Wireless Control of Lights

Independent Project in Electrical Engineering

Andreas Eckerblad
Sebastian From
Abstract

**Wireless Control of Lights**

*Andreas Eckerblad, Sebastian From*

Wireless control of lights is a 15HP project carried out at Uppsala University. Wireless control is used everywhere nowadays and the goal of this project is to create a prototype that can control the light intensity of an single phase lamp using wireless communication. Using this prototype, a solution of wireless control that can be applied for larger scale applications will be presented. By larger scale in this cases means more than the single lamp used for the prototype. The project uses WiFi and a server solution to communicate between the control unit and the light appliance. A small circuit was made to make it possible to control the grid voltage of 230 volt AC without damaging the microcontroller. The solution presented by this project works as intended. This solution can be modified to suit many different appliances, and not only control of lamps.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC</td>
<td>Direct Current</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IOS</td>
<td>iPhone Operating System</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IR</td>
<td>Infrared Radiation</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>VAC</td>
<td>Volts of Alternating Current</td>
</tr>
<tr>
<td>MOSFET</td>
<td>Metal Oxide Semiconductor Field Effect Transistor</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>SSID</td>
<td>Service Set Identifier</td>
</tr>
<tr>
<td>TV</td>
<td>Television</td>
</tr>
<tr>
<td>WiFi</td>
<td>Wireless Fidelity</td>
</tr>
</tbody>
</table>
Contents

1 Introduction ................................................................. 1
   1.1 Background .......................................................... 1
   1.2 Purpose ............................................................. 1
   1.3 Goal ................................................................. 1
   1.4 Boundaries .......................................................... 1
   1.5 Design .............................................................. 1

2 Theory ........................................................................... 3
   2.1 Microcontroller ......................................................... 3
   2.2 PWM signals .......................................................... 3
   2.3 Dimmer circuit ........................................................ 4
   2.4 Wireless communication .......................................... 4
       2.4.1 Radio Waves ...................................................... 4
       2.4.2 Bluetooth ......................................................... 5
       2.4.3 Wireless Fidelity ............................................... 6
       2.4.4 Infrared ........................................................... 6

3 Method and design .......................................................... 7
   3.1 Components .......................................................... 7
   3.2 Implementation ....................................................... 7
       3.2.1 Setting up a server ............................................ 7
       3.2.2 Server communication ...................................... 7
   3.3 Workspace and system setup .................................... 8

4 Results and discussion ...................................................... 9
   4.1 Wireless communication .......................................... 9
       4.1.1 Bluetooth solution ............................................ 11
       4.1.2 Infrared solution ............................................. 11
       4.1.3 Radio wave solution ......................................... 11
   4.2 Implementation for other appliances ............................. 11
   4.3 User interface ........................................................ 12
       4.3.1 Android app ..................................................... 12
       4.3.2 Blynk ............................................................. 12

5 Conclusion ...................................................................... 13

6 References ..................................................................... 14

Appendices ....................................................................... 15

A Arduino code .................................................................. 15
1 Introduction

1.1 Background
This 15HP project is carried out at Uppsala University in Uppsala, Sweden, for the course Independent Project in Electrical Engineering. The course will prepare students for working in projects within an area of electrical engineering. This project is about wireless control of electrical systems, specifically wireless control of lights connected to a single phase in a wall outlet.

1.2 Purpose
Today wireless control is becoming more and more popular. Controlling everything at home wirelessly is constantly being pushed by companies. The purpose of this project is to design a system to wireless communicate with and control lighting appliances.

1.3 Goal
The goal of this project is to create a prototype consisting of a small lighting appliance, in the form of a light bulb with corresponding circuit, which is wirelessly controlled using a smartphone. This means that single phase 230 VAC from the wall outlet has to be controlled wirelessly. Using this design, a solution for future wireless control of appliances will be suggested.

1.4 Boundaries
This project only contains work on single phase lighting appliances. This means three phase appliances can not be included in the conclusions and results of this project. Due to the lack of knowledge of smartphone programming, the wireless control will be limited to what can be done with the limited time frame.

1.5 Design
The circuit design (figure 1) was chosen due to stability and well tested proof of concept. The circuit can be said to consist of three parts. The Arduino (section 2.1) which controls when the bigger circuit leads using PWM signals (section 2.2), the rectifier which output is controlled by the Arduino and itself controls the MOSFET which in turn is what will actually dim the 230 VAC signal (section 2.3).

