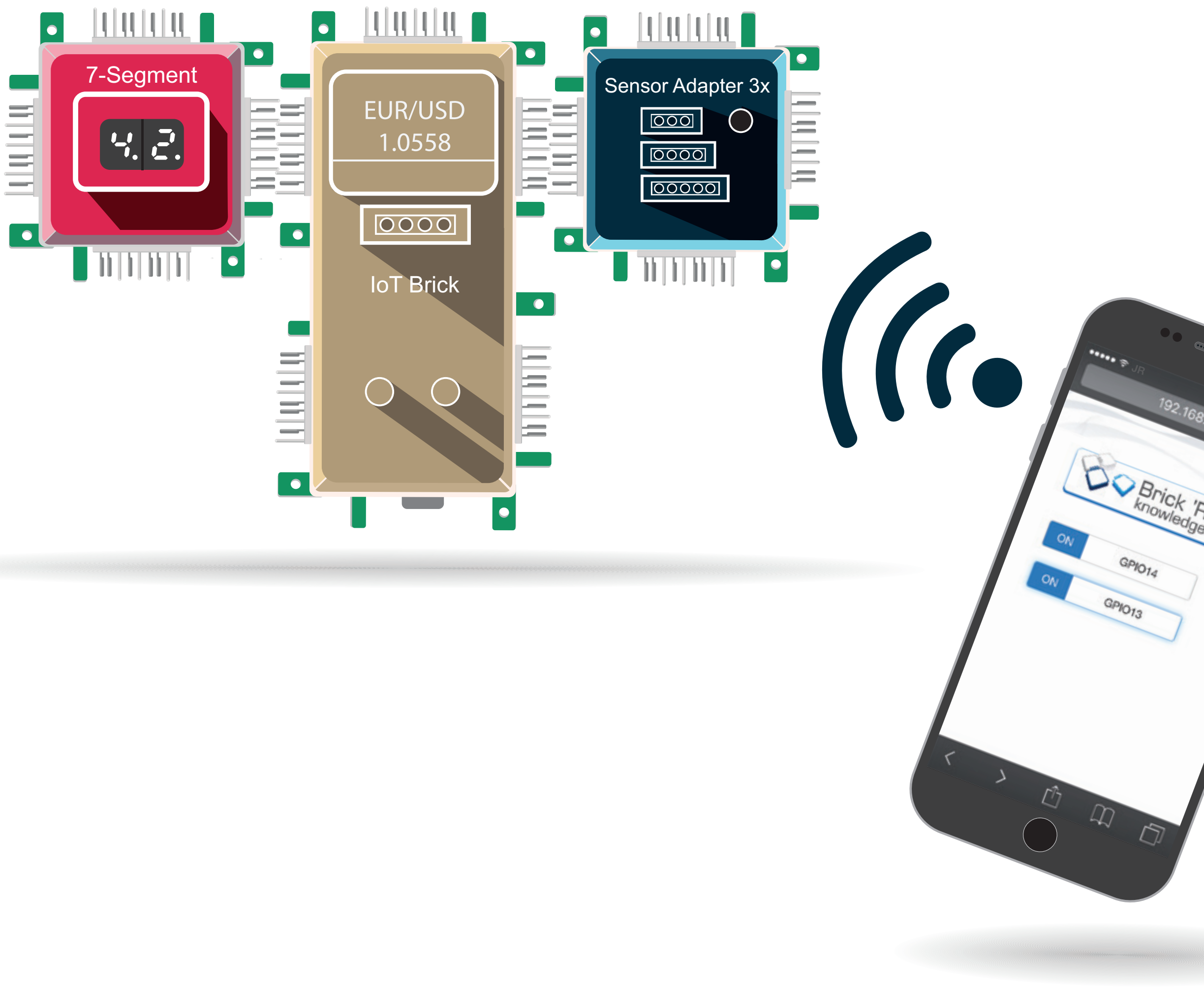
Internet of Things Set

Experimental kit by Brick'R'knowledge

Experimentierkasten von Brick'R'knowledge



www.brickrknowledge.de

Imprint

Brick'R'knowledge Internet of Things Set manual
Rev. 1.0
Date: 27.07.2017

ALLNET® GmbH Computersysteme
Brick'R'knowledge
Maistraße 2
D-82110 Germering

All information contained in this manual has been compiled with the utmost care and to the best of our knowledge. Nevertheless, mistakes can not be completely ruled out. We are always grateful for the communication of possible errors. Please send this to info@brickrknowledge.de.

DE13101093

Table of contents

1. Safety instructions 5

2. What does "Internet of Things" mean? 6

3. Basics of the Brick'R'knowledge system 7

3.1 Ground brick 7

3.2 Power supply 7

3.3 Connectors 8

3.4 Special connector brick for lower level 8

4. Hardware overview 9

5. The IoT brick and the Arduino IDE 13

5.1 The core of the Internet of Things set 13

5.1.1 Specifications of the IoT brick 13

5.1.2 The GPIO pins of the IoT brick 14

5.1.3 Pull-up resistors 14

5.2 The Arduino IDE 15

5.2.1 Installing libraries 16

5.2.1.1 Installing Arduino libraries 17

5.2.1.1.1 Library "NTPClient" 17

5.2.1.1.2 Library "Time" 17

5.2.1.1.3 Library "Json Streaming Parser" 18

5.2.1.1.4 Library "DHT Sensor Library" 18

5.2.1.1.5 Library "Adafruit Unified Sensor" 18

5.2.1.2 Installing external libraries 18

5.2.1.2.1 Library "esp8266-oled-ssd1306-master" 19

5.2.1.2.2 Library "MCP3421" 19

5.2.1.2.3 Library "BrickESP8266" 19

5.2.2 Virtual COM port driver 20

5.2.3 Serial monitor 20

5.3 First steps 21

5.3.1 Establishing a connection 21

5.3.2 Compiling and uploading code 22

5.3.3 Programming mode 22

6. Examples 23

6.1 "Hello World" (blinking-LED) 23

6.2 Switch and LED 24

6.3 I^2C-Bus 25

6.3.1 The 7-segment display 26

6.3.2 7-segment display as I^2C brick - structure and addresses 27

6.3.3 7-segment display as counter 29

6.3.4 7-segement display with button debouncing 30

6.4 OLED display - basics 32

6.4.1 OLED display - displaying text 33

6.5 Analogue inputs 35

6.5.1 A/D converter - basics 35

6.5.2 The A/D converter on the IoT brick 37

6.5.2.1 The 10bit A/D converter 37

6.5.2.2 The 18bit A/D converter 37

6.5.2.3 Voltage divider 37

6.5.2.4 Practical tip 38

6.5.2.5 Binary coding 38

6.5.3 A/D converter 10bit ____ 39
6.5.4 A/D converter 18bit ____ 41
6.6 IoT examples ____ 43
6.6.1 Setting up IoT brick as WLAN client ____ 44
6.6.2 Time from the internet ____ 46
6.6.3 Measuring temperature and humidity ____ 48
6.6.4 Dollar rate from the internet ____ 50
6.6.5 My first website ____ 52
6.6.6 Switching via website ____ 55
7. Brick Community ____ 60
8. Brick sets overview ____ 63

Foreword

The Brick'R'knowledge experimentation system was presented for the first time at the HAM radio exhibition on 28.06.2014 by Rolf-Dieter Klein (Amateurfunkzeichen: DM7RDK). The special feature of our electronics sets is that the individual components are connected via a plug system, in which the parts to be assembled are identical (Hermaphrodite). Thus, even tricky circuits can be realized. Also the putting together of the individual building blocks in different angles is possible! Two contacts are available for the return of the ground (0 volts)!

This makes it possible to build compact circuits in which the ground return ensures a stable voltage supply for the devices. Another particular feature is that such circuits can be easily explained and documented.

Rolf-Dieter Klein

Downloads:

- Sample code and libraries for the exercises in this tutorial:
  https://www.arduino.cc/en/Main/Software#
- Download Arduino development environment:
  http://www.brickrknowledge.de/downloads

1. Safety information

Note: Never connect the bricks directly to the mains power supply (115V/230V). There might be danger to life.

Please only use the included power supply-bricks. The voltage of our power-supply modules is 9V, which is not a health hazard. Please also ensure, that no openly wires are in contact with the mains power outlets. Otherwise there might be a danger of hazardous electric shocks. Never look straight into LEDs, since this may damage your eye retina.

Please connect the included polarized capacitors (tantalum/electrolytic) only with the positive side to the plus side of the power supply. If those polarized capacitors are connected not correctly, they can be destroyed and even explode!

Please remove the power supply brick everytime you finished expimenting, to avoid the risk af an electric fire.

2. What is the "Internet of Things"?

The Internet of Things (IoT) is the inter-networking of physical devices, vehicles (also referred to as "connected devices" and "smart devices"), buildings, and other items embedded with electronics, software, sensors, actuators, and network connectivity which enable these objects to collect and exchange data. In 2013, the Global Standards Initiative on Internet of Things (IoT-GSI) defined the IoT as "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies," and for these purposes a "thing" is "an object of the physical world (physical things) or the information world (virtual things), which is capable of being identified and integrated into communication networks.

(Source: https://de.wikipedia.org/wiki/Internet_der_Dinge)

Aha, now some readers will think, but what does that mean for me.

The Internet of things (IoT) is the inter-networking of physical devices, vehicles (also referred to as "connected devices" and "smart devices"), buildings, and other items embedded with electronics, software, sensors, actuators, and network connectivity which enable these objects to collect and exchange data. In 2013, the Global Standards Initiative on Internet of Things (IoT-GSI) defined the IoT as "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies," and for these purposes a "thing" is "an object of the physical world (physical things) or the information world (virtual things), which is capable of being identified and integrated into communication networks.

Some examples of the Internet of Things:

- So-called wearables, with sensors incorporated into clothing pieces
- Building Automation
- Temperature monitoring in the server room
- Energy management, smart metering
- Video monitoring

... and last but not least the Brick'R'knowledge Internet of Things Set with which you can now access your Bricks via the Internet.

First you will get an overview of the basics of the Brick'R'knowledge experimentation system. Then all the bricks and sensors contained in this set are presented. A special feature of this set is the combination of software and hardware. That is, we need a software to write programs, which brings the hardware to life. To do this, we use the Arduino development environment, which can be downloaded free of charge. After all preparations have been made, we can enter the world of the Internet of Things with numerous examples.

With the central IoT brick, you'll learn how to build your first website and control I/O pins with your smartphone. In addition, the set contains a temperature and humidity sensor, which values you can display. The first step towards your own home automation project! You can also view and display data, such as the dollar rate from the Internet. An I2C bus for connecting a 7-segment display or an 18-bit A/D converter is also included!

The Internet of Things is waiting to be discovered by you!

3. Basics of the Brick'R'knowledge system

3.1 Ground brick

One of the most important bricks is the so called ground brick. The ground brick has one connector with four contacts. Usually the middle two contacts are used for signal or power connection. But the outer contacts are intended for the so called ground level. Which means technically a level of 0V. The ground brick connects the both inner contacts with the outer contacts. Therefore it is possible to allow for a current return flow towards the 0V of a power supply invisible to the schematic symbols outside.

The power supply of course must also be connected at one pole (usually the minus pole) to the ground using the ground brick. There is a power brick with an internal ground connection already done and visible in the symbol, and also a battery brick, where both poles are open, and can be connected with a ground brick to the ground level.

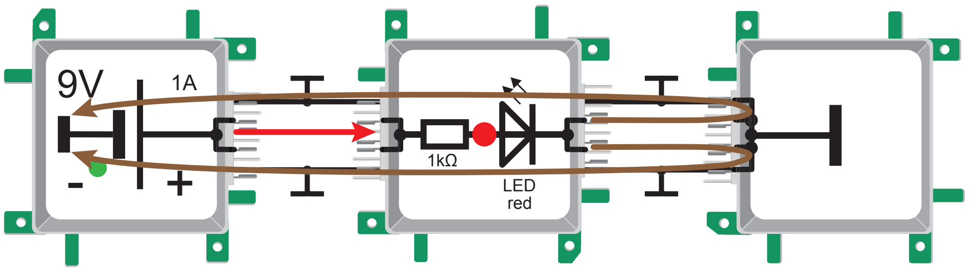


Abb. 1: Ground flow

3.2 The power supply

Abb. 2: Power supply adapter

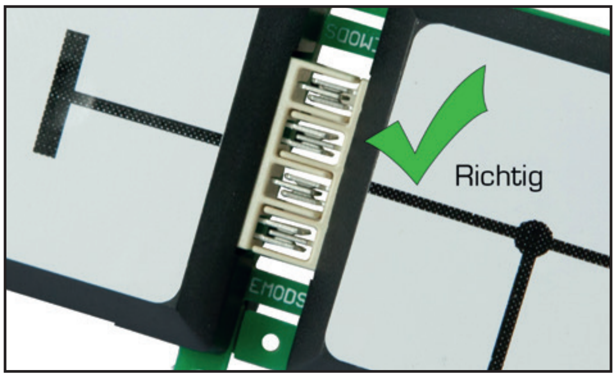
The power supply for the IoT set is provided by the supplied 9 V plug-in power supply (ALL-BRICK-0221). It provides a stabilized DC voltage of 9 V and a maximum current of 1 A. In the event of an overload, the power supply switches off, because it is short-circuit proof. An LED indicates when the brick voltage is available.

Optionally, a supply brick is also available via a 9 V block battery (ALL-BRICK-0001).

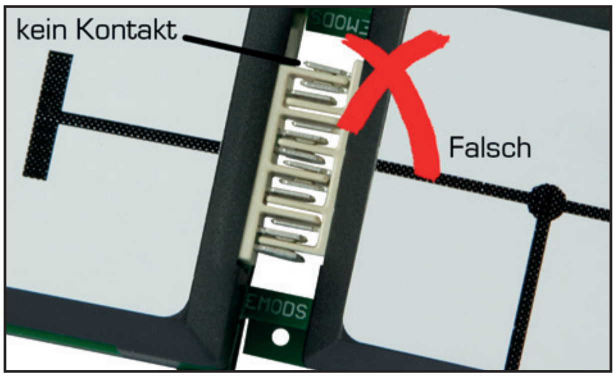
WIf you later put the bricks together in the practice examples, make sure you always plug the power brick as the last brick to your circuit after checking it again. At the end of the experiment, the power supply must be disconnected from the mains!

3.3 The connectors

When the bricks are plugged in, take care that the contacts are connected properly, as there is a risk of interruptions or even short-circuits!



Inserted correctly



Inserted incorrectly

Fig. 3: The connectors

In the left picture you see a correctly inserted connection. The connection consists in each case of small pins, which mechanically jam while making an electrical connection. In order to ensure an insulation between the contacts and to prevent a short circuit, there are little plastic webs inbetween, which do not conduct the electrical current.

An example of a faulty connection is shown in the right figure. Here, insulating strips meet contacts so that no current can flow. The circuit is "open" or unstable, and the circuit does not function.

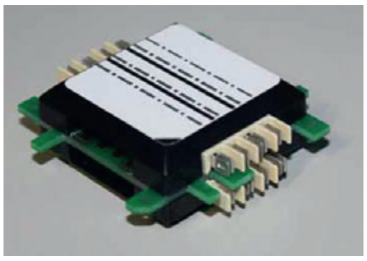
Caution: It is important to always check the correct position of the contact pins. If these diverge too far from each other, a short circuit may occur. Then the current flows through our components with the hoped-for effect, but the shortest path returns to the voltage source.

A short circuit leads to the maximum current, since the only resistance that the electrical current must overcome is the internal resistance of the voltage source. This resistance is clearly very small, so the short-circuit current can lead to overheating over a longer period of time. There is a fire hazard!

Important: Always check the correct position of the contacts!

3.4 Special connection bricks for lower level

The IoT brick also has a signal pin (GPIO12) on the lower pin plane. To get to this contact there are optionally available special bricks like "line downside up" (ALL-BRICK-0385) and "line 6-way straight " (ALL-BRICK-0383). A protective nose prevents the bricks fom above in order to avoid short circuits during the plugging process.



4. The hardware at a glance

The Internet of Things Set contains the following bricks, sensors and accessories, which are briefly presented.

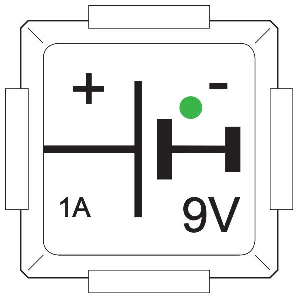
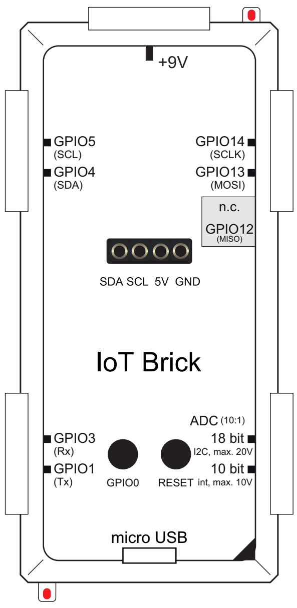
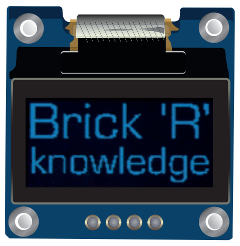
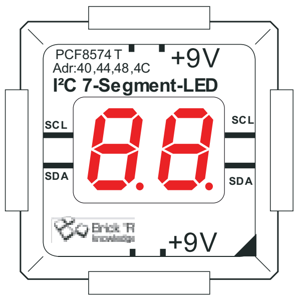
Illustration	Quantity	Art.-Nr. / Brick-ID	Short description
	1	Art.-Nr.: 118627 Brick-ID: ALL-BRICK-0221	**9 V power supply adapter** The 9 V mains adapter provides a maximum current of 1 A for the bricks. It is stabilized and protected against short circuits. An LED indicates when it is in operation. The positive pole is led out and the negative pole is connected to ground. Use this brick as the last one after you have checked the circuit again.
 +9V GPIO5 (SCL) GPIO4 (SDA) GPIO14 (SCLK) GPIO13 (MOSI) n.c. GPIO12 (MISO) SDA SCL 5V GND IoT Brick ADC (10:1) 18 bit I2C, max. 20V 10 bit int, max. 10V GPIO3 (Rx) GPIO1 (Tx) GPIO0 RESET micro USB	1	Art.-Nr.: 136716 Brick-ID: ALL-BRICK-0635	**IoT Brick ESP8266** The heart of the IoT set. ESP8266 module with WLAN interface, power supply: +9 V. 7 GPIOs, a 10 bit A / D converter, an 18 bit A / D converter, I2C interface, SPI interface For further details see chapter 5.1 on page 13.
	1	Art.-Nr.: 139779 Brick-ID: ALL-BRICK-0659	**OLED-Display for IoT Brick** Organic light-emitting diode monochrome. Exactly, 128 x 64 LEDs are arranged in a matrix and can be controlled individually by means of commands through the I2C bus. This makes it possible to display multiline texts, but also simple monochrome graphics. Standard I2C address: $78_{(16)}$
 PCF8574 T Adr:40,44,48,4C +9V I²C 7-Segment-LED SCL SDA Brick 'R' knowledge +9V	1	Art.-Nr.: 113713 Brick-ID: ALL-BRICK-0086	**I^2C 7-segment display** 7-segment display consisting of 7 light bars and one point. Behind it is a light-emitting diode. The LEDs are controlled by means of two 8574T modules(16 output lines to LEDs). The I^2C-address can be set whit small switches on the back. A maximum of four such devices can be operated on an I2C bus.

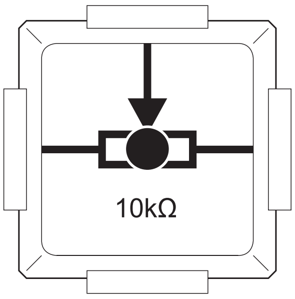
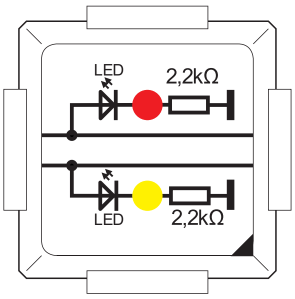
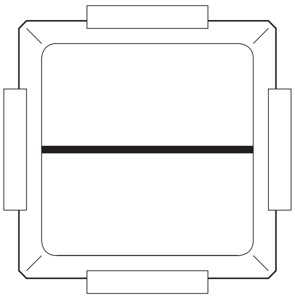
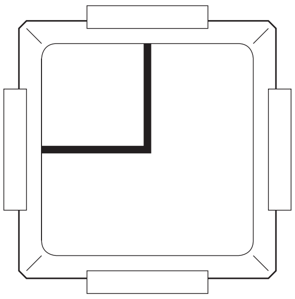
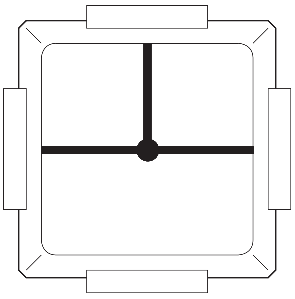
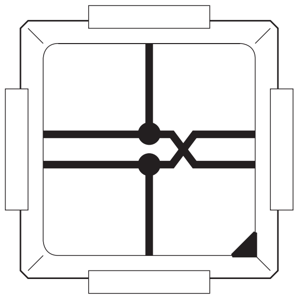
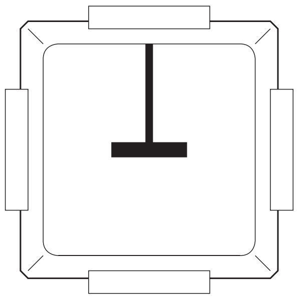
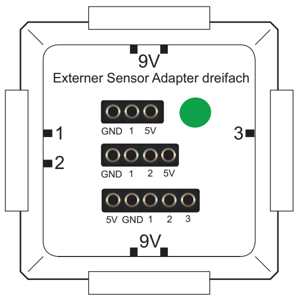
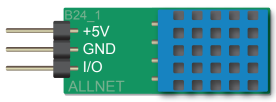
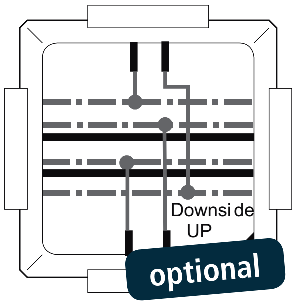
Illustration	Quantity	Art.-Nr. / Brick-ID	Short description
	1	Art.-Nr.: 113654 Brick-ID: ALL-BRICK-0027	**10 kΩ potentiometer** The potentiometer is a changeable resistor. There are three connections. The tapper can be changed mechanically and delivers a resistor size between the smallest and biggest value at the third clip. The potentiometer also has a maximum performance of 1/8 Watts if you connect the tapper with the voltage source and one of the other connectors..
	1	Art.-Nr.: 125693 Brick-ID: ALL-BRICK-0410	**Dual-LED to ground (red / yellow)** Two LEDs (red / yellow) are internally connected to ground. The signal lines are interconnected separately. Both LEDs are protected against excessive current currents via a 2.2 kΩ series resistor. They are designed for 2 mA current at 5 V voltage. The two resistors are internally grounded so that the module can be connected directly.
	1	Art.-Nr.: 113631 Brick-ID: ALL-BRICK-0004	**Straight connector** The straight connector brick connects two opposite bricks.
	3	Art.-Nr.: 113632 Brick-ID: ALL-BRICK-0005	**Corner Brick** You can connect your circuits with the corner brick across the corner.
	1	Art.-Nr.: 113633 Brick-ID: ALL-BRICK-0006	**T-crossing Brick** With the T-crossing brick you can connect your circuit's components like a "T".

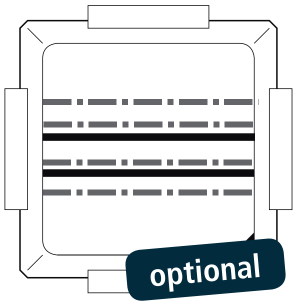
Illustration	Quantity	Art.-Nr. / Brick-ID	Short description
	1	Art.-Nr.: 113675 Brick-ID: ALL-BRICK-0048	**Wire double crossed** With this module you can forward the middle wires separately and cross at the same time.
	1	Art.-Nr.: 113630 Brick-ID: ALL-BRICK-0003	**Ground-Brick** Ground is responsible for leading and returning voltage. Put the ground brick on the end of each circuit in order to close it. It connects the middle circuit with the two external ground lines.
 9V Externer Sensor Adapter dreifach 1 2 3 GND 1 5V GND 1 2 5V 5V GND 1 2 3 9V	1	Art.-Nr.: 138408 Brick-ID: ALL-BRICK-0649	**External sensor adapter triple** Supply voltage: +9 V..
	1	Art.-Nr.: 139778 Brick-ID: ALL-BRICK-0658	**Temperature & humidity sensor** Temperature & humidity sensor Temperature/humidity sensor on PCB, Sensor Type: DHT11, Temp.: 0.50 ° C (± 2 ° C), rel. Humidity: 20..95% (± 5%), Power supply: 3-5,5V. Built-in pullup resistor
	1		**USB-cable** USB connection cable (plug type A to micro USB connector type B) between the development computer and IoT Brick.

Optional recommended bricks



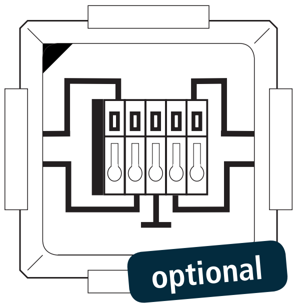
Art.-Nr.: 122448
Brick-ID: ALL-BRICK-0385

Wire Downside Up
A special module for accessing the Pin GPIO12 pin of the lower IoT Bricks. In addition to the contacts on the upper level, it also has four independent contacts on the bottom of the board. These are then routed to the upper level at the sides, where normal bricks can be used to get to the signals.



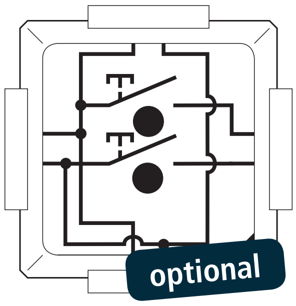
Art.-Nr.: 122446
Brick-ID: ALL-BRICK-0383

Wire 6-way straight
Line 6-way straight
The 6-fold straight line extends the four contacts at the top (including the ground at the external contacts) and the four signal lines at the lower level which are independent of each other.



Art.-Nr.: 125674
Brick-ID: ALL-BRICK-0407

Clamp 5-pole type 2
This allows wiring or components to be connected to the circuit. Use a small screwdriver to press the slot on the top. The contact then opens and the cable can be inserted laterally. When the screwdriver is released, the cable is secure. The middle contact is connected to ground..



