TURN ON YOUR CREATIVITY

FRANZIS MAKER KIT INTERNET OF THINGS

MANUAL

FRANZIS

CONTENTS

BEFORE YOU BEGIN

When attaching the board for the first time, it may happen that the computer does not automatically find the required driver for the USB-to-serial converter. In this case, you can download the driver from *www.iot.fkainka.de/driver* and install it manually. You can then select the port and the *Arduino Nano* board (processor: *Atmega328*) in the Arduino software. Now the controller should be fully operational.

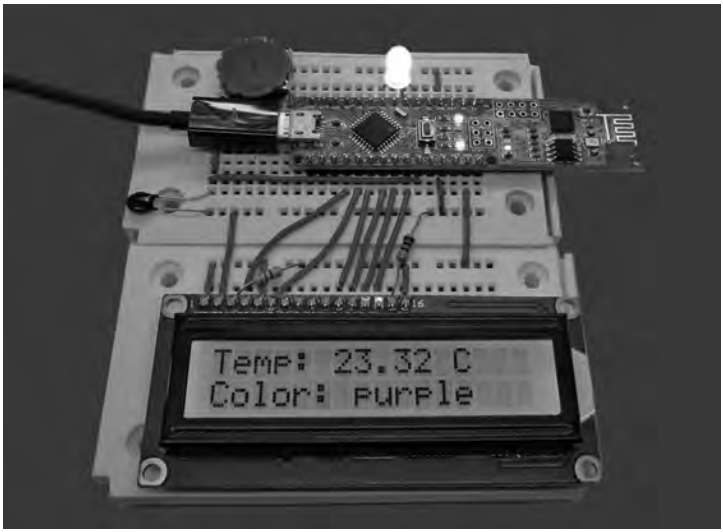
Correct settings for the Arduino environment

I used the Arduino IDE versions 1.64 and 1.6.5. Older versions may cause problems. Get the current version on *www.arduino.cc*.

For some Linux versions, there are currently only older IDE versions. Tips for solving this and other problems are provided on *www.iot.fkainka.de*.

There you will also find all of the examples in this book and additional useful information.

The tutorial kit comprises two breadboards, which can be combined. The best method is to attach the Pretzel board and the display to the breadboards as shown in the image below. This way, most of the space remains available for your own experiments, while the Wi-Fi module sticks out from the breadboard and the display has enough room on the other breadboard. The micro USB leads away from the breadboards and can thus cause only minimum trouble. Detailed wiring diagrams are given in the respective chapters.



Pretzel board and display on the breadboards

The Pretzel board

The main component of this tutorial kit is the Pretzel board (internally called NanoESP). As you can see in the picture, it consists of two parts. The left half is an Arduino compatible microcontroller system similar to the Arduino Nano. The right part is the *ESP8266* Wi-Fi module. These components communicate via a software-generated serial interface.

1 MEET THE MODULE

The first chapter deals with the basic features of the Wi-Fi module. This module is controlled by so-called AT commands. As stated in the introduction, all sample programs are provided on *www.buch.cd*. Just enter the code **65316-9**.

The easiest way is to download the complete zip directory and to copy the complete unzipped folder into your sketch folder. Then you can comfortably open the programs one by one in the Arduino user interface.

1.1 | Basic AT commands

Program file:
P01_SoftwareSerial.ino

The best way to get to know the usage of the AT commands is to have a go at them. Thus, this section introduces some of the basic commands for the module.

Open the *P01_SoftwareSerial* program in the Arduino IDE. This is a very basic program that uses a homemade software interface to simply pass all the data that were received via the serial hardware interface of the microcontroller to the ESP controller. This works also in the opposite direction. As you can see in the source code, the two ports for the software interface are the pins 11 and 12. You should not use them as GPIO pins in your own projects. These pins need the `SoftwareSerial` library, which is preinstalled in most Arduino versions. If not, download it via the manager.

Upload the program and start the serial monitor in the Arduino user interface. Before you can get started, you have to change two important settings in the serial monitor: Set the baud rate in the bottom left corner to *19200* and the box left of it to *CR and NL*.

Now you see the first messages, namely *AT* and a few lines below *OK*. The microcontroller has sent the *AT* command to the ESP module, and the module has responded *OK*. This means that the Wi-Fi module is working and ready for use.