The wireless communication for an Arduino device can be made using several technologies. One of the challenges of this project is to choose a fitting technology for the wireless communication, where allowances for each of the technologies pros and cons.
Figure 1: Ton Giesberts AC dimmer circuit configured to work with Arduino.
2 Theory

2.1 Microcontroller

A microcontroller is a system in a single integrated circuit that can be used as an embedded system to perform a specific task. It contains memory, programmable peripherals and a processor. Today, most microcontrollers are used in wide range of applications, such as machinery, consumer products and office products.

Microcontrollers are designed to be low cost, low power consumption and have a wide range of use. This makes microcontrollers very economical for control of electronic devices.[1]

An example of a programmable environment using microcontrollers is Arduino. Arduino is an open source electronics platform that focuses on easy-to-use hardware and software. They develop single-board microcontrollers and microcontroller kits used for electronic devices. Arduino boards are able to read inputs and turn it into outputs. The boards are programmable using the Arduino programming language and the Arduino Software (IDE).[5]

2.2 PWM signals

PWM is a technique to control analog circuits using a digital signal from a processor. An analog signal has a varying signal and infinite resolution, while a digital signal has a fixed value and resolution. Many microcontrollers include PWM controllers, making the use of digital signals to control analog signals possible without extra equipment.

PWM uses high resolution counters to modulate the duty cycle, the ratio of the on-time and period, of square waves out of the digital outputs of a processor. This means at any given time the digital output gives the voltage from the DC supply either fully turned on or off. Different duty cycles of a digital signal can be seen in figure 2.

One of the advantages of PWM is that the signals is digital, meaning that noise can only affect the signal if it is big enough to change the logic from a 1 to a 0, or vice versa. A use of PWM to control an analog signal is to control the duty cycle of an analog sine wave, creating a dimming effect. [3]

The duty cycle is given in % and is calculated using

\[ D = \frac{PW}{T} \cdot 100\% \] (1)

where PW is the pulse width active time, T the total period time and D is the duty cycle.[4]
2.3 Dimmer circuit

The circuit as seen in figure 1 is a dimmer circuit that uses PWM to control the duty cycle of the single phase AC from the wall socket. The operation relies on two transistors that modulates the AC signal from the socket. An optocoupler is used to isolate the microcontroller from the higher voltage of the analog AC signal. The optocoupler is used to control a MOSFET which is determining the duty cycle of the sine wave signal that goes through the lamp. The voltages and currents are limited by Ohm’s law,

\[ U = R \cdot I \] (2)

where \( U \) is the voltage, \( R \), the resistance and \( I \) the current, to keep the voltages and currents at specified levels.[6]

As a safety procedure, the microcontroller is protected by the optocoupler, which insulates the microcontroller from the higher voltages of the analog signals in the dimmer circuit. This has to do with the optocouplers class II insulation class. Class II appliances means that the user is protected by at least two layers of insulation and do not require a ground connection.[7]

2.4 Wireless communication

2.4.1 Radio Waves

Radio waves is, as microwaves, infrared radiation, X-rays and Gamma-rays a sort of electromagnetic radiation, also known as EM-radiation. The main purpose of radio waves is to send communication signals like TV, cellphones and radio, where you convert the radio waves to mechanical vibrations to produce sound or light. The radio
wave has the largest bandwidth of all EM-waves. They can vary between 3kHz all the way up to 300GHz depending on its purpose. See table [1] below. [9]

<table>
<thead>
<tr>
<th>Band</th>
<th>Frequency range</th>
<th>Wavelength range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extremely Low Frequency (ELF)</td>
<td>&lt;3 kHz</td>
<td>&gt;100 km</td>
</tr>
<tr>
<td>Very Low Frequency (VLF)</td>
<td>3 to 30 kHz</td>
<td>10 to 100 km</td>
</tr>
<tr>
<td>Low Frequency (LF)</td>
<td>30 to 300 kHz</td>
<td>1 m to 10 km</td>
</tr>
<tr>
<td>Medium Frequency (MF)</td>
<td>300 kHz to 3 MHz</td>
<td>100 m to 1 km</td>
</tr>
<tr>
<td>High Frequency (HF)</td>
<td>3 to 30 MHz</td>
<td>10 to 100 m</td>
</tr>
<tr>
<td>Very High Frequency (VHF)</td>
<td>30 to 300 MHz</td>
<td>1 to 10 m</td>
</tr>
<tr>
<td>Ultra High Frequency (UHF)</td>
<td>300 MHz to 3 GHz</td>
<td>10 cm to 1 m</td>
</tr>
<tr>
<td>Super High Frequency (SHF)</td>
<td>3 to 30 GHz</td>
<td>1 to 1 cm</td>
</tr>
<tr>
<td>Extremely High Frequency (EHF)</td>
<td>30 to 300 GHz</td>
<td>1 mm to 1 cm</td>
</tr>
</tbody>
</table>