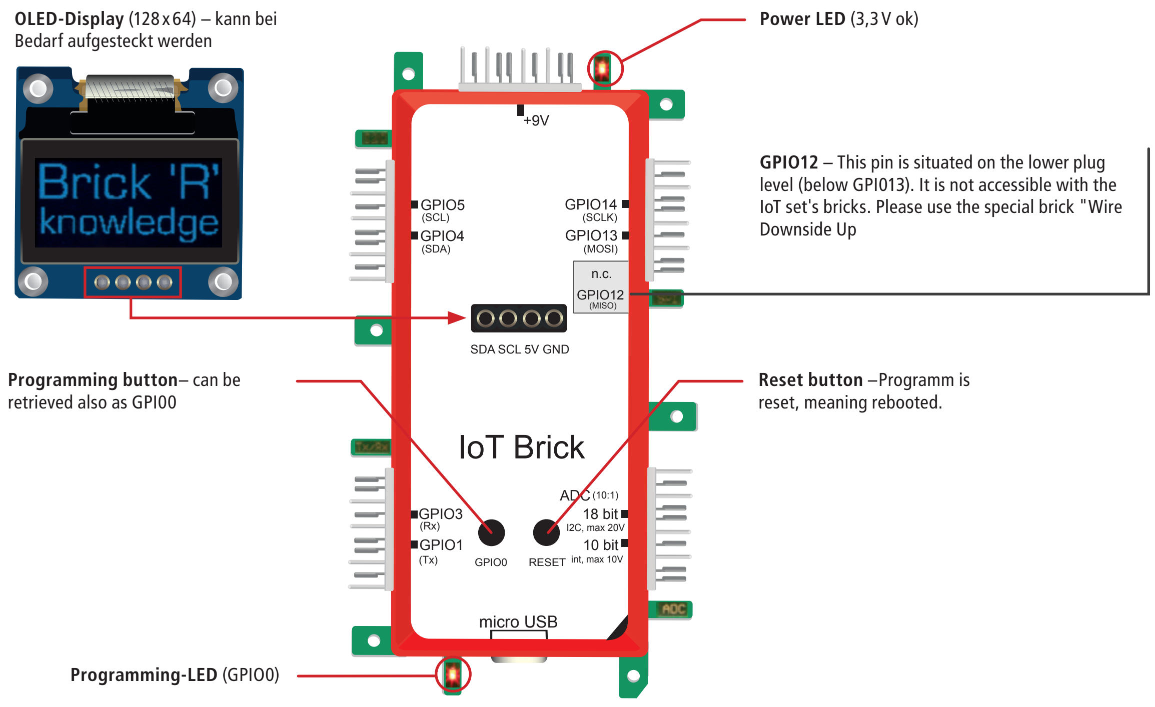
Art.-Nr.: 137824
Brick-ID: ALL-BRICK-0642

Double switch
This brick contains 2 single-pole NO contacts. This allows the inputs of gates or flipflops to be conveniently connected. In addition, the signal paths are connected separately from the upper to the lower connection. Close the power supply At the top and / or bottom connection, thus saving many line bricks in many situations.

5. The IoT brick and the Arduino IDE

5.1 The core of the Internet of Things Set

The IoT brick is the core of the Brick'R'knowledge Internet of Things set. Its central component is the ESP 12 F module by AI THINKER, consisting of a microcontoller type EsP8266 with integrated WLAN interface, a 4MB flash memory and an internal WLAN antenna.



OLED-Display (128 x 64) – kann bei Bedarf aufgesteckt werden
Brick 'R' knowledge
Power LED (3,3 V ok)
+9V
GPIO5 (SCL)
GPIO4 (SDA)
GPIO14 (SCLK)
GPIO13 (MOSI)
n.c.
GPIO12 (MISO)
GPIO12 – This pin is situated on the lower plug level (below GPI013). It is not accessible with the IoT set's bricks. Please use the special brick "Wire Downside Up
SDA SCL 5V GND
Programming button– can be retrieved also as GPI00
Reset button –Programm is reset, meaning rebooted.
IoT Brick
ADC (10:1)
18 bit I2C, max 20V
10 bit int, max 10V
GPIO3 (Rx)
GPIO1 (Tx)
GPIO0
RESET
ADC
micro USB
Programming-LED (GPIO0)

Fig. 4: The core of the Internet of Things set

5.1.1 Specifications of the IoT Brick

Element	Specification
Microcontroller	80 MHz Tensilica L106 Ultra-Low-Power 32 bit MCU
Flash memory	4 MByte for programs
WLAN-Interface	IEEE 802.11 b / g / n protocol, IPv4, IP address: is assigned via DHCP, integrated TCP / IP stack, ISMBand (2.4 GHz), supports WPA / WPA2 security modes, integrated antenna
GPIOs	7 GPIOs, standard function: digital in-/output, alternative functions: I^2C- and SPI-bus GPIO-autput level: +5 V; GPIO-input level: +5 V (the inputs are not 9V tolerant!)
Analogue inputs	**ADC 10 bit:** 10 bit A / D internal converter (type: SAR converter), sampling rate: max. 200 S / s, input voltage range: max. 10 V **ADC 18 bit:** 18 bit A / D converter MCP3421 connected via I2C (type: Delta-Sigma converter), sampling rate: max. 3.75 S / s, input voltage range: max. 20 V
Button	required for programming modus (after starting the ESP8266, button GPIO0 can be used as ordinary digital input.
Power supply	9 V supply (power LED at 3.3 V on board voltage)
Connections	1 x Brick connector for 9 V supply 4 x Brick connectors for a total of 7 GPIOs 1 x Brick connector for 2 analog inputs Micro-USB socket (type B) as programming interface
Display	I^2C-OLED-display monochrome (128 x 64 pixels) plug in via 4 pin female header (included in the set)

5.1.2 The IoT brick's GPIO pins

Analogue and digital inputs and outputs are necessary for a microcontroller to exchange data. The digital inputs and outputs are usually called GPIOs. GPIO is short for General Purpose Input/Output, meaning that such a connection can be used as either in- or output - according to former configuration. In coding this change of direction is called pinMode. The GPIOs voltage level complies with 5V for high level and 0V for low level. Alternatively, some GPIO pins can assume special functions like the data communication via I2C- or SPI-bus. To address the GPIO pin in your programm you need an index that you can take from the following table:

Standard-Function	Description	alternative function	Description (See chap. 5.1.3)	Index for pro-gramming
GPIO0	Index for programming	-	low: 40 kΩ, high: 4 kΩ	0
GPIO1	Digital-I/O 1	TxD	low: 40 kΩ, high: 4 kΩ	1
GPIO3	Digital-I/O 3	RxD	low: 40 kΩ, high: 4 kΩ	3
GPIO4	Digital-I/O 4	I^2C SDA	low: 40 kΩ, high: 4 kΩ	4
GPIO5	Digital-I/O 5	I^2C SCL	low: 40 kΩ, high: 4 kΩ	5
GPIO12**	Digital-I/O 12	SPI MISO	low: 40 kΩ, high: 4 kΩ	12
GPIO13	Digital-I/O 13	SPI MOSI	low: 40 kΩ, high: 4 kΩ	13
GPIO14	Digital-I/O 14	SPI SCLK	low: 40 kΩ, high: 4 kΩ	14
ADC0 (10 bit)	10 bit analog input (internal)	-	-	0
ADC1 (18 bit)	18 bit analog input (via I2C)	-	-	(via I^2C)
The GPIOs 2, 6, 7, 8, 9, 10, 11, 15 and 16 are not provided by the ESP-12-F module.				

* See next chapter. ** Pin on lower plug level only accessible with optional special brick (art.-no .: 122448, Brick ID: ALL-BRICK-0385)

Attention: Don't ever directly apply 9V to the GPIOs (for example, via a switch). Although you need 9V supply for the IoT brick and other active bricks, the GPIOs are designed for 5V level only. A GPIOs voltage level of more than 5V can lead to irreversible damages of the bricks!

Pull-up resistors

It is important that inputs always have a defined level in digital circuits. By installing so called Pull-up or Pull-down resistors, the input is transferred to high level (5V) or low level (0V). We use pull-up resistors with automatic impedance matching in our IoT brick. At low level the value is 40 kΩ (less current flows). At high level the value is 4 kΩ (for the high level to be recognized). Hence, the input level on the GPIOs always is clearly defined so that the logic level on a pinMode (GPIOx, INPUT_PULLUP) or pinMode(GPIOx, INPUT_PULLDOWN) does not have any effect on the GPIO pins as the pull-up resistors are always active. After having started the supply, GPIOs that were configurated as input are immediately converted to high level.

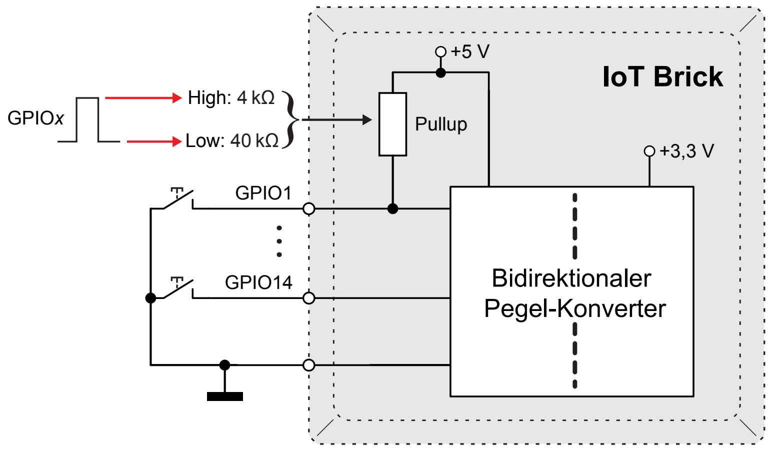
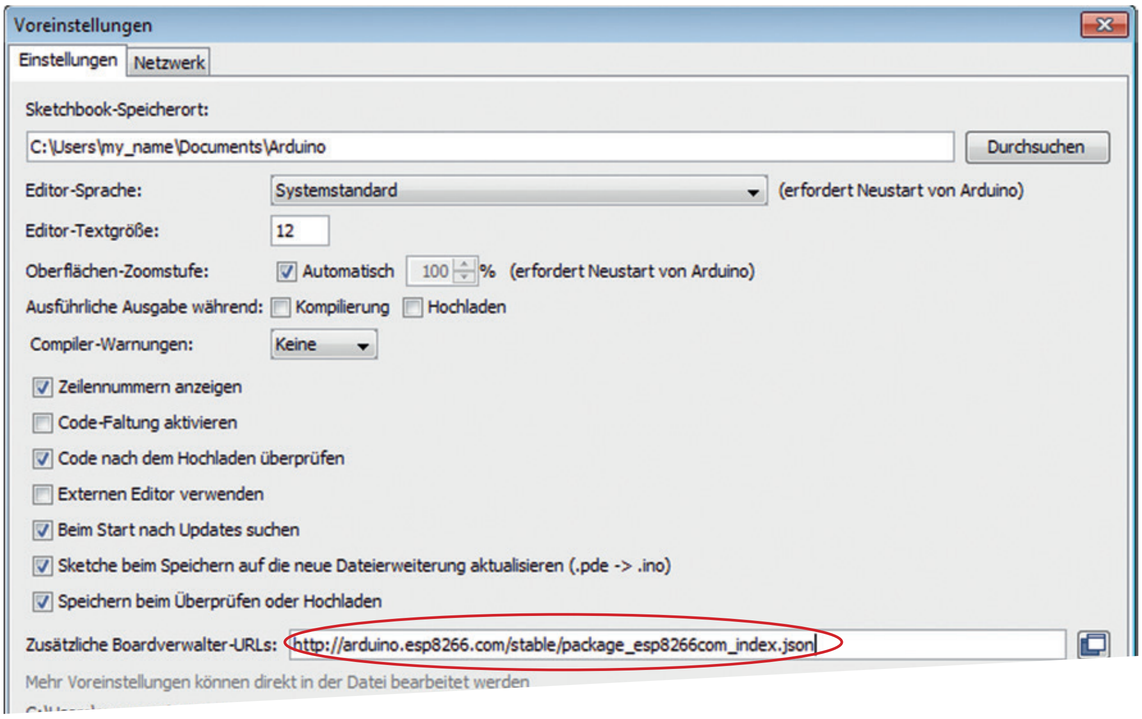


Fig. 5: Pull-up resistors

5.2 The Arduino integrated development environment

- To program the IoT brick we use the Arduino integrated development environment, also known as Arduino IDE. Many of you probably already know Arduino's free coding software. There are various Arduino projects on the internet and there is a huge community for it. We use a number of open source libraries in our examples in order to facilitate the programming of individual hardware components. There are installers for Windows, Linux, MAC OS X and a Windows app. This instruction's descriptions and screenshots refer to the Windows version.
- Download the installer for the current Arduino IDE here: https://www.arduino.cc
- Start installing the Arduino IDE by double clicking on the downloaded EXE file.
- To program the ESP8266 based microcontroller module with this software, you also have to install the respective core including some libraries. This is necessary as the Arduino IDE does not by default support the ESP8266.
- Start the Arduino IDE, for example via the start menu in Windows (differs according to operating system).
- Open the menu "Data - Presets" and add the following URL below "Additional board administrator URLs": http://arduino.esp8266.com/stable/package_esp8266com_index.json



Voreinstellungen

Einstellungen | Netzwerk

Sketchbook-Speicherort: C:\Users\my_name\Documents\Arduino [Durchsuchen]

Editor-Sprache: Systemstandard (erfordert Neustart von Arduino)

Editor-Textgröße: 12

Oberflächen-Zoomstufe: ✓ Automatisch 100 % (erfordert Neustart von Arduino)

Ausführliche Ausgabe während: ☐ Kompilierung ☐ Hochladen

Compiler-Warnungen: Keine

- ✓ Zeilennummern anzeigen
- ☐ Code-Faltung aktivieren
- ✓ Code nach dem Hochladen überprüfen
- ☐ Externen Editor verwenden
- ✓ Beim Start nach Updates suchen
- ✓ Sketche beim Speichern auf die neue Dateierweiterung aktualisieren (.pde -> .ino)
- ✓ Speichern beim Überprüfen oder Hochladen

Zusätzliche Boardverwalter-URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

Mehr Voreinstellungen können direkt in der Datei bearbeitet werden

Fig. 7: Board administrator

- Confirm with OK.
- Open "Board administrator" in the menu "Tools Board:.."

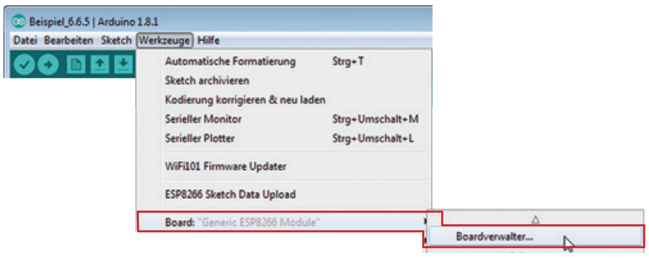
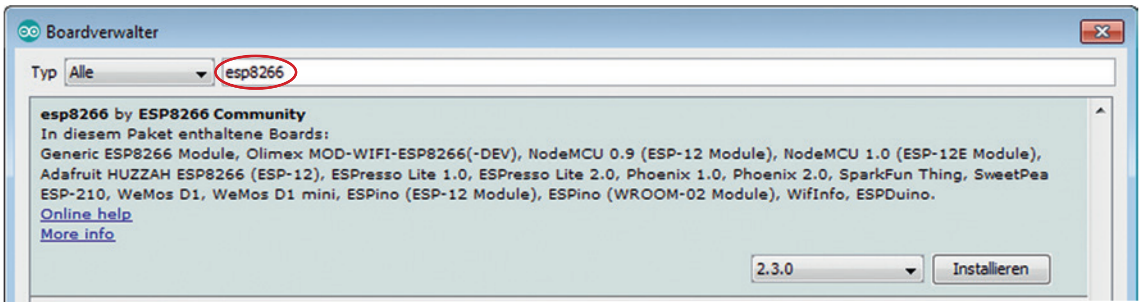

Beispiel_6.6.5 | Arduino 1.8.1
Datei
Bearbeiten
Sketch
Werkzeuge
Hilfe
Automatische Formatierung Strg+T
Sketch archivieren
Kodierung korrigieren & neu laden
Serieller Monitor Strg+Umschalt+M
Serieller Plotter Strg+Umschalt+L
WiFi101 Firmware Updater
ESP8266 Sketch Data Upload
Board: "Generic ESP8266 Module"
Boardverwalter...

Fig. 4: Board administrator

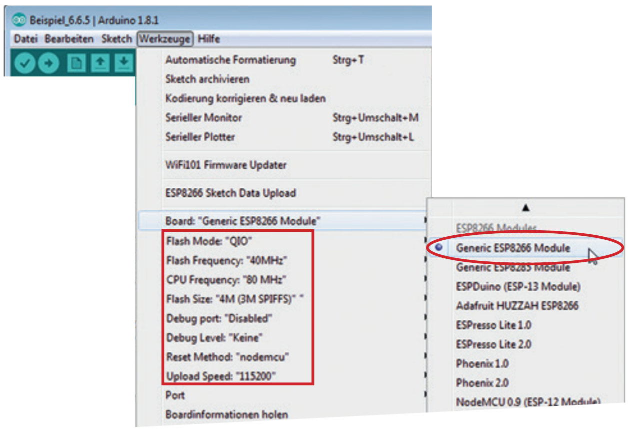
- Enter esp8266 in the search window. You should only see: "esp8266 by ESP8266 Community"



esp8266 by **ESP8266 Community**
In diesem Paket enthaltene Boards:
Generic ESP8266 Module, Olimex MOD-WIFI-ESP8266(-DEV), NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Adafruit HUZZAH ESP8266 (ESP-12), ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, SparkFun Thing, SweetPea ESP-210, WeMos D1, WeMos D1 mini, ESPino (ESP-12 Module), ESPino (WROOM-02 Module), WifInfo, ESPDuino.
Online help
More info

Fig. 5: Installation ESP8266 via board administrator

- Click on the entry and then click on "install". Installation can take some time.
- End the installation with "close"
- In the menu, choose "tools - board:...-"board administrator...".
- Choose the entry "Generic ESP8266 module". Once chosen, the menu below "board: ..." changes and displays various settings. Change them in such a way that they correspond to the screenshot (red triangle).



Board: "Generic ESP8266 Module"
Flash Mode: "QIO"
Flash Frequency: "40MHz"
CPU Frequency: "80 MHz"
Flash Size: "4M (3M SPIFFS)" "
Debug port: "Disabled"
Debug Level: "Keine"
Reset Method: "nodemcu"
Upload Speed: "115200"

Fig. 8: Installation ESP8266 via board administrator

- Go on with the following chapter "Installing libraries...

5.2.1 Installing libraries

We use so called libraries in our example programs in order to simplify coding. These collections of coding involve, for example, extensive tables that can define fonts or that encode 7 segment displays. Although most of the libraries are delivered with the Arduino IDE, they still have to be installed. Other libraries need to be downloaded from the internet and are usually integrated as ZIP files. Once you have installed all libraries that were listed here, you are fully prepared and don't have to install anymore for the exercises.Usually, you find your installed libraries in the follwing list on your PC: C:\Users\my_name\Documents\Arduino\libraries (substitute my_name with your user name).

You find advise for installing additional Arduino libraries here: https://www.arduino.cc/en/Guide/Libraries

5.2.1.1 Installing Arduino libraries

- Installing Arduino libraries

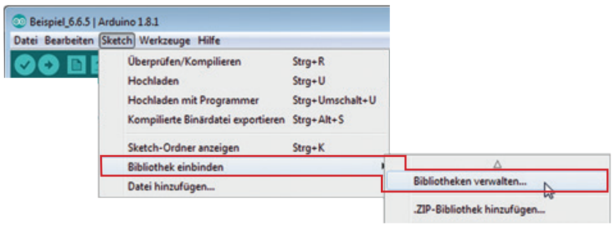
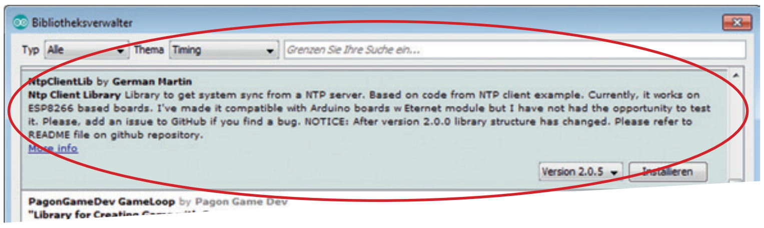

Beispiel_6.6.5 | Arduino 1.8.1
Datei Bearbeiten Sketch Werkzeuge Hilfe
Überprüfen/Kompilieren Strg+R
Hochladen Strg+U
Hochladen mit Programmer Strg+Umschalt+U
Kompilierte Binärdatei exportieren Strg+Alt+S
Sketch-Ordner anzeigen Strg+K
Bibliothek einbinden
Datei hinzufügen...
Bibliotheken verwalten...
.ZIP-Bibliothek hinzufügen...

Fig. 6: Bibliotheksverwalter

- Install the necessary libraries according to the following chapters. Limit the choice by using adequate search terms (see followinig screenshots). In case a version selection is offered, you can usually install the latest version. Click on the entry and then click on "install.

5.2.1.1.1 Library"NTPClient"



Typ Alle Thema Timing Grenzen Sie Ihre Suche ein...

NtpClientLib by **German Martin**
Ntp Client Library Library to get system sync from a NTP server. Based on code from NTP client example. Currently, it works on ESP8266 based boards. I've made it compatible with Arduino boards w Eternet module but I have not had the opportunity to test it. Please, add an issue to GitHub if you find a bug. NOTICE: After version 2.0.0 library structure has changed. Please refer to README file on github repository.
More info

Version 2.0.5 Installieren

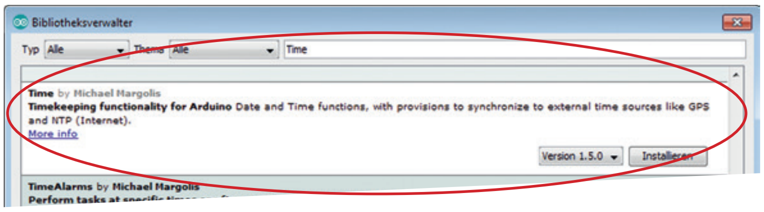
PagonGameDev GameLoop by Pagon Game Dev
"Library for Creating

Fig 11: Installating library "NTPClientLib"

- Choose the library with the latest version and click on "install".
- The library is needed to obtain time and date via Network Time Protocol (NTP) from the internet.

5.2.1.1.2 Library "Time"

The library is needed to obtain time and date via Network Time Protocol (NTP) from the internet.



Typ Alle Thema Alle Time

Time by Michael Margolis
Timekeeping functionality for Arduino Date and Time functions, with provisions to synchronize to external time sources like GPS and NTP (Internet).
More info

Version 1.5.0 Installieren

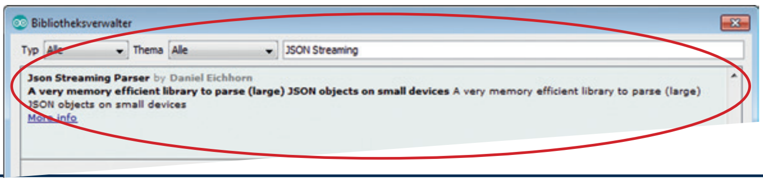
TimeAlarms by **Michael Margolis**
Perform tasks at

Fig. 12: Installation Library "Time"

- Choose the library with the latest version and click on "install".
- The header file is integrated with the instruction: #include <Time.h> and #include <TimeLib.h>.

5.2.1.1.3 Library "Json Streaming Parser"

The library is needed to obtain time and date via Network Time Protocol (NTP) from the internet.



Typ Alle Thema Alle JSON Streaming

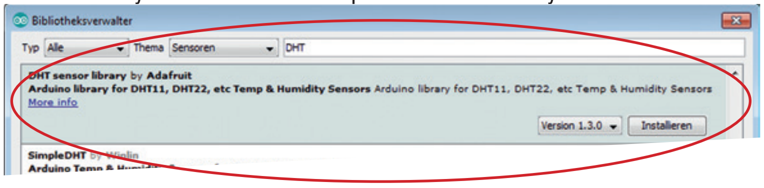
Json Streaming Parser by Daniel Eichhorn
A very memory efficient library to parse (large) JSON objects on small devices A very memory efficient library to parse (large) JSON objects on small devices
More info

Fig. 7: Installation library "Json Streaming Parser"

- <TimeLib.h>.
- The header files are integrated with the instructions #include <DHT.h> and #include <DHT_U.h>.

5.2.1.1.4 Library"DHT Sensor Library"

The library is needed to read temperature and humidity of sensor DHT11.



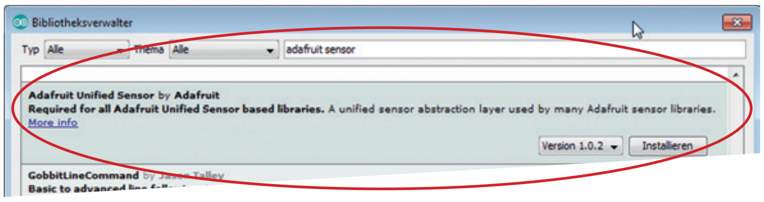
DHT sensor library by **Adafruit**
Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors
More info

Fig. 8: Installing Library "DHT Sensor Library"

- Select the library with the latest version and click on "Install".
- The header files are included with the instructions #include <DHT.h> and #include <DHT_U.h>.

5.2.1.1.5 Library "Adafruit Unified Sensor"

The library is needed to read temperature and humidity of sensor DHT11.



Adafruit Unified Sensor by **Adafruit**
Required for all Adafruit Unified Sensor based libraries. A unified sensor abstraction layer used by many Adafruit sensor libraries.
More info

Fig. 9: Installing library "Adafruit Unified Sensor"

- Choose the library with the latest version and click on "install".
- The header file is integrated with the instruction: #include <Adafruit_Sensor.h>.

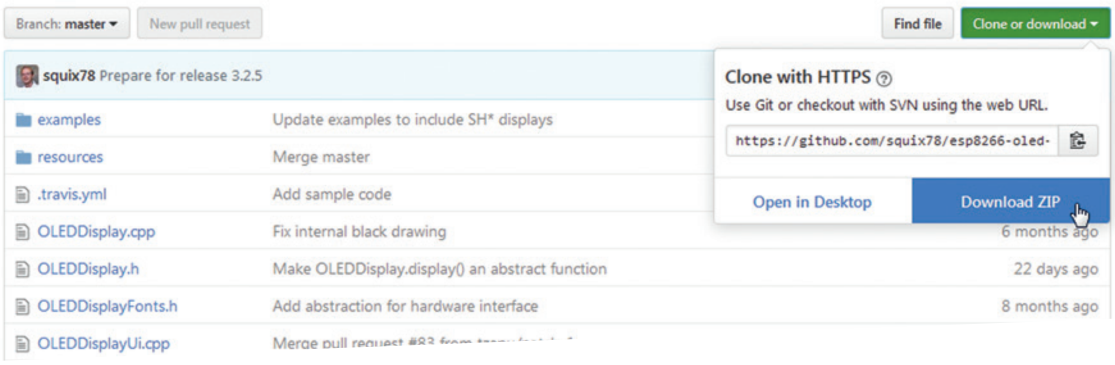
5.2.1.2 Installing external libraries

- The same procedure applies for each library
- You need to download the necessary library from the internet. You find the respective download pages in the following chapters.
- Open the menu "Sketch - integrate library - Add ZIP library...".
- Choose the downloaded ZIP file.

