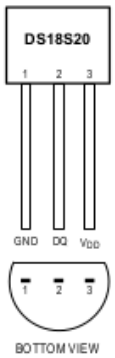
Transmitting sensor values over WiFi

The aim of this is to learn how to transmit a temperature value to a tablet via WiFi.
First we have a look at the sensor, then try to integrate this to a WiFi program.

1. Reading a Dallas temperature sensor, the easy way



Hardware connections:

- VDD to +3.3V from WEMOS chip
- DQ to any GPIO pin, here D7 for the code example
- pullup resistor 4.7 kΩ between VDD and DQ

The Arduino pinout is found here:
https://github.com/esp8266/Arduino/blob/master/variants/d1_mini/pins_arduino.h#L49-L61

D0 = 16	D4 = 2	D8 = 15	SDA = 4
D1 = 5	D5 = 14	RX = 3	SCL = 5
D2 = 4	D6 = 12	TX = 1	
D3 = 0	D7 = 13	LED_BUILTIN = 2	

There are two libraries needed:

- Onewire
  https://github.com/PaulStoffregen/OneWire
- DallasTemperature
  https://github.com/milesburton/Arduino-Temperature-Control-Library

The Onewire library must be one adapted to the ESP8266 chip, as is the one that Paul Stoffregen has written

```
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 13            // pin D7
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
float temp;

//--------------------------------------------------------------------
void setup(void){
 Serial.begin(9600);
 sensors.begin();
}
//--------------------------------------------------------------------
void loop(void){
   sensors.requestTemperatures();
   temp = sensors.getTempCByIndex(0);
   Serial.println(temp);
   delay(1000);
}
```

2. More sensors on the same bus

There can be more sensors connected to the same pin. They all have an internal address that allows to differentiate them. The DallasTemperature library automatically detects sensors on the bus, and manages them by using an index. The easyest way to know which sensor has which index is try and error: heat up the sensor with your finger and watch which value is changing.

```
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 13
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
float temp1;
float temp2;

//--------------------------------------------------------------------------
void setup(void){
 Serial.begin(9600);
 sensors.begin();
}
//--------------------------------------------------------------------------
void loop(void){
   sensors.requestTemperatures();
   temp1 = sensors.getTempCByIndex(0);
   temp2 = sensors.getTempCByIndex(1);
   Serial.print(temp1);
   Serial.print("\t");
   Serial.println(temp2);
   delay(1000);
}
```

3. Do more with the DS1820

The DallasTemperature library is very easy to use. If you need more control, you can use the OneWire library and do things like read the ROM address of the sensor, write and read bytes of the scratchpad and so on.

https://www.pjrc.com/teensy/td_libs_OneWire.html

Also you have more control over the timing. Remember that the sensors nee up to 750ms to acquire the temperature value.

4. A webpage displaying the temperature

The following sketch is based on WiFi experiments described here:

http://staff.ltam.lu/feljc/electronics/arduino/WiFi_01.pdf

In these examples we had used a static webpage transmitted by the ESP8266.

For our purpose, we need the webpage to be regularly updated with the temperature value.
To do this, we create two halfs of the html webpage string (these are the parts that remain constant), and

put another string with the temperature value inbetween them. This is updated in the loop function of the program, so the new HTML page with the new temperature value is sent to the tablet in regular time intervals.

But there is a catch: the webbrowser will load the web page into the cache and always display the old page, except if it is explicitely told to do so.

Fortunately there is a simple solution to this: the <head> section of the HTML page must contain something like this:

```
<head>
    <title>TEMPERATURE</title>
    <meta http-equiv="refresh" content="2" />
  </head>
```

The meta tag tells the browser to refresh the display in an interval of (in our example) 2 seconds.

This solution is maybe not so elegant, as the update of a whole page produces some flickering, but it has the advantage of being simple.

For better overview our sketch is divided into several tabs in the Arduino editor, that means several files in the sketch folder:

The declarations and setup part of our sketch is nearly the same as in the static webpage sketch:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

#include "html.h"
#include "DS1820.h"

float temp;
String temperature;


/* Change these: */
const char* ssid = "myAccesspoint";
const char* password = "myPassword";
IPAddress IPaddr (192, 168, 168, 168);
IPAddress IPmask(255, 255, 255, 0);

ESP8266WebServer server(80);
# include "functions.h"


void setup() {
      delay(1000);
      Serial.begin(115200);
```

```
      Serial.println();
      Serial.print("Configuring access point...");

      WiFi.softAP(ssid, password);
      WiFi.softAPConfig(IPaddr, IPaddr, IPmask);

      IPAddress myIP = WiFi.softAPIP();
      Serial.print("AP IP address: ");
      Serial.println(myIP);
      server.on("/", handleRoot);
      // eventually add other functions to handle

      server.begin();
      Serial.println("HTTP server started");
}
```

There is an include for the webpage (in an extra tab for better overview):

```
String html;

String html1 = R"***(
<!DOCTYPE html>

<html>
  <head>
    <title>TEMPERATURE</title>
    <meta http-equiv="refresh" content="2" />
  </head>

  <body>
    <h1>TEMPERATURE</h1>
)***";

// temperature will be inserted between html1 and html2

String html2 =  R"***(
  </body>
</html>
)***";
```

Another include contains the DS1820 function to read the temperature:

```
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 13
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

//--------------------------------------------------------------------------
float readtemperature(byte index){
   sensors.requestTemperatures();
   float temp = sensors.getTempCByIndex(index);
   return temp;
   }
```

The loop function reads the temperature and puts together the strings html1, temperature and html2 to a big string html containing the dynamic webpage.

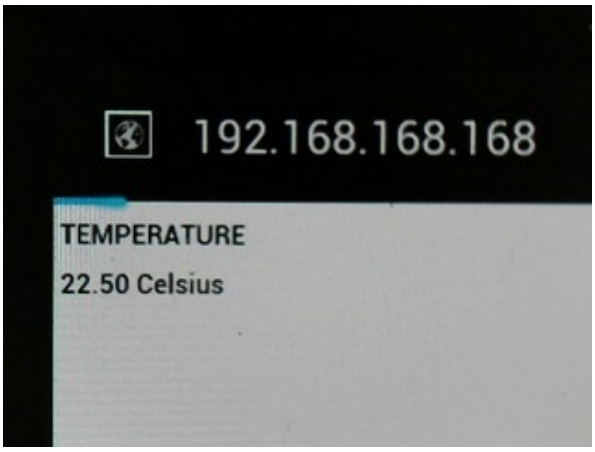
```
void loop() {
    temp = readtemperature(0);
    temperature = "<h1> " + String(temp) + " Celsius </h1>";
    html = html1 + temperature + html2;
    server.handleClient();
    Serial.println(temp);
}
```

The sending of the webpage is done in the function handleRoot, every time the browser calls the root page by requesting “192.168.168.168/”.

This function just sends the html string to port 200 of the browser:

```
void handleRoot() {
  server.send(200, "text/html", html);
}
```

On my smartphone, the result looks like this:



Every 2 seconds, as told by the the HTML code, the display is updated.

The complete example code is found here:

http://staff.ltam.lu/feljc/electronics/arduino/8266/WiFiAccessPoint_temperature_01.zip