Table 1: How the Frequency changes with wavelength

A radio wave transmitter does not need to be in direct view of the transceiver. They can go through hills and radio waves in the lowest frequency can be reflected on the Earths upper atmosphere called the ionosphere which makes the waves even go over hills. [8]

The main problem with radio waves is interference if two sources sends on the same frequency the receiver will get both signals. The problem can be solved by slightly adjusting the signal frequency sending and receiving.

2.4.2 Bluetooth

Bluetooth also uses radio waves. The main difference between normal radio waves and Bluetooth is that Bluetooth devices need to pair with each other before they can be used, this solves the previous problem with interference between sources with the same frequency mentioned in section [2.4.1]. Bluetooth can transfer data between two devices with a distance of up to 60 meters between them. [10]

The main problem with Bluetooth is also part of its strength. One can at the moment only connect to one device at once. This makes Bluetooth a inferior choice if multiple
2.4.3 Wireless Fidelity

Like Bluetooth (section 2.4.2), Wireless Fidelity, or WiFi for short, is using radio waves (section 2.4.1). Like Bluetooth, some way is needed to connect the source of the signal to the receiver. The WiFi solution is an antenna, which is more commonly known as the router. The router creates a hotspot which is the area where you, with a WiFi module, can connect to the network created by the router. Multiple modules can be connected to the same network at the same time which makes WiFi a good solution if you want to send signals to multiple devices at the same time.

WiFi work on frequency levels of either 2.4GHz or 5GHz depending on the amount of data you want to send. The network using the 802.11 networking standards will vary depending on the users needs, going from 802.11a to the faster 802.11n with 5GHz frequency, and 802.11b to the faster 802.11g with 2.4GHz frequency.[11]

2.4.4 Infrared

Infrared radiation or IR-radiation is as radio waves (section 2.4.1) a sort of EM-radiation with an frequency below radio waves but higher then visible light. IR-radiation is simply light that you can not see but can still use to transfer data.

IR-radiation are commonly used in remote controllers for the TV. It often operate in the 38kHz area where there are very few natural sources of that frequency. IR-communicating uses an IR-LED on the transmitter and an IR-receiver will pick up the signal and decode it. The LED will blink in a different pattern depending on what you want the reciever to do.[12]
3 Method and design

3.1 Components

The microcontroller chosen for this project is called Adafruit Feather Huzzah ESP8266 which features Arduino compatible hardware as well as a WiFi module called ESP8266. The ESP8266 module is a WiFi module compatible with Arduino which can connect to a network and send and transmit data wirelessly over said network. Since the chosen Arduino board features a built-in WiFi module, there is no need to add extra modules to the project. The ESP8266 has its own Arduino library which can be used to program it.

This microcontroller is connected to the circuit shown in figure [1], where the Adafruit Feather Huzzah ESP8266 replaces the generic Arduino board in the figure. Because the Adafruit Feather Huzzah ESP8266 uses 3.3 V logic instead of the standard 5 V logic featured on common Arduino boards, the current-limiting resistor has to be changed. Using equation [2], a new resistor value of 180 Ω is used to have the same voltage across the optocoupler.

3.2 Implementation

3.2.1 Setting up a server

Programming the Huzzah to connect to a network means any device connected to the network can communicate with it. This is essential for the wireless communication between the smartphone and the Arduino. To be able to send and transmit data between the devices, a server is set up. The server is web-based and is accessible by the Arduinos IP adress. This is done using any web browser by a device connected to the same network. The IP adress is accumulated by a computer physically connected to the Arduino, and is displayed when the Arduino is connected to the desired network. The Arduino is programmed to connect to a specific network chosen in the code, where the SSID and password of the network to connect to is programmed into the Arduino.

