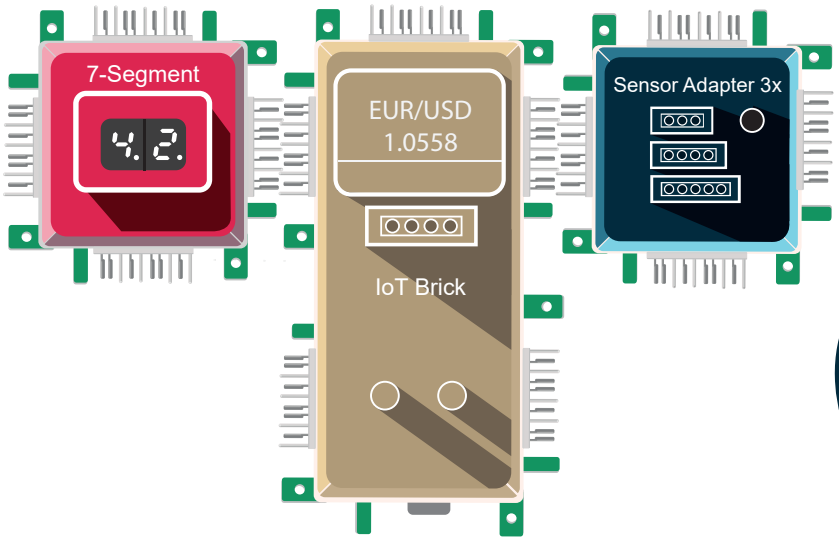
Brick 'R' knowledge

Set Internet de las Cosas

Kit experimental de Brick'R'knowledge
Experimentierkasten von Brick'R'knowledge



Aviso legal

Brick'R'knowledge, manual del set Internet de las Cosas
Rev. 1.0
Fecha: 27.07.2017

ALLNET® y Brick'R'knowledge® son marcas registradas de ALLNET® GmbH Computersysteme.

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

© Copyright 2017 ALLNET GmbH Computer systems. Todos los derechos reservados.

Toda la información contenida en este manual ha sido recopilada con el mayor cuidado y a nuestro leal saber y entender. Sin embargo, no se pueden descartar completamente los errores. Estamos siempre agradecidos por la notificación de posibles errores. Por favor, envíelos a info@brickrknowledge.de.

Índice

Prólogo

El sistema de experimentación Brick'R'knowledge fue presentado por primera vez en la exposición de radio HAM el 28.06.2014 por Rolf-Dieter Klein (Amateurfunkzeichen: DM7RDK). La característica principal de nuestros sets electrónicos es que los componentes individuales están conectados a través de un sistema de conectores, en el que las piezas a ensamblar son idénticas (hermafrodita). De este modo, incluso los circuitos más complicados se pueden realizar. También es posible el montaje de bloques de construcción individuales en diferentes ángulos! Hay dos contactos disponibles para el retorno a tierra (0 voltios)!

Esto permite construir circuitos compactos en los que el retorno a tierra garantiza una alimentación de tensión estable para los dispositivos. Otra característica particular es que tales circuitos pueden ser fácilmente explicados y documentados.

Rolf-Dieter Klein

Descargas:

- Código de muestra y bibliotecas para los ejercicios de este tutorial: https://www.arduino.cc/en/Main/Software#
- Descargar el entorno de programación de Arduino: http://www.brickrknowledge.de/downloads

1. Instrucciones de seguridad

Atención: los Bricks del set de electrónica nunca se deben conectar directamente a la red eléctrica de (230V), existe peligro de muerte!

Para su alimentación (9V) sólo se puede utilizar la fuente (Brick adaptador de Pila) que va incluida. La tensión de 9V y un flujo de corriente de aprox. 1A no suponen ningún riesgo para la salud. Por favor, tenga especial cuidado de que los cables abiertos no entren en contacto o caigan dentro de una regleta de enchufes ya que existe el riesgo de recibir una descarga eléctrica o de electrocutarse. Nunca mire directamente a un diodo de emisor de luz (LED), ya que existe el riesgo de dañar la retina (deslumbrar).

El condensador polarizado (condensador de tantalio/condensador electrolítico) incluido en el set nunca se debe conectar con el contacto marcado en “Plus/Positivo” de manera directa o indirecta al conector de la fuente de alimentación (9V) marcado con “Minus/Negativo”, sino directa o indirectamente al conector de la fuente de alimentación (9V) marcado con “Plus/Positivo”. A esto se le llama polaridad. Si el condensador electrolítico tiene la polaridad invertida, es decir que no se tiene en cuenta lo anterior descrito, existe el riesgo de que se pueda estropear e incluso haber riesgo de explosión.

Es importante asegurarse de que la fuente de alimentación (Brick adaptador de Pila) está desconectada después de cada experimento, ya que sino existe el riesgo de un incendio eléctrico.

2. ¿Qué significa "Internet de las cosas"?

El Internet de las cosas (IdC) es la interconexión de dispositivos físicos, vehículos (también denominados "dispositivos conectados" y "dispositivos inteligentes"), edificios y otros elementos incorporados en la electrónica, el software, los sensores, los actuadores y la conectividad de red que permiten a estos objetos recopilar e intercambiar información. En 2013, el comité de Estándares Globales para el Internet de las cosas (IdC-GSI) definió el IdC como "una infraestructura global para la sociedad de la información, que permite la provisión de servicios avanzados a través de la interconexión de cosas (físicas y virtuales) basadas en tecnologías de la información y comunicación existentes, interoperables y en continua evolución," y con este fin, una "cosa" es "un objeto del mundo físico (cosas físicas) o del mundo de la información (cosas virtuales), que es capaz de ser identificada e integrada en redes de comunicación.

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

Ajá, ahora algunos pensarán, pero ¿qué significa esto para mí?

El Internet de las cosas (IdC) es la interconexión de dispositivos físicos, vehículos (también denominados "dispositivos conectados" y "dispositivos inteligentes"), edificios y otros elementos incorporados en la electrónica, el software, los sensores, los actuadores y la conectividad de red que permiten a estos objetos recopilar e intercambiar información. En 2013, el comité de Estándares Globales para el Internet de las cosas (IdC-GSI) definió el IdC como "una infraestructura global para la sociedad de la información, que permite la provisión de servicios avanzados a través de la interconexión de cosas (físicas y virtuales) basadas en tecnologías de la información y comunicación existentes, interoperables y en continua evolución," y con este fin, una "cosa" es "un objeto del mundo físico (cosas físicas) o del mundo de la información (cosas virtuales), que es capaz de ser identificada e integrada en redes de comunicación.

Ejemplos sobre el Internet de las Cosas:

- Los llamados vestibles, con sensores incorporados en las prendas de vestir
- Automatización de edificios
- Control de la temperatura en la sala de servidores
- Gestión de la energía, medición inteligente
- Videovigilancia

... y por último, el set Internet de las Cosas de Brick'R'knowledge con el que ahora puedes acceder a tus Bricks a través de Internet.

En primer lugar, obtendrás una visión general de los fundamentos del sistema de experimentación del Brick'R'knowledge. A continuación, se introducen todos los bricks y sensores incluidos en este set. Una característica especial de este set es la combinación de software y hardware. Es decir, necesitamos un software para crear programas, que dé vida al hardware. Para ello, utilizamos el entorno de programación de Arduino, que puede descargarse gratuitamente. Después de haber realizado todos estos pasos, podemos entrar en el mundo del Internet de las Cosas con numerosos ejemplos.

Con el brick central del IdC, aprenderás a construir tu primera página web y a controlar los pines E/S con tu smartphone. Además, el set contiene un sensor de temperatura y humedad, cuyos valores son visualizables. El primer paso hacia tu propio proyecto de domótica! También puedes ver y visualizar información, como la cotización del dólar, desde Internet. También incluye un bus I2C para conectar una pantalla de 7 segmentos o un convertidor A/D de 18 bits.

El Internet de las Cosas te está esperando!

3. Fundamentos del sistema Brick'R'knowledge

3.1 Brick de tierra

El Brick de tierra es un componente especial en nuestro set electrónico. Ahorra conexiones adicionales con la ayuda de otros Bricks o líneas. Aquí revelamos el secreto de nuestros conectores de cuatro polos. Los dos contactos centrales están reservados a la transmisión de señal, tal y como lo indica la etiqueta. Los contactos exteriores se utilizan para cerrar el circuito de corriente, es decir, el retorno de corriente a la fuente de alimentación. Esto se logra mediante el Brick de Tierra. Por lo tanto, este Brick se denomina Brick de Tierra porque en electrónica la designación "tierra" no describe el peso inercial del propio Brick, sino el potencial de referencia al cual se designan todos los demás potenciales. Así que, nuestro Brick proporciona exactamente esta conexión a 0V. En nuestro circuito esto son 9 voltios frente a 0 voltios: simplemente hablamos de "nueve voltios". Los circuitos electrónicos están diseñados de tal manera que después de que todos los componentes se hayan conectado a circuitos más o menos complejos, estén conectados a la "tierra".

Nuestro Brick de Tierra conecta los dos contactos del medio con los dos contactos exteriores. Con esto no provocamos ningún cortocircuito, porque la corriente aún fluye en los componentes en el interior del Brick.

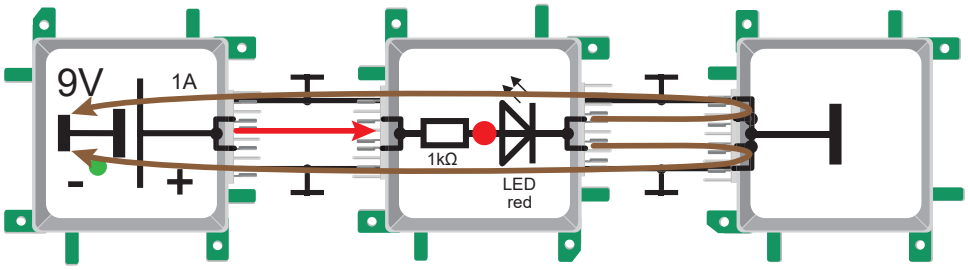


Fig. 1: Flujo de tierra

3.2 Fuente de alimentación

Fig. 2: Adaptador de alimentación

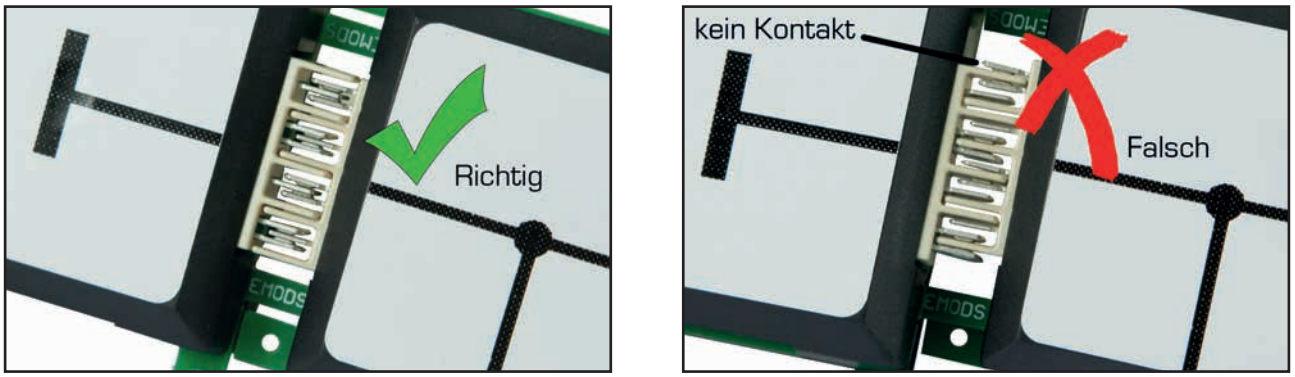
El set IdC se alimenta a través de la fuente de alimentación enchufable de 9V (ALL-BRICK-0221) suministrada. Proporciona una tensión continua estabilizada de 9V y una corriente máxima de 1A. En caso de sobrecarga, la fuente de alimentación se desconecta, ya que es a prueba de cortocircuitos. Un LED indica cuando el voltaje del brick está disponible.

Opcionalmente, también está disponible un brick de alimentación a través de una batería de bloque de 9V (ALL-BRICK-0001).

Cuando conectes los bricks para hacer los ejemplos de práctica, asegúrate siempre de conectar el brick de alimentación como el último brick de tu circuito. Al final del experimento, la fuente de alimentación debe desconectarse de la red!

3.3 Conectores

Al interconectar los Bricks hay que asegurarse de que los contactos se tocan correctamente, ya que de lo contrario podrían producirse interrupciones o incluso cortocircuitos!



Insertado correctamente

Insertado incorrectamente

Fig. 3: Conectores

En la imagen de la izquierda tenemos un ejemplo de una conexión realizada correctamente. La conexión consiste en pequeños pines, que se unen mecánicamente y por lo tanto conducen la electricidad. Para garantizar un aislamiento entre los contactos y evitar un cortocircuito, se introducen bandas de plástico que no conducen la corriente eléctrica.

En la imagen de la derecha tenemos un ejemplo de una conexión incorrecta. Hay espacios entre los contactos, que no pueden garantizar el flujo de corriente seguro. El circuito permanece "abierto" o es inestable y no se proporciona la función del circuito.