5.2.1.2.1 Library "esp8266-oled-ssd1306-master"

To simplify controlling the OLED display, we use the library "esp8266-oled-ssd1306-master". You can download it from the GitHub page as a ZIP file.

Go to: https://github.com/squix78/esp8266-oled-ssd1306. Below "Clone or download" choose the option "Download Z



squix78 Prepare for release 3.2.5

```
https://github.com/squix78/esp8266-oled-
```

examples	Update examples to include SH* displays	
resources	Merge master	
.travis.yml	Add sample code	
OLEDDisplay.cpp	Fix internal black drawing	6 months ago
OLEDDisplay.h	Make OLEDDisplay.display() an abstract function	22 days ago
OLEDDisplayFonts.h	Add abstraction for hardware interface	8 months ago
OLEDDisplayUi.cpp	Merge pull request #83 from	

Fig. 16: Installing library "esp8266-oled-ssd1306-master"

- Install the library via the menu "Sketch - include library - add ZIP library"
- The header file is integrated with the instruction: #include <SSD1306Wire.h>.

5.2.1.2.2 Library "MCP3421"

To simplify communication with the 18 bit A/D-converter MCP3421, we use the library "MCP3421". It can be downloaded as a ZIP file.

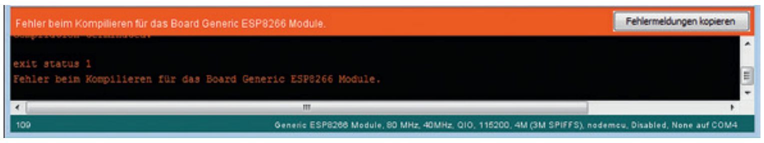
- Folow the link: http://interface.khm.de/index.php/lab-log/connect-a-mcp3421-18-bit-analog-to-digital-converter-to-an-arduino-board/. Below "Library download" at the end of this website you will find the link for the necessary library.
- Download the file MCP3421.zip and save it on your PC.
- Install the library via the menu "Sketch - Include library - add ZIP library".
- The header file is integrated with the instruction: #include <MCP3421.h>.

5.2.1.2.3 Library "BrickESP8266"

The library is needed, among other things, to retrieve the current US dollar exchange from the internet. It can be downloaded as ZIP file.

- Folow the link: http://www.brickrknowledge.de/downloads.
- Download the file BrickESP8266.zip and save it on your PC.
- Install the library via the menu "Sketch - Include library - add ZIP library".
- The header file is included with the instruction: #include <CurrencylayerClient.h>.

In case a library is missing, it will be indicated by an error report while compiling:



Fehler beim Kompilieren für das Board Generic ESP8266 Module.

```
exit status 1
Fehler beim Kompilieren für das Board Generic ESP8266 Module.
```

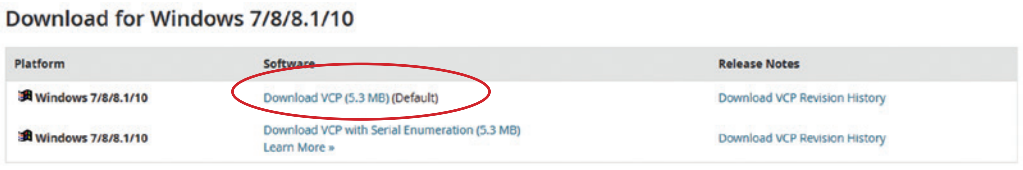
109 Generic ESP8266 Module, 80 MHz, 40MHz, QIO, 115200, 4M (3M SPIFFS), nodemcu, Disabled, None auf COM4

In case you haven't installed the driver for the USB-to-UART-bridge, please continue with 5.2.2.

5.2.2 Virtual COM-Port-Driver

For your development computer to communicate with the IoT brick, you need to select your PC's interface in the Arduino IDE to establish the connection to the IoT brick. For this, you can select a serial port (also called COM port) in the Arduino IDE. Typically, these are RS-232-ports but in modern computers they are hardly to be found. Instead, we use a free USB port of your computer and lead the Arduino IDE to believe in it being a serial port. We have to install a virtual COM-port.driver in order to make this clear to the operating system. This is a requirement for the communication between Arduino IDE and the USB port of the IoT brick.

- Download the current driver for your operating system (Windows, MAC OS X, Linux) from the Silicon Labs
- https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx

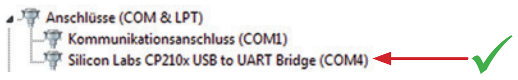


Download for Windows 7/8/8.1/10

Platform	Software	Release Notes
Windows 7/8/8.1/10	Download VCP (5.3 MB) (Default)	Download VCP Revision History
Windows 7/8/8.1/10	Download VCP with Serial Enumeration (5.3 MB) Learn More »	Download VCP Revision History

Fig. 18: Screenshot of the download page including a link for current Windows versions

- Unpack the packed file on your PC.
- Start installation by double clicking. According to the Windows version you need to start the respective installer: CP210xVCPInstaller_x86.exe for a 32 bit Windows version or CP210xVCPInstaller_x64.exe for a 64 bit version.
- In the Windows device manager below "Connectors (COM & LPT)" .
- In the Windows device manager below "Connectors (COM & LPT)".The number of COM ports and the index depend on your PC's configuration.

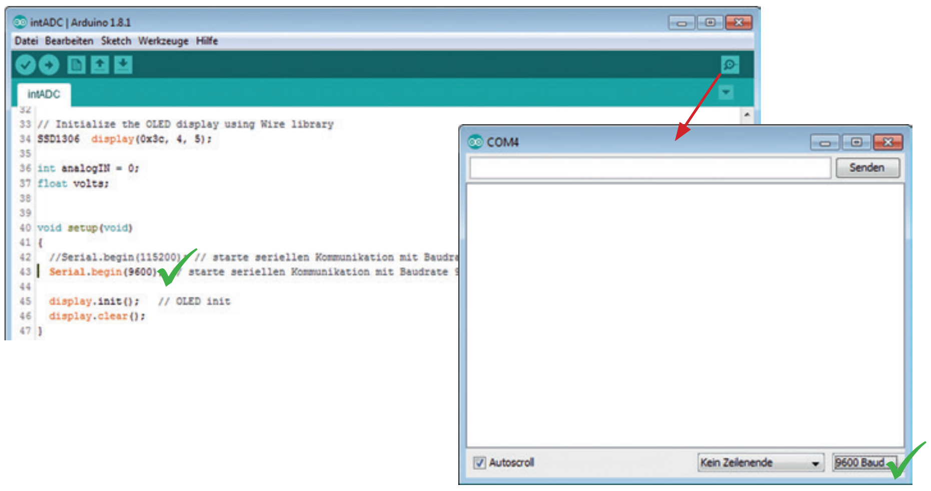


Anschlüsse (COM & LPT)
Kommunikationsanschluss (COM1)
Silicon Labs CP210x USB to UART Bridge (COM4)

Fig. 19: Virtual COM port in the device manager

5.2.3 Serial monitor

Some sample programs use the so called serial monitor to directly display values and notifications at the development computer. Click on the magnifier icon in the Arduino IDE to open the serial monitor. The baud rate in the monitor window (downright) and in the sketch need to be consistent (see Fig. 20).



```
// Initialize the OLED display using Wire library
SSD1306  display(0x3c, 4, 5);

int analogIN = 0;
float volts;


void setup(void)
{
  //Serial.begin(115200); // starte seriellen Kommunikation mit Baudra
  Serial.begin(9600); // starte seriellen Kommunikation mit Baudrate 9

  display.init();   // OLED init
  display.clear();
}
```

9600 Baud

Fig. 20: Issue data on a serial monitor

5.3 First steps

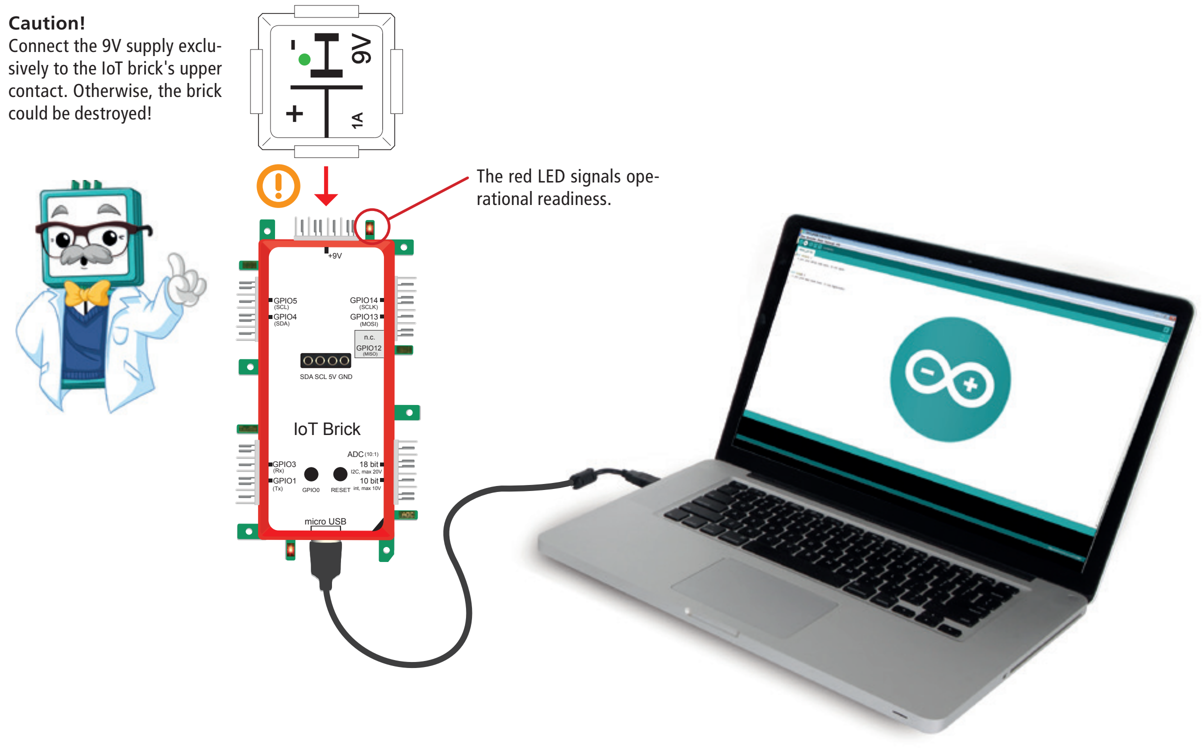
Now it's getting exciting! After having installed the Arduino IDE including the libraries and the virtual COM port driver, you will start operating.

5.3.1 Establish connection

Connect the IoT brick with the enclosed 9V power supply at the upper port (see Fig. 21).

Caution: Don't ever connect the 9V supply to the other IoT brick's contacts as this could destroy the brick!

Connect your development computer and the IoT brick (Micro USB port) with the included USB cable.



Caution!
Connect the 9V supply exclusively to the IoT brick's upper contact. Otherwise, the brick could be destroyed!
9V
1A
The red LED signals operational readiness.
IoT Brick
micro USB

Fig. 21: Basic connection of the IoT brick

The computer should have a new, so called virtual COM port now via that the Arduino IDE loads the programs on the IoT brick. You can easily work out the COM port's index via the Arduino IDE.

- First, disconnect the USB cable from the IoT brick.
- Start the Arduino software and examine which COM port(s) are displayed below "tools - port:...".
- Connect the USB cable with the IoT brick again. Below "tools - port:.." you should see an additional COM port. Select this port. In case this should not be possible, uninstall the COM port in the device manager and reinstall the "CP210x USB to UART Bridge" driver. Compare ch. 5.2.2. on page 20.

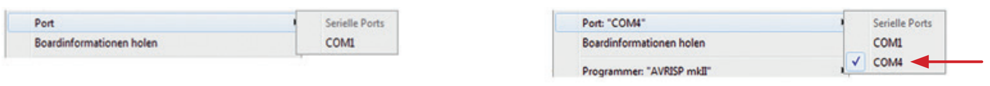
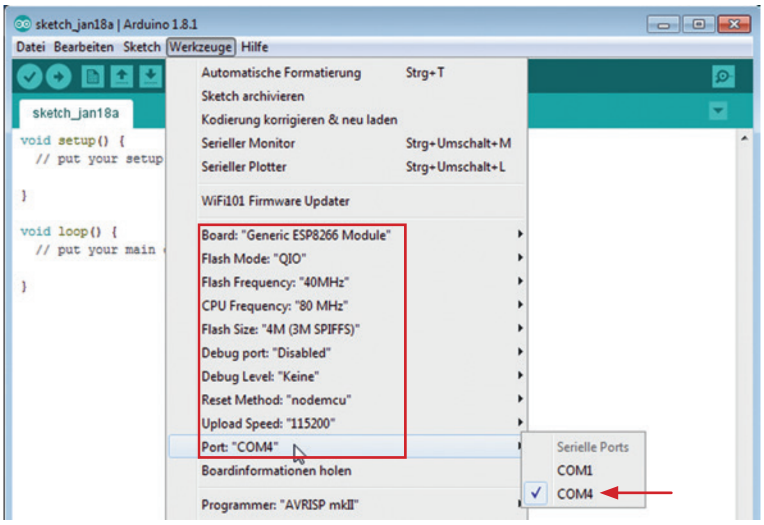


Fig. 22: Left: USB cable not connected, right: USB cable connected (virtual COM port selected)

You have to select or examine more parameters in the menu "tools". The settings displayed in the following screenshot need to be selected. Below "port:" you have to select the COM port that was assigned to the virtual COM port before.


sketch_jan18a | Arduino 1.8.1
Datei Bearbeiten Sketch Werkzeuge Hilfe
sketch_jan18a
Automatische Formatierung Strg+T
Sketch archivieren
Kodierung korrigieren & neu laden
Serieller Monitor Strg+Umschalt+M
Serieller Plotter Strg+Umschalt+L
WiFi101 Firmware Updater
Boardinformationen holen
Serielle Ports
COM1

```
void setup() {
  // put your setup
}

void loop() {
  // put your main
}
```

Board: "Generic ESP8266 Module"
Flash Mode: "QIO"
Flash Frequency: "40MHz"
CPU Frequency: "80 MHz"
Flash Size: "4M (3M SPIFFS)"
Debug port: "Disabled"
Debug Level: "Keine"
Reset Method: "nodemcu"
Upload Speed: "115200"
Port: "COM4"
Programmer: "AVRISP mkII"
✓ COM4

Fig. 10: Settings for IoT brick

5.3.2 Compiling and uploading program code

By clicking on the button with the checkmark, the program code - the Arduino world calls it Sketch - is examined for mistakes in syntax and is compiled. Mistakes in syntax are typos in your Sketch or violations against rules of the programming language. While compiling, your program code is translated into machine code that our microcontroller can perform. This is done automatically by a compiler (translator) that is integrated into the Arduino IDE.

After having successfully compiled your program code, you can upload the program to your IoT brick by clicking on the arrow icon. The red LED on the lower left close to the USB port on the IoT brick will light up while the program is loaded into the flash memory. Beginning and ending of the loading process is signaled by a short flashing. Once the upload is done, the LED will turn off.

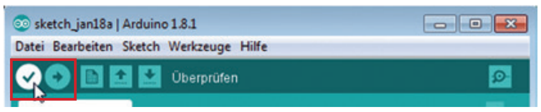

sketch_jan18a | Arduino 1.8.1
Datei Bearbeiten Sketch Werkzeuge Hilfe
Überprüfen

Fig. 24: Examine sketch, compile and upload

In the next chapter you will get to know how to start the programming mode manually.

5.3.3 Programming mode

Use this practice in case you want to start the programming mode manually. This is only necessary if the communication between Arduino IDE and IoT brick does not work.

1. Keep pressing the programming key (GPIO0) (lower left red LED lightens up)
2. Press reset key at the same time
3. Stop pressing reset key
4. Stop pressing programming key (lower left red LED is only slightly illuminate

6. Examples

6.1 "Hello World" (blinking LED)

Open the sketch in the Arduino IDE: Example_6.1.ino

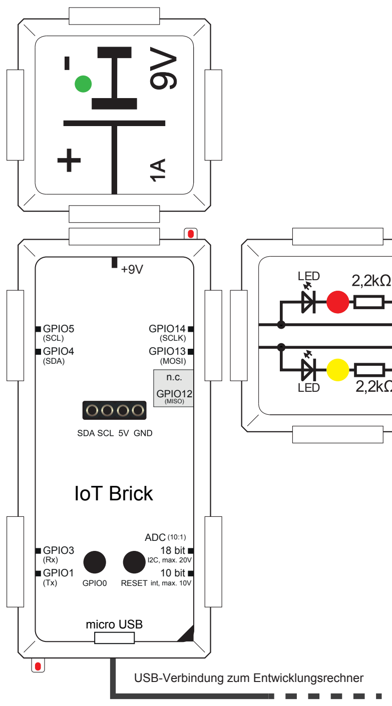
As usual in the coding world, it starts with a "Hello World" example. In this case, we will have two LEDs blink alternately. A program - also called sketch in the Arduino world - consists of at least two parts.

The Setup-Routine void setup()

First, there is the part void setup() {…}. When starting the program, all commands within this brace are carried out exactly once. In our example, the GIPI pins' direction is defined, meaning (mode: input) or, as in our example: two outputs (mode: output)

Program loop `void loop()`

The actual program is included in the brace of void loop() {…} and is permanently repeated. The commands are carried out in an infinite loop until the program is stopped by pressing the reset button, for example. In our first coding example, (Beispiel_6.1.ino) the double LED brick is connected to the IoT brick's GPIOs 13 and 14.



```
void setup() {
  pinMode(14, OUTPUT);     // Pin 14 is output
  pinMode(13, OUTPUT);     // Pin 13 is output
}

void loop() {
  digitalWrite(14, HIGH); // LED on Pin 14 an
  digitalWrite(13, LOW);  // LED on Pin 13 aus

  delay(1000);            // wait for 1000ms

  digitalWrite(14, LOW);  // LED on Pin 14 aus
  digitalWrite(13, HIGH); // LED on Pin 13 an

  delay(1000);            // wait for 1000ms
}
```

Fig. 25: Example: "Hello World" (blinking LED

GPIO 13 and 14 are defined as output in the setup routine with pinMode(GPIOx, OUTPUT) .

Hence, these pins are able to issue a high level to have both LEDs blink alternately.

In the loop, the pin with digitalWrite(GPIOx, HIGH) is set on high level.

This means that each GPIO issues 5V which is why the LED turns on. With the command digitalWrite(GPIOx, LOW) the pin is set to 0V – the LED turns off.

The program stops for the respective number of milliseconds with the command delay(…).

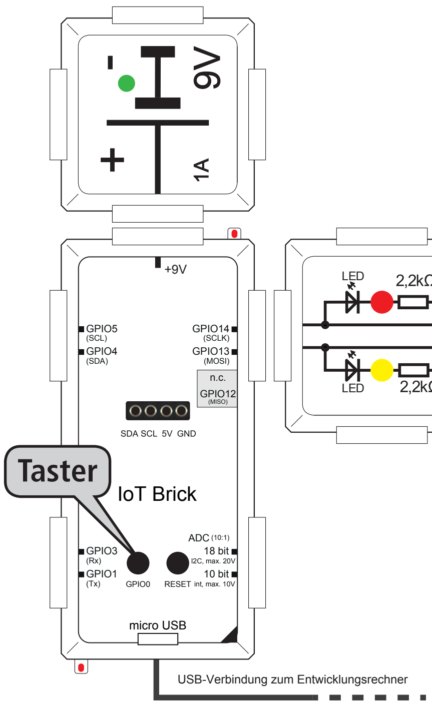
In this example it is 1000 milliseconds that complies with one second. After having uploaded the coding example to your IoT brick (see chapter 5.3.2 on page 22), both LEDs blink alternately. Play with the code a little bit. Change the order in which the LEDs turn on and off and change the time intervals inbetween.

6.2 Button and LED

Open the following sketch in the Arduino IDE

In the following example, we want the LEDs to lighten up dependent on a button. In addition to the reset button, the IoT brick includes a further button that is connected to the GPIO0. Hence, we use the GPIO0 as input and read the status of the button. To prevent an undefined level at this input as long as you don't push the button, an internal pull-up resistor was included that moves the input to high level (see chapter 5.1.3 on page 14). As soon as the button is pushed there are 0V at the input. This way, you can clearly tell if the button is pushed or not.

Don't get confused that the coding LED bottom left is lightened up as long as you push the button. The LED and the button are connected and don't have anything to do with your code.



```
void setup() {
  pinMode(0, INPUT);    // Pin 0 ist Tastereingang

  pinMode(14, OUTPUT);    // Pin 14 ist Ausgang
  pinMode(13, OUTPUT);    // Pin 13 ist Ausgang
}

void loop() {
  if (digitalRead(0)==LOW)
  {              // wenn Taste gedrückt wird

    digitalWrite(14,HIGH);     // LED an Pin 14 an
    digitalWrite(13,LOW);      // LED an Pin 13 aus

  }
  else{         // wenn Taste NICHT gedrückt wird
    digitalWrite(14,LOW);      // LED an Pin 14 aus
    digitalWrite(13,HIGH);     // LED an Pin 13 an
  }

  delay(100);           // warte für 100ms
}
```

Fig. 26: Example: Button and LED

As long as the button is not pushed, the yellow LED lightens up at GPIO13. As soon as you push the button, the red LED lightens up at GPIO14.

The command delay(100) at the end of the loop is responsible for the program waiting 100 milliseconds at this point. The microcontroller can "take a break" during this time.Now it's your turn again! Vary the code a little in order to better understand it. For example, make the LEDs lighten up in sequence as soon as you push the button.

Caution: Don't us an external button that is directly connected to 9V as the GPIOs are only made for 5V levels. We advise to use the integrated button GPIO0.

At a voltage level of more than 5V on the GPIOs, the IoT brick can get irreversible damages!

6.3 I^2C-bus

The I²C bus is a serial interface that gets by with two lines, the clock line SCL (Serial Clock) and the data line SDA (Serial Data). The lines work bidirectional. We differentiate between master and slave among the network devices. In this case, the IoT brick is the master and the other bricks are the slaves. The network devices are addressed via I²C addresses. There are 128 per bus possible. Individual network devices can cover various addresses. Some network devices have little DIP switches on the back, making it possible to change the address range when several network devices are used at the same bus. Basically, the bus can be used with different speed:

Mode	Speed
Standard Mode (Sm)	0,1 Mbit/s
Fast Mode (Fm)	0,4 Mbit/s
High Speed Mode (HS-Mode)	1,0 Mbit/s
Ultra Fast-Mode (UFm)	5,0 Mbit/s

Many microcontrollers are able to control the first two modes only (for example, the controller of the Arduino Nano), some of them also control the third mode. The same is true for the companion devices. Naturally, the modes need to fit together. The master, meaning the microcontroller, sets the pace on the SCL line. The acutal datat is transferred via the SDA line.

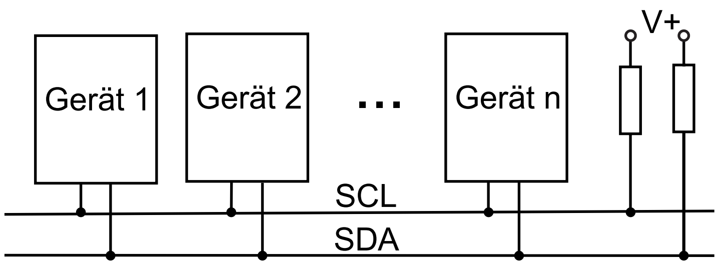


Fig.11: I^2C-Busstruktur

You can connect a maximum of 128 devices to an I²C bus as long as each device occupies one address only, otherwise accordingly less. The devices are connected via two bus lines. Both pull-up resistors (in the kΩ unit) are already integrated in the IoT brick.

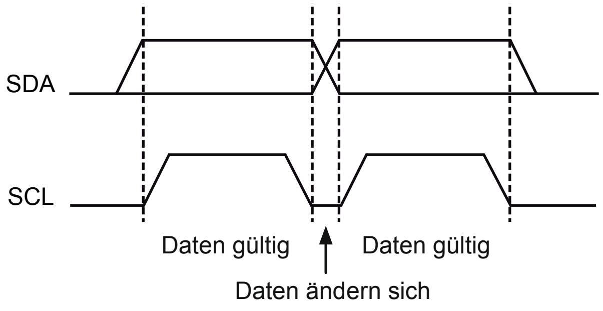


Fig. 12: Valid data at the I^2C- bus

The clock indicates when there is valid data received. As you can see in fig. 28, this is always the case at the high level of the SCL line. The receiver can now sample and evaluate the data. The master sets the pace, either creating data itself or expecting this data from the respective device.

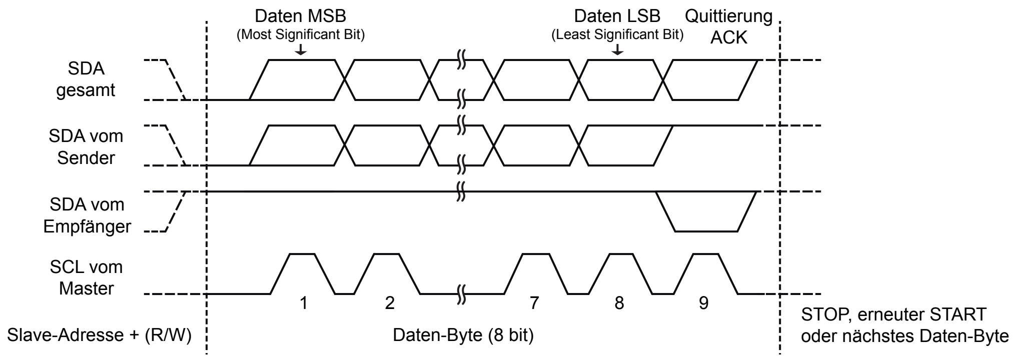


Fig. 29: Data transfer on the I²C bus

In Fig. 29 you can see the temporal course of a data transfer with the following signals (from bottom to top): the clock signal SCL (indicated by the master), the data line from receiver's perspective (receiver) is not directly activated as it is low-active. The data bits, as they were sent by the transmitter (low-active) and at the very top the SDA signal in the overall view. The synchranisation is important. A receiver (whether master or slave) sends an acknowledgement signal (ACK = acknowledge) at the end of each data packet by moving the SDA line to low. As this is equivalent to a Wired-OR, it is sufficient that a slave sends the ACK signal.