When the server is set up, it needs to have a GUI for the user to interact with. This interface is implemented to the server using JavaScrip. This allows for an interactive interface that can be used by any device with internet browser support.

3.2.2 Server communication

To send and transmit data to and from the server, the interactions used by the user on the web server will be detected and recorded on the Arduino. The Arduino will not read the JavaScrip directly, but will instead put the data in a string and split it in to smaller parts to be able to react to actions made on the server, this is made with the server code directly implemented in to the Arduino code. The interface consists of
sliders, which are called PWM Outputs, which goes from values 0 to 1023. The slider values determines the duty cycle of which the Arduino controls the lamp, meaning a duty cycle of 0 means the lamp is completely off and a value of 1023 means the lamp is completely on.

The server is programmed to support more than one slider, even though the prototype only requires one slider as only one of the digital outputs is used to control a lamp. Each slider would represent a digital output from the Arduino and can therefore be controlled by adding more sliders and implementing those into the code.

3.3 Workspace and system setup

The system is built on a breadboard instead of on a through hole PCB for easy assemble, disassemble and components replacement. The breadboard can have the Arduino connected directly to it, instead of using extra wiring. This makes testing easy, as the Arduino can be easily removed. A computer is used to program the Arduino in the Arduino IDE, this makes transferring the code over to the Arduino easy. To test the Arduino, a generic smartphone with a web browser is used.

Figure 3: System setup. 1) Computer, 2) Circuit, 3) Smartphone, 4) Lamp
4 Results and discussion

The resulting prototype works. There is a running server that can be accessed by any device on the same wireless network and it responds to commands by clients on the network. These commands can be interpreted by the Arduino to control the lamp. The dimming is not continuous when the slider value is altered, but instead only reads the value of the slider when it is released. There is a small problem with the bootup of the Arduino. The current limiting resistor on the digital output causes the digital pin to float, which causes the Arduino to not load the program properly. Booting the Arduino separate and then connecting it to the circuit will however make the system load fine and work as intended.

4.1 Wireless communication

The final product uses WiFi (section 2.4.3) to communicate between the smartphone and the Arduino instead of Bluetooth (section 2.4.2), regular radio waves (section 2.4.1) or IR-radiation (section 2.4.4).

The WiFi solution makes the whole system password protected and easy to connect multiple modules to the same network. WiFi lets the user control the system in a larger area depending on your routers range, which makes it easy to control several modules on the same network. The WiFi solution has a GUI designed to be easy to use by any device with web browser support, as seen on a smartphone in figure 5.
Processing Status

Figure 5: Server GUI as seen from a smartphone
4.1.1 Bluetooth solution

A Bluetooth solution would have looked similar to the WiFi solution except there would have been a Bluetooth chip on the Arduino for communication instead of the WiFi chip, and a sever solution would not have been an option instead an Android app would have been needed. The reason that Bluetooth was not chosen was because of its limitation to only be able to be connected to one receiver module at once and one of the projects goals was to be able to come up with a solution to implement the solution to a larger scale.

4.1.2 Infrared solution

A IR-radiation solution would have used an IR-module on the Arduino for the wireless communication. Since most smartphones nowadays do not have IR-transmitters, an IR controller would have been needed. IR-radiation is limited to line of sight with the receiver which can be problematic to use when there are many objects nearby the transmitter and reciever. IR-radiation also has a very limited amount of data that can be sent which in itself becomes a limitation if the final product needs to transfer a lot of data.

4.1.3 Radio wave solution

The radio wave solution would have been very similar to the IR-radiation solution except that all the receiver and transmitter modules would have been radio wave versions. A radio wave solution makes the whole system exposed from outside sources which makes it an inferior choice, as it will be easier for other sources to interfere with the signals. Some smartphones have radio wave transmitters, which can be used for this purpose.

4.2 Implementation for other appliances

The Adafruit Feather Huzzah ESP8266 supports nine digital outputs, meaning this microcontroller can control up to nine different appliances at a time. Each output can also control more than one lamp, several lamps connected in serial or parallel can be connected to each output, making the single microcontroller able to control larger lighter systems.

The final product can be used in many different areas. It is not always preferred to be able to dim electrical appliances, but that can easily be modified in the prototype so that instead of controlling a dimmer circuit, relay switches are controlled with an Arduino. This opens up a lot of possible areas of usage. For example are most industrial lighting appliances made up of fluorescent lamps, which can not be dimmed. These can however be controlled using relay switches, making an implementation of
our system to industrial appliances and offices not too difficult.