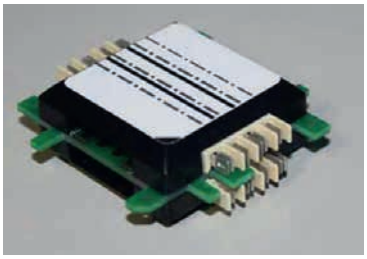
Prudencia: Es muy importante controlar el ajuste correcto de los pines, ya que si están muy separados, pueden provocar un cortocircuito.Entonces, el flujo de corriente no se produce a través de nuestros componentes con el efecto deseado, sino que busca el camino más corto de regreso a la fuente de alimentación.

Un cortocircuito lleva a un flujo de corriente máximo porque la única resistencia que debe superar la corriente eléctrica es la resistencia interna de la fuente de alimentación. Esta resistencia es claramente muy pequeña, por lo que la corriente de cortocircuito puede provocar un sobrecalentamiento durante un período de tiempo más prolongado. ¡En este caso, existe riesgo de incendio!

Importante: Verificar siempre la posición correcta de los contactos!

3.4 Bricks de conexión especiales para niveles inferiores

El brick de IdC también dispone de una clavija de señal (GPIO12) en el nivel inferior de la clavija. Bricks especiales como "Downside Up cable" (ALL-BRICK-0385) y "6-way straight cable" (ALL-BRICK-0383) están disponibles opcionalmente para acceder a este contacto. Se debe tener especial cuidado al juntarlos. Una lengüeta de protección impide que los bricks se introduzcan desde arriba para evitar cortocircuitos durante el proceso de conexión.



4. Visión general del Hardware

El set Internet de las Cosas contiene los siguientes bricks, sensores y accesorios, brevemente introducidos.

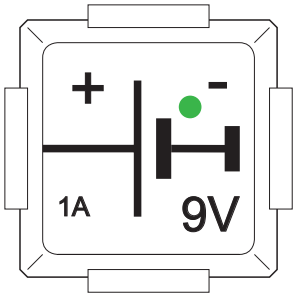
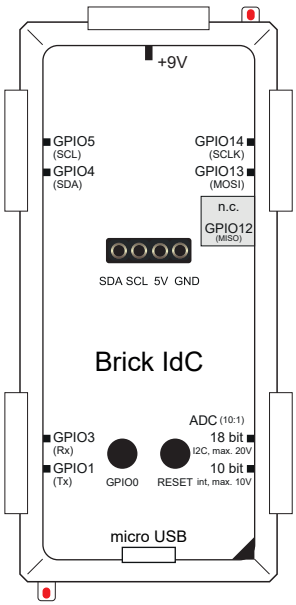
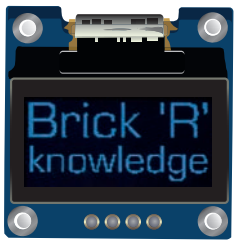
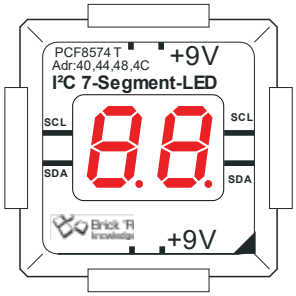
Ilustración	Volumen	Art.-Nº /ID del Brick	Breve descripción
	1	Art.-Nº: 118627 ID del Brick: ALL-BRICK-0221	**Adaptador de alimentación de 9V** El adaptador de red de 9 V proporciona una corriente máxima de 1 A para los bricks. Está estabilizado y protegido contra posibles corto-circuitos. Un LED indica cuando está en funcion-amiento. El polo positivo se extrae y el negativo se conecta a tierra. Utilice este brick como el último, una vez hayas comprobado el circuito de nuevo.
	1	Art.-Nº: 136716 ID del Brick: ALL-BRICK-0635	**Brick IdC ESP8266** El núcleo del set IdC. Módulo ESP8266 con interfaz WLAN, alimentación: +9 V. 7 GPIOs, convertidor A/D de 10 bits, convertidor A/D de 18 bits, interfaz I2C, interfaz SPI Para más detalles consulte el capítulo 5.1 en la página 13.
	1	Art.-Nº: 139779 ID del Brick: ALL-BRICK-0659	**Pantalla OLED para Brick IdC** Pantalla con iluminación orgánica monocromáti-ca. Exactamente, 128 x 64 LEDs están dispuestos en una matriz y pueden ser controlados indivi-dualmente mediante comandos a través del bus I2C. Esto permite mostrar textos multilínea, pero también gráficos monocromáticos simples. Dirección I2C estándar: $78_{(16)}$
	1	Art.-Nº: 113713 ID del Brick: ALL-BRICK-0086	**Pantalla de 7 segmentos I2C** Pantalla de 7 segmentos compuesta por 7 barras luminosas y un punto. Detrás de cada uno hay un LED. Los LED se controlan mediante dos módulos 8574T (16 líneas de salida a los LED). La dirección I2C se puede ajustar con pequeños interruptores en la parte posterior. Un máximo de cuatro, de estos dispositivos, pueden ser operados en un bus I2C.

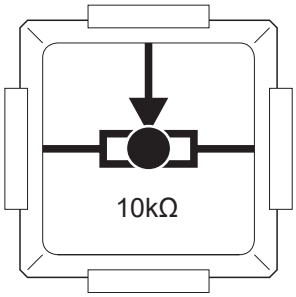
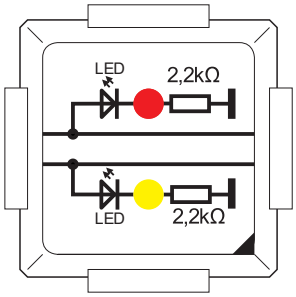
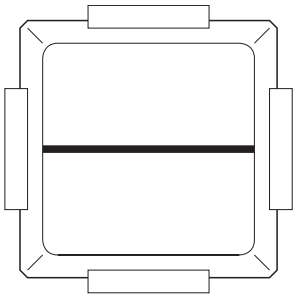
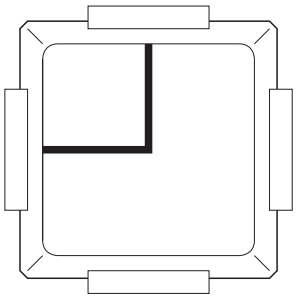
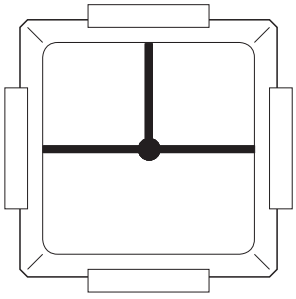
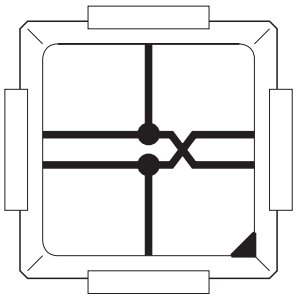
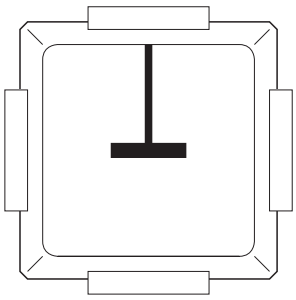
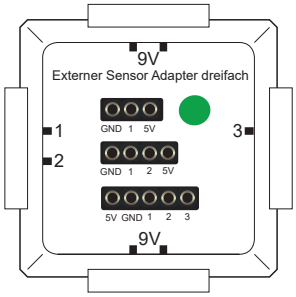
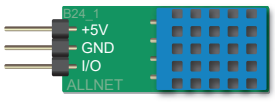
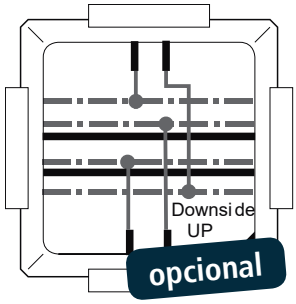
Ilustración	Volumen	Art.-Nº /ID del Brick	Breve descripción
	1	Art.-Nº: 113654 ID del Brick: ALL-BRICK-0027	**10 kΩ potenciómetro** El potenciómetro es una resistencia ajustable manualmente. En este caso, un tercer contacto (deslizador) se desplaza a lo largo de la resistencia y, por lo tanto, cambia la intensidad del valor de la resistencia eléctrica en su conexión. Se puede ajustar en el rango de 0 a 10kΩ. Se produce un cortocircuito si el deslizador o también llamado contacto central, o uno de los otros contactos está conectado directamente a la fuente de alimentación. Esto debe evitarse por completo! El potenciómetro tiene una potencia máxima de aprox. 1/8 W.
	1	Art.-Nº: 125693 ID del Brick: ALL-BRICK-0410	**Doble LED a tierra (rojo / amarillo)** El módulo contiene dos LED (rojo/amarillo) que están conectados internamente a tierra. Las líneas de señal se conectan por separado. Ambos LED están protegidos contra el flujo excesivo de corriente por una resistencia en serie de 2.2 kΩ. Están diseñados para una corriente de 2 mA a una tensión de 5 V. Las dos resistencias están conectadas internamente a tierra, de modo que el módulo puede conectarse directamente.
	1	Art.-Nº: 113631 ID del Brick: ALL-BRICK-0004	**Cable recto** El brick de conector recto conecta dos bricks opuestos.
	3	Art.-Nº: 113632 ID del Brick: ALL-BRICK-0005	**Brick de esquina** El brick de esquina une dos partes adyacentes.
	1	Art.-Nº: 113633 ID del Brick: ALL-BRICK-0006	**Brick cruce en T** Con el brick de cruce en "T" puedes conectar los componentes de tu circuito como una "T".

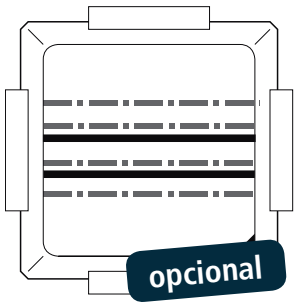
Ilustración	Volumen	Art.-Nº /ID del Brick	Breve descripción
	1	Art.-Nº: 113675 ID del Brick: ALL-BRICK-0048	**Cable cruzado doble** Con este módulo se pueden enviar los cables centrales por separado y cruzarlos al mismo tiempo.
	1	Art.-Nº: 113630 ID del Brick: ALL-BRICK-0003	**Brick de tierra** Con el brick de tierra se puede conectar fácilmente a tierra en cualquier punto del circuito. Coloca el brick de tierra al final de cada circuito para cerrarlo. Este brick conecta los dos contactos centrales de la conexión con las dos líneas de tierra externas.
	1	Art.-Nº: 138408 ID del Brick: ALL-BRICK-0649	**Adaptador triple de sensor externo** Tensión de alimentación: +9 V.
	1	Art.-Nº: 139778 ID del Brick: ALL-BRICK-0658	**Sensor de temperatura y humedad** Sensor de temperatura/humedad de platino, Tipo de sensor: DHT11, Temp.0,50 °C (± 2 °C), rel. Humedad: 20..95% (± 5%), Alimentación: 3-5,5V. Resistencia de arranque incorporada.
	1		**Cable USB** Cable de conexión USB (enchufe tipo A para conector micro USB tipo B) entre el ordenador de desarrollo y el brick IdC.

Bricks opcionales recomendados



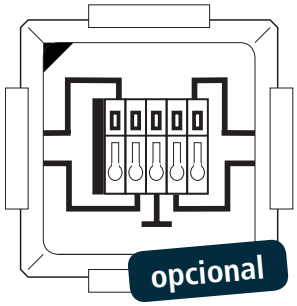
Art.Nº: 122448
ID del Brick: ALL-BRICK-0385

Cable "Downside up"
Un módulo especial para acceder al pin GPIO12 de los bricks inferiores del IdC. Además de los contactos en el nivel superior, también tiene cuatro contactos independientes en la parte inferior de la placa. Estos se dirigen al nivel superior en los laterales, donde los bricks normales pueden ser utilizados para llegar a las señales.



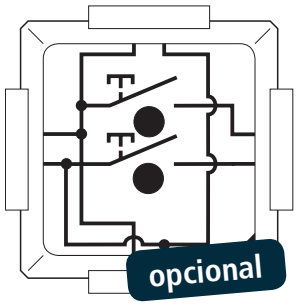
Art.-Nº: 122446
ID del Brick: ALL-BRICK-0383

Cable de 6 vías recto
Línea recta de 6 vías.
La línea recta de 6 vías extiende los cuatro contactos en la parte superior (incluyendo el suelo en los contactos externos) y las cuatro líneas de señal en el nivel inferior que son independientes entre sí.



Art.-Nº: 125674
ID del Brick: ALL-BRICK-0407

Abrazadera de 5 polos tipo 2
Esto permite que los cables o componentes se conecten al circuito. Utilice un destornillador pequeño para presionar la ranura en la parte superior. A continuación, el contacto se abre y el cable se puede insertar en el lateral del mismo. Al soltar el destornillador, el cable queda fijo. El contacto central está conectado a tierra.



Art.-Nº: 137824
ID del Brick: ALL-BRICK-0642

Interruptor doble
Este brick contiene 2 contactos NA unipolares. Esto permite conectar convenientemente las entradas de puertas o flipflops. Además, las vías de señal se conectan por separado de la conexión superior a la inferior. Conecte la fuente de alimentación al contacto superior y/o inferior, ahorrando así numerosos bricks de línea en distintas situaciones.

