style.color = "#FFFFFF";
                }
            }
        </script>
    </head>
    <body onload="avviaAggiornamento()">
        <div id="placeholder" style="display: none;"><?php echo $misure ?></div>
        <div id="scritte">
            <h2 style="display:flex;justify-content: center;padding-top:20px">
                ITT G. Fauser - OpenDay 18 gennaio 2020
            </h2>
            <h5 style="display:flex;justify-content: center;" id="h5">
                Progetto in Arduino: Misurazioni dei parametri ambientali
            </h5>
            <h2 style="display:flex;justify-content: center;">Laboratorio <?php echo $LABORATORIO ?></h2>
            <h4 style="display:flex;justify-content: center;" id="sottotitolo">
                Lettura dati in corso... Attendi...
            </h4>
        </div>
        <div class="container">
            <h1 id="condizione" class="row align-items-center justify-content-center"></h1>
            <h5 id="ultimaUmidita" class="row align-items-center justify-content-center"></h5>
            <h5 id="ultimaLuminosita" class="row align-items-center justify-content-center"></h5>
            <div id="immagine" class="row align-items-center justify-content-center"></div>
            <div id="tabella" class="row align-items-center justify-content-center"></div>
        </div>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
        <script type="text/javascript">
            // Funzione JQuerry di aggiornamento dei dati e aggiornamento dinamico (ogni 2 secondi), per evitare il ricaricamento della pagina
            $(document).ready(function () {
                function aggiornaDati() {
                    $.ajax({
                        type: 'GET', // Richiesta GET allo script "richiedi.php"
                        url: 'richiedi.php?nomeFile=misure<?=$LABORATORIO?>.txt',
                        success: function (data) {
                            $('#placeholder').html(data);
                            var placeholderMisure = document.getElementById("placeholder").innerText.replace("<br/>", "");
                            Main(tornaArrayMisure(placeholderMisure));
                        }
                    });
                }

                aggiornaDati();
                setInterval(function () {
                    aggiornaDati();
                }, 2000);
            });
        </script>
    </body>
</html>
```

Componente di indicizzazione (“arduino.php”)

```php
<?php
// Richiesta di inserimento nella richiesta (GET o POST) di uno o due Laboratori validi per la visualizzazione delle misurazioni
$LAB1 = "LS";
$LAB2 = "";
if (empty($_GET["lab1"]) == false) {
    $labUtente = $_GET["lab1"];
    // Validazione della richiesta
    if ($labUtente == "LS" || $labUtente == "LM" || $labUtente == "LI" || $labUtente == "LE" || $labUtente == "MB" || $labUtente ==
"N3") {
        $LAB1 = $labUtente;
    } else {
        echo "<p>Laboratorio 1 indicato non valido. Riprova.</p>";
        return;
    }
} else {
    echo "<p>Indicare almeno un laboratorio valido. Riprova.</p>";
    return;
}
if (empty($_GET["lab2"]) == false) {
    $labUtente = $_GET["lab2"];
    if (($labUtente == "LS" || $labUtente == "LM" || $labUtente == "LI" || $labUtente == "LE" || $labUtente == "MB" || $labUtente ==
"N3") && $labUtente != $LAB1) {
        $LAB2 = $labUtente;
    } else {
        echo "<p>Laboratorio 2 indicato non valido. Riprova.</p>";
        return;
    }
} else
    $LAB2 = false;
?>
<!-- Componente di indicizzazione -->
<!DOCTYPE html>
<html lang="it">
    <head>
        <meta charset="UTF-8">
        <title>Progetto in Arduino - ITT G. Fauser - OpenDay 18 gennaio 2020</title>
        <link rel="apple-touch-icon" sizes="180x180" href="/apple-touch-icon.png">
        <link rel="icon" type="image/png" sizes="32x32" href="/favicon-32x32.png">
        <link rel="icon" type="image/png" sizes="16x16" href="/favicon-16x16.png">
        <link rel="manifest" href="/site.webmanifest">
        <link rel="mask-icon" href="/safari-pinned-tab.svg" color="#5bbad5">
        <meta name="msapplication-TileColor" content="#ffc40d">
        <meta name="theme-color" content="#ffffff">
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
    </head>
    <body>
        <h3 class="row align-items-center justify-content-center" style="padding-top: 40px; padding-bottom: 40px;">
            Scegli il Laboratorio di cui visualizzare le misurazioni:
        </h3>
        <!-- Generazione dinamica della tabella con le immagini rappresentanti i laboratori indicati -->
        <table border="0" class="row align-items-center justify-content-center">
            <tr>
                <td>
                    <a href="misurazioni.php?lab=<?php echo $LAB1 ?>">
                        <img src="<?php echo $LAB1 ?>.jpg" alt="<?php echo $LAB1 ?>" height="300" width="300"/>
                    </a>
                    <?php
                    if ($LAB2 != false) {
                        echo
"                        
";
                    }
                    ?>
                </td>
                <?php
                if ($LAB2 != false) {
                    echo "<td>";
                    echo
"                        
";
                    echo "<a href=\"misurazioni.php?lab=" . $LAB2 . "\">";
                    echo "<img src=\"" . $LAB2 . ".jpg\" alt=\"" . $LAB2 . "\" height=\"300\" width=\"300\"/>";
                    echo "</a>";
                    echo "</td>";
                }
                ?>
            </tr>
        </table>
        <!-- Immagine rappresentante il meccanismo di funzionamento -->
        <h3 style="display: flex; justify-content: center">
            Schema del meccanismo di funzionamento:
        </h3>
        <figure style="display: flex; justify-content: center;">
            <img src="schema.png" alt="Schema del meccanismo di funzionamento" height="50%" width="50%"/>
        </figure>
        <!-- Link alla documentazione -->
        <div style="display: flex; justify-content: center; padding-top: 50px; padding-bottom: 50px;">
            <a href="https://github.com/arduinofauser2019/Documentazione-Progetto">Documentazione del progetto</a>
        </div>
    </body>
</html>
```

Componente di richiesta/aggiornamento dati (“richiedi.php”)