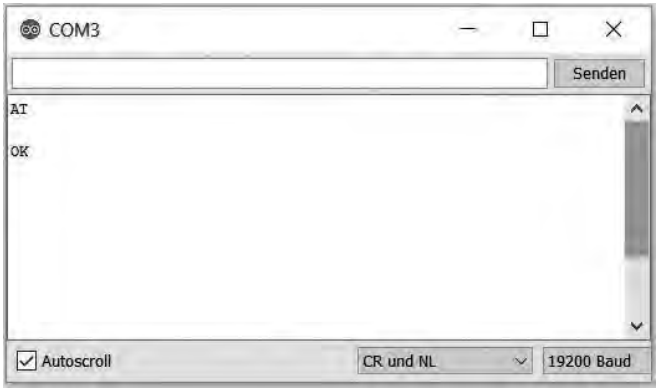


Figure 1.1: Terminal settings CR and NL and a baud rate of 19200

1.1.1 | Basic commands

You can easily test some of the basic commands for the module by entering them and then hitting Enter to send them to the module. It is essential to enter the commands in upper case. As a first command, enter

```
AT
```

in the serial monitor and hit Enter. The uploaded program passes the command to the ESP module, which responds with `AT`, followed by `OK`. Next, try the following command:

```
AT+GMR
```

This command returns the current firmware and version number. The command

```
AT+RST
```

resets the module. In this case, you first see a few cryptic characters in the terminal, followed by `ready` to indicate that the module is again ready for use. Another command is:

```
ATE0
```

2 UDP AND IP

This chapter discusses the basic data transfer between two systems via a Wi-Fi network. We will look at topics like IP, ports, and UDP. However, at first we have to explain these basic terms.

What is an IP address?

An IP address works like a mail address. It allows to identify and to address a computer in the network. An IP address as per the still current IPv4 standards looks like the following example:

```
192.168.4.1
```

It is made up from four numbers or to be more precise, four bytes. This means that the numbers can only have a maximum value of 255. As a general distinction, there a local IP addresses that you can assign to computers and other devices in your home network, and global IP addresses.

Local IP addresses are usually assigned by your router. In most cases, they begin with 192.168. The third number differs between routers. When the Pretzel board is used as access point, any computers that enter its network get an address beginning with 192.168.4. This also creates a subnet. Fritz!Box routers usually assign local IP addresses in the format 192.168.178.X. In Windows, you can discover your IP address by entering `ipconfig` at the command prompt. The command returns a long list that contains the entry *IPv4 address* stating your local IP address in the network.

Global IP addresses are usually assigned by your internet service provider (ISP). These addresses allow your router to be accessible in the worldwide network. The router creates the local network and distributes any data to the clients. One possibility to find out your global IP address is to navigate to the website *http://whatismyipaddress.com/*. Here you see several data that a web server can see. You are not as anonymous in the Internet as you may have believed.

What is a port?

When we stay with the analogy of the mail address, a port can be thought of as something like the entry door of an apartment house. A computer with a unique IP address can offer services via different ports. You access the server by means of the IP address, but with the port, you chose the service. This can be e.g. port 20 for FTP data transfer or port 23 for a telnet connection. In general, you are free to choose the port number, although there are some standard ports to facilitate the use of web application. A list of these standard ports can be found at *https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers*.

What is UDP?

UDP is short for User Datagram Protocol. This is a minimalistic connectionless network protocol. It is more minimalistic and simpler than other Internet protocols like e.g. TCP, which we will discuss later. At this point, it is not easy to compare the protocols, but bear in mind the following properties of UDP:

- UDP is able to broadcast
- UDP does not check the data for correctness and does not correct any errors
- Successful data transfer is not guaranteed
- There is also no guarantee that the data are not tampered with or intercepted by others during transfer
- It is not necessary that a connection is established up front; instead, a fast data transfer is possible
- There are nearly no variations in transfer delay
- The format can be used e.g. for VoIP (Voice over IP, i.e. phone connections via the Internet)

These are the most important basic information about the concepts used in the following projects. Whenever it is appropriate, I will give you further information to address these topics in more detail. But first, we will do some hands-on experiments.

3 TCP CLIENT

The previous chapter dealt with UDP, which allows sending and receiving data in a simple manner. With this protocol, you can implement many applications. However, in this chapter we will cover TCP (Transmission Control Protocol) with the module taking on the role of the TCP client. This is the role that your PC plays in dealing with a web server. The differences between TCP and UDP can be roughly summarized as follows:

- The connection links exactly two devices.
- The sent packages are checked for transmission errors and corrected if needed.
- TCP is in particular used for Internet browsing.
- The protocol is a little slower than UDP but more reliable.

When you want to download a web page from a web server, both your PC and the web server use this protocol to establish a connection to each other. The contents of the web page are then transferred via HTTP (Hypertext Transfer Protocol). This will be explained in more detail in the following chapter.

3.1 | A browser

This experiment uses again the existing setup with the LC display. The objective is to get to know the basics of TCP communication with the help of the serial monitor.