5. El brick IdC y el Arduino IDE

5.1 El núcleo del set Internet de las Cosas

El brick IdC es el núcleo del set Internet de las Cosas de Brick'R'knowledge. Su componente central es el módulo ESP 12 F de AI THINKER, compuesto por un microcontolador tipo EsP8266 con interfaz WLAN integrada, una memoria flash de 4 MB y una antena WLAN interna.

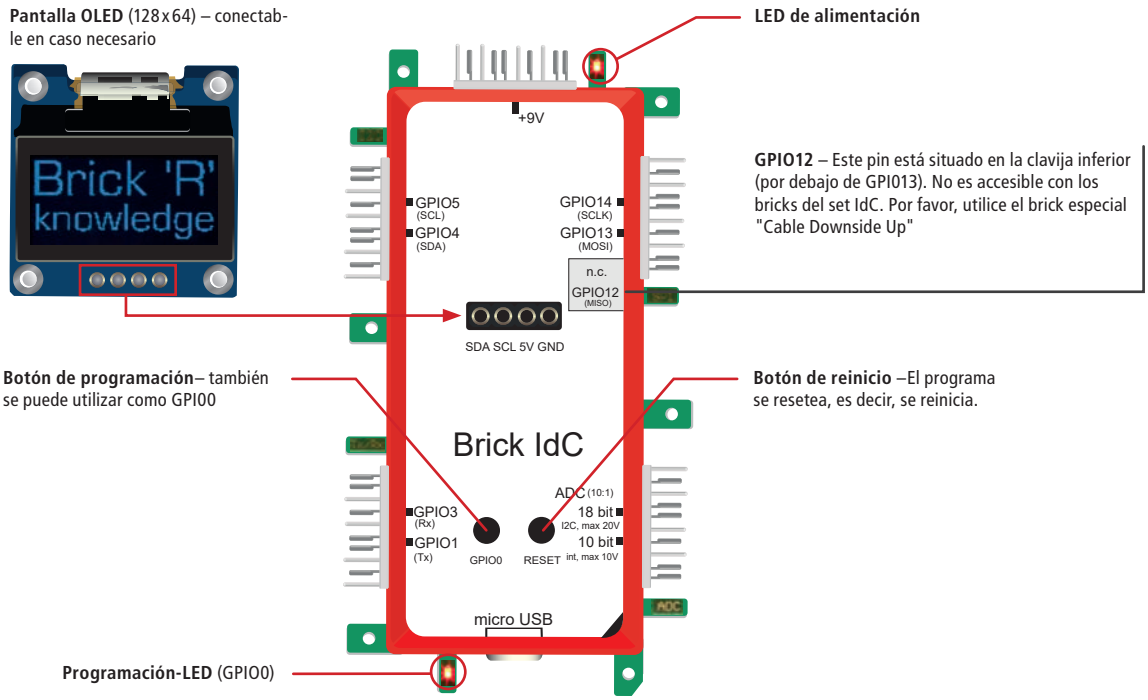


Fig. 4: El núcleo del set Internet de las Cosas

5.1.1 Especificaciones del Brick IdC

Elemento	Especificación
Microcontrolador	80 MHz Tensilica L106 Ultra-Low-Power 32 bit MCU
Memoria Flash	4 MByte para programas
Interfaz WLAN	Protocolo IEEE 802.11 b / g / n, IPv4, dirección IP: asignado a través de DHCP, pila TCP / IP incorporada, banda ISM (2.4 GHz), soporta modos de seguridad WPA / WPA2, antena incorporada
GPIO	7 GPIOs, función estándar: entrada/salida digital, funciones alternativas: I^2C y SPI-bus Nivel de salida GPIO: +5 V; nivel de entrada GPIO: +5 V (las entradas no son tolerantes a 9V!)
Entradas análogas	**ADC 10 bit:** Convertidor interno A / D de 10 bit (tipo: convertidor SAR), frecuencia de muestreo: máx. 200 S / s, rango de tensión de entrada: máx. 10 V **ADC 18 bit:** Convertidor A / D de 18 bit MCP3421 connected vía I2C (tipo: convertidor Delta-Sigma), frecuencia de muestreo: máx. 3.75 S / s, rango de tensión de entrada: máx. 20 V
Botón	2 botones (botón de programación GPI00 y Reset) : necesarios para el modo de programación (después de iniciar el ESP8266, se puede utilizar el botón GPIO0 como entrada digital normal)
Fuente de alimentación	Alimentación 9V (LED de alimentación a 3.3 V en la tensión de la placa)
Tomas de corriente	1 x conector Brick para alimentación de 9 V 4 x conectores Brick para un total de 7 GPIO 1 x conector Brick para 2 entradas analógicas Enchufe micro-USB socket (tipo B) como interfaz de programación
Pantalla	pantalla-I^2C-OLED monocromática (128 x 64 pixels) conexión mediante conector de 4 polos

5.1.2 Los pines GPIO del brick IdC

Las entradas y salidas analógicas y digitales son necesarias para que un microcontrolador pueda intercambiar información. Las entradas y salidas digitales suelen denominarse GPIO. GPIO es la abreviatura de General Purpose Input/Output (Entrada/Salida de Propósito General), lo que significa que tal conexión se puede utilizar como entrada o salida, según la configuración anterior. En programación, este cambio de dirección se denomina pinMode. El nivel de tensión de los GPIO en el brick IdC corresponde a 5V para el nivel alto y 0V para el nivel bajo. Alternativamente, algunos pines GPIO pueden realizar funciones especiales, como la comunicación de información vía bus I²C o SPI. Para acceder al pin GPIO deseado en su programa, necesita un índice que puede encontrar en la siguiente tabla:

Función estándar	Descripción	Función alternativa	Descripción (capítulo. 5.1.3))	Índice para programar
GPIO0	Índice para programar	-	bajo: 40 kΩ, alto: 4 kΩ	0
GPIO1	E/S Digital 1	TxD	bajo: 40 kΩ, alto: 4 kΩ	1
GPIO3	E/S Digital 3	RxD	bajo: 40 kΩ, alto: 4 kΩ	3
GPIO4	E/S Digital 4	I^2C SDA	bajo: 40 kΩ, alto: 4 kΩ	4
GPIO5	E/S Digital 5	I^2C SCL	bajo: 40 kΩ, alto: 4 kΩ	5
GPIO12**	E/S Digital 12	SPI MISO	bajo: 40 kΩ,alto: 4 kΩ	12
GPIO13	E/S Digital 13	SPI MOSI	bajo: 40 kΩ, alto: 4 kΩ	13
GPIO14	E/S Digital 14	SPI SCLK	bajo: 40 kΩ, alto: 4 kΩ	14
ADC0 (10 bit)	Entrada analógica de 10 bits (interna)	-	-	0
ADC1 (18 bit)	Entrada analógica de 18 bits (vía I2C)	-	-	(via I^2C)

Los GPIO 2, 6, 7, 8, 9, 10, 11, 15 y 16 no están suministrados por el módulo ESP-12-F.

Véase el siguiente capítulo. Pin en el nivel de conexión inferior sólo accesible con brick especial opcional (art.-nº: 122448, ID del Brick: ALL-BRICK-0385))

Atención: Nunca aplique directamente 9V a los GPIO (por ejemplo, a través de un interruptor). Aunque necesites 9V para el brick IdC y otros bricks activos, los GPIO se diseñan solamente para el nivel 5V. Un nivel de tensión GPIO superior a 5V puede provocar daños irreversibles en los bricks!

5.1.3 Resistencia Pull-Up

Es importante que las entradas tengan siempre un nivel definido en los circuitos digitales. Mediante la instalación de las denominadas resistencias Pull-up o Pull-down, la entrada se transfiere a nivel alto (5V) o bajo (0V). En nuestro brick IdC utilizamos resistencias pull-up con adaptación automática de impedancia. A nivel bajo el valor es 40 kΩ (menos los flujos de corriente). En el nivel alto el valor es 4 kΩ (para que el nivel alto sea reconocido). Por lo tanto, el nivel de entrada en los GPIOs siempre está claramente definido para que el nivel lógico en un pinMode (GPIOx, INPUT_PULLUP) o pinMode (GPIOx, INPUT_PULLDOWN) no tenga ningún efecto en los pines GPIO, ya que las resistencias de pull-up están siempre activas. Después de haber iniciado el suministro, los GPIO configurados se conectan inmediatamente a un nivel alto como entradas.

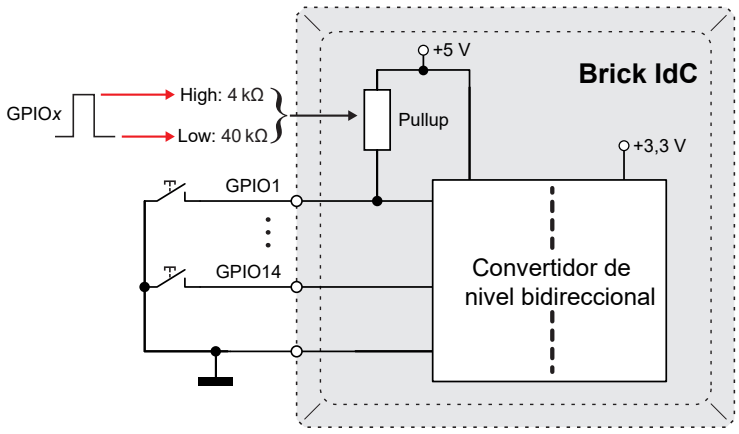


Fig. 5: Resistencia Pull-Up

5.2 Entorno de programación integrado de Arduino

- Para programar el brick IdC utilizamos el entorno de programación integrado de Arduino, también conocido como Arduino IDE. Muchos de vosotros probablemente ya conocéis el software de codificación de Arduino. Hay varios proyectos de Arduino en internet y hay una gran comunidad para ello. En nuestros ejemplos utilizamos una serie de librerías de código abierto para facilitar la programación de componentes de hardware individuales. Hay programas de instalación para Windows, Linux, MAC OS X y una aplicación de Windows. Las descripciones y capturas de pantalla de este manual hacen referencia a la versión de Windows.
- Descarga el instalador para el IDE de Arduino actual, aquí: https://www.arduino.cc
- Comienza la instalación del IDE de Arduino haciendo doble clic en el archivo EXE descargado.
- Para programar el módulo del microcontrolador basado en ESP8266 con la ayuda de este software, debe instalarse el núcleo correspondiente, incluyendo algunas librerías. Esto es necesario porque el IDE de Arduino, por defecto, no soporta el ESP8266.
- Inicie el IDE de Arduino, por ejemplo a través del menú de inicio de Windows (difiere según el sistema operativo).
- Abra el menú "Archivo - Preferencias" e introduzca la siguiente URL debajo de "URL del administrador de placas adicionales": http://arduino.esp8266.co/stab/package_esp8266com_index.json

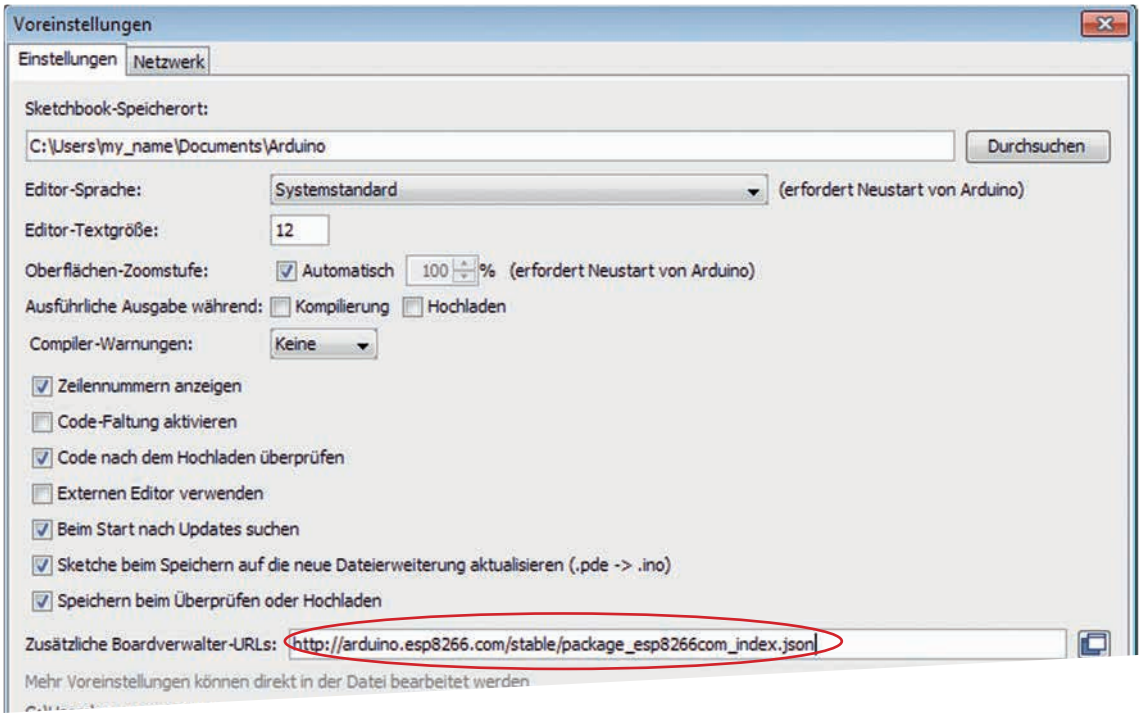


Fig. 6: Panel de control

- Confirmar con OK.
- Abra la "Placa" en el menú "Herramientas"