```php
<?php
try {
    $misure = "";
    // Viene restituito il file richiesto, se non contiene caratteri "/" (per
evitare tentativi di accesso ad altre cartelle)
    if (file_exists($_REQUEST["nomeFile"]) && (strpos($_REQUEST["nomeFile"],
'/') !== false) == false) {
        $misureFile = fopen($_REQUEST["nomeFile"], "r");
        while (!feof($misureFile)) {
            $linea = fgets($misureFile);
            if (empty($linea) == false)
                $misure .= $linea;
        }
    } else {
        echo "<p>File non specificato o inesistente.</p>";
        return;
    }
} catch (Exception $e) {

}
echo $misure;
```

Componente Arduino di rilevazione misurazioni

```
#include <DHT.h>
#include <DHT_U.h>

#include <Adafruit_Sensor.h>

/* ATTENZIONE: Sono richieste le seguenti librerie Arduino:
 * DHT Sensor Library: https://github.com/adafruit/DHT-sensor-library
 * Adafruit Unified Sensor Lib: https://github.com/adafruit/Adafruit_Sensor
 */

#include "DHT.h"

#define DHTPIN 2     // PIN digitale connesso al PIN "dati" del Sensore DHT 11

#define PHOTOSENSPIN A2

#define DHTTYPE DHT11 // Definizione tipo di sensore utilizzato
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  pinMode(PHOTOSENSPIN, INPUT); // Definizione del PIN di input per il sensore di luminosità.
  Serial.begin(9600); // Baud Rate pari a 9600
  dht.begin(); // Inizializzazione lettura
}

void loop() {
  delay(500); // Attesa di 500 ms tra le misurazioni
  /*
   * NB: La lettura delle misure impiega almeno 250 ms e può richiedere anche 2 secondi, per via di problemi di bufferizzazione.
   */

  float umidita = dht.readHumidity(); // Lettura Umidità (metodo già pronto)
  float temperatura = dht.readTemperature(); // Lettura Temperatura (metodo già pronto, gradi Celsius)

  // Se la lettura è fallita...
  if (isnan(umidita) || isnan(temperatura)) {
    Serial.println(F("LETTURA FALLITA!")); // ...lo indico sulla porta seriale
    return;
  }

  // Calcola l'indice di calore in Gradi Celsius
  float indiceCalore = dht.computeHeatIndex(temperatura, umidita, false);

  int luminosita = 100 - (analogRead(PHOTOSENSPIN)/6);

  Serial.print(F("Umidita': "));
  Serial.print(umidita);
  Serial.print(F(" % Temperatura: "));
  Serial.print(temperatura);
  Serial.print(F(" °C "));
  Serial.print(F("Indice di calore: "));
```

```
  Serial.print(indiceCalore);
  Serial.print(F(" °C "));
  Serial.print(F("Luminosita': "));
  Serial.print(luminosita);
  Serial.println(F(" %"));
}
```