Progam file:
P10_TCPBrowser.ino

The program

The program works similar to the `SoftwareSerial` program in the first experiment. One of the differences is that it establishes the Wi-Fi connection during startup without user interaction. This saves you a lot of typing work and you can get up and running in less time. Do not forget to enter the data of your home network in the program! After you have done that, enter the following command line in the serial monitor:

```
AT+CIPSTART="TCP","www.example.com",80
```

This command establishes a TCP connection to the website *www.example.com*. Port 80 is the standard port for HTTP requests. After acknowledging the connection with *OK*, you can enter the following familiar command:

```
AT+CIPSEND=40
```

This is because you now want to send a message via the newly established connection. When the character > prompts you to provide the text, you first enter the following:

```
GET / HTTP/1.1
```

Then press Enter. The Enter command is not shown in the serial module but is received by the module. Continue to enter the message as follows:

```
Host:www.example.com
```

Press Enter twice. A long text is returned. The first part is the response of the server and contains some information for the browser. The text following <!document html> is the web page that you will also see if you open *www.example.com* directly. However, here it is displayed as text only. A browser can interpret the text and display it in the familiar form.

```
+IPD,1452:HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html
Date: Thu, 18 Jun 2015 15:05:48 GMT
Etag: "359670651"
Expires: Thu, 25 Jun 2015 15:05:48 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (iad/182A)
X-Cache: HIT
x-ec-custom-error: 1
Content-Length: 1270

<!doctype html>
<html>
<head>
    <title>Example Domain</title>
```

Figure 3.1: Excerpt from the answer of the web server

4 TCP SERVER

In the previous chapter, you have learned about using the module as TCP client. Here you will use the module as an independent TCP server. There is a simple AT command that comes in handy to start this complex server application. The module then behaves like a TCP server in the Internet, except that you have to program the web page transfer by yourself.

4.1 | TCP web server

For the first attempts to set up a TCP web server you do not need additional hardware. First, you simply test the important commands in the serial monitor.

The program

Program file:
P14_TCPServer.ino

Change the Wi-Fi data as before and upload the program to your board. Start the monitor. It may take some seconds until you see the message that the board is connected. When you see this success message and the IP address of the module, you can start with the first command in the serial monitor:

```
AT+CIPMUX=1
```

This command allows multiple connections to the module, so that several computers can access the web server. In order to start the server, you use the following command:

```
AT+CIPSERVER=1,80
```

The parameter 1 means that the server has to be started. In order to quit the server you use 0. 80 represents the port for accessing the server. HTTP requests by a browser are usually sent via port 80.

Open a browser of your choice, enter the IP address of the module in the address bar and press Enter. The browser first displays a load message. In the serial monitor, however, you can see a change:

Now there is a request like the one you sent manually in a previous experiment.

```
AT+CIPSERVER=1,80

OK
0,CONNECT

+IPD,0,363:GET / HTTP/1.1
Host: 192.168.178.58
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Accept-Encoding: gzip, deflate, sdch
Accept-Language: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
```

Figure 4.1: The browser request

The browser waits for an answer and shows the loading icon until the connection times out. In order to send a message to the browser, you can use a variation of the known command:

```
AT+CIPSEND=0,7
```

The `0` parameter represents the client to which the message is addressed. This is necessary because multiple connections are allowed and thus different clients may be connected. The second parameter again specifies the number of characters to send, in this case 7. The prompt > is displayed. Now you can enter a simple message like

```
Hello
```

and send it by pressing Enter. At first, you will not see any changes in your browser. You have to close the connection first. To this end you use the following command:

```
AT+CIPCLOSE=0
```

Now you see *Hello* in your browser. You have just implemented your very first web server application!

This first test tells us a lot about the detailed communication processes in the Internet. However, displaying *Hello* in the browser is a very simple task, as you only send a short text instead of a complex HTML page. For a real HTML page you would have to write manually headers for the `Get` request as well as for the HTML text, but I did not want to bother you with this for the first test.

5 THINGSPEAK

In the last chapter, we will discuss a completely new topic, namely the ThingSpeak platform. This web site was developed specifically for the Internet of Things and provides several useful functions. Among others, you can log and neatly display long-term monitoring of your sensors. As long as you are connected to the Internet, you do not have to bother with saving the values on external media. Furthermore, you can implement diverse control facilities via this web site. In this chapter, you get more detailed information about the individual features.

5.1 | ThingSpeak

Before you set up the experiment and upload the program, you have to create an account on the following website:

`www.ThingSpeak.com`

When you have an account, use the *Sign In* link to log on with your user name. The page that is now displayed shows your channels. For now, the page is rather empty as you do not yet have any channels. Click *New Channel* and specify a name, e.g. *Light* as in this project, we will measure the brightness. In *Field1* you can specify a name for the respective field, e.g. *Brightness*. Leave the rest of the fields untouched for now.