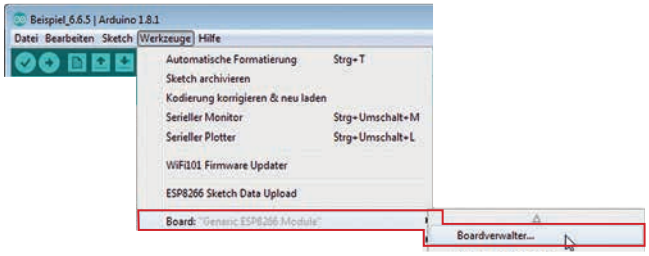


Fig. 7: Panel de control

- Introduzca esp8266 en la ventana de búsqueda. Sólo debería ver: "esp8266 by ESP8266 Community"

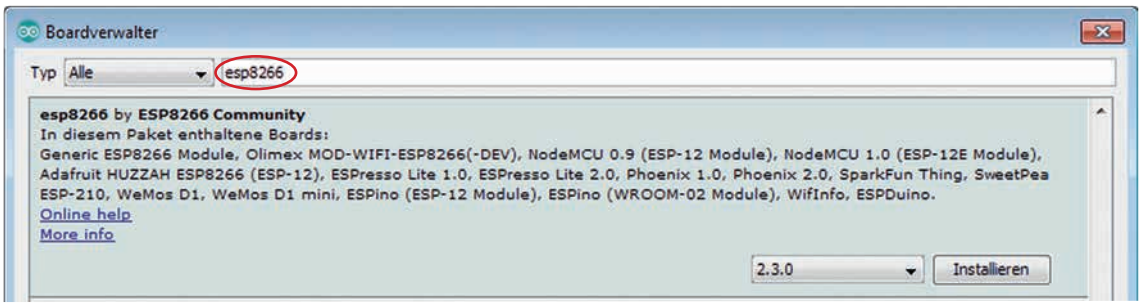


Fig. 8: Instalación del paquete de la placa ESP8266

- Haga clic en la entrada y, a continuación, haga clic en el botón "instalar". La instalación puede llevar algún tiempo.
- Finalizar la instalación con el botón "cerrar".
- En el menú, seleccione "Herramientas - Placa:...-"administrador de placas...".
- Seleccione la entrada "Módulo genérico ESP8266". Una vez seleccionado, el menú debajo de "placa: ..." cambia y muestra varios ajustes. Cámbielos de manera que correspondan con la captura de pantalla (triángulo rojo).

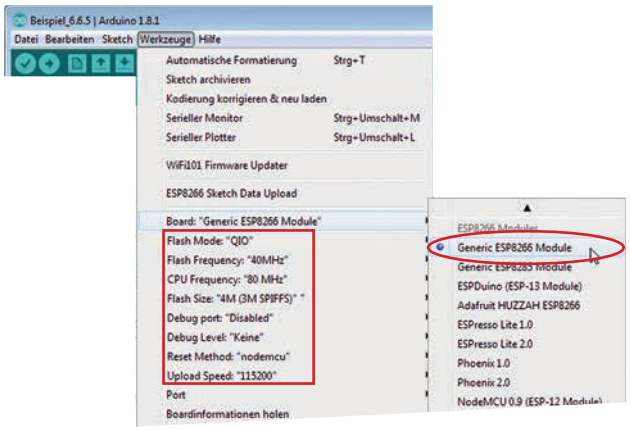


Fig. 9: Instalación del paquete de la placa ESP8266

- Continúe con el siguiente capítulo "Instalar bibliotecas...

5.2.1 Instalación de librerías

Utilizamos las llamadas librerías en nuestros programas de ejemplo para simplificar la codificación. Estas colecciones de codificación incluyen, por ejemplo, extensas tablas que pueden definir fuentes o que codifican visualizaciones de 7 segmentos. Aunque la mayoría de las librerías se entregan con el IDE de Arduino, siguen teniendo que ser instaladas. Otras librerías necesitan ser descargadas de Internet y normalmente están integradas como archivos ZIP. Una vez que haya instalado todas las librerías listadas, estará preparado y no tendrá que instalar nada más para los ejercicios. Encontrará las librerías instaladas en la siguiente lista de su PC: C: Usuarios\Mi_nombre\Documentos\Arduino (sustituya mi_nombre por su nombre de usuario).

Puedes encontrar consejos para instalar librerías adicionales de Arduino aquí: https://www.arduino.cc/en/Guide/Libraries

5.2.1.1 Instalando librerías de Arduino

- Instalando librerías de Arduino

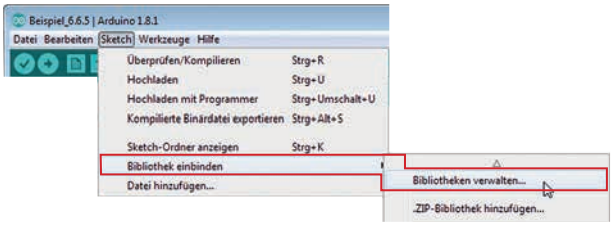


Fig. 10: Administrador de librerías

- Instale las bibliotecas necesarias acorde con los capítulos siguientes. Limite la elección utilizando términos de búsqueda adecuados (consulte las siguientes capturas de pantalla). En caso de que se ofrezca una selección de versión, puede instalar la última versión. Haga clic en la entrada y, a continuación, haga clic en "instalar".

5.2.1.1.1 Librería "NTPClient"

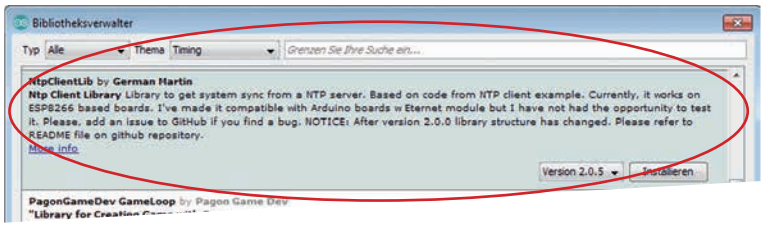


Fig 11: Instalación de la librería "NTPClientLib"

- Elija la librería con la última versión y haga clic en "instalar".
- La librería es necesaria para obtener la hora y la fecha mediante el Network Time Protocol (NTP) de Internet.

5.2.1.1.2 Librería "Tiempo"

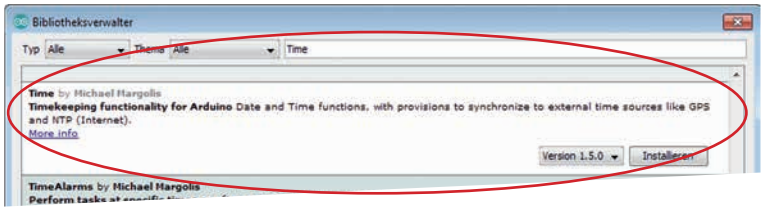


Fig. 12: Instalación librería"Tiempo"

- Elija la librería con la última versión y haga clic en "instalar".
- El archivo de cabecera está integrado con la opción #include <Time.h> y #include <TimeLib.h>.

5.2.1.1.3 Librería "Json Streaming Parser"

La librería es necesaria para obtener la hora y la fecha mediante el Network Time Protocol (NTP) de Internet.

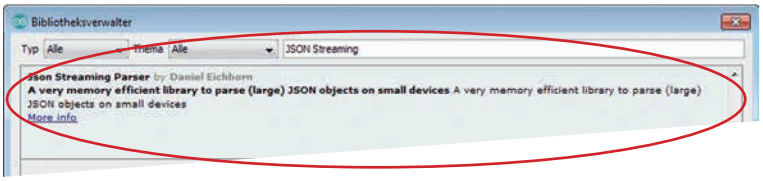


Fig.13 : Instalación librería"Json Streaming Parser"

- <TimeLib.h>.
- Los archivos de cabecera están integrados con las instrucciones #include <DHT.h> y #include <DHT_U.h.

5.2.1.1.4 Librería "librería de sensores DHT"

La biblioteca es necesaria para leer la temperatura y la humedad del sensor DHT11.

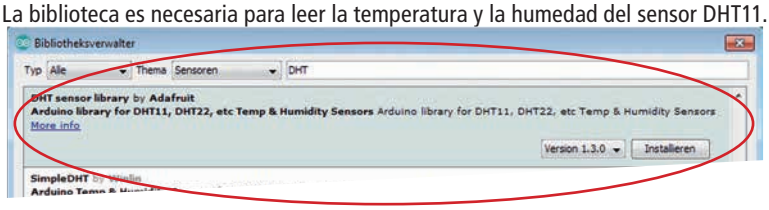


Fig. 14: Instalación de la librería " Librería de sensores DHT ".

- Seleccione la librería con la última versión y haga clic en "Instalar".
- Los archivos de cabecera están integrados con las instrucciones #include <DHT.h> y #include <DHT_U.h

5.2.1.1.5 Librería "Adafruit Unified Sensor"

La biblioteca es necesaria para leer la temperatura y la humedad del sensor DHT11.

Fig.15: Instalación de la librería "Adafruit Unified Sensor"

- Seleccione la librería con la última versión y haga clic en "Instalar".
- El archivo de cabecera está integrado con la instrucción: #include <Adafruit_Sensor.h>.

5.2.1.2 Instalación de librerías externas

- El mismo procedimiento se aplica a cada librería.
- Necesita descargar la librería necesaria de Internet. Encontrará las respectivas páginas de descarga en los siguientes capítulos.
- Abra el menú "Programa - Incluir librería - Añadir librería ZIP...".
- Elija el archivo ZIP descargado.

5.2.1.2.1 Librería "esp8266-oled-ssd1306-master"

Para simplificar el control de la pantalla OLED, utilizamos la librería "esp8266-oled-ssd1306-master". Puedes descargarlo desde la página de GitHub como un archivo ZIP.

Visite: https://github.com/squix78/esp8266-oled-ssd1306. En la opción "Clonar o descargar" seleccione la opción "Descargar Z"

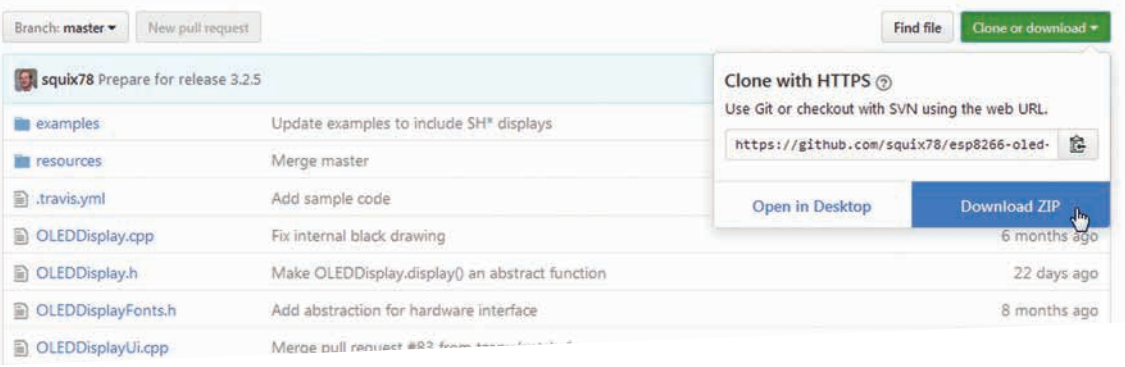


Fig. 16: Instalación librería "esp8266-oled-ssd1306-master"

- Instale la librería a través del menú "Programa - incluir librería - añadir librería ZIP".
- El archivo de cabecera está integrado con la instrucción: #include <SSD1306Wire.h>.

5.2.1.2.2 Librería "MCP3421"

Para simplificar la comunicación con el convertidor A/D de 18 bits MCP3421, utilizamos la librería "MCP3421". Puede ser descargada como un archivo ZIP.

- Haga click en el enlace: http://interface.khm.de/index.php/lab-log/connect-a-mcp3421-18-bit-analog-to-digital-converter-to-an-arduino-board/. En el apartado "Descarga de la librería" al final de esta página web encontrará el enlace a la librería necesaria.
- Descargue el archivo MCP3421.zip y guárdelo en su PC.
- Instalar la librería a través del menú "Programa- Incluir librería - añadir librería ZIP".
- El archivo de cabecera está integrado con la instrucción: #include <MCP3421.h>>.

5.2.1.2.3 Librería "BrickESP8266"

La librería es necesaria, entre otras cosas, para recuperar el cambio actual del dólar de los EE.UU. de Internet. Se puede descargar como archivo ZIP.

- Haga click en el enlace: http://www.brickrknowledge.de/downloads.
- Descargue el archivo BrickESP8266.zip y guárdelo en su PC.
- Instalar la librería a través del menú "Programa- Incluir librería - añadir librería ZIP".
- El archivo de cabecera está integrado con la instrucción: : #include <CurrencylayerClient.h>.

En caso de que falte una librería, se indicará mediante un mensaje de error durante la compilación.

En caso de que no haya instalado el controlador para el USB-to-UART-bridge, por favor continúe con la sección 5.2.2.

