# Laboratorium 4

# Interfejs webowy do komunikacji z systemem wbudowanym

**Zakres:** Laboratorium obrazuje podstawy komunikacji z systemem wbudowanym za pomocą interfejsu webowego.
**Zasady zaliczenia:**
zadanie 1 – 20%; Utworzenie serwera WWW z wykorzystaniem Arduino IDE
zadanie 2 – 20%; Utworzenie serwera WWW z poziomu systemy Linux.
zadanie 3 – 30%; Zadanie do samodzielnej realizacji Arduin IDE.
zadanie 4 – 30%; Zadanie do samodzielnej realizacji Linux.

## 1. Utworzenie serwera WWW z wykorzystaniem Arduino IDE

Do obsługi Ethernetu została stworzona biblioteka (Ethernet library) umożliwiająca w prosty sposób zestawienie połączenia i postawienie np. serwera www wyświetlającego stronę zapisaną w przykładowym kodzie źródłowym. Zapoznaj się z przykładowymi tutorialami:

http://startingelectronics.org/tutorials/arduino/ethernet-shield-web-server-tutorial

http://startingelectronics.org/articles/arduino/switch-and-web-page-button-LED-control

Do inicjalizacji serwera w kodzie używamy metody Ethernet.begin(). Wymaga ona konfiguracji MAC adres i IP. Należy je odczytać wcześniej np. z poziomu linuxa. Dodatkowo należy skonfigurować port na którym będzie nasłuchiwał serwer. W tym momencie możliwe jest już wysyłanie standardowych pakietów http:

```
#include <Ethernet.h>

// MAC ADRES
byte mac[] = {0xEA,0xFE,0xED,0xED,0xAB,0xCD};
// IP: 192.168.0.100
byte ip[] = {192,168,0,100};
EthernetServer server(80);

void setup()
{
   Ethernet.begin(mac, ip);
   server.begin();
   Serial.begin(9600);
   Serial.print("Serwer wystartowal z IP: ");
   Serial.println(Ethernet.localIP());
}

void loop ()
```

```
{
   EthernetClient client = server.available();
   if (client)
   {
     while (client.connected())
     {
       if (client.available())
      {
            // Wysyla standardowy naglowek HTTP
            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
            client.println("Connection: close");

         // Standardowy kod HTML
            client.println("<!DOCTYPE HTML>");
            client.println("<html><head><title>Serwer na
Arduino</title></head>");
            client.println("<body><h1>Witaj Swiecie</h1></body></html>");
            break;
        }
     }
   }
     // konczymy polaczenie
     client.stop();
}
```

## 2. Utworzenie serwera WWW z poziomu systemy Linux.

W systemie Linux jest już zainstalowany pakiet Node.js. Node.js jest platformą, która pozwala uruchomić kod JavaScript. Node.js oferuje on kilkadziesiąt modułów, dzięki którym można zbudować bardzo ciekawe aplikacje. Jest między innymi moduł HTTP, dzięki któremu można utworzyć serwer www. Dostępny jest też moduł File System, dający dostęp do systemu plików.
Dzięki globalnej bibliotece modułów **npm**, każdy może łatwo zainstalować niestandardowe moduły np. poprzez **npm install *nazwaModulu***, jak i również dodać swoje produkcje do niej.
W ramach tego laboratorium do tworzenia aplikacji potrzebne będą moduły socket.io, express i now. W celu ich instalacji wykonaj polecenie:

```
npm install socket.io
npm install now
npm install express
```

Zapoznaj się z dokumentacją:
http://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm
http://socket.io/

https://zeit.co/now#whats-now

Po zainstalowaniu tych pakietów przykładowy skrypt serwera server.js może wyglądać następująco:

```
var express = require('express');
var app = express();

// This responds with "Hello World" on the homepage
app.get('/', function (req, res) {
   console.log("Got a GET request for the homepage");
   res.send('Hello GET');
})


// This responds a POST request for the homepage
app.post('/', function (req, res) {
   console.log("Got a POST request for the homepage");
   res.send('Hello POST');
})

// This responds a DELETE request for the /del_user page.
app.delete('/del_user', function (req, res) {
   console.log("Got a DELETE request for /del_user");
   res.send('Hello DELETE');
})

// This responds a GET request for the /list_user page.
app.get('/list_user', function (req, res) {
   console.log("Got a GET request for /list_user");
   res.send('Page Listing');
})

// This responds a GET request for abcd, abxcd, ab123cd, and so on
app.get('/ab*cd', function(req, res) {
   console.log("Got a GET request for /ab*cd");
   res.send('Page Pattern Match');
})


var server = app.listen(8081, function () {

  var host = server.address().address
  var port = server.address().port

  console.log("Example app listening at http://%s:%s", host, port)
```

```
})
```

W tym momencie możemy uruchomić serwer poprzez polecenie:

```
node server.js
```

Ważne jest aby pamiętać że na Galileo działa serwer WWW i nasłuchuje na porcie 80. W przypadku serwera node w przykładzie zdefiniowano wykorzystanie portu 8081, i należy ten port podawać przy otwieraniu strony:

```
http://192.168.137.2:8081/
```

## 3. Zadanie do samodzielnej realizacji Arduino IDE

Napisz aplikację sterującą przełączaniem przekaźnika, jasnością diody Led za pomocą PWM, wyświetlając status przycisku oraz bieżącą temperaturę i natężenie światła.

## 4. Zadanie do samodzielnej realizacji Node.js

Napisz aplikację wyświetlając status przycisku oraz bieżącą temperaturę i natężenie światła , a także sterującą przełączaniem przekaźnika, wychyleniem serwomechanizmu za pomocą kontrolki jQuery,: http://anthonyterrien.com/demo/knob/

Do dostępu do urządzeń wykorzystaj funkcje systemu plików:

```
fs.writeFile('/sys/class/gpio/export', '18', 'utf-8', function (err) {if (err) console.log('Port 18 already exported.');});
```