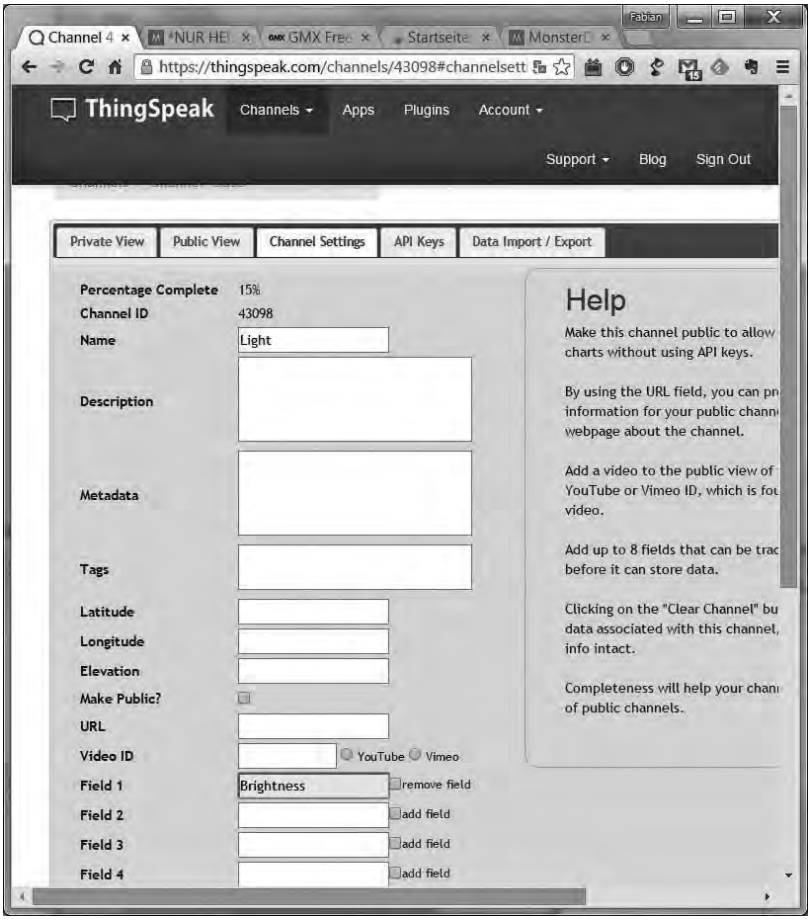


Figure 5.1: Creating a channel on the ThingSpeak website

Click *Save Channel* to save the settings. Now the page of your channel is opened, but it only shows an empty diagram. Click on the *API Keys* tab above the diagram. Under *Write API Key*, you can see a string of numbers and characters, which you will need soon.

The setup of this experiment comprises a sensor at the analog input A6 and the display. A voltage divider with a 10 kΩ resistor and a photo transistor allow for measuring the current brightness. You can also use the temperature sensor. The circuit remains nearly the same; just replace the photo transistor by the black NTC. The polarity of the NTC does not matter.

6 APPENDIX

Command	Notation
General	
Test command	AT
Reset	AT+RST
Firmware info	AT+GMR
Echo on/off	ATE<1/0>
Wi-Fi commands	
Wi-Fi mode (1 = client, 2 = AP, 3 = dual)	AT+CWMODE=<mode>
Discover WLANs	AT+CWLAP
Connect to WLAN	AT+CWJAP="<ssid>","<pass>"
Disconnect	AT+CWQAP
Access point settings	AT+CWSAP="<ssid>","<pass>"[,<chan>,<enc>]
Display IP address	AT+CIFSR
Activate/deactivate DHCP	AT+CWDHCP=<1/0>
Connect automatically to WLAN	AT+CWAUTOCONN=<1/0>
Change MAC address of the station	AT+CIPSTAMAC=<MAC>
Set IP address (station)	AT+CIPSTA=<IP>
Set IP address (access point)	AT+CIPAP=<IP>
Start SmartConfig	AT+CWSTARTSMART=<typ>
Stop SmartConfig	AT+CWSTOPSMART
Communication	
Ping	AT+PING=<HOST>
Allow several connections	AT+CIPMUX=<mode>
Data mode (0 = transparent, 1 = data mode)	AT+CIPMODE=<mode>
Structure of received data	+IPD,<id>,<len>:<data>
Establish connection	AT+CIPSTART="<type>","<address>",<port>
Send data	AT+CIPSEND=<id>,<len>
Disconnect	AT+CIPCLOSE=<id>
Server commands	
Start server	AT+CIPSERVER=1,<port>
Quit server	AT+CIPSERVER=0
Server state and connected clients	AT+CIPSTATUS
Set server timeout	AT+CIPSTO=<timeout>
Show connected clients	AT+CWLIF