5.2.2 Controlador de puerto virtual COM

Para que tu ordenador se comunique con el brick IdC, necesitas seleccionar la interfaz de tu PC en el IDE de Arduino para establecer la conexión con el brick. Para esto, puede seleccionar un puerto en serie (también llamado puerto COM) en el IDE de Arduino. Normalmente, se trata de puertos RS-232, pero en los ordenadores modernos apenas se encuentran. En su lugar, usamos un puerto USB libre de tu ordenador y hacemos que el IDE de Arduino crea que es un puerto en serie. Tenemos que instalar un puerto virtual COM para dejar esto claro al sistema operativo. Este es un requisito para la comunicación entre Arduino IDE y el puerto USB del brick IdC.

- Descargue el controlador actual para su sistema operativo (Windows, MAC OS X, Linux) de Silicon Labs
- https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx

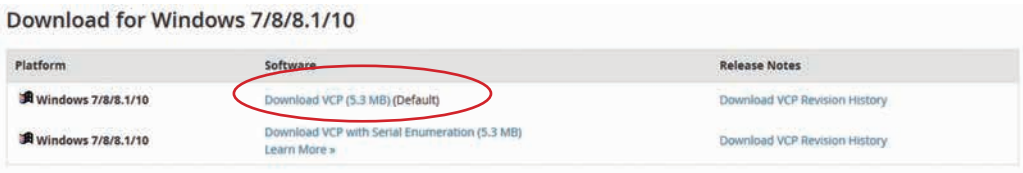


Fig. 17: Captura de pantalla de la página de descarga incluyendo un enlace a las versiones actuales de Windows

- Desempaquetar el archivo empaquetado en su PC.
- Inicie la instalación haciendo doble clic. Según la versión de Windows, debe iniciar el instalador correspondiente: CP210xVCPInstaller_x86.exe para una versión de Windows de 32 bits o CP210xVCPInstaller_x64.exe para una versión de 64 bits.
- En el administrador de dispositivos de Windows encontrará "Conectores (COM & LPT)".
- En el administrador de dispositivos de Windows encontrará "Conectores (COM & LPT)". El número de puertos COM y el índice dependen de la configuración de su PC.

Fig. 18: Puerto COM virtual en el administrador de dispositivos

5.2.3 Monitor serie

Algunos programas de ejemplo utilizan el llamado monitor serie para mostrar directamente los valores y notificaciones en el ordenador. Haz clic en el icono de la lupa en el IDE de Arduino para abrir el monitor serie. La velocidad en baudios en la ventana del monitor y en el sketch deben ser consistentes (ver Fig. 19).

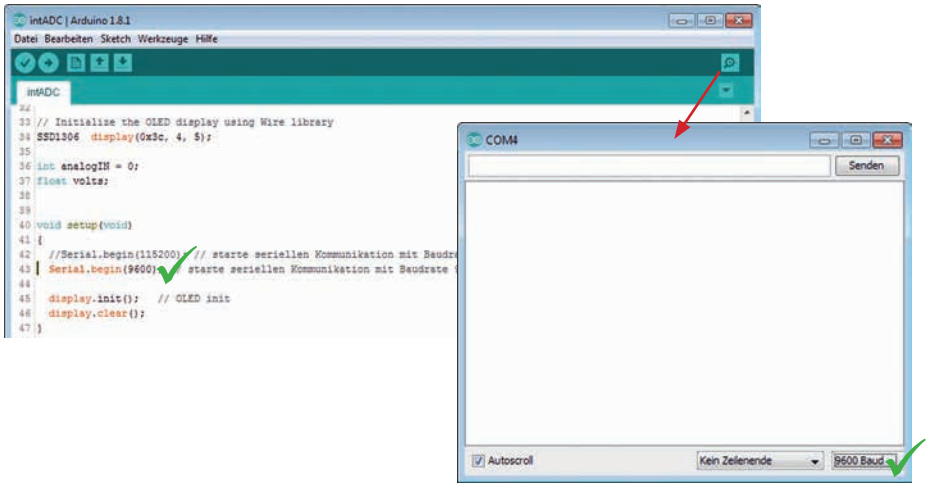


Fig. 19: Emisión de información mediante un monitor serie

5.3 Primeros pasos

Ahora se está poniendo interesante! Después de haber instalado el IDE de Arduino incluyendo las librerías y el controlador del puerto virtual COM, empezarás a operar.

5.3.1 Establecer conexión

Conecte el brick IdC con la fuente de alimentación de 9V incluida en el puerto superior (ver Fig. 20).

Precaución: ¡Nunca conecte el suministro de 9V a los contactos del otro brick IdC ya que esto podría destruir el brick!

Conecte su ordenador y el brick IdC (puerto Micro USB) con el cable USB incluido.

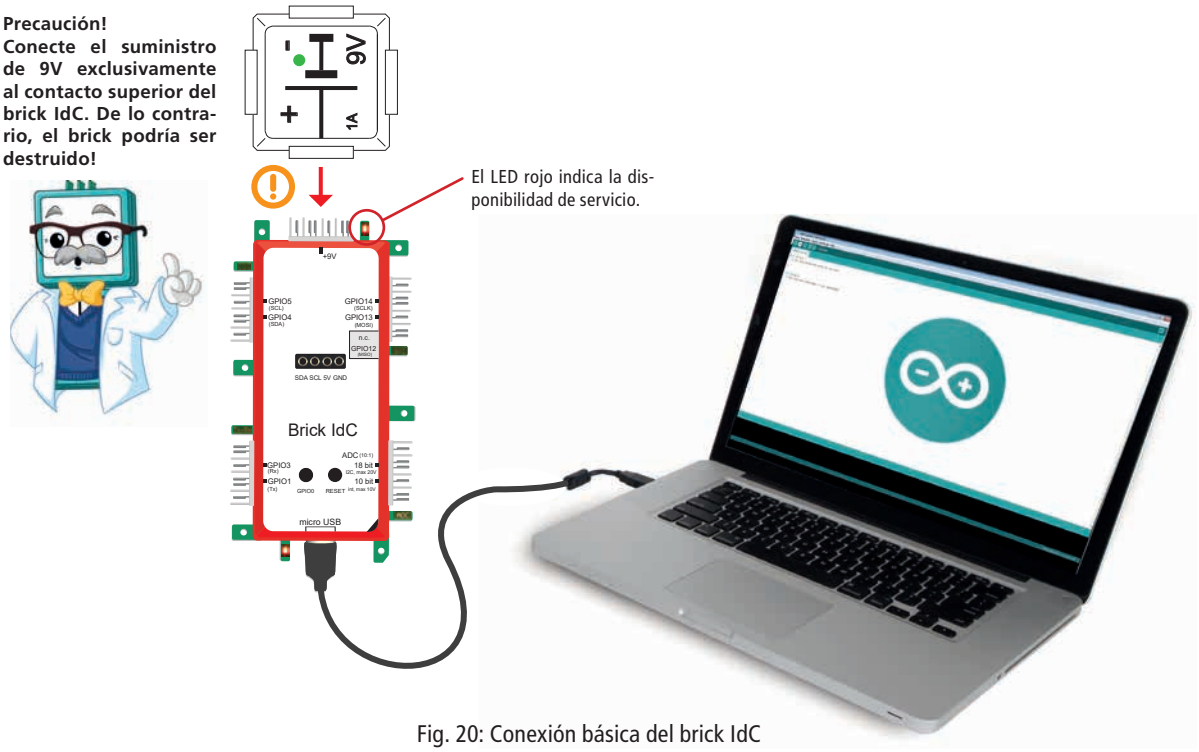


Fig. 20: Conexión básica del brick IdC

El ordenador debería tener ahora un nuevo puerto virtual COM a través del cual el IDE de Arduino cargue los programas en el brick IdC. Puedes calcular fácilmente el índice del puerto COM a través del IDE de Arduino.

- En primer lugar, desconecte el cable USB del brick IdC.
- Inicie el software de Arduino y examine qué puertos COM se muestran en " herramientas - puerto:...".
- Conecte el cable USB con el brick IdC de nuevo. En la sección " herramientas - puerto:..." debería ver un puerto COM adicional. Seleccione este puerto. Si esto no fuera posible, desinstale el puerto COM en el administrador de dispositivos y vuelva a instalar el controlador "CP210x USB to UART Bridge". Compare la sección 5.2.2. en la página 20.

Fig. 21: Izquierda: cable USB no conectado, derecha: cable USB conectado (puerto COM virtual seleccionado)

Tiene que seleccionar o examinar más parámetros en el menú "Herramientas". Es necesario seleccionar la configuración que se muestra en la siguiente captura de pantalla. En la sección " puerto " hay que seleccionar el puerto COM que ha sido asignado anteriormente al puerto COM virtual.

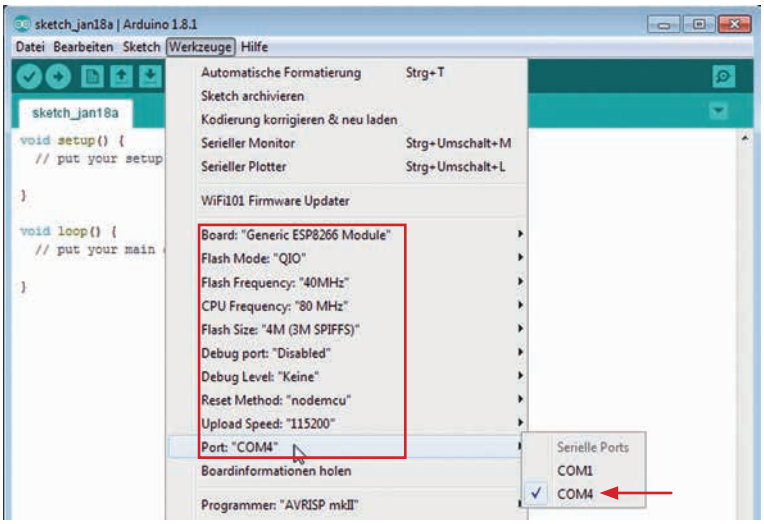


Fig. 22 : Ajustes para el brick IdC

5.3.2 Compilación y carga del código del programa

Haciendo clic en el botón con la marca de verificación, el código del programa - que en el mundo Arduino es conocido como Sketch - se examina en busca de errores de sintaxis y se compila. Los errores de sintaxis son errores tipográficos en su Sketch o violaciones de las reglas del lenguaje de programación. Durante la compilación, el código de su programa se traduce a código de máquina que nuestro microcontrolador pueda realizar. Esto se realiza automáticamente a través de un compilador (traductor) que está integrado en el IDE de Arduino.

Después de haber compilado con éxito su código de programa, puede cargar el programa a su brick IdC haciendo clic en el icono de la flecha. El LED rojo en la parte inferior izquierda cerca del puerto USB en el brick de IdC, se encenderá mientras el programa se está cargando en la memoria flash. El inicio y el final del proceso de carga se indican con un breve parpadeo. Una vez finalizada la carga, el LED se apagará.

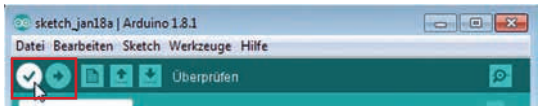


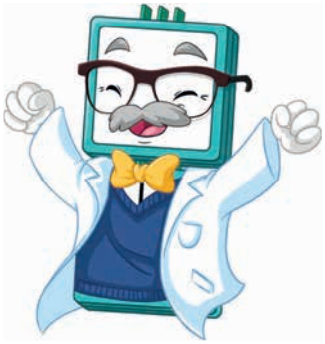
Fig. 23: Examinar el sketch, compilar y cargar

En el siguiente capítulo aprenderá a activar el modo de programación manualmente.

5.3.3 Modo de programación

Utilice esta práctica en caso de que desee activar el modo de programación manualmente. Esto sólo es necesario si la comunicación entre el IDE de Arduino y el brick IdC no funciona.

1. Mantenga presionada la tecla de programación (GPIO0) (el LED rojo de la parte inferior izquierda se ilumina).
2. Presione la tecla de reinicio al mismo tiempo.
3. Deje de presionar la tecla de reinicio.
4. Deje de pulsar la tecla de programación (el LED rojo de la parte inferior izquierda está muy poco iluminado).



6. Ejemplos

6.1 "Hola Mundo" (LED parpadeante)

Abra el sketch en el IDE de Arduino: Example_6.1.ino

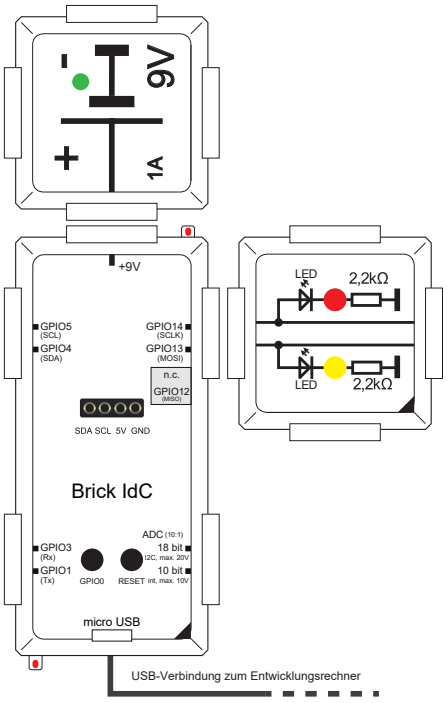
Como es habitual en el mundo de la codificación, empieza con un ejemplo de "Hello World". En este caso, tendremos dos LED parpadeando alternativamente. Un programa - también llamado sketch en el mundo Arduino - consta de al menos dos partes.