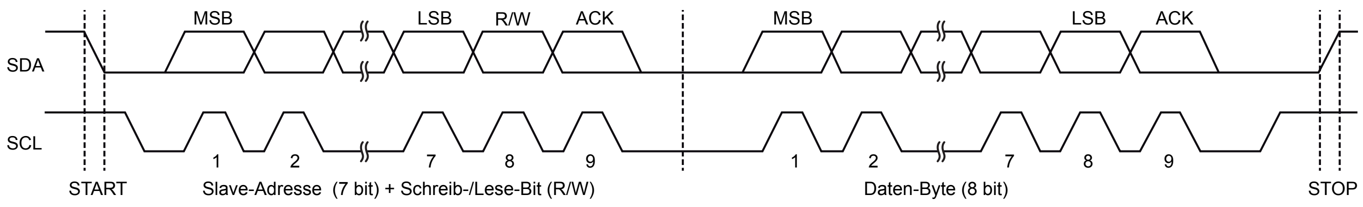


Fig. 30: transmission cycle at the I²C bus

You can see the complete transmission cycle in Fig. 30. First, a package including the address is sent. The address consists of 7 bits, including an additional one, the so called R/W (Read/Write) bit. All network devices compare the emitted address with their own. When they correspond, the respective slave acknowledges this with an ACK signal by briefly setting the SDA line on low. Depending on the R/W bit, the addressed slave knows whether to start a sending- or receiving cycle. Afterwards, the acutal data transfer can begin. In the end, a stop cycle is initiated. For this, the clock is put on high and the SDA line is released. The lines SDA and SCL are both on high level, meaning that the I²C bus can be used freely. It would be possible that a different master starts a new cycle (in case there are more than one at the bus).

The I²C bus is easy to use for us as the Arduino library provides various commands to control the bricks with I²C interface (e.g. 7 segment indicator brick) via the IoT brick.

6.3.1 The 7 segment indicator

In the early days of computer technology they cared about how to display numbers. The easiest way to do that was with 10 lights that were labelled from 0 to 9. Afterwards, they used the lights for illuminating little glass plates with the respective holes. At the same time, the so called "Nixi tubes" were introduced, the numbers were made of wire and started to lighten up when applying a higher voltage under protective gas atmosphere. Then they came up with the idea to devide the numbers into segments. You can display all numbers between 0 and 9 with seven segments. The first displays still used glow wire but it became easier when the LEDs were invented. Behind each segment there is a LED that illuminates the bars. The individual segments are frequently indicated with "a to g ". In addition, there is a single LED for the decimal point (dp).

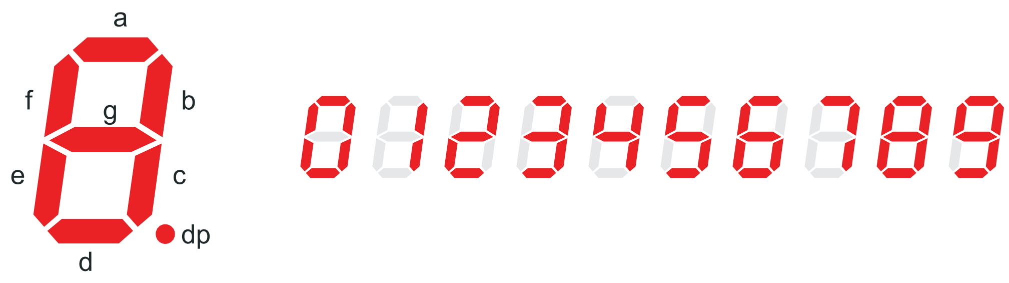


Fig. 31: The 7 segment indicator

Like this, you can easily display the numbers 0 to 9. We will get to know a more elegant way of displaying when dealing with the OLED display (see chapter 6.4 on page 32). With it you can also depict simple graphics.

There are two 7 segment indicators included in our 7 segment indicator brick that are connected via the I²C bus. The display on the left is called MSB sign (Most Significant Bit) while the right display is called LSB sign (Least Significant Bit) - see chapter 6.5.2.5 on page 38. Both 7 segment indicators are controlled each via an I/O extender type 8574T. This component is responsible for decoding the address on the I²C bus and the data bytes (here: numbers) for controlling the seven segments including the LED driver stage.

6.3.2 7 segment indicator as I²C brick - Structure and addresses

In the Arduino IDE, open the sketch: Beispiel_6.3.2.ino. You need to include the following header file: Wire.h. In case you haven't installed the necessary libraries yet, see chapter 5.2.1 on page 16 and follow the instructions.

We will realise a simple circuit with the 7 segment indicator. You can adjust the brick's I²C address on the back via two little switches (possible addresses hexadecimal: 40(16), 44(16), 48(16), 4C(16))

Hence, you can use a maximum of four of these bricks on a I²C bus.

Caution: Usually, the address 40(16) (see Fig. 32) is preset but you possibly need to control and adjust the settings (see sketch Beispiel_6.3.2.ino). The switches are on the back of the board and are accessible via a hole at the bottom. If you are skilled, you can adjust the little slide switches with a toothpick, for example.

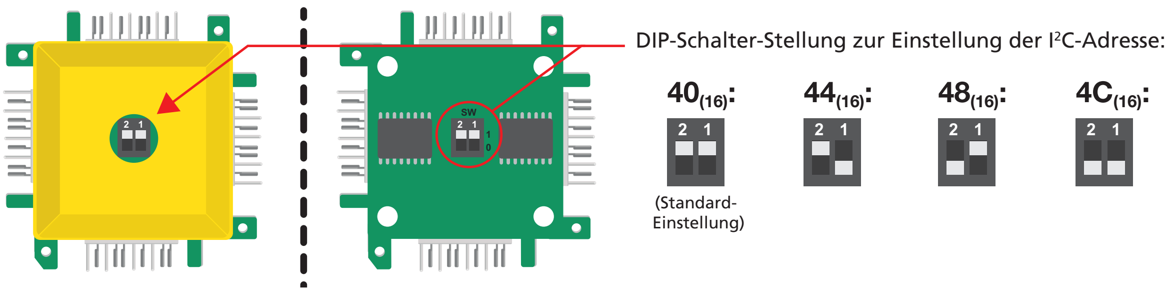


Fig. 32: Setting of addresses I²C 7 segment brick

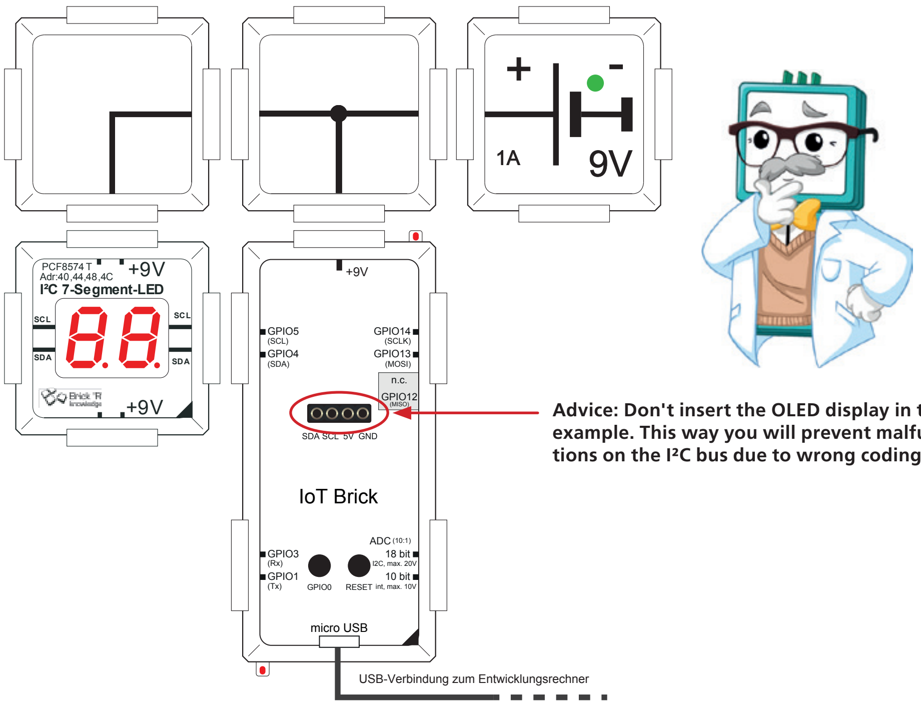
Caution: All bus subscribers at the I²C bus have to use a different address. Otherwise malfunctions can be caused!

We prepared a little library for the Arduino IDE in which all segments are coded. This is realized via a so called character table. With one byte, the combination of segments in a table is assigned to the numbers and even letters from A to Z. To make it easier, we already prepared several subprograms for it.

The function display_seg1x() displays one individual segment. For this, you transfer the I²C address of the respective driver component. The function get_7seg () converts the ASCII code into the index of the table including the segment assignments. You can directly control the segments with display_seg1xbin (). There are two of these driver components for both numbers that are addressed with a rising address and with a gap of two numbers. The sketch normally uses the I²C address for the low digit (called "LSB signs" in the sketch):

40(16), and for the high-order digit 42(16) (called "MSB signs" in the sketch).

Please take a closer look at and experiment with our coding example.



Advice: Don't insert the OLED display in this example. This way you will prevent malfunctions on the I²C bus due to wrong coding.

Fig. 33: Example: shows I²C address of the 7 segment brick

Program detail

```
void loop() {
// 7 segment display with driver part 8574T
// Export all potential addresses for identification
// This way you can see which address is set
   display_seg1x(i2cseg7x2amsb1,'4'); // own address
   display_seg1x(i2cseg7x2alsb1,'0'); // export
   display_seg1x(i2cseg7x2bmsb1,'4'); // are always pairs
   display_seg1x(i2cseg7x2blsb1,'4'); // two commands for a brick
   display_seg1x(i2cseg7x2cmsb1,'4');
   display_seg1x(i2cseg7x2clsb1,'8'); // from 0x40 to 0x4C
   display_seg1x(i2cseg7x2dmsb1,'4');
   display_seg1x(i2cseg7x2dlsb1,'C');
}
```

What's happening?

The sample program (see sketch Beispiel_6.3.2.ino) shows the I²C address on the display by sending the address as data bytes to the corresponding address. This is repeated with all useful addresses (40(16), 44(16), 48(16), 4C(16)). As soon as the correct address is sent, the brick feels addressed and displays the respective address. This way, you can easily find and note the value.

6.3.3 7 segment display as counter

Open the sketch Beispiel_6.3.3.ino in the Arduino IDE. Include the following header file: Wire.h. In case you haven't installed the respective libraries yet, see chapter 5.2.1 on page 16 and install it.

We want to realize a simple counter in the following example. You can use the same circuit as in Fig. 33. The count is saved in the variable counter. The program increases the value every 500ms by one. But on a two-digit 7 segment display a maximum of 99 can be displayed. In an if request the value of counter is requested to greater than 99. If this is the case the display will be set back to 00. The variable counter counts from 0 to 99 and then restarts from 0.

Program excerpt

```
void loop() { // In the loop

  char buffer[10];            // Select a character buffer of a specific value

  static int counter = 0;     // Set counter variable to 0

  sprintf(buffer,"%02d",counter++); // Convert Integer to sign

  if (counter >99) counter = 0;  // Counter should count between 0 and 9

     // Export count with two numbers, that´s why buffer is 0 and 1
     display _ seg1x(i2cseg7x2alsb1,buffer[1]); // LSB signs to address 0x40
     display _ seg1x(i2cseg7x2amsb1,buffer[0]); // MSB signs to address 0x42

     delay(500); // count up circa every 500ms.

} // End of loop
```

Normally, for the low digit (called LSB signs in the sketch) it uses the I²C address: 40(16) and for the high-order digit 42(16) (called MSB sign in the sketch).

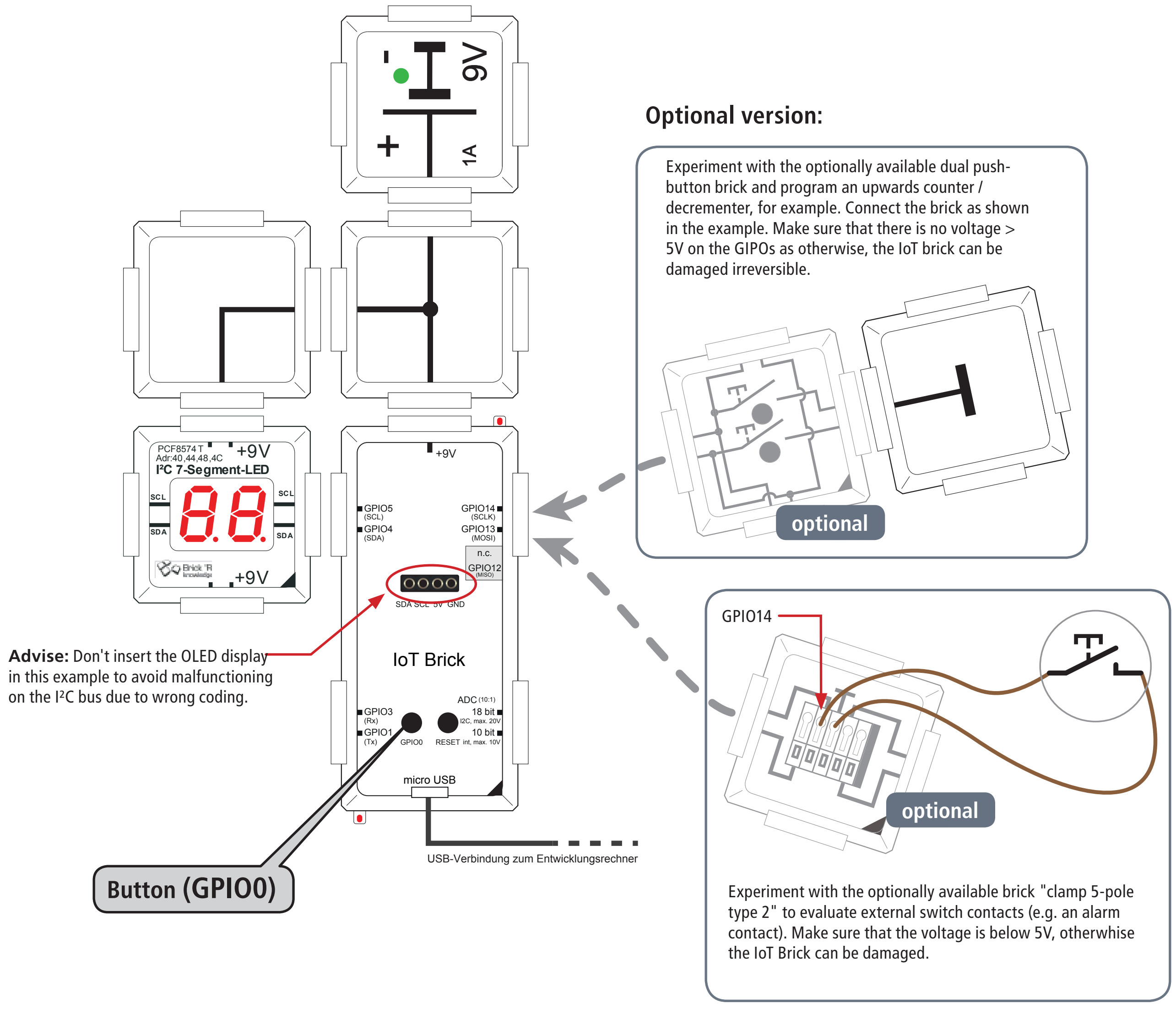
Please note that the loop will take more time than 500 ms. Striktly speaking, the time that is needed to carry out the other commands has to be added to the actual command delay(500). If you would like to work more precise, you have to use and request a timer.

6.3.4 7 segment display with debounced button

Open the sketch Beispiel_6.3.4.ino in the Arduino IDE. You have to include the following header file: Wire.h. In case you haven't yet installed the necessary libraries, see chapter 5.2.1 on page 16.

Buttons and switches have the disadvantage that the mechanical contact (frequently a spring) when being used causes a repeated closing and opening. This disruptive effect is called "bouncing" in digital technology. This problem can be solved, however, with a simple RS flip-flop (learn more about this topic with the Brick'R'knowledge Logic Set). In this task, alternatively, you get to know button debouncing via software. The main idea is to include a waiting period in the software that lasts at least as long as a bouncing cycle.

We use our counter again that you already know from example 6.3.3.



Advise: Don't insert the OLED display in this example to avoid malfunctioning on the I²C bus due to wrong coding.

Experiment with the optionally available dual push-button brick and program an upwards counter / decrementer, for example. Connect the brick as shown in the example. Make sure that there is no voltage > 5V on the GIPOs as otherwise, the IoT brick can be damaged irreversible.

Experiment with the optionally available brick "clamp 5-pole type 2" to evaluate external switch contacts (e.g. an alarm contact). Make sure that the voltage is below 5V, otherwhise the IoT Brick can be damaged.

Fig. 34: with debounced switch

Careful: Don't use external switches that are directly connected with 9V as the GPIOs are designed for 5V levels only. We advise to use the integrated switch GPIO0. **At a voltage level > 5V on the GIPOs as otherwise, the IoT brick can be damaged irreversible!**

Test different values in the delay command to determine a short but still reliably debouncing waiting time. Experiment with optionally available bricks as in Fig. 34 ("Dual push-button brick" and "Clamp 5-pole type 2"). Make sure that you connect the GPIOs (that are configured as input) only to ground. As long as the button or switch is open, there is high level on the pin as pull-up resistors are equipped internally (see chapter 5.1.3 on page 14).

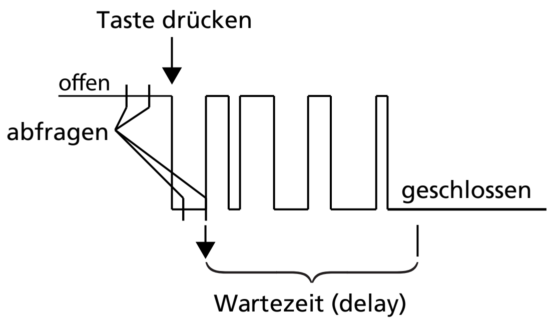


Fig. 35: Button debouncing via software

Algorithm to debounce

The button is low active, that means when pushing it there is a low level at the input (example GPIO0). The signal is requested during the first transition from high to low (if query). Then there is a pause until it gets to high level (supposedly, the button is not pushed anymore) - the while-loop is left. The delay time starts from here (40ms) to finally issue the new valid count. The next keystroke is expected, namely, high-to-low transition. Be aware that the delay time depends on the type of key that is being used. The optimal value needs to be determined experimentally to avoid evaluating pulse bounces.

Program excerpt

```
void loop() { // Loop

  char buffer[10];                    // Use character buffer of a defined value

  static int counter = 0;            // Set counter variable to 0

  sprintf(buffer,"%02d",counter);    // Converter Integer to sign

  // Interogate button, can bounce
  if (digitalRead(0)==LOW) { // Ceck High->low
    // Wait for the low-high transition
    // You can also do that after the delay, is not critical.

    while (digitalRead(0)==LOW) {
     // Wait until the button is released
    }
    counter++; // Count up
    delay(40); // delay, so that the queries are not done too quickly
  } // end of the if query High->Low

  if (counter > 99) counter = 0; // Counter should count between 0 and 99

  // Export count as two digits, that´s why the buffer is 0 and 1
     display _ seg1x(i2cseg7x2alsb1,buffer[1]); // LSB number to address 0x40
     display _ seg1x(i2cseg7x2amsb1,buffer[0]); // MSB number to address 0x42

} // End of loop
```

What's happening?

With every keystroke the display should count up exactly "1", that means, 00, 01, 02...99. Afterwards, the display returns to 00.

Don't get confused due to the coding LED lightening up in the low left as long as you push the GPIO0 button. The LED and the button are connected and don't have anything to do with your code.

6.4 OLED-Display – Basics

7 segment displays are usually used for numbers only and very rarely for letters. With 14 and 16 segment displays letters can be displayed adequately. After that, there were the first Grid Guide Displays including a matrix of 5x7 points that displayed texts in a much better quality and even first graphic characters could be displayed. The displays were based on LEDs, then LCDs (Liquid Crystal Displays) and recently OLEDs (Organic Light Emitting Diodes) were introduced to the market. The latters are similar to LEDs as they are able to lighten up. Our set contains a monochromatic OLED display with 128x64 pixels with which you cannot only display individual numbers but also multiline text and simple graphics. To display characters in this way you need a character generator or a table similar to the one of the 7 segment display. The numerical code is translated into the state of on /off for the segments. The character generator or the character chart needs comparably more memory. The value depends on the number of pixels per character. For the smallest with about 5x7 pixels, about 5 bytes per character are needed. If you want to display the whole ASCII set (128 characters incl. spaces) you need 128x5 bytes = 640 bytes.

With the 4-pole male connector, you put the included OLED display on the IoT brick as shown in the Fig..

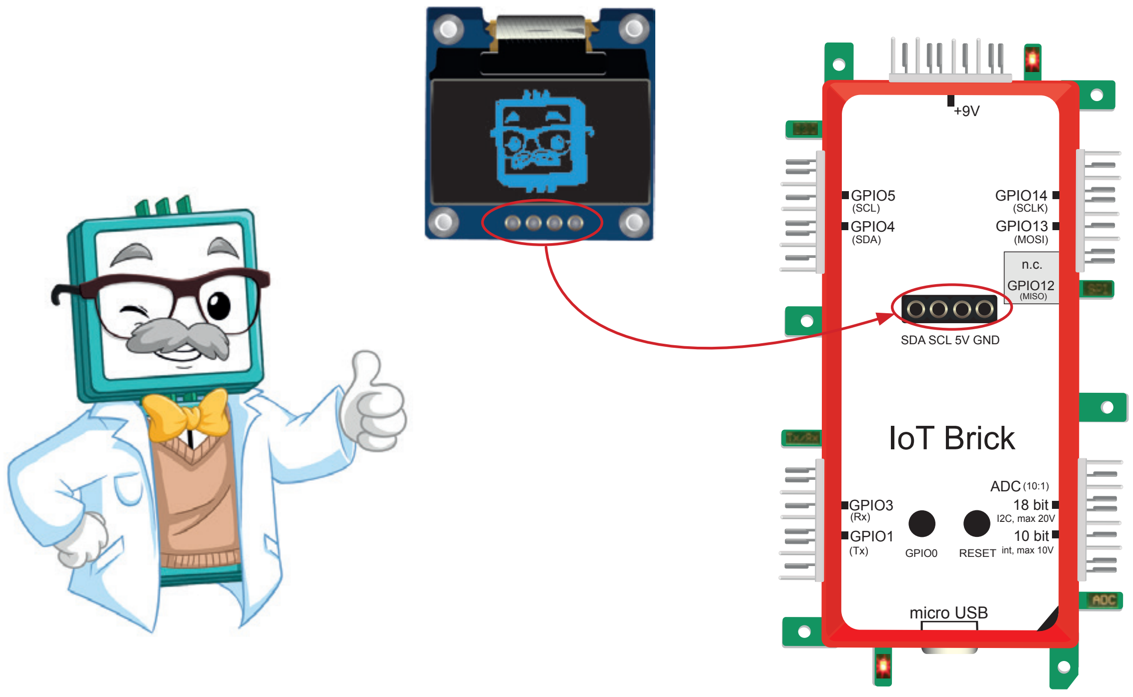


Fig. 36: OLED display on IoT brick

Open the following sketch in the Arduino IDE: Beispiel_6.4.ino. Include the following header file: #include <SSD1306Wire.h> and #include "Fig.s.h". The file Fig.s.h needs to be included in the project list of this sketch. In case you haven't installed the library SSD1306Wire.h yet, see chapter 5.2.1 on page 16. To simplify controlling the OLED displays we use a completed library. In case you haven't installed it yet, see chapter 5 on p.13.

The library for controlling our OLED display is included with #include "SSD1306Wire.h" and with initialized with SSD1306Wire display(0x3c, 4, 5);. This means, to get the correct address for the function display(), you need to move the hardware-I²C address of our OLED displays (standard: 0x78) for one bit to the right. The programmer writes: (0x78>>1) - the result is "0x3c". The parameters including the values 4 and 5 indicate the GPIOs for the I²C bus.

Upload this sketch on your IoT brick to get a first impression. It shows you everything that is possible, from different font sizes, via graphics, to uploading progress bars.

- Take your time to have a look on this and the other examples. Don't get confused, the examples involve code that is partly irrelevant for us. In the following OLED example you will see that it is basically easy to show a text on the display.
- Find more examples on the OLED display in the Arduino IDE:
- "Datei – Beispiele – ESP8266 Oled Driver for SSD1306 display – …"
- Control the following lines of code to make them function correctly with the IoT brick:
- Allocating the I²C pins for the OLED needs to include: SSD1306Wire display(0x3c, 4, 5);
- or: SSD1306 display(0x3c, 4, 5); depending on #include command at the beginning of the sketch.
- Commenting out the line including the command display.fl ipScreenVertically(); there need to be two forward slashes // prefixed to avoid displaying it the wrong way.

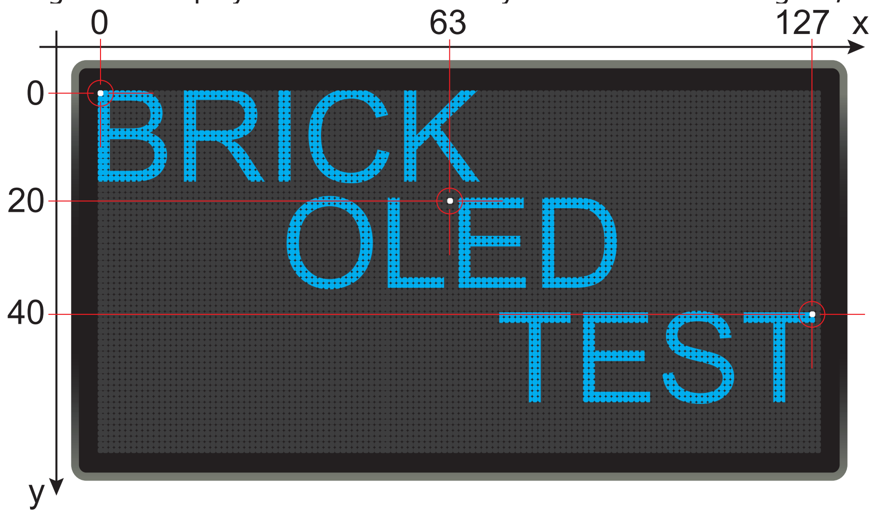
6.4.1 OLED-Display – display text

Open the following sketch in the Arduino IDE: Beispiel_6.4.1.ino. Include the following header file: In case you haven't yet installed the respective library, see chapter 5.2.1 on page 16.

In this simple example we want to issue a text on the OLED display by using the OLED library SSD1306Wire.h.

Determine the following parameters:

- **Coordinates**
  Originating from the coordinat (x, y) that was defined here you can determine the point of reference of the text and its content with the command…
- display.drawString(x, y, "Beispieltext");
- **Focus**
  Originating from the defined point of reference, with the command display.setTextAlignment(TEXT_ALIGN_x); you can determine whether the text is aligned to the left (TEXT_ALIGN_LEFT), centered (TEXT_ALIGN_CENTER) or to the right (TEXT_ALIGN_ RIGHT).
- **Size of text**
  This library natively supports the three standard font sizes 10, 16 und 24 pixels. Define the size with the command display.setFont(ArialMT_Plain_X);.
- Imagine the display to be a coordinate system that has the origin 0,0 on the upper left.