To implement wireless control of electrical appliances in a room, the Arduino needs to be connected to a relay for each phase or number of appliances to control. The Arduino will have to be programmed in a slightly different way for this to work, depending on the appliances. For appliances that can not be dimmed, a simple ON/OFF switch can replace the sliders on the server. This requires the dimming code itself to be changed so that the digital pins instead outputs a HIGH signal when the button is ON, and a LOW signal when the button is OFF, instead of a duty cycle that is used for the dimmer circuit.

4.3 User interface

Mentioned in section 3.2 a server solution is used to control the intensity of the light. Several solutions were tested during the course of this project where an Android app and a Blynk solution was the biggest contenders, but ultimately the sever solution was chosen.

4.3.1 Android app

A solution using a self-designed Android app is probably the best solution. The GUI will be designed for a smartphone and lets the person making the interface more in control of the design. The Android app solution was not chosen because of lack of knowledge in programming Android software and WiFi communication between Android apps and microcontrollers.

4.3.2 Blynk

Blynk is a platform for both Android and IOS which lets you control your microcontroller easily with minimal coding needed. The only requirement for the Blynk solution is that your unit is connected to the internet. Blynk was not used as a solution for this project for several reasons. Blynk has its own library for Arduino programming, meaning the code is already premade and doesn’t offer too many options for the programmer. To use Blynk, an app is downloaded to the smartphone. When started, the wanted sliders or buttons are chosen depending on the users needs. Then premade code are accessed on the Blynk’s own servers, and as long internet connection is up everything works. Because Blynk uses their own server for the communication, the solution strictly depends on the Blynk server runtime.
5 Conclusion

The final product works. The system connects to a WiFi network and can be controlled by devices, such as a smartphone, connected to the same network. The system has a GUI that can be used by a smartphone, but also by devices with a web browser. The product can be used for many electric appliances that has to be controlled wirelessly. The system can be altered for none-dimming purposes by using relay switches instead of dimming circuits for easier implementation, as many appliances does not support varying input voltages.

Wireless control of appliances is in line with the future, as many devices aims for wireless control, such as home automation and wireless communication between devices. This solution of wireless control can be implemented for many different applications, but will need some improvements to work flawlessly.
6 References


Appendices

A Arduino code

```c
#include <ESP8266WiFi.h>

const char* ssid = "network";
const char* password = "password";

// Create an instance of the server
// specify the port to listen on as an argument
WiFiServer server(80);

int pwm = 0;
int PwmPin = 0;

void setup() {
  Serial.begin(115200);
  Serial.setTimeout(50);
  delay(10);
  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected");
  // Start the server
  server.begin();
  Serial.println("Server started");
  // Print the IP address
  Serial.println(WiFi.localIP());
}

void loop() {
  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }
  // Wait until the client sends some data
  Serial.println("new client");
```

while (!client.available()) {
    delay(1);
}

// Read the first line of the request
String command1 = client.readStringUntil('/');
String command = client.readStringUntil('/');
Serial.println(command1);
Serial.println(command);

if (command == "pwm") {
    int pin, value;
    pin = client.parseInt();
    if (client.read() == '/') {
        value = client.parseInt();
        pwm = value;
analogWrite(pin, value);
    } else {
        value = pwm;
    }
    client.print(F("pwm,"));
    client.print(pin);
    client.print(F(""));
    client.println(value);
} else if (command == "status") {
    int pin, value;
    client.print(F("status"));
    client.println("");
} else { // Prepare the response
    String s = "HTTP/1.1 200 OK\r\nContent-Type:  text/html\r\n\r
";
s += "<html>\r
<head>\r
<title>Wireless I/O Demo</title>\r
<script type="text/javascript">\r
window.onload=Pinstatus;\r
function Pinstatus() {\r
    morestatus();\r
}
function morestatus() {
    setTimeout(morestatus, 4000);
    document.getElementById("description").innerHTML = "Processing Status\r
";
    server = "status/99";\r
    request = new XMLHttpRequest();\r
    request.onreadystatechange = updateasyncstatus;\r
    request.open("GET", server, true);\r
    request.send(null);
    }\r
    function updateasyncstatus() {\r
    if ((request.readyState == 4) && (request.status == 200))\r
    "}\r
";
}
result = request.responseText;