El Setup-Routine void setup()

Primero, está la parte void setup() {...}. Al iniciar el programa, todos los comandos dentro de esta llave se ejecutan exactamente una vez. En nuestro ejemplo, se define la dirección de los pines GIPI, es decir (modo: entrada) o, como en nuestro ejemplo: dos salidas (modo: salida)

Bucle de programa `void loop()`

El programa actual se incluye en la abrazadera de void loop() {...} y se repite permanentemente. Los comandos se llevan a cabo en un bucle infinito hasta que el programa se detiene pulsando el botón de reinicio, por ejemplo. En nuestro primer ejemplo de codificación (Beispiel_6.1.ino), el brick LED doble está conectado a los GPIO 13 y 14 del brick IdC.




```
void setup() {
  pinMode(14, OUTPUT);     // Pin 14 es salida
  pinMode(13, OUTPUT);     // Pin 13 es salida
}

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

  delay(1000);            // esperar 1000ms

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

  delay(1000);            // esperar 1000ms
}
```

Fig. 24: Ejemplo: "Hola Mundo" (LED parpadeante)

Los GPIO 13 y 14 se definen como salida en la setup routine con pinMode (GPIOx, OUTPUT)

Por lo tanto, estos pines son capaces de emitir un nivel high para que ambos LED parpadeen alternativamente. En el bucle, el pin con digital Write (GPIOx, HIGH) se ajusta a nivel high.

Esto significa que cada GPIO emite 5V, por lo que el LED se enciende. Con el comando digital Write (GPIOx, LOW) el pin se ajusta a 0V, el LED se apaga.

El programa se detiene durante el número de milisegundos correspondiente con el comando delay(....).

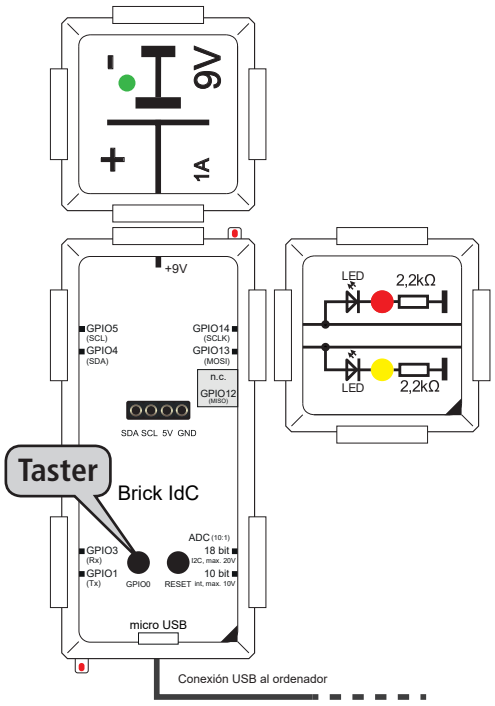
En este ejemplo son 1000 milisegundos, lo que corresponde a un segundo. Después de haber cargado el ejemplo de codificación a su brick IdC (vea el capítulo 5.3.2 en la página 22), ambos LED parpadean alternativamente. Juega un poco con el código. Cambie el orden en que se encienden y apagan los LED y cambie los intervalos de tiempo entre ellos.

6.2 Botón y LED

Abra el siguiente sketch en el IDE de Arduino

En el siguiente ejemplo, queremos que los LED se iluminen en función de un botón. Además del botón de reset, el brick IdC incluye otro botón que está conectado al GPI0. Por lo tanto, usamos el GPIO0 como entrada y leemos el estado del botón. Para evitar un nivel indefinido en esta entrada mientras no se pulsa el botón, se ha incluido una resistencia pull-up interna que mueve la entrada a un nivel high (véase el capítulo 5.1.3 en la página 14). En cuanto se pulsa el botón, hay 0V en la entrada. De esta manera, puede saber claramente si el botón está oprimido o no.

No te confundas, el LED de codificación en la parte inferior izquierda se ilumina mientras presionas el botón. El LED y el botón están conectados y no tienen nada que ver con tu código.




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

  pinMode(14, OUTPUT);    // Pin 14 es Salida
  pinMode(13, OUTPUT);    // Pin 13 es Salida
}

void loop() {
  if (digitalRead(0)==LOW)
  {              // al pulsar el botón

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

  }
  else{         // si el botón no está pulsado
    digitalWrite(14,LOW);      // LED an Pin 14 aus
    digitalWrite(13,HIGH);     // LED an Pin 13 an
  }

  delay(100);           // espera 100ms
}
```

Fig. 25: Ejemplo: Botón y LED

Mientras el botón no esté presionado, el LED amarillo se ilumina en el GPIO13. En cuanto aprietas el botón, el LED rojo se ilumina en el GPIO14.

El comando delay(100) al final del bucle es responsable de que el programa espere 100 milisegundos en este punto. El microcontrolador puede " descansar " durante este tiempo, ¡ahora es su turno de nuevo! Varíe un poco el código para entenderlo mejor. Por ejemplo, haga que los LED se iluminen en secuencia en cuanto pulse el botón.

Precaución: No utilice un botón externo que esté directamente conectado a 9V ya que los GPIO sólo están hechos para niveles de 5V. Aconsejamos utilizar el botón integrado GPIO0.

A un nivel de tensión de más de 5V en los GPIO, el brick IdC puede sufrir daños irreversibles!

6.3 Bus I²C

El bus I²C es una interfaz serial que funciona con dos líneas, la señal de reloj (SCL, Serial Clock) y la línea de datos SDA (Serial Data). Las líneas trabajan bidireccionalmente. Diferenciamos entre master (maestro) y slave (esclavo) entre los dispositivos de red. En este caso, el brick IdC es el maestro y los otros bricks son los esclavos. Los dispositivos de red se direccionan a través de direcciones I²C. Se pueden tener hasta 128 dispositivos conectados al mismo bus. Los dispositivos de red individuales pueden cubrir varias direcciones. Algunos dispositivos de red tienen pocos conmutadores DIP en la parte posterior, lo que permite cambiar el rango de direcciones cuando se utilizan varios dispositivos de red en el mismo bus. En general, el bus se puede utilizar con diferentes velocidades:

Mode	Velocidad
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

Muchos microcontroladores son capaces de controlar sólo los dos primeros modos (por ejemplo, el controlador del Arduino Nano), algunos de ellos también controlan el tercer modo. Lo mismo ocurre con los dispositivos complementarios. Naturalmente, los modos deben encajar. El maestro, es decir, el microcontrolador, marca el ritmo en la línea SCL. La transmisión de datos actual se realiza a través de la línea SDA

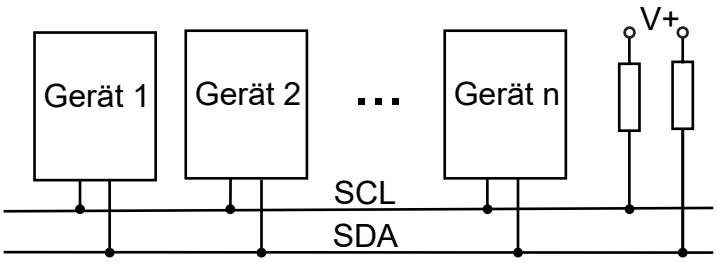


Fig. 26 : Estructura del Bus I²C

Puede conectar un máximo de 128 dispositivos a un bus I²C siempre y cuando cada dispositivo ocupe una sola dirección, de lo contrario, menos. Los dispositivos se conectan a través de dos líneas de bus. Ambas resistencias pull-up (en la unidad kΩ) ya están integradas en el brick IdC.

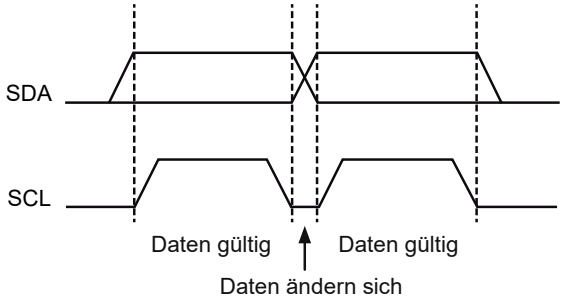


Fig. 27 : Datos válidos en el bus I²C

El reloj indica cuando hay datos válidos recibidos. Como puede ver en la fig. 27, este es siempre el caso en el nivel high de la línea SCL. El receptor puede ahora muestrear y evaluar los datos. El maestro establece el ritmo, ya sea creando los datos por sí mismo o esperando estos datos del dispositivo en cuestión.

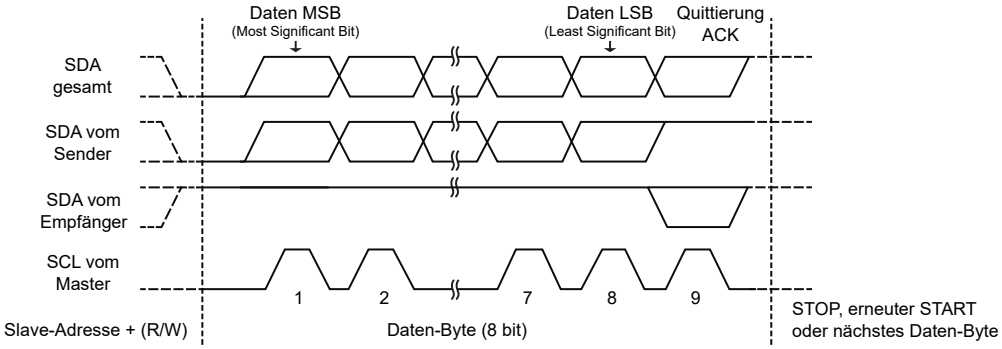


Fig. 28: Transmisión de datos en el bus I²C

En la Fig. 28 se puede ver el curso temporal de una transferencia de datos con las siguientes señales (de abajo hacia arriba): la señal de reloj SCL (indicada por el maestro), la línea de datos desde la perspectiva del receptor (receptor) no se activa directamente ya que es de baja actividad. Los bits de los datos, tal y como fueron enviados por el transmisor (baja actividad) y en la parte superior, la señal SDA en la visión general. La sincronización es importante. Un receptor (ya sea maestro o esclavo) envía una señal de acuse de recibo (ACK = acknowledge) al final de cada paquete de datos, moviendo la línea SDA a un nivel low. Como esto es equivalente a un Wired-OR, es suficiente que un esclavo envíe la señal ACK.

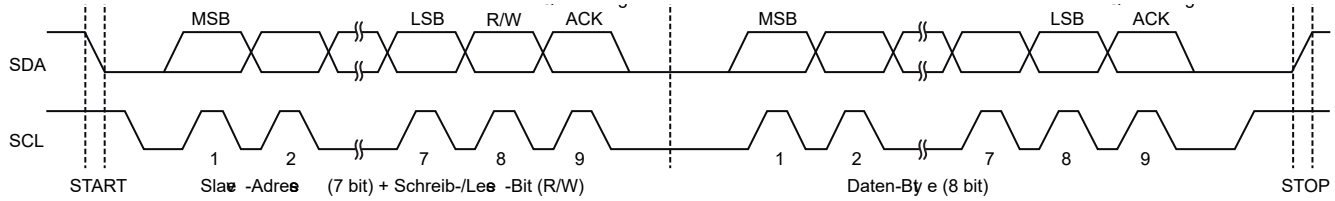


Fig. 29: Ciclo de transmisión en el bus I²C

En la Fig. 29 se puede ver el ciclo completo de transmisión. En primer lugar, se envía un paquete que incluye la dirección. La dirección consta de 7 bits, incluyendo uno adicional, el llamado bit R/W (read/write). Todos los dispositivos de red comparan la dirección emitida con la suya propia. Cuando se corresponden, el respectivo esclavo lo reconoce con una señal ACK ajustando brevemente la línea SDA a un nivel low. Dependiendo del bit R/W, el esclavo direccionado sabe si debe iniciar un ciclo de envío o de recepción. A continuación se puede iniciar la transferencia de datos. Al final, se inicia un ciclo de parada. Para ello, el reloj se pone a un nivel high y se libera la línea SDA. Las dos líneas, SDA y SCL, son de nivel high, lo que significa que el bus I²C puede utilizarse libremente. Es posible que un maestro diferente inicie un nuevo ciclo (en caso de que haya más de uno en el bus).

El bus I²C es fácil de usar para nosotros ya que la librería Arduino proporciona varios comandos para controlar los bricks con interfaz I²C (ej. Brick indicador de 7 segmentos) a través del brick IdC.

6.3.1 El indicador de 7 segmentos

En los comienzos de la tecnología informática había una cierta preocupación por cómo mostrar los números. La manera más fácil de hacerlo fue con 10 luces que fueron etiquetadas de 0 a 9. Después, utilizaron las luces para iluminar pequeñas placas de vidrio con sus respectivos agujeros. Al mismo tiempo, se introdujeron los llamados "tubos Nixi", los números estaban hechos de alambre y comenzaron a iluminarse cuando se aplicaba un voltaje más alto en una atmósfera de gas protectora. Más tarde se les ocurrió la idea de dividir los números en segmentos. Se pueden mostrar todos los números entre 0 y 9 con siete segmentos. Las primeras pantallas todavía utilizaban hilo incandescente, pero se hizo más fácil cuando se inventaron los LED. Detrás de cada segmento hay un LED que ilumina las barras. Los segmentos individuales se indican frecuentemente con "a hasta g". Además, hay sólo un LED para el punto decimal (dp).

Fig. 30 : El indicador de 7 segmentos

De esta manera, puede mostrar fácilmente los números del 0 al 9. Conoceremos una forma más elegante de visualizar la pantalla OLED (vea el capítulo 6.4 en la página 32). De este modo también se pueden representar gráficos simples.

Hay dos indicadores de 7 segmentos incluidos en nuestro brick indicador de 7 segmentos que están conectados a través del bus I²C. La pantalla de la izquierda se denomina signo MSB (Most Significant Bit), mientras que la de la derecha se denomina signo LSB (Least Significant Bit) - véase el capítulo 6.5.2.5 en la página 38. Los dos indicadores de 7 segmentos se controlan a través de un extensor E/S de tipo 8574T. Este componente se encarga de decodificar la dirección en el bus I²C y los bytes de los datos (aquí: números) para el control de los siete segmentos, incluida la etapa del LED controlador.

6.3.2 Indicador de 7 segmentos como brick I²C - Estructura y direcciones

En el IDE de Arduino, abra el sketch: Beispiel_6.3.2.ino. Necesitas introducir el siguiente archivo de cabecera: Wire.h. En el caso de no haber instalado las librerías necesarias aún, véase el capítulo 5.2.1 en la página 16 y siga las instrucciones.

Realizaremos un circuito simple con el indicador de 7 segmentos. Puede ajustar la dirección I²C del brick en el reverso mediante dos interruptores (posibles direcciones hexadecimales: 40(16), 44(16), 48(16), 4C(16).

Por lo tanto, puede utilizar un máximo de cuatro de estos bricks en un bus I²C.

Atención: Normalmente, la dirección 40(16) (ver Fig. 31) está preajustada, pero es posible que sea necesario controlar y ajustar los ajustes (ver esquema Beispiel_6.3.2.ino). Los interruptores se encuentran en el reverso de la placa y son accesibles a través de un orificio en la parte inferior. Si usted es experto, puede ajustar los pequeños interruptores deslizantes con un palillo de dientes, por ejemplo.

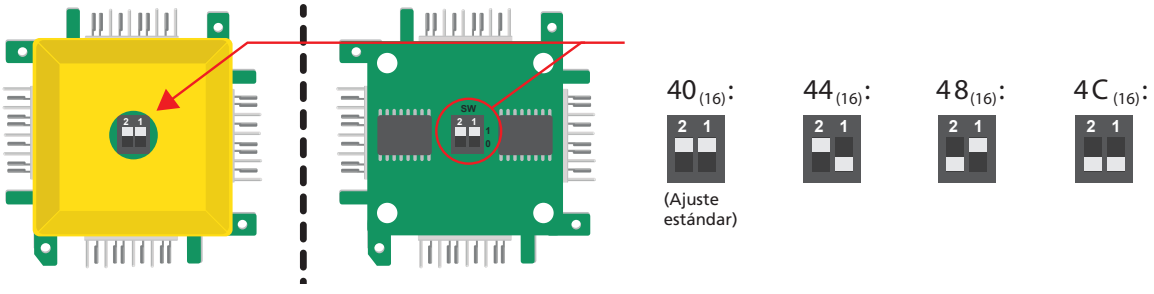


Fig. 31: Ajuste de direcciones I²C del brick de 7 segmentos

Atención: Todos los usuarios del bus I²C deben utilizar una dirección diferente. De lo contrario, pueden producirse fallos de funcionamiento!

Hemos preparado una pequeña librería para el IDE de Arduino en la que todos los segmentos están codificados. Esto se realiza a través de una llamada tabla de caracteres. Con un byte, la combinación de segmentos de una tabla se asigna a los números e incluso a las letras de la A a la Z. Para facilitarlo, hemos preparado varios subprogramas.

La función display_seg1x() muestra un segmento individual. Para esto, transfiera la dirección I²C del correspondiente componente controlador. La función get_7seg () convierte el código ASCII en el índice de la tabla incluyendo las asignaciones de segmentos. Puede controlar directamente los segmentos con display_seg1x-bin (). Hay dos de estos componentes controladores para ambos números que se dirigen con una dirección ascendente y con una separación de dos números. El sketch utiliza normalmente la dirección I²C: 40(16) para el dígito inferior (llamado "signo LSB" en el sketch), y 42(16) para el dígito superior (también llamado "signo MSB" en el sketch).

Por favor, eche un vistazo y experimente con nuestro ejemplo de codificación.

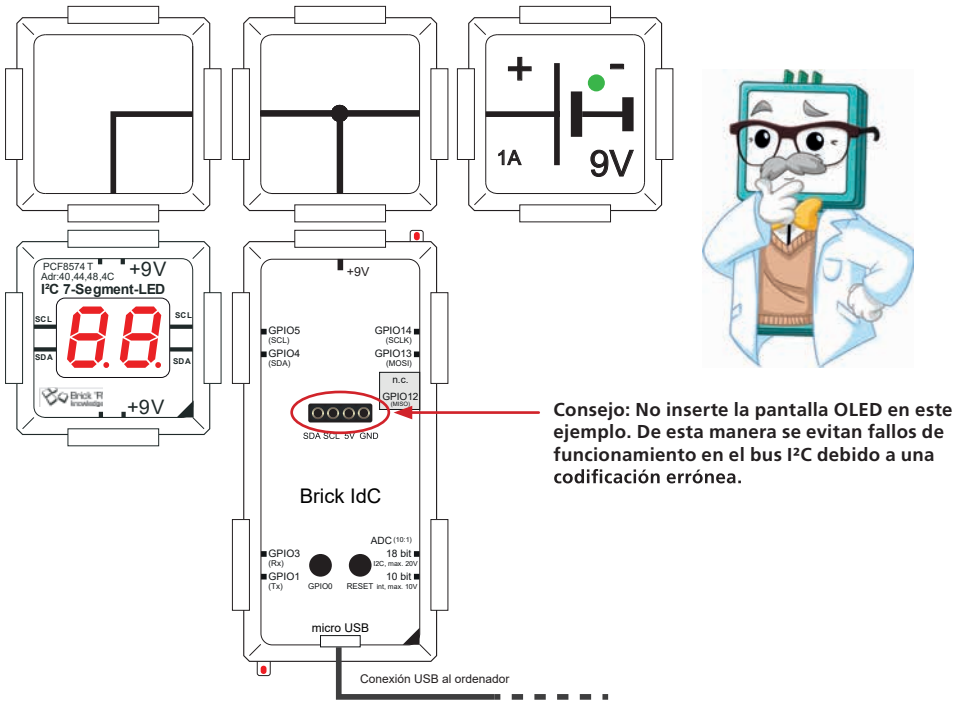


Fig. 32: Ejemplo: muestra la dirección I²C del brick de 7 segmentos

Detalles del programa