-

Fig. 37: The coordinate system of the OLED matrix

Program excerpt

```
void loop() {

  display.clear();                                        // Clear display

  display.setTextAlignment(TEXT _ ALIGN _ LEFT);          // Align text on the left
  display.setFont(ArialMT _ Plain _ 16);                  // Font size 16pixels
  display.drawString(0, 0, "BRICK");                      // Position 0,0  Text: "BRICK"

  display.setTextAlignment(TEXT _ ALIGN _ CENTER);        // Align text centred
  display.setFont(ArialMT _ Plain _ 16);                  // Font size 16 pxls
  display.drawString(63, 20, "OLED");                     // Position 63,20  Text: "OLED"

  display.setTextAlignment(TEXT _ ALIGN _ RIGHT);         // Align text on the right
  display.setFont(ArialMT _ Plain _ 16);                  // Font size 16 pxls
  display.drawString(127, 40, "TEST");                    // Position 127,40  Text: "TEST"

  display.display();                                      // Issued on display

  delay(1000);
}
```

If you want to try a different font size or even type, the OLED library's developer offers an online font editor: http://oleddisplay.squix.ch/#/home

After having defined the text and its position, focus, size and content, you can display it on the OLED with the command display.display();.

And now feel free to write different texts in different sizes to different positions.

6.5 Analog inputs

6.5.1 A/D-converter – basics

A/D conversion stands for analog-to-digital conversion. The aim is to measure analog values such as an unknown voltage and convert them into a digital value. The computer can then process this value further. With analog values, a normal computer can not do anything. Two important steps are carried out here, a quantization of the amplitude, for example, the voltage and a quantization of the time in the case of a chronologically variable course of the analog value.

What does this mean?

The quantization in amplitude is easy to understand. Assuming an analog voltage can take any value between 0 and 5V. So 2,3 V or 2,31 V or 2,315 V ... etc. So the question arises, how exactly I want to measure and how fine and how many steps I want my signal to dissolve? Finally, the numerical values have to be prepared for processing in a digital calculation system.

Example:

We want to digitize a voltage range from 0 to 5V in 6 steps. What digital value is then given for 2.1V? The diagram below shows the assignment at the red points. The value 2.1 is closer to 2 than to 3, so the digital value 2 will be selected. At a value of 2.5, you can assign 3 as a digital value if you round the figure 2.5 commercially.

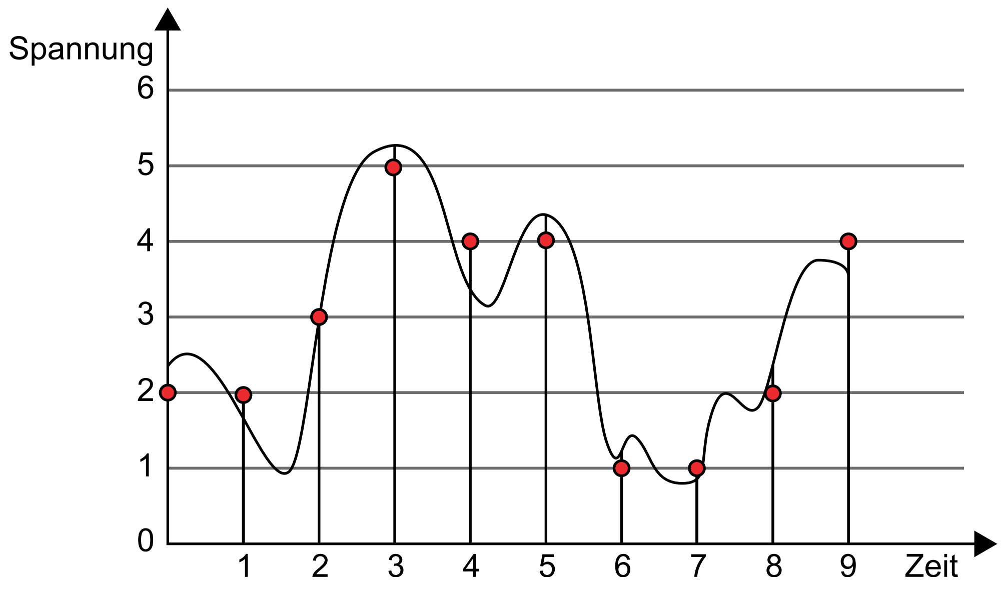


Fig. 38: Scan analog signal

In the above figure, we recorded the waveform over time of a voltage signal (in the x-axis we see the time in seconds and the y-axis the voltage in volts). When the signal is converted into a digital number sequence, a measured value (vertical lines) is sampled once per second, and the rounding is then performed on a single point. This results in the following measurement series:

Time	0 s	1 s	2 s	3 s	4 s	5 s	6 s	7 s	8 s	9 s
Voltage	2 V	2 V	3 V	5 V	4 V	4 V	1 V	1 V	2 V	4 V

There are two interesting effects:

1. We lose information in amplitude because the smallest detectable voltage unit in our example is assumed to be 1V. This effect is also called quantization error, which in our case can be up to 0.5 V deviation from the true value in the positive or negative direction. With a higher resolution of the conversion, we would also have more information on the original signal.
2. We also lose time-relevant information. Thus, in the range between 1 and 2 seconds, we see a voltage drop of up to 1V, which is not visible in the measurement series. Those skilled in the art will now note that the so-called Nyquist-Shannon sampling theorem has been violated. This means that the sampling rate for a periodic signal must be at least twice as high as its maximum frequency component (also called oversampling). This criterion is violated in the case of the shorter over- and under-oscillators.

If you now connect the red dots, you get the "visible" signal for the computer, which was scanned in a time raster of 1 second. The original curve is no longer exactly reconstructible. However, the more the number of sampling points is increased, the better the signal can be reconstructed (see sampling theorem).

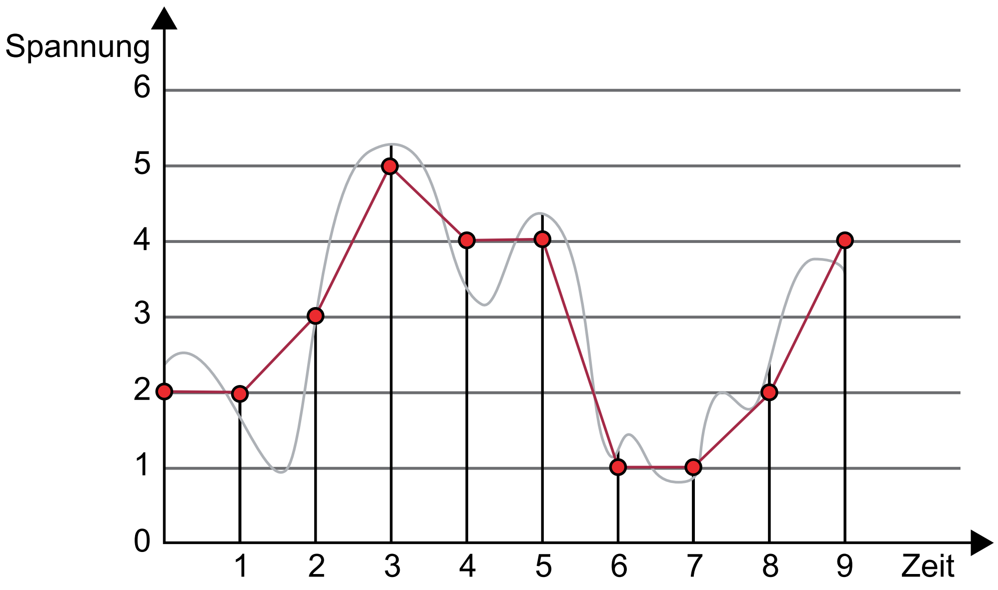


Fig. 39: Reconstruction of the sampled signal

The resolution of the amplitude is determined by the number of discrete steps assigned to a numerical value. In our example, we distinguish 6 voltage levels. For comparison, a 10-bit A / D converter already provides 1024 stages, which are generally binary-coded (see also section 6.5.2.5 on page 38). In the IoT brick we have the choice between a 10 bit and an 18 bit A / D converter.

10 bit and 18 bit A / D conversion in comparison

	10 bit A/D converter	18 bit A/D-converter
Number of stages (resolution)	2^{10} = 1024	2^{18} = 262144
Measuring range and corresponding digital value (decimal)	0…9,99 V 0…$1023_{(10)}$	0… 19,999924 V 0…$262143_{(10)}$
Minimum voltage level (so-called "Least Significant Bit" (LSB))	10 V / 1024 = 0,0098 V (= 9,8 mV)	20 V / 262144 = 0,000076 V (= 76 µV)
Quantization	±4,9 mV	±38 µV
Sampling rate max.	200 Samples/second (= 200 S/s)	3,75 Samples/second (= 3,75 S/s)
Sampling rate max.	SAR-converter	Delta-Sigma-converter (ΔΣ-converter)

To convert the analog signals into digital, binary coded values, there are different conversion methods that would go beyond the scope of this manual. As a keyword for our own research we would like to mention only the most important procedures:

- **Integrating converters (counting methods)**
  Slower than weighing, interference-free, low hardware, realization: single-, dual- and quad-slope converter, application example: Multimeter
- **Feedback transducers (weighing)**
  Good compromise between speed and hardware expenditure, realization:
  - delta-Sigma-converter (ΔΣ-converter), application exemple: 1-bit A/D-converter in audio technology - SAR (Successive Approximation Register) converter, application example: measuring technology with very high resolution.
- **Parallel-converter (Flash- end Pipeline-converter)**
  Very fast, very expensive, realisation: Flash and pipeline converter, Example: radar engineering.

6.5.2 The A/D converters on the IoT Brick

6.5.2.1 The 10 bit A/D converter

The ESP8266 offers an integrated analog-to-digital (A/D) converter with a resolution of 10 bits, which corresponds to a resolution of the voltage value of 210 = 1024 steps. The input voltage range of the converter itself is from 0V to 1V. For a better realisation, a 10-to-1 (10: 1) voltage divider was connected at the brick input "ADC 10 bit", so that a voltage of 0V to 10V can be measered directly. This allows you to build a simple voltmeter that can measure DC voltages from 0V to 10V.

6.5.2.2 The 18 bit A/D-converter

Age range from 0V to 2V with a 10-to-1 voltage converter connected upstream. This results in a practical input voltage range of 0V to 20V, so the input voltage at the brick input "ADC 18 bit" has an input voltage of max. 20V.

6.5.2.3 The voltage divider

A voltage divider with a 10-to-1 divider ratio is installed in the IoT brick at both voltage converter inputs "ADC 10 bit" and "ADC 18 bit":

Make sure that no more than 10V is present at the "ADC 10 bit" input and never more than 20V at the "ADC 18 bit" input. Otherwise, the A/D converter can be damaged!

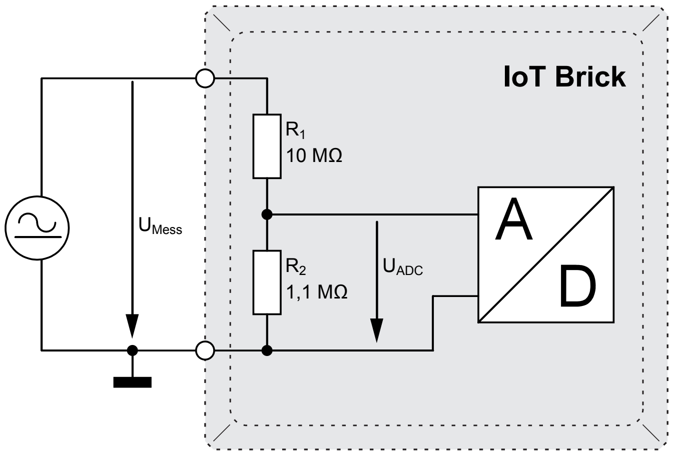


Fig. 40: Voltage splitter (10.1)

The division ratio is calculated as follows: $$\frac{U_{Mess}}{U_{ADC}} = \frac{R_1 + R_2}{R_1} = \frac{11{,}1\ \text{M}\Omega}{1{,}1\ \text{M}\Omega} = 10{,}09\ \approx 10$$

6.5.2.4 Practice Tip: correction factor

Note that inaccuracies in the measurement result can occur due to tolerances of the internal resistor advantages at the inputs "ADC 10 bit" and "ADC 18 bit". This deviation can be compensated by a correction factor. To determine the correction factor, for example, use a precise multimeter and check the voltage U_{Mess}. Then you calculate the correction factor from the ratio of the voltage measured with the multimeter and the voltage displayed on the IoT Brick without correction! In the sample program, we have prepared the appropriate code for this (Remove the two slashes (comment) only after you have made the measurement without correction). The correction factor is defined as a constant for the sake of clarity at the beginning of the program (see the following program section for the 10-bit A / D converter).

$$\textit{correction factor} = U_{Mess,\ Multimeter} / U_{Display\ Brick}$$

Program section for correction factor `(see example_6.5.3.ino)`

```
//The procedure is the same for 10bit and 18bit ADC
...
//define the corpuscleactor for 10bit ADC
const float Correction _ 10bit = 1.046;
...
//Correction factor (determined with multimeter measurement)
UMess _ 10bit = UMess _ 10bit * Correction _ 10bit;
...
```

Since each resistor has different manufacturing tolerances, the correction factor for the two A / D converters should also be determined separately.

6.5.2.5 Binary encoding

An A/D converter converts an analog measured value at the input into a binary coded value at the output. One also speaks of binary word whose width, ie the number of individual binary characters is given in bits (as a unit, incidentally, small: 1 bit). In our case, we have a 10 bit or 18 bit binary word depending on the A/D converter resolution. The least significant bit is often referred to as a "Least Significant Bit" (LSB) and the most significant bit is a "Most Significant Bit" (MSB).

Highest bit
(Most Significant Bit (MSB))

Least significant bit
(Least Significant Bit (LSB))

Number of bits	Number of encodable states	binary number (Number of codable states - 1)	Decimal number	Hexadecimal
		2^{17} … Valence … 2^0		
(0)	$1 = 2^0$	00 0000 0000 0000 0000	0	0
1 bit	$2 = 2^1$	00 0000 0000 0000 0001	1	1
2 bit	$4 = 2^2$	00 0000 0000 0000 0011	3	3
3 bit	$8 = 2^3$	00 0000 0000 0000 0111	7	7
4 bit	$16 = 2^4$	00 0000 0000 0000 1111	15	F
…	…	…		
8 bit	$256 = 2^8$	00 0000 0000 1111 1111	255	FF
10 bit	$1024 = 2^{10}$	00 0000 0011 1111 1111	1023	3FF
12 bit	$4096 = 2^{12}$	00 0000 1111 1111 1111	4095	FFF
14 bit	$16.384 = 2^{14}$	00 0011 1111 1111 1111	16.383	3FFF
16 bit	$65.536 = 2^{16}$	00 1111 1111 1111 1111	65.535	FFFF
18 bit	$262.144 = 2^{18}$	11 1111 1111 1111 1111	262.143	3FFFF

6.5.3 A/D-converter 10 bit

Open the sketch in the Arduino IDE: Beispiel_6.5.3.ino. The following header files must be included: SSD1306Wire.h. If you do not have the appropriate libraries installed, go to chap. 5.2.1 on page 16 and retrieve this.

In this example, we measure a voltage across a variable resistor (potentiometer, often called only Poti).

The resistance value of the potentiometer varies between 0 Ω and 10 kΩ depending on the position.

The voltage that drops across the potentiometer is linear to the resistance value. When the potentiometer is in the clockwise direction, the resistance value is approx. 10 kΩ between ground and the sliding contact, which is connected to the analog input (ADC 10 bit) of the IoT bricks. Now the full supply voltage is present, which should be about 9V. If we now turn the potentiometer counterclockwise to the stop, then there is 0V (ground) at the input of the A / D converter. In the middle position of the potentiometer, approximately half the supply voltage will be present, H. About 4.5V.

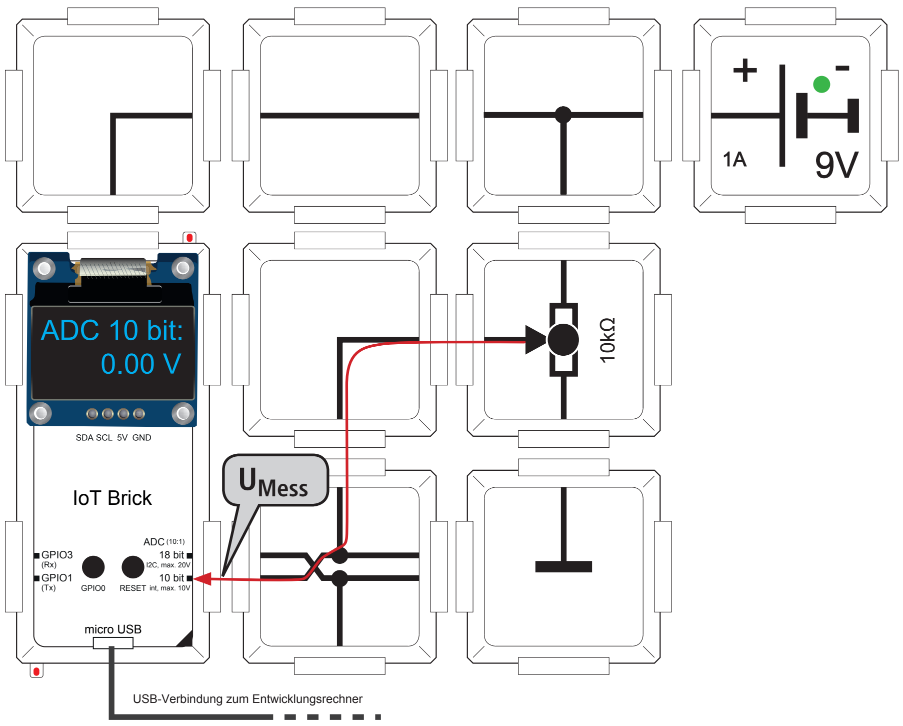


Fig. 41: Brick circuit 10 bit A / D converter

The measured voltage UMess is displayed on the OLED display and updated every second. Furthermore, the digital raw value and the calculated voltage value is output to the serial monitor of your computer. To open the serial monitor, simply click on the magnifying glass symbol in the Arduino IDE (see chapter 5.2.3 on page 20). You can use the following formula to convert the digital value from the A / D converter to the analog voltage value (see also sample program Beispiel_6.5.3.ino).

$$U_{Mess} = (analogIN / 1024) * 10\,V$$

By setting 1023 for analogIN, you get the maximum value of the input voltage range. In Our case is 9.99 V.

➔ See next page for program details.

Program excerpt

```
void loop(void)
{
// Reads voltage as raw value: 0 = 0V bis 1023 = 1V-1LSB
   analogIN = analogRead(adc _ esp8266);
// Convert to Volt with analog IN / resolution (* 1 Volt), * 10 because of 10: 1
divider at the ADC
   UMess = ((float)analogIN/1024)*10.0;
// Convert voltage value to string for OLED output
   String UMess _ str = String(UMess, 2); // 2 decimal

   display.clear();      //delete OLED

   display.setTextAlignment(TEXT _ ALIGN _ LEFT);
   display.setFont(ArialMT _ Plain _ 24);
   display.drawString(0, 0, "ADC 10 bit:");
   display.setTextAlignment(TEXT _ ALIGN _ RIGHT);
   display.setFont(ArialMT _ Plain _ 24);
   display.drawString(120, 32, UMess _ str + " V");

  display.display();    // output to OLED
  delay(1000);
```

Using the command analogIN = analogRead (adc_esp8266); reads the digital raw value from the internal A / D converter of the ESP8266. The value range of the 10-bit converter is from 0 to 1023 (decimal). In our example, the value 0 corresponds to the voltage 0V and the raw value 1023 corresponds to the maximum value of the input voltage range.

For syntactic reasons, we must first convert the analogue ININg.variable, which returns the raw value of the A / D converter, to the float variable UMESS to convert it into the correct voltage value. For the output, we need the string variable UMess_str

6.5.4 18 bit A / D converter

The brick circuit for this exercise differs only by the brick bottom in the middle as opposed to the previous exercise. Thus, the same voltage UMess is present at both analog inputs, which can be varied via the potentiometer. This allows a simple comparison of the two A / D converters. The resolution of the 18-bit A / D converter, which is 256-fold better than the 10-bit A / D converter, is clearly visible in the measurement result.

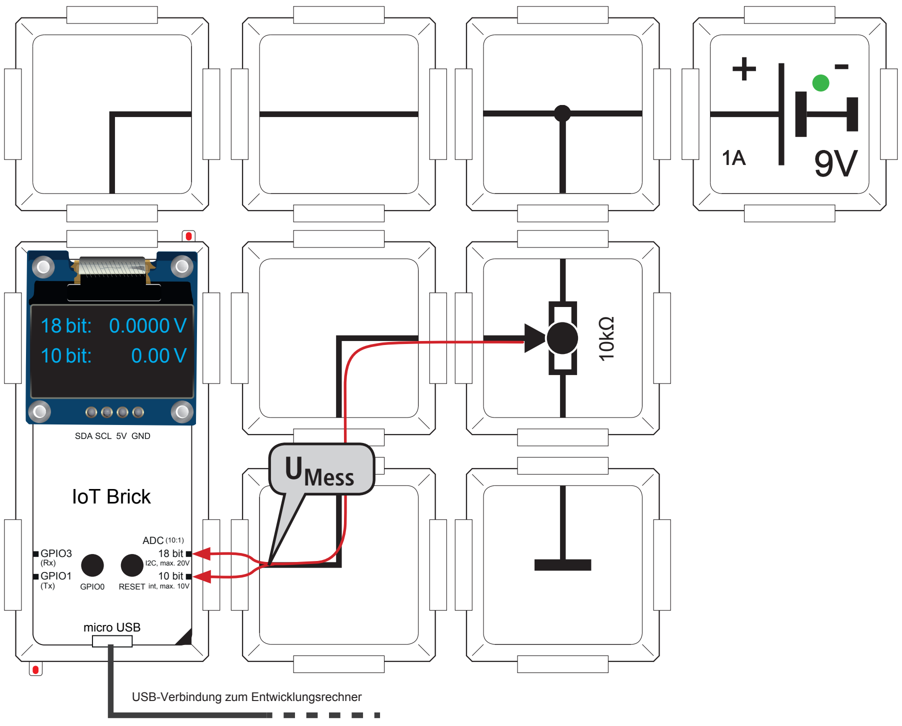


Fig. 42: Brick circuit 18 bit A / D converter

From the belly we would expect identical voltage values for this circuit. But the saying "Who measures, measures dung" also applies here. As already mentioned in the previous chapter, component tolerances play a role here. The default tolerance of 10% of the two 10-to-1 voltage dividers at the inputs of the A/D converters has a major effect (see section "6.5.2.3 the voltage divider" on page 37).

➔ For program details, see next page.

Programmausschnitt

```
void setup(void)
{
…
// Init MCP3421: I2C address, 18 bit mode, no gain
   MCP.init(0x68,3,0);
…
}
void loop(void)
{
…
// Reads voltage as double values of MCP3421, input range: 0 to 2.048V
   analogIN _ 18bit=MCP.getDouble();
   UMess _ 18bit = analogIN _ 18bit*10.0; // * 10 because of 10: 1 divider
// The next line for determining the correction factor using
// Please measure the multimeter measurement!
UMess _ 18bit = UMess _ 18bit * Correction _ 18bit; // Correction factor
// Convert voltage value to string for OLED output
   String UMess _ 18bit _ str = String(UMess _ 18bit, 4);
…
  display.display();    // Output of both voltage values to OLED
  delay(1000);
}
```

We use the library MCP3421.h to simplify the programming of the MCP3421. The A / D converter is initially set with MCP.init (0x68,3,0); initialized. With the MCP.get-Double () function, the voltage value is already returned as a double value, ie as a floating point number.

Since the MCP3421 sweeps an input voltage range from 0 to 2.048 V, the value must be multiplied by a factor of 10. Next, as described in chap. 6.5.2.4 on page 38 - a correction factor which is defined as a constant at the beginning of the sketch by the float. For output to the OLED display or serial monitor, we have the floating point numbers UMess_10bit and convert UMESS_18bit to corresponding strings. The values are updated approximately in the secondary cycle.

6.6 IoT-example

Control your IoT brick via your smartphone or any other WiFi-enabled device and discover new possibilities. Program your small website and realize an own control centre.

You will learn...

...how to synchronize the time and date with the internet (example 6.6.2)

...how to import the data of the temperature and humidity sensor (example 6.6.3)

...how to request the current Dollar exchange rate from the internet (example 6.6.4)

Later (see example 6.6.5) you will learn how to program a small website as the basis for your IP measurement central. Via the universal sensor adaptor (ALL-BRICK-0649) you can connect various sensors in an easy way. Last but not least you will learn how to control LEDs via the internet. This is a basic feature which is needed in various applications of home automation.

Because of the simple network integration of the IoT brick, there are a lot of new possibilities.

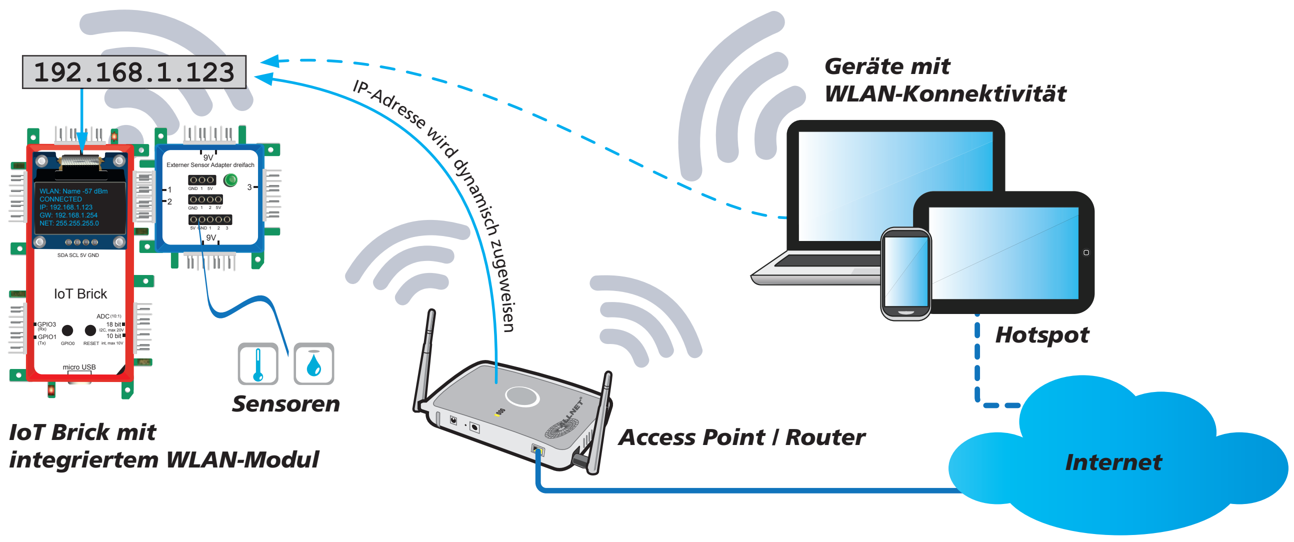


Figure 43: The IoT brick in the network

The big advantage of the ESP-12-F module - which is integrated in the IoT brick - compared to the Arduino is the networking possibility thanks to the WiFi module. The ESP 8266 micro controller can be embedded easily into an existing WiFi network. The IP address is normally allocated dynamically. The hotspot can be the router of your local networks or your smartphone.