document.getElementById("description").innerHTML = result;

fullset = result.split("=");

document.getElementById("description").innerHTML = fullset;

for (i = 1; i < fullset.length; i++) { 
    PinPair = fullset[i];
    singleset = PinPair.split("=");
    PN = singleset[0];
    Pinstatus = singleset[1];
    if (PN.length > 11) {
        ActNum = "action" + PN;
        TxtNum = "text" + PN;
        if (Pinstatus == 0) {
            PinAct = "1";
            text = "Off";
        } else {
            PinAct = "0";
            text = "On";
        }
        document.getElementById(ActNum).value = PinAct;
        document.getElementById(TxtNum).innerHTML = text;
    }
    if (PN.length == 4) {
        TxtNum = "text" + PN;
        if (Pinstatus == 1) {
            text = "On";
        } else {
            text = "Off";
        }
        document.getElementById(TxtNum).innerHTML = text;
    }
    if (PN.length == 0) {
        PinVal = parseInt(singleset[1]);
        PwmNum = "pwm" + PN;
        ValNum = "valuePwm" + PN;
        document.getElementById(PwmNum).value = PinVal;
        document.getElementById(ValNum).innerHTML = PinVal;
    }
    if (PN.substr(0, 1) == "A") {
        PinVal = parseFloat(singleset[1]);
        AnalogNum = "analog" + PN.substr(1, 2);
        document.getElementById(AnalogNum).value = PinVal;
    }
}
}
function updateasyncbutton() {
  if (request.readyState == 4 & request.status == 200)
  result = request.responseText;
  document.getElementById("description").innerHTML = result;
  singleSet = result.split("",");
  document.getElementById("description").innerHTML = result;
}

function sendPwm(Pin, value) {
  ValNum = valuePwm + Pin;
  document.getElementById(ValNum).innerHTML = value;
  document.getElementById("description").innerHTML = "Processing Slider";
  server = "pwm/" + Pin + "/" + value + ";
  request = new XMLHttpRequest();
  request.onreadystatechange = updateasyncPwm;
  request.open("GET", server, true);
  request.send(null);
}

function updateasyncPwm() {
  if (request.readyState == 4 & request.status == 200)
  result = request.responseText;
  singleSet = result.split("",");
  PinType = singleSet[0];
  PinNum = singleSet[1];
  PinVal = parseInt(singleSet[2]);
  PwmNum = "pwm" + PinNum;
  ValNum = valuePwm + PinNum;
  document.getElementById(PwmNum).value = PinVal;
  document.getElementById(ValNum).innerHTML = PinVal;
  document.getElementById("description").innerHTML = result;
}
</script>
</head>

<table name="Table" border="1" cellspacing="0">
<tr><th align="center" colspan="6">PWM Output</th></tr>
<tr><td align="center">
  GPIO 0<br>
  <input type="hidden" name="pin" value="0" id="pin0" />
  <input type="range" style="width: 400px; height: 60px;" id="pwm0" min="0" max="1023" value="0" step="1" onchange="sendPwm(document.getElementById("pin0").value, this.value)" /></td>
</tr>
<tr><th align="center" colspan="6">PWM Output 2</th></tr>
<tr><td align="center">
  GPIO 1<br>
  <input type="hidden" name="pin" value="1" id="pin1" />
  <input type="range" style="width: 400px; height: 60px;" id="pwm1" min="0" max="1023" value="0" step="1" onchange="sendPwm(document.getElementById("pin1").value, this.value)" /></td>
</tr>
</table>
"GPIO 16
"<br/>
"<input type="hidden" name="pin" value="16" id="pin16" />
"<input type="range" style="width: 400px; height: 60px;" id="pwm0"
min="0" max="1023" value="0" step="1" onchange="sendPwm{
document.getElementById('pin16').value,
"
"this.value);"/>
"<br/>
"<span id="valuePwm16">0</span>
"</td>
"</tr>
"</table>
<br>
<br>
<p id="description">
</font>
"</p>
"</html>
client.flush();
// Send the response to the client
while(s.length() >2000)
{
  String dummy = s.substring(0,2000);
  client.print(dummy);
  s.replace(dummy, " ");
}
client.print(s);
delay(1);
Serial.println("Client disconnected");
// The client will actually be disconnected
// when the function returns and 'client' object is destroyed