```
void loop() {
// Pantalla de 7 segmentos con módulo conductor 8574T
// Exportar todas las direcciones potenciales para la identificación
// De esta manera puede ver qué dirección está configurada
   display_seg1x(i2cseg7x2amsb1,'4'); // dirección propia
   display_seg1x(i2cseg7x2alsb1,'0'); // exportar
   display_seg1x(i2cseg7x2bmsb1,'4'); // siempre son pares
   display_seg1x(i2cseg7x2blsb1,'4'); // dos comandos por brick
   display_seg1x(i2cseg7x2cmsb1,'4');
   display_seg1x(i2cseg7x2clsb1,'8'); // de 0x40 a 0x4C
   display_seg1x(i2cseg7x2dmsb1,'4');
   display_seg1x(i2cseg7x2dlsb1,'C');
}
```

¿Qué está pasando?

El programa de muestra (ver esquema Beispiel_6.3.2.ino) muestra la dirección I²C en la pantalla enviando la dirección como bytes de datos a la dirección correspondiente. Esto se repite con todas las direcciones útiles (40(16), 44(16), 48(16), 4C(16)). En cuanto se envía la dirección correcta, el brick se siente dirigido y muestra la dirección correspondiente. De esta manera, puede encontrar y anotar fácilmente el valor.

6.3.3 Display de 7 segmentos como contador

 Abra el sketch Beispiel_6.3.3.ino en el IDE de Arduino. Incluya el siguiente archivo de cabecera: Wire.h. Si aún no ha instalado las librerías correspondientes, consulte el capítulo 5.2.1 en la página 16 e instálelo.

En este ejemplo queremos realizar un contador simple. Puede utilizar el mismo circuito que en la Fig. 32. El valor del contador se almacena en el contador de variables. El programa aumenta el valor en uno cada 500ms. En una pantalla de dos dígitos de 7 segmentos, sin embargo, se puede mostrar un máximo de 99. En una petición If, el valor del contador solicitado es superior a 99. Si este es el caso, la pantalla volverá a ajustarse a 00. La variable de conteo cuenta de 0 a 99 y luego reinicia de 0.

Extracto del programa

```
void loop() { // En el bucle

  char buffer[10];             // Utilizar búferes de caracteres de tamaño definido

  static int counter = 0;     // Fijar variable de contador a 0

  sprintf(buffer,"%02d",counter++); // Convertir un número entero en un signo

  if (counter >99) counter = 0;  // El contador debe contar entre 0 y 99

     // Contador de salida de dos dígitos, por lo que el buffer es 0 y 1
     display _ seg1x(i2cseg7x2alsb1,buffer[1]); //signos LSB en la dirección 0x40
     display _ seg1x(i2cseg7x2amsb1,buffer[0]); //signos MSB en la dirección 0x42

     delay(500); //  incremento aprox. cada 500ms.