All of the IoT examples need an internet connection, that´s why you will see the following symbol:

In the examples 6.6.5 and 6.6.6 you can access the webserver which is integrated into the IoT brick. Theoretically also via internet if you can set up your network for this case. The keywords here are port-forwarding and firewall. Standard routers normally offer the possibility to forward the specific access from the internet to an internal port. This internal port is actually the IoT brick with its local IP address. It is required that your firewall allows this kind of access from the "outside". You can find further information about the relevant configurations of your router in the documentation or on the internet.

6.6.1 Configuring the IoT brick as WiFi client

Open the sketch in the Arduino IDE: `Beispiel_6.6.1.ino`. You have to integrate the following header files: `ESP8266WiFi.h` and `SSD1306Wire.h`.In case you have not yet installed the libraries, go back to chapter 5.2.1 on page 16 and install them.

If you want to communicate with your IoT brick via WiFi, you have to integrate it into your local network fist. You will need the WiFi name (calles SSID) of your access point or router and the password. Since there is no possibility to type in the information manually, we have to consign the access data in our sketch:

Search the green lines at the beginning of the sketch file and replace mein_wlan_name by the name (SSID) of your WiFi and instead of mein_wlan_passwort enter your corresponding password (quotes must remain)

Programmausschnitt

```
//Insert the WiFi name (SSID) here:
const char* ssid = "mein _ wlan _ name";
//Insert your WiFi password here:
const char* password = "mein _ wlan _ passwort";
...
void setup() {
...
}
```

In the void loop() funktion, you can see how the following strings are assembled and line by line how they are prepared to display with the command display.drawString(x,y,"String"). The following functions are used to investigate the parameter values.

`wlan_oled` (WiFi name and strength), funktion `WiFi.SSID()` and `WiFi.RSSI()`.

- `state` (link state), Funktion `WiFi.state`
- `ip_oled` (IP adress of the IoT brick), Funktion `WiFi.localIP()[x]`
- `gw_oled` (gateway-IP), funktion `WiFi.gatewayIP()[x]`
- `mask_oled` (sub mask), funktion `WiFi.subnetMask()[x]`

The output is finally done with the command `display.display();`

Parallel to this, the network connection information is also displayed on the serial monitor of your computer. To open the serial monitor, just click on the magnifying glass symbol (refer to chap. 5.2.3 on page 20).

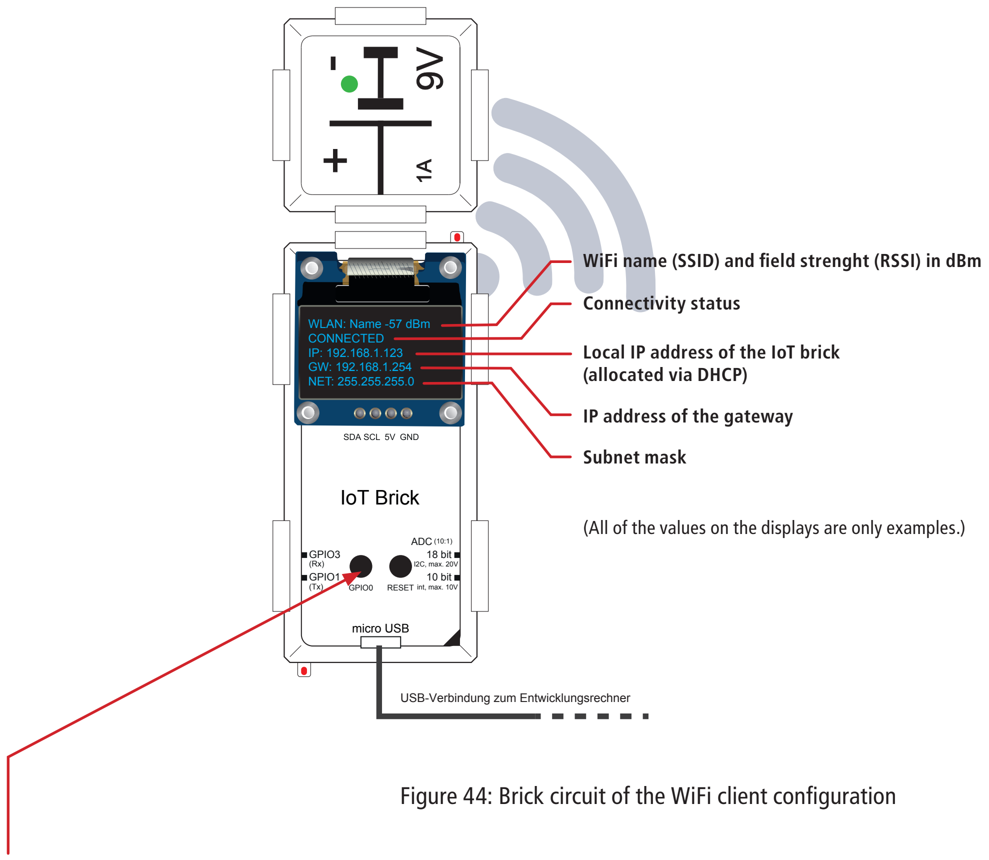
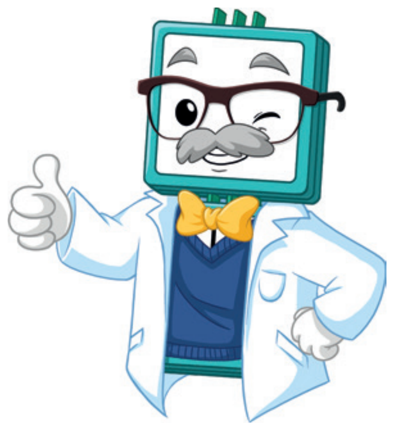


Figure 44: Brick circuit of the WiFi client configuration

The WiFi configuration can also be displayed by pushing the button GPIO0 (figure 44)!



6.6.2 Time from the internet

Open the following sketch in the Arduino IDE: `Beispiel_6.6.2.ino`. You have to integrate the following header files: `ESP8266WiFi.h`, `SSD1306Wire.h`, `TimeLib.h` and `NtpClientLib.h`. In case you have not yet installed the libraries, go back to chapter 5.2.1 on page 16 and install them.

For this example, you have to enter your WiFi name (SSID) and the corresponding password in the example program first. To do so follow the instructions as in chap. 6.6.1 . On the internet, there are so-called "time servers", which can be accessed via NTP (Network-Time-Protocol) to determine the current time and date. Via the domain `pool.ntp.org` everyone can access a free pool of time servers. To process the information transmitted by the time server, we use predefined functions `NtpClientLib.h`. in the libraries of this exercise. We use an OLED-display to visualize the data.

program sample

```
...
void setup() {
...
//if the WiFi is connected, connect to the NTP time server
{
    NTP.begin("pool.ntp.org", 1, true);  ////connect to the NTP time server
    NTP.setInterval(63);          //syncronize every 63 seconds
    }

NTP.onNTPSyncEvent([](NTPSyncEvent_t event) {
    ntpEvent = event;
    syncEventTriggered = true;     //connected to the time server
   });
...
}

void loop() {
...
    String time = NTP.getTimeStr();  //save the time of day in the variable time
    String date = NTP.getDateStr();  //save the date in the variable date
    display.clear();
    display.setTextAlignment(TEXT_ALIGN_LEFT);
    display.setFont(ArialMT_Plain_24);
    display.drawString(15, 5, time);   //prepare the output of the time of day
    display.drawString(5, 35, date);   //prepare the output of the date
    display.display();                 //update the OLED display
...
}
```

As soon as the WiFi connection is established, the function `NTP.begin("pool.ntp.org",1, true);` makes the connection to the NTP server. This can take some seconds. The first parameter assigns the URL to the time server pool, the second one is the deviation of our time zone to the "Universal Time Coordinated" (UTC) of +1 hour and the `true` " in the third parameter indicates that in our time zone we switch between summer and winter time.. defines the interval in seconds in which the synchronization with the NTP server should be updated and `syncEventTriggered=true` reports a successful synchronization with the NTP server.

In `void loop()` the `NTP.getTimeStr()` and the `NTP.getDateStr()` functions are used to read the time/date as a string and prepares it for output. The actual output on the OLED display is done with- `display.display();`. `display.clear();` is also important in this example, in order to clear the display and not to just override it in every loop pass.

Parallel to the display on the OLED display the output is also send to the serial monitor of your computer. To open the serial monitor, simply click the magnifying glass symbol (see chap. 5.2.3 on page 20). Beside the time and date other information like summer or winter time, the time elapsed since the initial synchronization (uptime) and the time of the initial synchronization is displayed on the serial monitor.

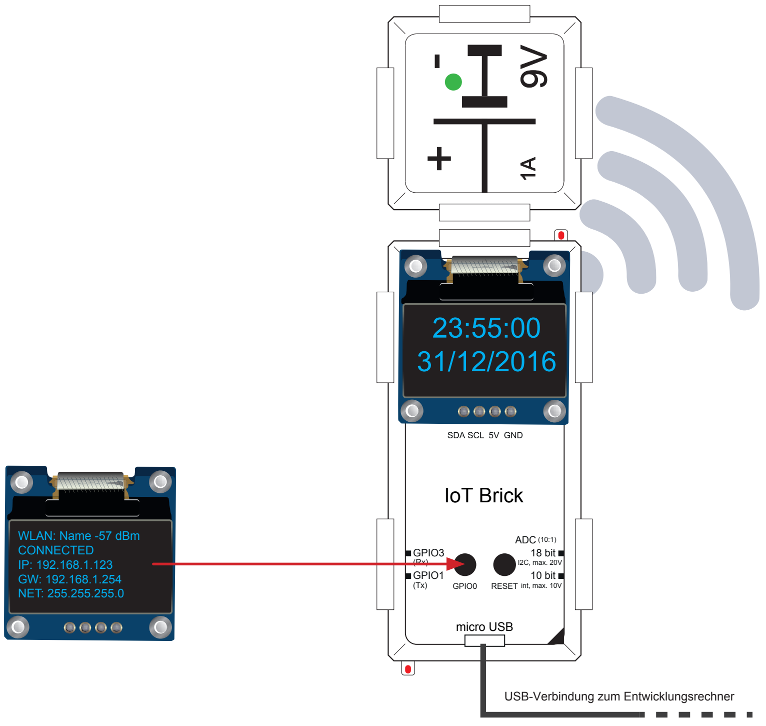
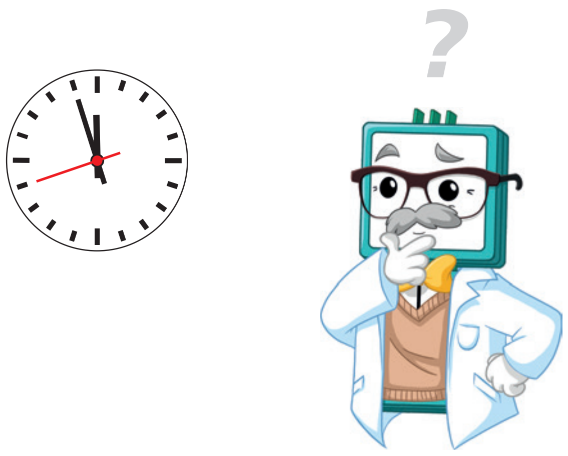


Figure 45: Brick circuit of the time from the internet configuration

TIPP: The connection status of the IoT brick can always be displayed by pushing the button GPIO0 (see figure 44)!



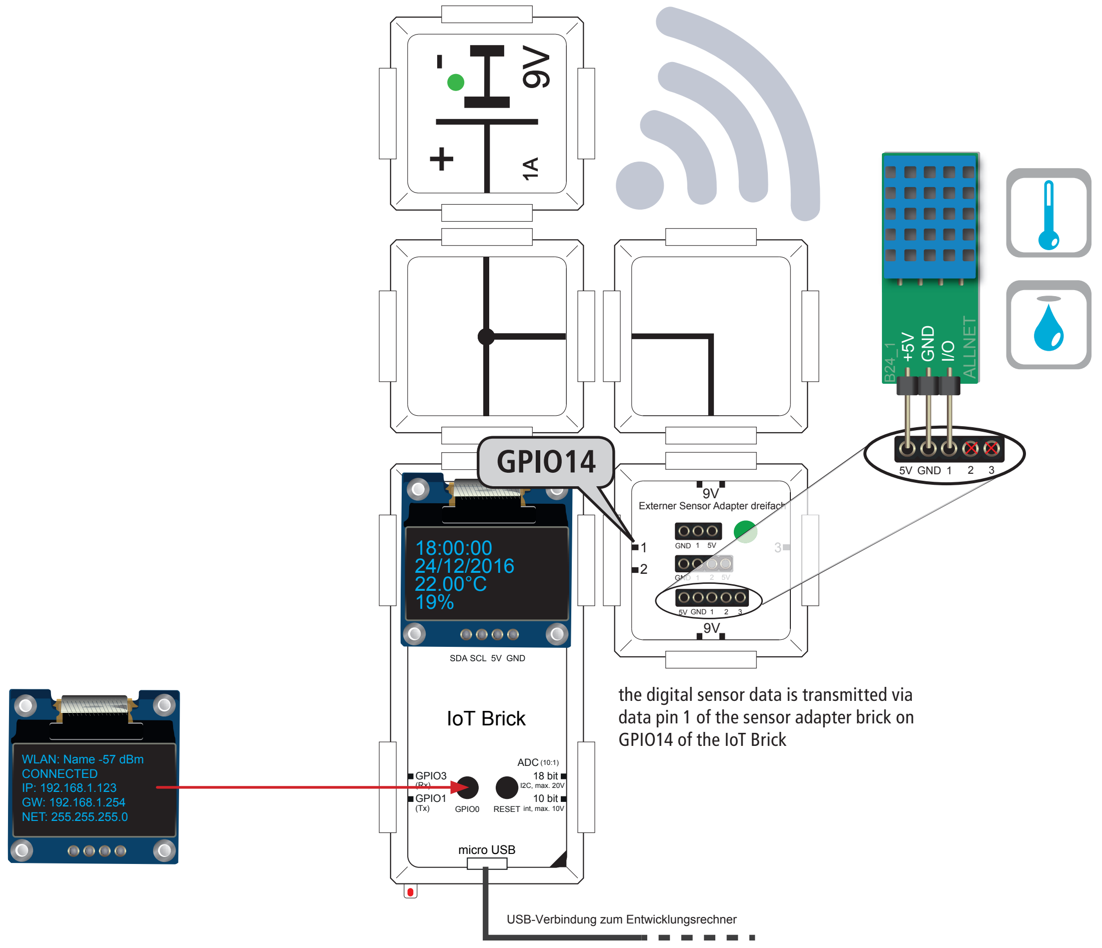
6.6.3 Measuring temperature and humidity

open the sketch following in the Arduino IDE `Beispiel_6.6.3.ino`. You have to integrate the following header files: `ESP8266WiFi.h`, `SSD1306Wire.h`, `TimeLib.h`, `NtpClientLib.h` and `DHT.h`. If you do not already have the appropriate libraries installed, go to chap. 5.2.1 on page 16 and get this done first.Also for this example you must first enter your WiFi name (SSID) and the corresponding password in the sample program. Go to as in chap. 6.6.1 described.For temperature and humidity measurement, we use the widely used combination sensor of the type DHT11, for which there is already a library (DHT.h) and examples.In addition, there are many different sensors, for example from the Arduino-World, such as:

- temperature sensors
- infrared light barrier
- motion detector (IR sensor)
- light sensor (LDR)
- gas-sensor
- hall-sensor
- shock sensor
- touch sensitive button

The universal sensor adapter brick (ALL-BRICK-0649) allows you to simply connect numerous commercially available sensors. For many sensors there are already examples and libraries, so that these can also be easily integrated into own projects.

First put the bricks together as shown. Pay attention to the correct connection of the supplied DHT11 sensor. Insert the sensor exactly as shown in fig. 46 to the far left in the lower 5-pin connector of the sensor adapter brick. The label "5V" on the sensor must be at the far left of the socket marked "5V" otherwise the sensor and the IoT brick are irreversibly damaged.



the digital sensor data is transmitted via data pin 1 of the sensor adapter brick on GPIO14 of the IoT Brick

Fig. 46: Brick circuit meassure temperature and humidity

Tip: The connection status of the IoT Brick can be displayed at any time by pressing the GPIO0 button (see also fig. 44).

program sample

```
#define DHT_TYPE DHT11      // define sensortype: DHT11
const int DHT_PIN = 14;    //data pipe of the sensor to GPIO4 of the IoT Brick
char temp[20];             //define string variable for the temperature
char humi[20];             //define string variable for the humidity

DHT dht(DHT_PIN, DHT_TYPE);    //variable of type DHT

...
void setup() {
...
   dht.begin();                    //initialize sensor
}

void loop() {
...
 if (counter%1000==0){               //Check the sensor approximately once per second
     float t = dht.readTemperature(); //Read Temperature (Celsius)
     float h = dht.readHumidity();     //Read the humidity

     sprintDouble(temp,t,2);  //Convert temp with 2 decimal places to string
     sprintDouble(humi,h,0);  //Convert humidity without decimal to string
     strcat(temp," °C");      //add °C to string
     strcat(humi," %");       //add % character to string

   }
  display.drawString(5, 30, temp);   //Prepare temperature output
  display.drawString(5, 45, humi);   //Prepare humidity output
  display.display();                 //update OLED display
...
}
```

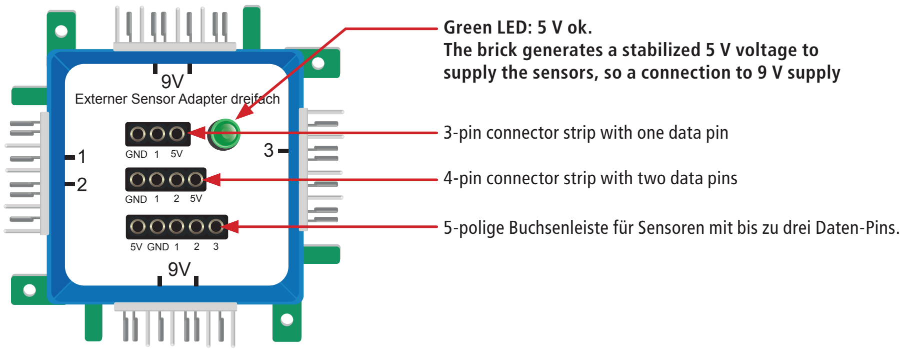
At the beginning of the sketch, the sensor type DHT11 and the data line GPIO pin (here GPIO4) are defined. A special variable is the type DHT, with the help of which the sensor is initialized in void setup()

But now for the actual measurement in the section void loop (). With the two functions calls off the sensor library dht.readTemperature() and dht.readHumidity() the temperature and the humidity are read from the sensor and stored in the two floating point variables t and h. With the help function sprint Double () the floating-point numbers are converted in strings with the desired number of decimals and the string operation strcat () adds the unit to the string.

In addition to the values for temperature and humidity, the display is supplemented with time and date– as already shown in exercise 6.6.2. The actual output of all values to the OLED display is as usual with the command `display.display();`.

Parallel to this, the output is also on the serial monitor of your computer. To open the serial monitor, just click on the magnifying glass symbol of the Arduino IDE (see chap. 5.2.3 on page 20) In this example, the output of example 6.6.2 was supplemented by temperature and humidity.

By adapting the slot and de sketch numerous sensors on the market can be connected.



9V
Externer Sensor Adapter dreifach
GND 1 5V
GND 1 2 5V
5V GND 1 2 3
9V
1
2
3
Green LED: 5 V ok.
The brick generates a stabilized 5 V voltage to supply the sensors, so a connection to 9 V supply
3-pin connector strip with one data pin
4-pin connector strip with two data pins
5-polige Buchsenleiste für Sensoren mit bis zu drei Daten-Pins.

Fig. 47 Connection options for "Sensor-Adapter-Brick dreifach"

Please take care of the right connection to the adapter brick. If you are not sure, refer to the documentation of the sensor. Otherwise there is a risk of irreversibly damaging the sensor and / or IoT brick!

Dollar exchange rate from the internet

open the sketch following in the Arduino IDE: `Beispiel_6.6.4.ino`. You have to integrate the following header files: `ESP8266WiFi.h`, `SSD1306Wire.h`, `TimeLib.h`, `NtpClientLib.h` and `CurrencylayerClient.h`. If you do not already have the appropriate libraries installed, go to chap. 5.2.1 on page 16 and get this done first.

Also for this example you must first enter your WiFi name (SSID) and the corresponding password in the sample program Go to as in chap. 6.6.1 described.

In order to retrieve the dollar exchange rate (EUR-USD) from the internet you need a special library "Brick-ESP8266 ", which you can download at http://www.brickrknowledge.de/downloads . Unless you have already installed all libraries in chap. 5.2.1, please install them first as discribed in chap. 5.2.1.2.3 on page 19.

program sample

```
#include <CurrencylayerClient.h>
...

if(counter%5000==0){      //update exchange rate approximately every 5 seconds
   currencylayer.getLastChannelItem();
   counter = 0;
   }
display.setTextAlignment(TEXT_ALIGN_LEFT);
display.setFont(ArialMT_Plain_16);
display.drawString(0, 45, currencylayer.getFieldValue(0));
display.setTextAlignment(TEXT_ALIGN_RIGHT);
display.drawString(127, 45, " EUR/USD");
display.display();
...
```

In this example, we get the current dollar rates from the internet to calculate how much euro I have to pay for a dollar (or vice versa).

$$EUR = USD * exchange\ rate$$

To get the conversion rate, we call the function currencylayer.get- LastChannelItem (). The time interval for updating can be controlled over the counter in the If statement if (counter% 5000 == 0) The display is formatted as usual – However, the call currencylayer.getFieldValue (0) is important.. Only at this position, the dollar rate rounded to 4 decimals is available as a string in the sketch.

In addition to the current conversion rate, the display is also supplemented with time and date - as already showed in exercise 6.6.2. The actual output of all values to the OLED display is done as usual with the command display.display ()..

Parallel to this, the output is also on the serial monitor of your computer. To open the serial monitor, just click the magnifying glass symbol of the Arduino IDE (see chap. 5.2.3 on page 20) The output of example 6.6.2 was supplemented by some status messages.

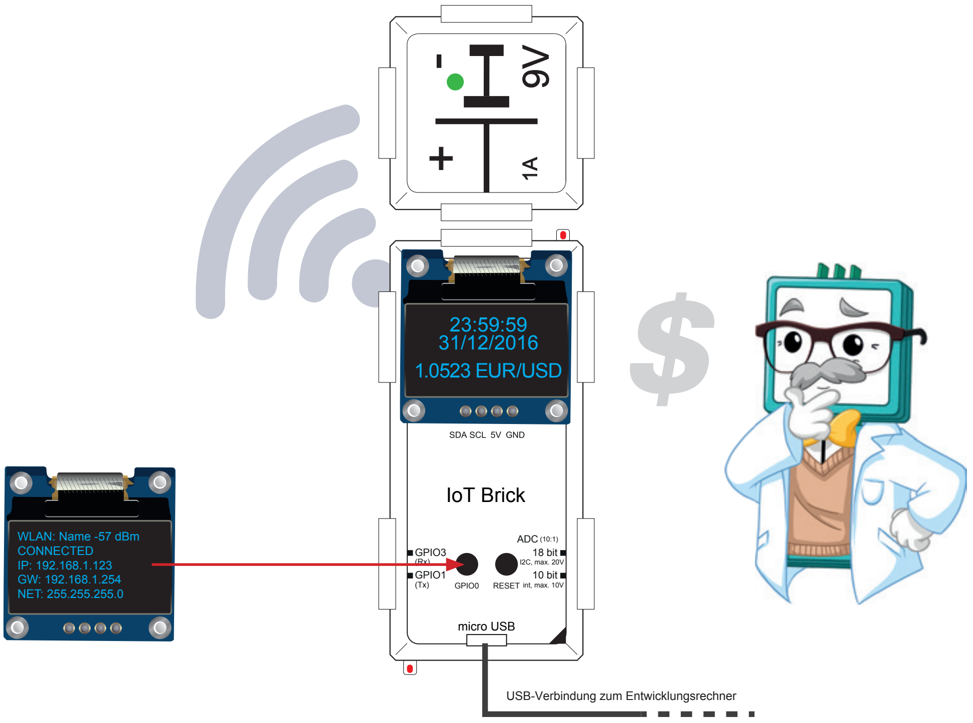


Fig. 48 Brick circuit dollar exchange rate from the internet

TIP: The connection status of the IoT brick can always be displayed by pushing the button GPIO0 (see figure 44)!

6.6.4 My first website

open the following sketch in the Arduino IDE: `Beispiel_6.6.5.ino`. You have to integrate the following header files: `ESP8266WiFi.h`, `SSD1306Wire.h`, `TimeLib.h`, `NtpClientLib.h`, `ESP8266WebServer.h` and `DHT.h`. If you do not already have the appropriate libraries installed, go to chap. 5.2.1 on page 16 and get this done first.

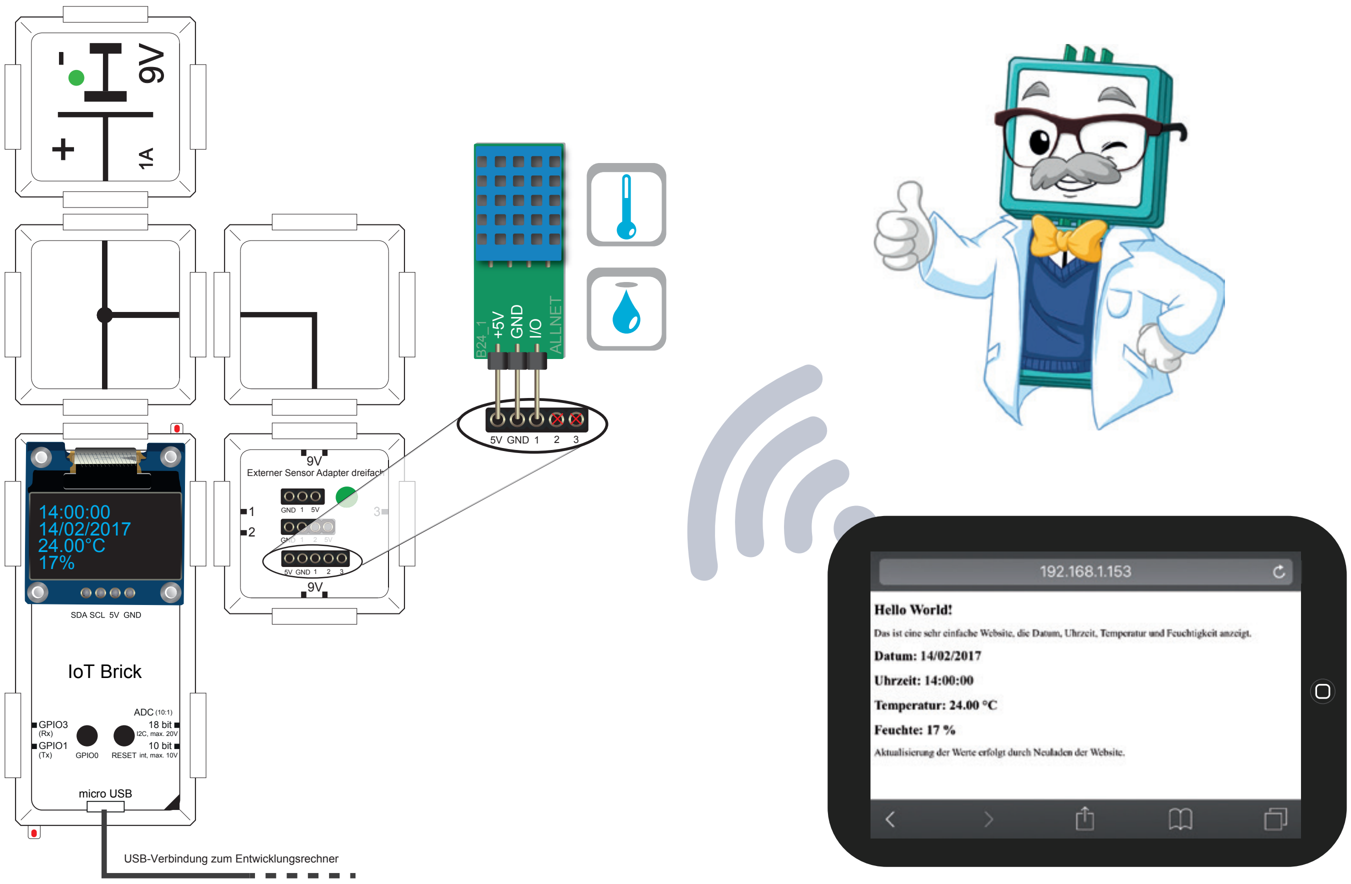
Also for this example you must first enter your WiFi name (SSID) and the corresponding password in the sample program.

In this example, we will build a simple website that will give us time, date, temperature and humidity.The basis for each website is Hypertext Markup Language, abbreviated: HTML. You can use the WYSIWYG HTML editor to code your website. You can download various free HTML editors, e.g., NVU for Windows, Linux (see: www.nvu.com) or BlueGriffon for MAC OS X (see: www.bluegriffon.org).

HTML basics are useful for this exercise. You can find comprehensive HTML basics in the frame of the web project SELFHTML at www.selfhtml.org.

The brick circuit and the sketch of this exercise are based on the example 6.6.3. The WiFi status can be displayed by pushing the button GPIO0 (figure 44)!

First put the bricks together as shown in fig. 49, otherwise the sketch can not be loaded correctly. Be sure to connect the supplied DHT11 sensor correctly. Insert the sensor exactly as shown in fig. 46 to the far left in the lower 5-pin connector of the sensor adapter brick The label "5V" on the sensor must be at the far left of the socket marked "5V" otherwise the sensor and the IoT Brick are irreversibly damaged.



Hello World!

Das ist eine sehr einfache Website, die Datum, Uhrzeit, Temperatur und Feuchtigkeit anzeigt.

Datum: 14/02/2017

Uhrzeit: 14:00:00

Temperatur: 24.00 °C

Feuchte: 17 %

Aktualisierung der Werte erfolgt durch Neuladen der Website.

Fig. 49: Brick circuit"My first website"

program sample

```
...
ESP8266WebServer server(80); //start web server on port 80
...
void setup() {
...
//when a browser directly accesses the root directory,
//execute handleRoot (see below).
   server.on("/", handleRoot);
   server.begin();            //from now on, the server listens for HTTP requests
   Serial.println("HTTP server started");
}

void loop() {

   server.handleClient(); //handle the HTTP request
...
}

//the handleRoot () function is used to deliver the site
//as soon as a request arrives from a browser
void handleRoot() {
  String content;
  String time _ web = NTP.getTimeStr();
  String date _ web = NTP.getDateStr();
  String temp _ web = temp;
  String humi _ web = humi;
  content = "<!DOCTYPE html>";
  content += "<html>";
  content += "<head>";
 //next line is important for the degree character "°" to be displayed correctly
  content += "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\">";
  content += "<title>Meine erste IoT Brick-Website</title>";
  content += "</head>";
  content += "<body>";
  content += "<h1> Hello World! </h1>";
  content += "<p>Das ist eine sehr einfache Website von deinem IoT Brick, die
             Datum, Uhrzeit, Temperatur und Feuchtigkeit anzeigt.</p>";
  content += "<h2>Datum: "+date _ web+"</h2>";
  content += "<h2>Uhrzeit: "+time _ web+"</h2>";
  content += "<h2>Temperatur: "+temp _ web+"</h2>";
  content += "<h2>Feuchte: "+humi _ web+"</h2>";
  content += "<p>Durch Neuladen der Website können die Werte jederzeit aktuali-
             siert werden.</p>";
  content += "</body>";
  content += "</html>";
  server.send(200, "text/html", content);
}
```

At the beginning of the sketches, the web server is started and standard port 80 is assigned for queries via HTTP (eg:Http://www.brickrknowledge.com). As soon as you type the local IP address of your IoT brick in your browser with prefixed http: //, (eg http://192.168.1.153), you access the root directory of your web server so that the handleRoot () function is called. As you may have already read on www.selfhtml.org, the basic structure of each website consists of so-called HTML-Tags. For each element, there is usually a start tag and an end tag (to be recognized by the preceding slash "/"), which are embedded in the characteristic pointed brackets.. For example`<p>Normaler Absatz</p>`. In the function handle root (), the entire content of our website is saved in the string variable content.In order to make the whole picture clearer, the string assignment of the HTML code is spread to several lines in the sketch. This includes all lines of the form content + = "...";

Another difficulty is that in the sketch programming start and end of a string with quotes (").Because the HTML code also contains quotation marks, within the string for the correct coding a so-called Escape Sequence \ " is used for the quotation mark. The string itself is started and endet with a normal quotation mark.

For example, the line ...

```
content += "<p>\"Dieser Absatz steht in Anführungszeichen\"</p>";
```

…equals the HTML code (UTF-8 coded):

```
<p>"Dieser Absatz steht in Anführungszeichen"</p>
```

Without the backslash at the two green marked places, the first green one would already close the string - instead of the fourth in the line.

Finally, the server.send (200, "text / html", content); delivers the website.The first parameter defines the HTTP status code, which is returned when the execution is successful. The code 200 means OK (the request has been successfully processed).Next, the type of the delivered content is defined by the type "Text" and the third parameter passed through the string with the content of the website.

6.6.5 Switch via website

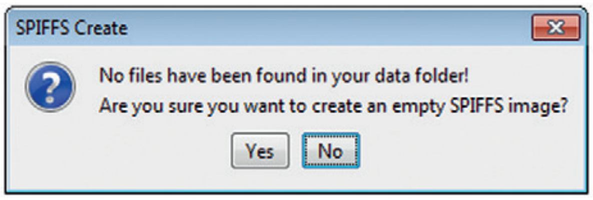
open the following sketch in the Arduino IDE: `Beispiel_6.6.6.ino`. You have to integrate the following header files `ESP8266WiFi.h`, `SSD1306Wire.h`, `TimeLib.h`, `NtpClientLib.h`, `ESP8266WebServer.h` and `FS.h`. If you do not already have the appropriate libraries installed, go to chap. 5.2.1 on page 16 and get this done first.

Also for this example you must first enter your WiFi name (SSID) and the corresponding password in the sample program. Go to as in chap. 6.6.1 described.

As a completion of the Internet of Things examples you will learn how to trigger an action on the IOT brick or control GPIO pins. As the basis for the programming of the website, we need something more than basic HTML tags this time. For this, we use the free CSS framework called Bootstrap, with which you can design relatively simple responsive webdesigns, which then are optimized on mobile devices (such as your smartphone or tablet).There are forms, buttons, tables, navigation and a grid system for layouts as well as various CSS classes and javascript components available. For further information on Bootstrap, go to https://www.bootstrapworld.de.

So that you can get started with sketch programming, you have to do some assiduity work. We need to get access to the SPIFFS file system (ESP8266 File System) of the IoT bricks, in order to be able to copy files there. Proceed as follows:

1. Download the Arduino ESP8266 File Uploader from: https://github.com/esp8266/arduino-esp8266fs-plugin/releases The sketch has been tested with the version 0.2.0 https://github.com/esp8266/arduino-esp8266fs-plugin/releases/download/0.2.0/ESP8266FS-0.2.0.zip
2. Unzip the ZIP file to a directory on your computer. The file Esp8266fs.jar contained in it will by default be extracted to the subdirectory \ ESP8266FS-0.2.0 \ ESP8266FS \ tool.
3. Check if there is already a directory "tools" on your computer in the path "Documents - Arduino" available. If not, create a new folder named "tools" There you can install various tools, which can be operated from the Arduino IDE, like our ESP8266 File Uploader
4. Create another subfolder named "ESP8266FS" in the "tools" directory..
5. In the directory "ESP8266FS" create another subfolder with the name "tool" (without "s" on end).
6. Copy the file esp8266fs.jar (see point 2) to the path "Documents -Arduino - tools - ESP8266FS - tool ".
7. Restart the Arduino IDE
8. The "ESP8266 Sketch Data Upload" option appears in the "Tools" menu.
9. If you have a project with SPIFFS File System support, all the files in the subdirectory "data" of the project folder (standard path: "Documents - Arduino") are converted into a binary format and uploaded to the SPI flash memory of the ESP8266 module.
10. Start the file upload in the "Tools" menu with the option "ESP8266 Sketch Data Upload". If the first upload returns the message"SPIFFS Create", confirm with "Yes".


SPIFFS Create

No files have been found in your data folder!
Are you sure you want to create an empty SPIFFS image?

Uploading the files can take several minutes!

In this example, we will connect the double LED brick to GPIO14 (red LED) and GPIO13 (yellow LED) as shown. Via two buttons on our website, we can then switch the LEDs on and off - theoretically worldwide. The OLED display shows us additional time and date as we do already know from the previous exercises.

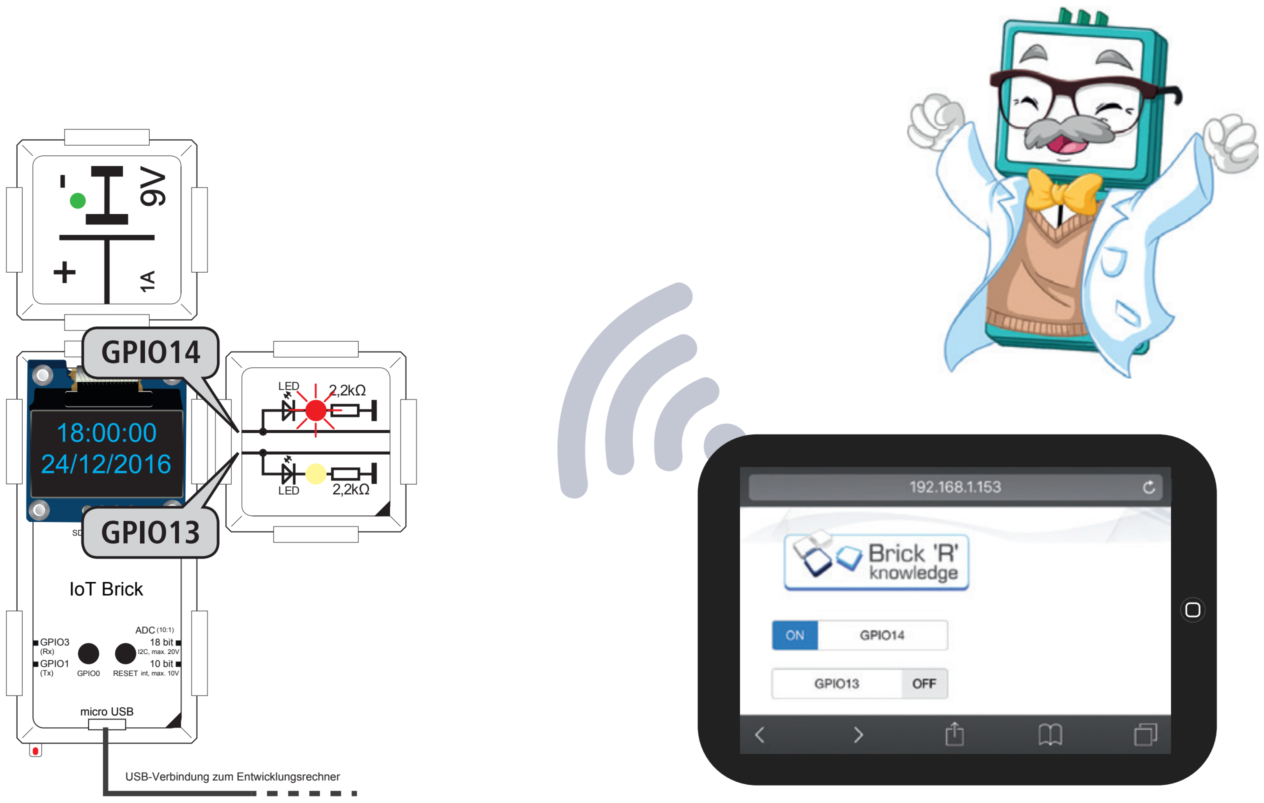


Fig. 13: Brick circuit "Switching via website"

TIP: The connection status of the IoT brick can always be displayed by pushing the button GPIO0 (see figure 44)!

As mentioned, in this example we use advanced web technologies like the Bootstrap framework as well as various CSS classes (CSS = Cascaded Style Sheets) and javascript components (Javascript is a scripts language used frequently in websites). This means that the code is much more extensive than in the previous exercises, but builds on this.. By asking how to use Bootstrap, CSS or javascript we recommend you to search the inter-net accordingly or to buy a corresponding textbook, as this would burst the scope of this Brick'R'knowledge guide.

TIP:

For those who already have good knowledge of HTML, bootstrap, CSS and javascript and even want to code the website themselves, we have put the HTML code from the function handleRoot () in the plaintext ascom-ment. For editing, you can download various free HTML editors such as NVU for Windows, Linux or MAC OS X (see: www.nvu.com) or BlueGriffon (see: www.Bluegriffon.org).

For this example to work correctly, you need to enable javascript in your browser.

This is usually by default.

program sample

```
...
ESP8266WebServer server(80); //start web server on port 80
...
void setup() {
...
//when a browser directly accesses the root directory,
//execute handleRoot.
   server.on("/", handleRoot);
    // JS (Javascript)
    server.on("/js/jquery", handleJsJquery);
    server.on("/js/bootstrap", handleJsBootstrap);
    server.on("/js/bootstrap-switch", handleJsBootstrapSwitch);
    // CSS (Cascaded Style Sheets)
    server.on("/css/bootstrap", handleCssBootstrap);
    server.on("/css/bootstrap-switch", handleCssBootstrapSwitch);
    // Fig.s (Bilder)
    server.on("/img/bg.png", handleImgBg);
    server.on("/img/brklogo.png", handleImgLogo);

    // web server handles for the GPIO switching funktion
    server.on("/switch13-on", handleGpio13On);
    server.on("/switch13-off", handleGpio13Off);
    server.on("/switch14-on", handleGpio14On);
    server.on("/switch14-off", handleGpio14Off);

    server.begin();
    Serial.println("HTTP server started");
    setupPins();
}
...
//activate handle funktion for GPIO13
 void handleGpio13On() {
  digitalWrite(gpio13, dOn);
  server.send(200, "text/html", gpio13Name+" on");
```

Continued on next page ..

In the setup routine of our sketches we first begin with the numerous server.on () statements, which determine which functions are executed as soon as certain links are called in the browser.If, for example the IoT Brick has the IP address 192.168.1.153, given from the DHCP server, then calling the address http://192.168.1.153/img/ brklogo.png will cause the handleImgLogo () function to execute, which returns the Fig. filebrklogo.png from the internal file system (SPIFFS) to the web server. Just like the log file any Java script, CSS or Fig. file that is stored in the internal file system (SPIFFS) is required to have a file handle, so we can access it.In addition, handles are defined with which the web server responds when a browser request arrives to enable or disable a GPIO.After that the web server is started with server.begin.. The handles themselves are defined at the end of the sketche.. In the example code above, we have vicarious the handle function handleGpio13On () to turn on the LED on GPIO13.

program sample (continuation)

```
...
void loop() {

server.handleClient(); //handle the HTTP request

//the handleRoot () function is used to deliver the web site
//as soon as a request arrives from a browser
void handleRoot() {
  String content;
  String network(ssid);
  content += "<html>";

  content += "<head>";
  //... various meta tags, that are not relevant to the understanding
  content += "<title>IoT Brick via WLAN "+network+" schalten</title>";
  content += "<link rel=\"stylesheet\" href=\"/css/bootstrap\">";
  content += "<link rel=\"stylesheet\" href=\"/css/bootstrap-switch\">";
  content += "<style>body{background-Fig.:url(/img/bg.png);margin:0;
             padding:20px;background-size:100% auto;background-repeat:no-repeat;
             font-family:\"Helvetica Neue\",Helvetica,Arial,sans-serif;
             font-size:14px;line-height:1.5;color:#333;background-color:#fff}
             .col-sm-2{margin-top:20px}.img-thumbnail{border:0}</style>";
  content += "</head>";

  content += "<body>";
  content += "<div class=\"container-fluid\">";
  content += "<div class=\"row\">";
  content += "<div class=\"col-sm-2\"><img class=\"img-thumbnail\"
             src=\"/img/brklogo.png\"></div>";
  content += "</div>";
  content += "<div class=\"row\">";
  content += "<div class=\"col-sm-2\"><input type=\"checkbox\" id=\"switch14\"
             data-label-text=\""+gpio14Name+"\" data-label-width=\"120\"></div>";
  content += "<div class=\"col-sm-2\"><input type=\"checkbox\" id=\"switch13\"
             data-label-text=\""+gpio13Name+"\" data-label-width=\"120\"></div>";
  content += "</div>";
  content += "</div>";
  content += "<script src=\"/js/jquery\"></script>";
  content += "<script src=\"/js/bootstrap\"></script>";
  content += "<script src=\"/js/bootstrap-switch\"></script>";
  content += "<script type=\"text/javascript\">";
  content += "$('input[type=\"checkbox\"]').bootstrapSwitch({onSwitchChange:
             function(){$.ajax({url:'/'+$(this).prop('id')+'-'+($(this).prop('checked')?
             'on':'off')});}});";
  content += "var setAdc=function(){$.ajax({url:'/adc'}).done(function(data)
             {data=data||{};if(data.hasOwnProperty('data')){$('#adc')
             .text(data.data);}}).fail(function(jqxhr,textStatus,error){})
             .always(function(){setTimeout(setAdc,1000);});};
             setAdc();";
  content += "</script>";
  content += "</body>";

  content += "</html>";
  server.send(200, "text/html", content);  // HTTP statuscode 200 = ok
  }
}
```

In the function handle Root (), - as already practiced in example 6.6.5 - inside the string variable content the content of our website is saved. Again, it should be noted that in the sketch-programming start and end of a string must be marked with quotation marks at the top ("). Because the HTML code also contains quotation marks, within the string for the correct coding a so-called Escape Sequence \ " is used for every the quotation mark. For the sake of clarity, we write the code snippets cited in the following text in "plain text"..

With the style element `<style>body{background-Fig.:url(/img/bg.png);...` the background picture and other parameter, which change the look of the website, are defined In the body area, you see nested Div elements that are structuring of the website.In a Div element, in turn, various elements such as text, Fig., or forms are summarized and their properties are controlled. For example, the Div element `<div class="col-sm-2"><img class="img-thumbnail" src="/img/brklogo.png">` integrates the logo.

The buttons for switching the LEDs on and off are controlled with an input element of type "checkbox".

The formatting is done by stylesheets from the bootstrap framework. Various script elements towards the end of the sketch, specify the relative path to javascript files, which the site needs to access. The script element <script type = "text / javascript"> finally leads to one longer section with JavaScript code, in which with onSwitchChange the event of the actuation of a button is requested and the handle function is activated to switch the LEDs on or off.

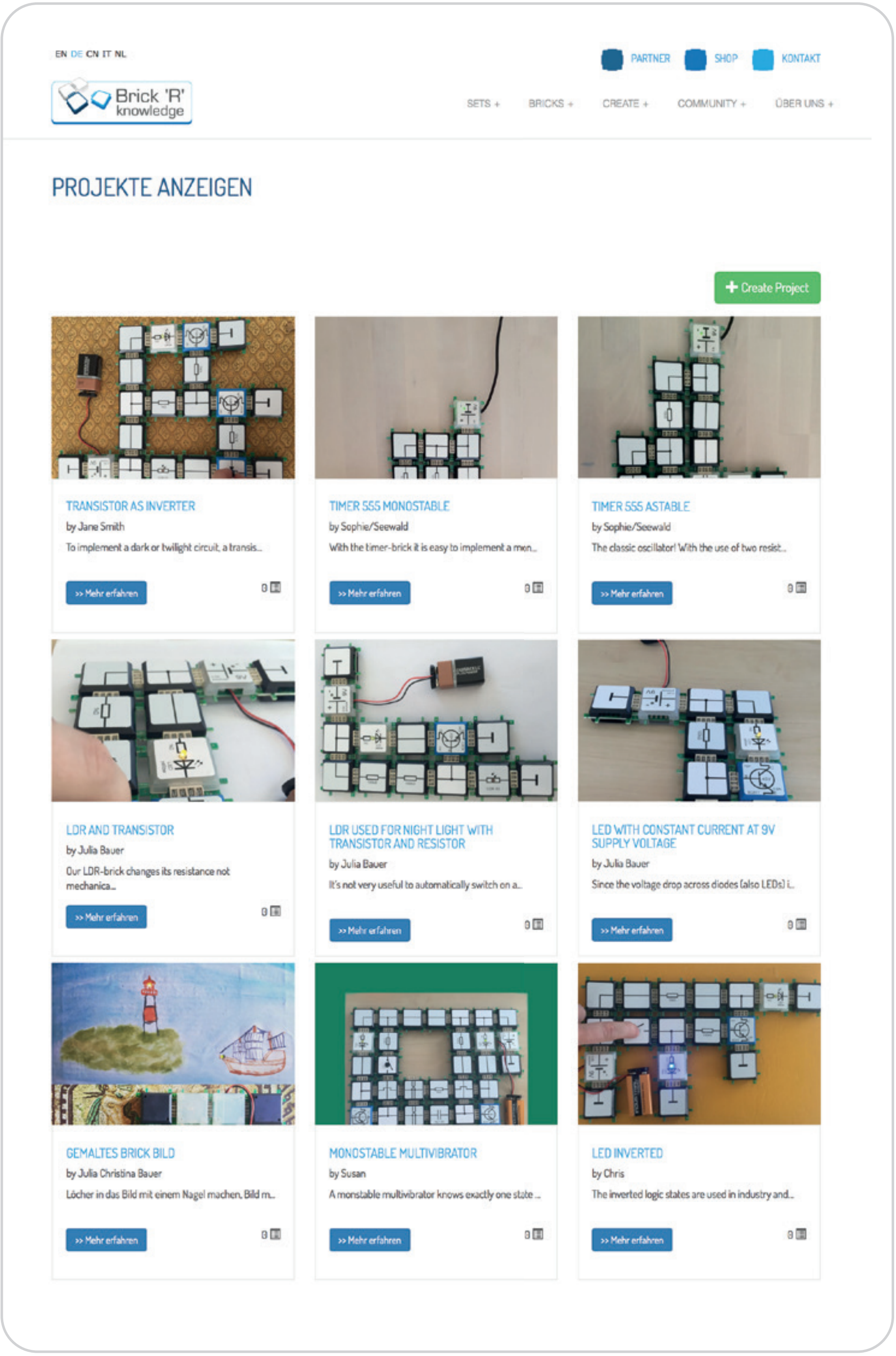
With the command server.send (200, "text / html", content); the site is dynamic delivered to the browser.

7. Brick Community

The brick universe is expanding: You can find more ideas, experiments and bricks at fairs, on our website, on youtoube or social media networks. Boost your creativity!

More projects

By clicking on "Create" you can try out experiments from other users or show your own cool circuits.



PROJEKTE ANZEIGEN

TRANSISTOR AS INVERTER
by Jane Smith
To implement a dark or twilight circuit, a transis...
>> Mehr erfahren 0

TIMER 555 MONOSTABLE
by Sophie/Seewald
With the timer-brick it is easy to implement a mon...
>> Mehr erfahren 0

TIMER 555 ASTABLE
by Sophie/Seewald
The classic oscillator! With the use of two resist...
>> Mehr erfahren 0

LDR AND TRANSISTOR
by Julia Bauer
Our LDR-brick changes its resistance not mechanica...
>> Mehr erfahren 0

LDR USED FOR NIGHT LIGHT WITH TRANSISTOR AND RESISTOR
by Julia Bauer
It's not very useful to automatically switch on a...
>> Mehr erfahren 0

LED WITH CONSTANT CURRENT AT 9V SUPPLY VOLTAGE
by Julia Bauer
Since the voltage drop across diodes (also LEDs) i...
>> Mehr erfahren 0

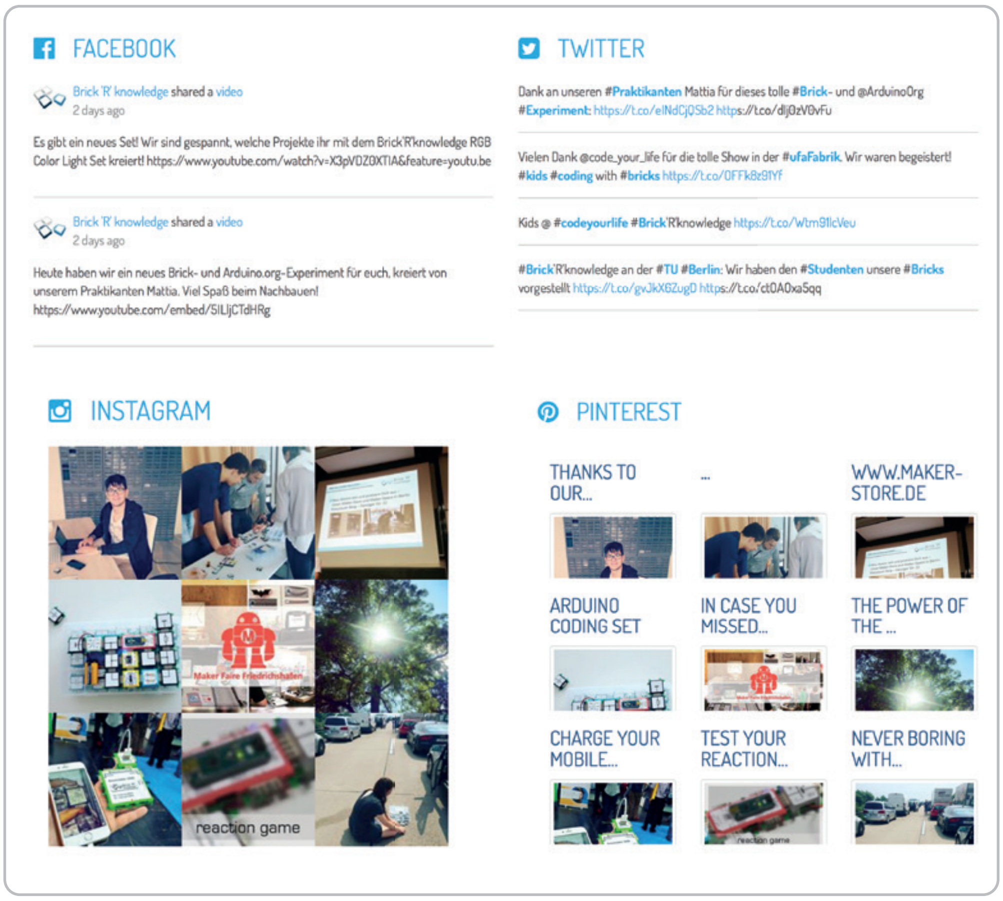
GEMALTES BRICK BILD
by Julia Christina Bauer
Löcher in das Bild mit einem Nagel machen, Bild m...
>> Mehr erfahren 0

MONOSTABLE MULTIVIBRATOR
by Susan
A monstable multivibrator knows exactly one state ...
>> Mehr erfahren 0

LED INVERTED
by Chris
The inverted logic states are used in industry and...
>> Mehr erfahren 0

Social Media

By clicking on "Community" you can find all of our social media networks. Stay up-to-date!



FACEBOOK

Brick 'R' knowledge shared a video
2 days ago

Es gibt ein neues Set! Wir sind gespannt, welche Projekte ihr mit dem Brick'R'knowledge RGB Color Light Set kreiert! https://www.youtube.com/watch?v=X3pVDZ0XTIA&feature=youtu.be

Brick 'R' knowledge shared a video
2 days ago

Heute haben wir ein neues Brick- und Arduino.org-Experiment für euch, kreiert von unserem Praktikanten Mattia. Viel Spaß beim Nachbauen!
https://www.youtube.com/embed/5ILIjCTdHRg

TWITTER

Dank an unseren #Praktikanten Mattia für dieses tolle #Brick- und @ArduinoOrg #Experiment: https://t.co/eINdCjQSb2 https://t.co/dIj0zV0vFu

Vielen Dank @code_your_life für die tolle Show in der #ufaFabrik. Wir waren begeistert! #kids #coding with #bricks https://t.co/0FFk8z91Yf

Kids @ #codeyourlife #Brick'R'knowledge https://t.co/Wtm91lcVeu

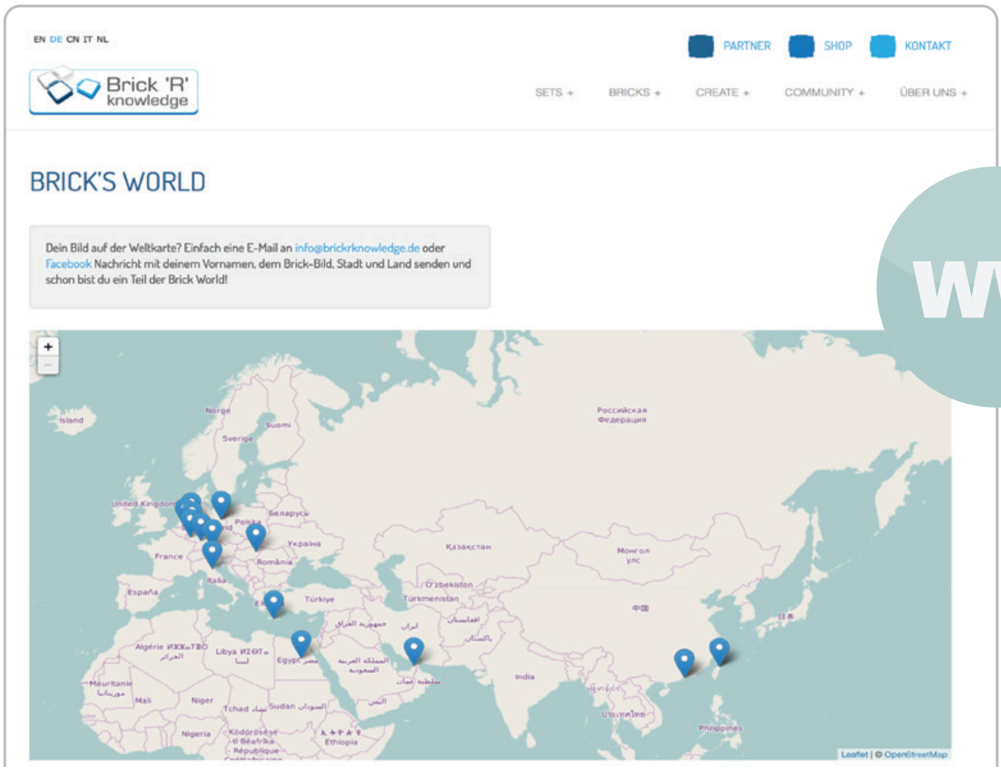
#Brick'R'knowledge an der #TU #Berlin: Wir haben den #Studenten unsere #Bricks vorgestellt https://t.co/gvJkX6ZugD https://t.co/ct0A0xa5qq

INSTAGRAM

PINTEREST

Worldwide

At "Community" you can also nd out where our bricks were already. Do you have a nice picture of bricks in your town? Just send it to us and soon you will nde it on our website!

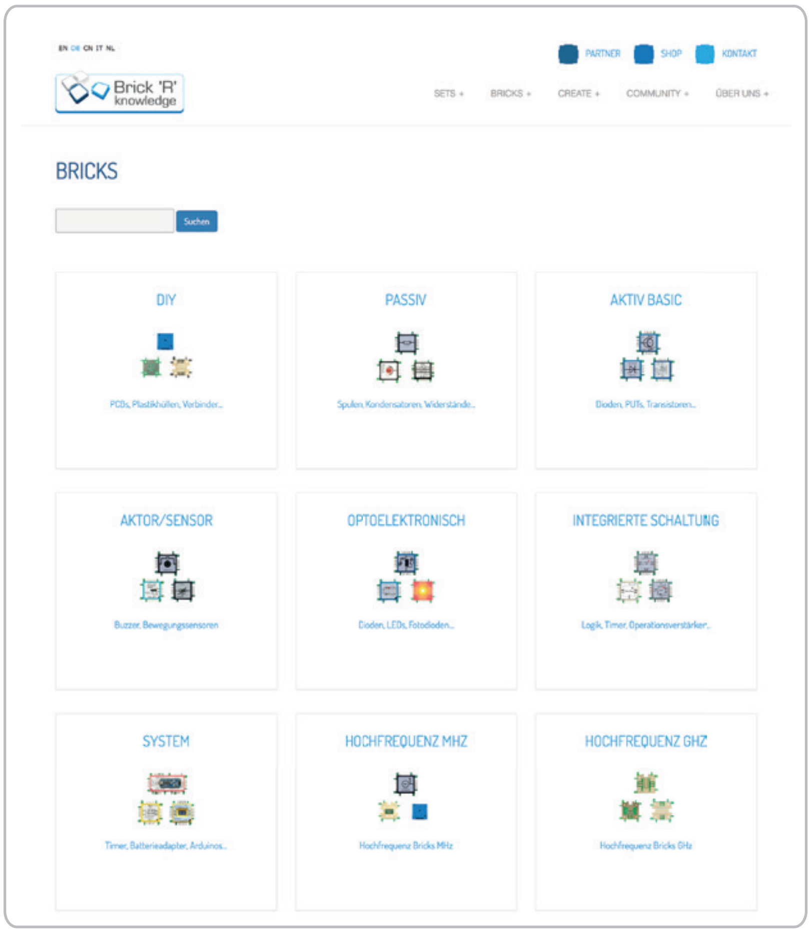


BRICK'S WORLD

Dein Bild auf der Weltkarte? Einfach eine E-Mail an info@brickrknowledge.de oder Facebook Nachricht mit deinem Vornamen, dem Brick-Bild, Stadt und Land senden und schon bist du ein Teil der Brick World!

Even more bricks!

By clicking on "Bricks" you can find all of the available bricks with information and ideas for experiments.



BRICKS

DIY
PCBs, Plastikhüllen, Verbinder...

PASSIV
Spulen, Kondensatoren, Widerstände...

AKTIV BASIC
Dioden, PUTs, Transistoren...

AKTOR/SENSOR
Buzzer, Bewegungssensoren

OPTOELEKTRONISCH
Dioden, LEDs, Fotodioden...

INTEGRIERTE SCHALTUNG
Logik, Timer, Operationsverstärker...

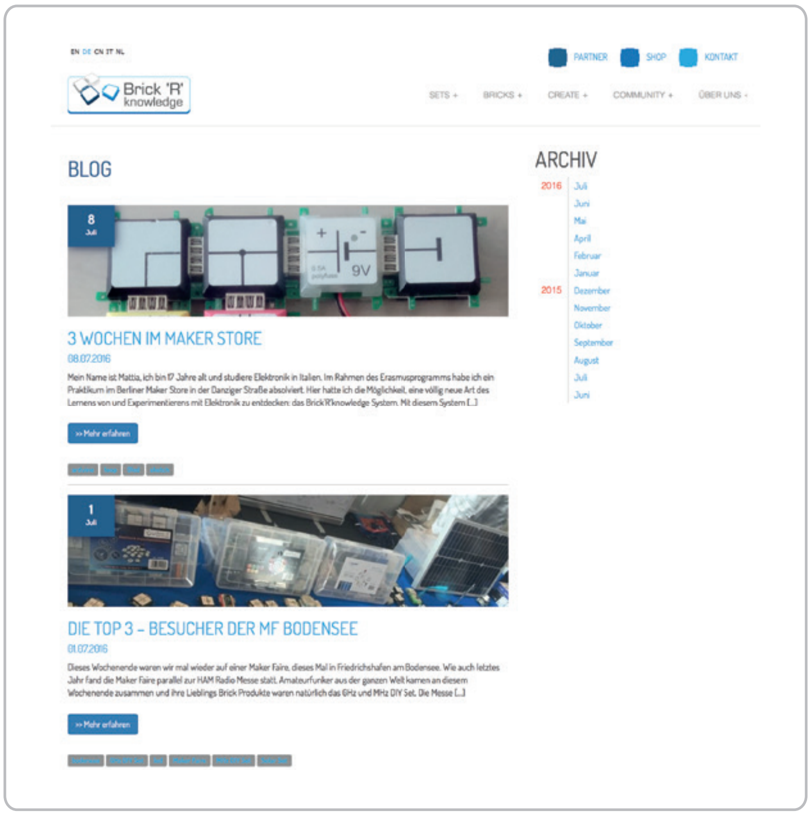
SYSTEM
Timer, Batterieadapter, Arduinos...

HOCHFREQUENZ MHZ
Hochfrequenz Bricks MHz

HOCHFREQUENZ GHZ
Hochfrequenz Bricks GHz

Brick Blog

Each week we post a new blog post. You can read about our experiences at fairs, new circuits, funny stories and information of the world of electronics.



BLOG

3 WOCHEN IM MAKER STORE

08.07.2016

Mein Name ist Mattia, ich bin 17 Jahre alt und studiere Elektronik in Italien. Im Rahmen des Erasmusprogramms habe ich ein Praktikum im Berliner Maker Store in der Danziger Straße absolviert. Hier hatte ich die Möglichkeit, eine völlig neue Art des Lernens von und Experimentierens mit Elektronik zu entdecken: das Brick'R'knowledge System. Mit diesem System [...]

DIE TOP 3 – BESUCHER DER MF BODENSEE

01.07.2016

Dieses Wochenende waren wir mal wieder auf einer Maker Faire, dieses Mal in Friedrichshafen am Bodensee. Wie auch letztes Jahr fand die Maker Faire parallel zur HAM Radio Messe statt. Amateurfunker aus der ganzen Welt kamen an diesem Wochenende zusammen und ihre Lieblings Brick Produkte waren natürlich das GHz und MHz DIY Set. Die Messe [...]

ARCHIV

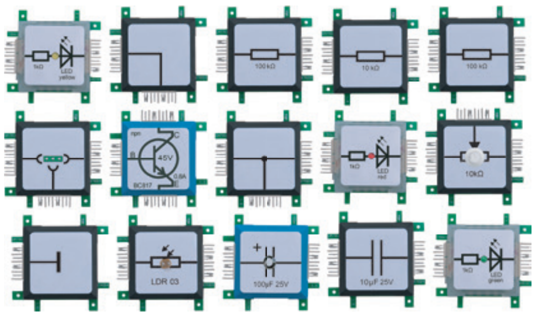
2016 Juli
Juni
Mai
April
Februar
Januar
2015 Dezember
November
Oktober
September
August
Juli
Juni

8. Brick sets overview

Basic Set

ALL-BRICK-0374

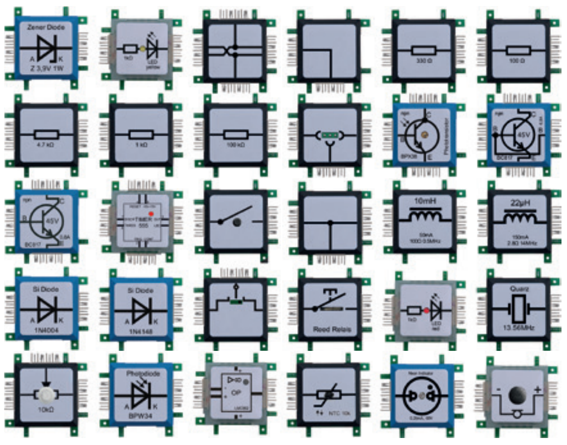
The basic set contains 19 selected bricks to offer a fast and easy start into the world of Brick `R` knowledge as well as the possibility to create numerous circuits. The basic set is a perfect support for kids gaining their rst experiences with electronic and technical experiments.



Advanced Set

ALL-BRICK-0223

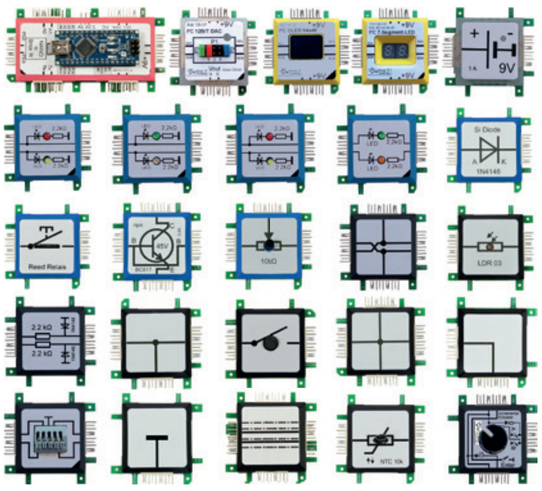
Our Advanced Set contains 111 components that allow you to build more complicated and complex solutions. Thanks to the educational system, knowledge can be gathered, so that not only you but also our next generation can prot from it. You can build individual circuits by plugging di erent bricks together. Simple as well as complex electronic and technological topics can be experienced in a totally new way. Due to the open-source factor, you can create your own bricks and develop your own solutions.
Brick'R'knowledge isn't all about basic electronic engineering, also RF experiments can be realized, which makes it a unique system worldwide.



Arduino Coding Set

ALL-BRICK-0414

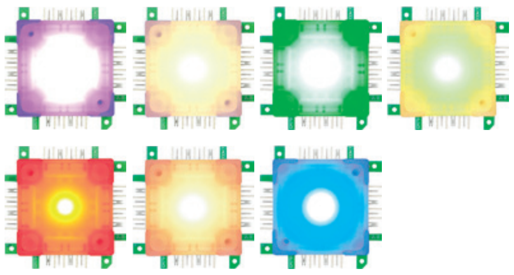
Get in touch with digital electronics and start understanding programming with the Arduino® Nano, which is included in the kit. It is our rst kit with digital components, such as 7-segment displays, OLED display, D/A converter or I2C Bricks, complementary to all analog bricks. To get you started with the popular microcontroller, we support you with various programming examples.



7 Color Light Set

ALL-BRICK-0398

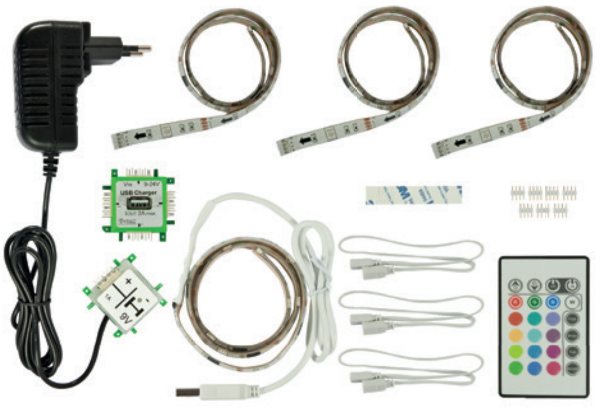
The 7 Color Light Set contains 28 LED bricks in 7 different colors to create stunning light effects in a horizontal and vertical architecture. The red, yellow, blue, orange, violet, green and warm white 1 watt LEDs are perfect for individual lighting characters or as a mobile lighting solution.



RGB Color Light Set

ALL-BRICK-0619

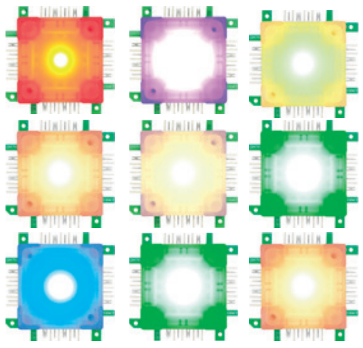
Create your light show! The RGB Color Light Set comes with four exible LED strips containing 36 LEDs in total that can be controlled with the included infrared remote control. You can glue, cut and connect the LED strips however you want. The infrared remote control has 16 different color keys and 4 light programs.



Programmable LED Set

ALL-BRICK-0483

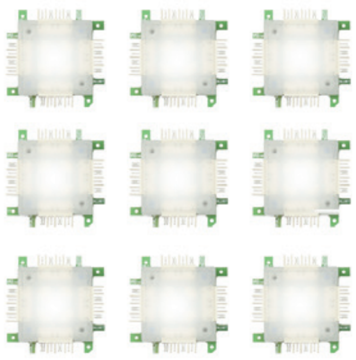
The kit contains 49 programmable and controllable RGB LED bricks, each with two or three connectors and a conjunction-brickfor Arduino management and power supply. Furthermore, the Brick'R'knowledge Programmable LED Set includes an Arduino adapter brick and an Arduino Nano. With this kit you can realize colorful LED animations and other individual ideas. And the best thing about it: by performing di erent projects, you can easily learn the programming of microcontrollers.



Highpower LED Set

ALL-BRICK-0399

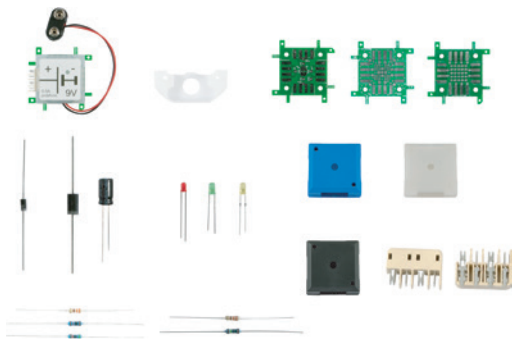
Powered by 1 Watts, each of the 50 High Power LED bricks contained in the kit irradiate the whole surrounding area in bright white. Build individual solutions in every imaginable architecture and invent Brick nightlights, Brick table lamps or any other creative illuminant. The power supply with 12V 8A supports the intensive luminosity to o er a stylish and cozy atmosphere. The High Power LED Set 50 allows you to deal with modern light design and simultaneously learn about electronics.



DIY Set

ALL-BRICK-0397

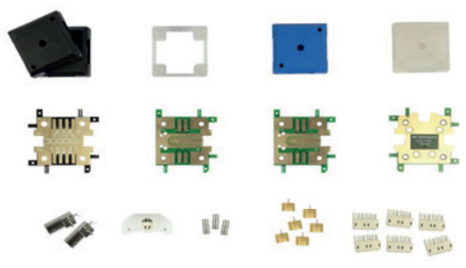
The DIY set goes even a step further. The included components offer a much more detailed insight into the brick architecture and allow even the production of individual bricks. The DIY set o ers an enormous exibility for the maker generation or for people creating individual bricks.



MHz DIY Set

ALL-BRICK-0457

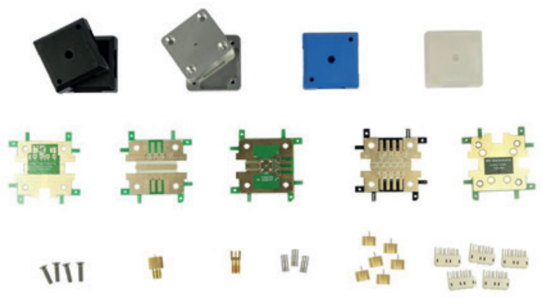
Individual challenging projects within the MHz frequency range can be created with the MHz DIY set. Three different grid and experimentation boards, BNC sockets, P-SMP plugs and suitable connectors make the kit perfect for any high frequency experiment. The kit contains hermaphrodite connectors and a soldering jig for SMD plugs to develop your own bricks or other components for the Brick system.



GHz DIY Set

ALL-BRICK-0458

Realize advanced and complicated experiments in the high frequency range up to GHz frequencies. In addition, the kit offers four diferent PCBs, P-SMP, SMA sockets, P-SMP connectors and brick-specic hermaphrodite connectors. The GHz DIY Set is perfect for HAM radio operators and fans of measuring.



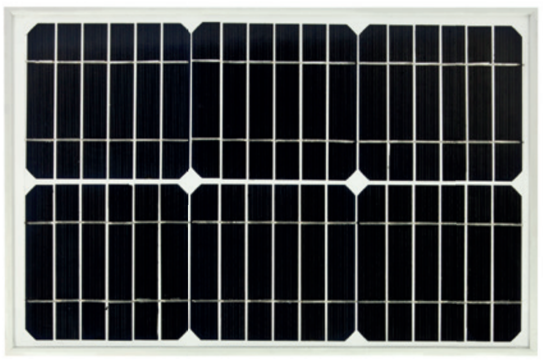
Solar Set

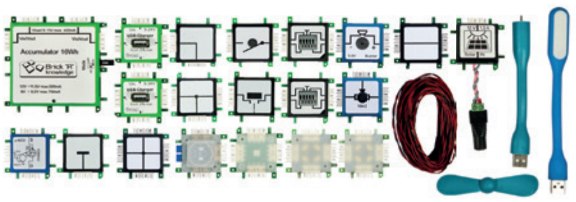
ALL-BRICK-0484

Das Solar Set von Brick'R'knowledge garantiert Experimentierspaß für die ganze Familie und bringt Kindern erneuerbare Energien auf spielerische Art und Weise näher.

- Wie funktioniert eine Solarzelle?
- Wie speichert ein Akku Strom?
- Wie baut man ein Nachtlicht mit Bewegungsmelder?

Auf diese und weitere Fragen gibt das Solar Set Antworten. Mit diesem Set sind Sie und Ihre Kinder offizielle Mitglieder der Maker-Generation.





Measurement Set One

ALL-BRICK-0637

The Measurement Set ONE enables you to measure the voltage, current and other measured variables with standard measuring instruments. The set contains measuring adapters (3x2mm), with additional cable clamp and measuring adapters (4mm) with a yellow endpoint.

Measurement Set Two

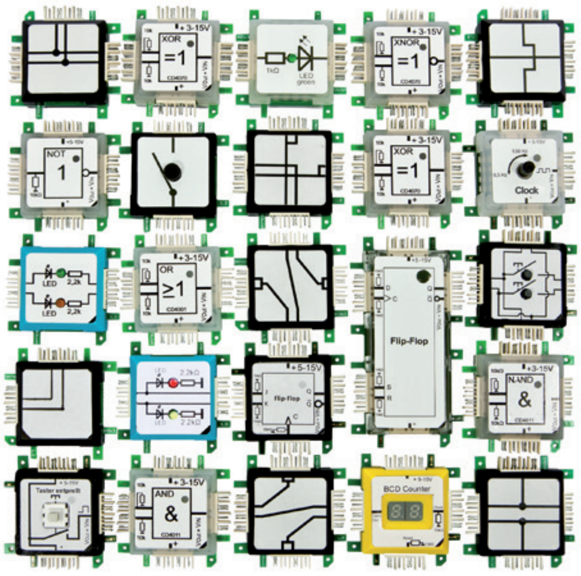
ALL-BRICK-0638

The Measurement Set Two enables you to measure the voltage, current and other measured variables with standard measuring instruments. It contains measuring adapters (4mm) with closed end GND, measuring adapters (4mm) inline red and measuring adapters (4mm) with open end GND black.

Logic Set

ALL-BRICK-0630

The Logic Set is ideally suited for a quick start into the digital circuit technology. While working with the manual, which includes didactically structured examples of circuits, students learn about the most important digital circuits like adder, shift register and numerator. The comprehensively equipped Logic Set provides teachers with a practical basis for daily teaching. Plugging the bricks together and experimenting with them is fun and encourages building your own circuit variants. The Logic Set's scope of supply ranges from easy logic bricks (AND, OR, NAND, NOR, XOR, XNOR, NOT), to a variety of flip flop bricks (D-, RS- and JK-type), to an impulse brick (alternatively a debounced switch for single pulses) up to a BCD counter brick with an integrated 7 segment display. A wide range of LED bricks, switch bricks and wire bricks make the set complete.



Brick 'R' knowledge

ALLNET© GmbH Computersysteme
Maistrasse 2
D-82110 Germering
www.brickrknowledge.com

Telefon: +49 (0)89 894 222 921
Fax: +49 (0)89 894 222 33
info@brickrknowledge.com

Maker Store & Maker Space
Danziger Straße 22
D-10435 Berlin
www.maker-store.de

Telefon: +49 (0)30 473 756 80
service@allknow.de

www.brickrknowledge.de