} // Fin del bucle
```

El sketch utiliza normalmente la dirección I²C: 40(16) para el dígito inferior (llamado "signo LSB" en el sketch), y 42(16) para el dígito superior (también llamado "signo MSB" en el sketch).

Tenga en cuenta que el bucle tardará más de 500 ms, ya que el retraso causado por el comando delay(500) debe añadirse estrictamente a los tiempos de ejecución de los otros comandos. Si desea trabajar con mayor precisión, debe utilizar y solicitar un temporizador.

6.3.4 Display de 7 segmentos con botón de desbloqueo

Abra el sketch Beispiel_6.3.4.ino en el IDE de Arduino. Debe incluir el siguiente archivo de cabecera: Wire.h. Si aún no ha instalado las bibliotecas necesarias, consulte el capítulo 5.2.1 en la página 16.

Los botones e interruptores tienen la desventaja de que cuando se accionan, el contacto mecánico (a menudo un muelle) causa múltiples cierres o aperturas. En la tecnología digital, este efecto disruptivo se denomina "rebote" . Este problema puede ser resuelto con un simple flip-flop RS (aprender más sobre este tema con el Set Lógico de Brick'R'knowledge). En esta tarea, alternativamente, podrá familiarizarse con la eliminación de rebotes de los botones a través del software. La idea principal es incluir un periodo de espera en el software que dure por lo menos lo mismo que un ciclo de rebote.

Volvemos a usar nuestro contador, que ya conoces del ejemplo 6.3.3.

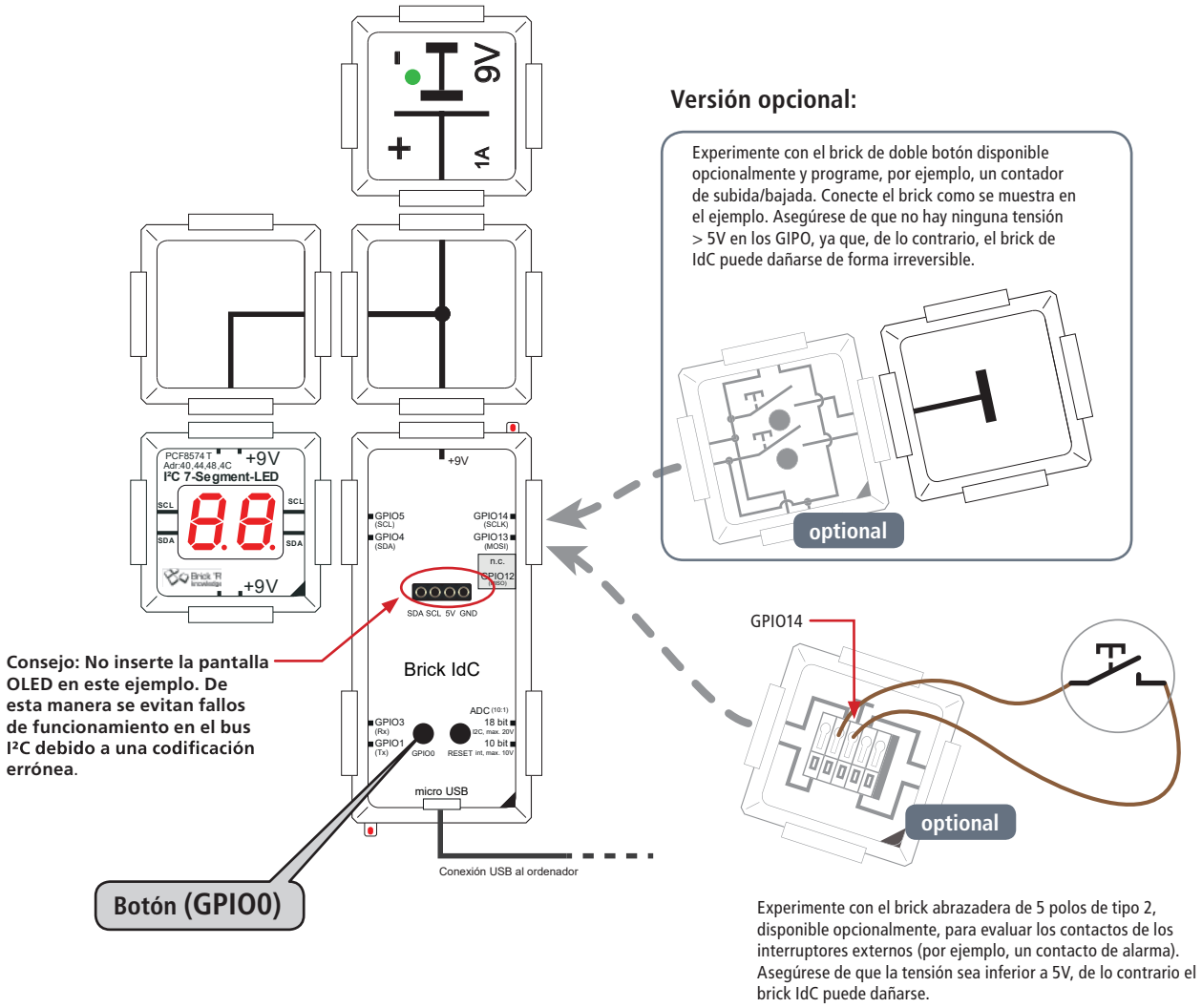


Fig. 33: con interruptor desbloqueado

Atención: No utilice interruptores externos que estén conectados directamente con 9V ya que los GPIO están diseñados para niveles de 5V solamente. Aconsejamos utilizar el interruptor integrado GPIO0. A un nivel de tensión > 5V en los GIPO, ya que, de lo contrario, el brick de IdC puede sufrir daños irreversibles!

Pruebe diferentes valores en el comando delay para determinar un tiempo de espera de rebote corto pero fiable. Experimente con bricks disponibles opcionalmente como en la Fig. 34 ("brick de doble botón" y " abrazadera de 5 polos de tipo 2"). Asegúrese de conectar los GPIO (que están configurados como entrada) sólo a tierra. Mientras el botón o el interruptor estén abiertos, hay un nivel high en el pin, ya que las resistencias pull-up están equipadas internamente (véase el capítulo 5.1.3 en la página 14).

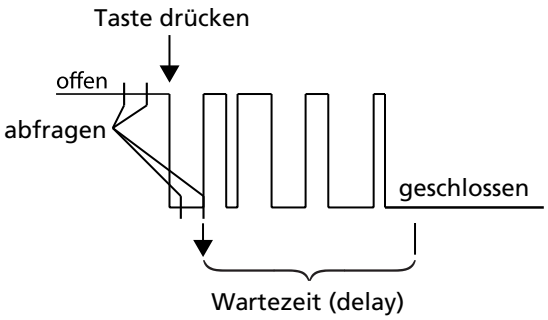


Fig. 34: Eliminacion del rebote de botones mediante software

Algoritmo para eliminar el rebote

El botón está bajo activo, es decir, al pulsarlo hay un nivel low en la entrada (ejemplo GPIO0). La señal se solicita durante la primera transición de high a low (en caso de petición). Luego hay una pausa hasta que llega a un nivel high (supuestamente, el botón no se presiona más) - el bucle while se abandona. El tiempo de retraso comienza desde aquí (40ms) para finalmente emitir el nuevo conteo válido. Se espera la siguiente pulsación, es decir, la transición de high a low. Tenga en cuenta que el tiempo de retraso depende del tipo de llave que se esté utilizando. El valor óptimo debe determinarse experimentalmente para evitar la evaluación de los impulsos de rebote.

Extracto del programa

```
void loop() { // Bucle

  char buffer[10];                  // Utilizar buffer de caracter de valor definido

  static int counter = 0;           // Fijar variable de contador a 0

  sprintf(buffer,"%02d",counter);   // Convertir un número entero en un signo

  // la llave, puede rebotar
  if (digitalRead(0)==LOW) { // Compruebe High->low
    // Esperar la transición low-high
    // También puede hacerlo después del delay, no es crucial.

    while (digitalRead(0)==LOW) {
     // Esperar hasta que se suelte el botón
    }
    counter++; // incrementar el contador
    delay(40); // esperar para que la petición no suceda demasiado rápido
  } // fin de la petición if High->Low

  if (counter > 99) counter = 0; // El contador debe contar entre 0 y 99

  // Contador de salida de dos dígitos, por lo que el buffer es 0 y 1
     display_seg1x(i2cseg7x2alsb1,buffer[1]); //signos LSB en la dirección 0x40
     display_seg1x(i2cseg7x2amsb1,buffer[0]); //signos MSB en la dirección 0x42

} // Find del bucle
```

¿Qué está pasando?

Por cada pulsación, la pantalla debe contar exactamente "1", es decir, 00, 01, 02...99. A continuación, la pantalla vuelve a mostrar 00.

No se confunda si el LED de codificación se ilumina en la parte inferior izquierda mientras presione el botón GPIO0. El LED y el botón están conectados y no tienen nada que ver con el código.

6.4 Pantalla OLED - Conceptos básicos

Las pantallas de 7 segmentos se utilizaban normalmente sólo para dígitos y muy pocas veces para letras. Con pantallas de 14 y 16 segmentos, las letras se pueden mostrar adecuadamente. Después de eso, hubo las primeras Grid Guide Displays que incluían una matriz de 5x7 puntos y que mostraban textos de mejor calidad e incluso mostraron los primeros carácteres gráficos. Las pantallas se basaron en LED, más tarde se introdujeron los LCD (Liquid Crystal Displays) y recientemente los OLED (Organic Light Emitting Diodes). Estos últimos son similares a los LED ya que son capaces de iluminarse. Nuestro set contiene una pantalla monocromática OLED de 128x64 píxeles con la que no sólo se pueden visualizar números individuales, sino también texto multilineal y gráficos simples. Para visualizar carácteres de esta manera se necesita un generador de carácteres o una tabla similar a la de la visualización de 7 segmentos. El código numérico se traduce al estado de encendido/apagado de los segmentos. El generador de carácteres o el gráfico de carácteres necesita comparativamente más memoria. El valor depende del número de píxeles por carácter. Para los más pequeños, con unos 5x7 píxeles, se necesitan unos 5 bytes por carácter. Si desea visualizar todo el conjunto ASCII (128 carácteres con espacios incluidos) necesita 128x5 bytes = 640 bytes.

Con el conector macho de 4 polos se coloca la pantalla OLED incluida en el brick IdC como se muestra en la figura.

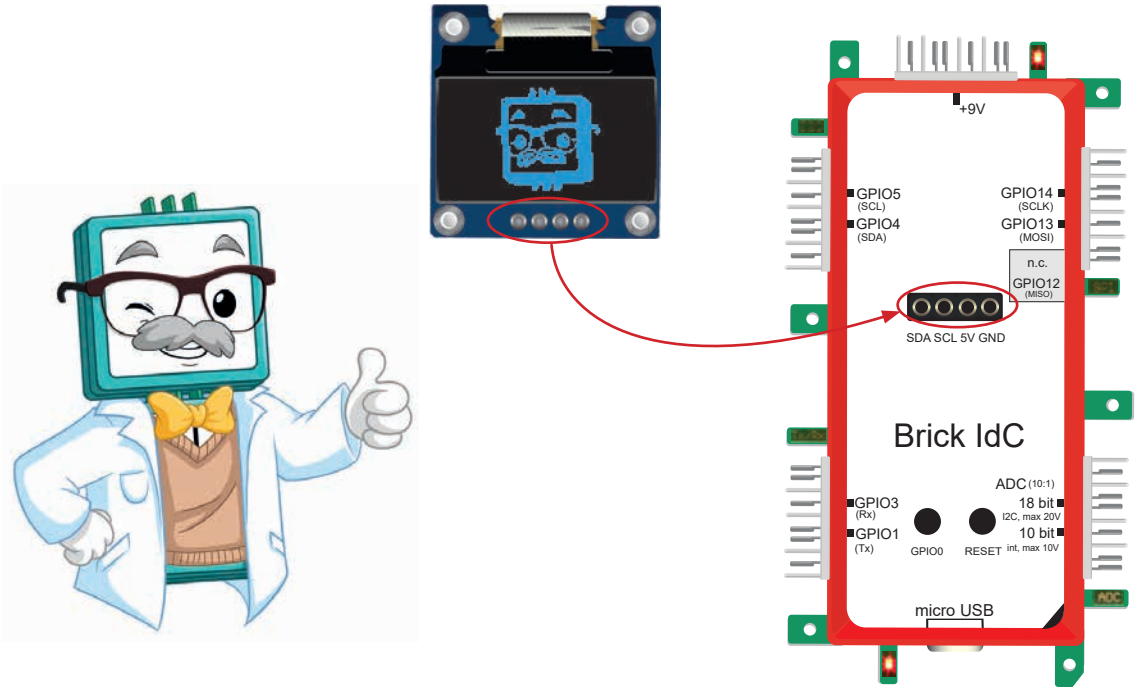


Fig. 35: pantalla OLED en el brick IdC

Abra el siguiente sketch en el IDE de Arduino: Beispiel_6.4.ino. Incluya el siguiente archivo de cabecera: #include <SSD1306Wire.h> y #include "Fig.s.h". El archivo Fig.s.h necesita ser incluido en la lista de proyectos de este sketch. Si aún no ha instalado la librería SSD1306Wire.h, consulte el capítulo 5.2.1 en la página 16. Para simplificar el control de las pantallas OLED utilizamos una librería completa. Si aún no lo ha instalado, consulte el capítulo 5 en la página 13.

La librería para controlar nuestra pantalla OLED está incluida con #include "SSD1306Wire.h" y con SSD-1306Wire display(0x3c, 4, 5);. Esto significa que, para obtener la dirección correcta para la función display(), necesita mover la dirección I²C del hardware de nuestras pantallas OLED (estándar: 0x78) un bit a la derecha. El programador escribe: (0x78>>1) - el resultado es "0x3c". Los parámetros que incluyen los valores 4 y 5 indican los GPIO para el bus I²C.

Cargue este sketch en su brick IdC para obtener una primera impresión. Te muestra todo lo que es posible, desde diferentes tamaños de fuente, pasando por gráficos, hasta cargar barras de progreso.

Tómese su tiempo para echar un vistazo a este y otros ejemplos. No se confunda, los ejemplos implican código que es, en parte, irrelevante para nosotros. En el siguiente ejemplo de OLED verá que es bastante fácil mostrar un texto en la pantalla.

Encuentra más ejemplos de la pantalla OLED en el IDE de Arduino:

"Datei – Beispiele – ESP8266 Oled Driver para SSD1306 display – ..."

Controle las siguientes líneas de código para que funcionen correctamente con el brick IdC:

- La distribución de los pines I²C para el OLED debe incluir: SSD1306Wire display (0x3c, 4, 5);
  o : SSD1306 display(0x3c, 4, 5); dependiendo del comando #include al principio del sketch.
- Comentando la línea incluyendo el comando display.fl ipScreenVertically(); debe haber dos barras oblicuas hacia adelante // prefijadas para evitar que se muestre de forma incorrecta.

6.4.1 Pantalla OLED - mostrar texto

Abra el siguiente sketch en el IDE de Arduino: Beispiel_6.4.1.ino. Incluya el siguiente archivo de cabecera: En caso de que aún no haya instalado la librería correspondiente, consulte el capítulo 5.2.1 en la página 16.

En este sencillo ejemplo queremos emitir un texto en la pantalla OLED utilizando la librería OLED SSD1306Wire.h.

Determine los siguientes parámetros:

- **Coordenadas**
  A partir de la coordenada (x, y) definida previamente se puede determinar el comando…display. drawString(x, y, "Beispieltext"); determine el punto de referencia del texto y su contenido.
- **Alineación**
  Procedente del punto de referencia definido previamente, con el comando display. setTextAlignment(TEXT_ALIGN_x); puede determinar si el texto está alineado a la izquierda (TEXT_ ALIGN_LEFT), centrado (TEXT_ALIGN_CENTER) o a la derecha (TEXT_ALIGN_RIGHT).
- **Tamaño de la fuente**
  Los tres tamaños de fuente estándar admitidos de forma predeterminada en esta librería son 10, 16 y 24 píxeles. Defina el tamaño con el comando display.setFont(ArialMT_Plain_X);

Imagine la pantalla como un sistema de coordenadas con origen 0.0 en la parte superior izquierda.

Fig. 36: El sistema de coordenadas de la matriz OLED

Extracto del programa

```
void loop() {

  display.clear();                                    // Display claro

  display.setTextAlignment(TEXT_ALIGN_LEFT);          // Alinear texto a izquierda
  display.setFont(ArialMT_Plain_16);                  // Tamaño de fuente 16 píxeles
  display.drawString(0, 0, "BRICK");                  // Posición 0,0  Texto: "BRICK"

  display.setTextAlignment(TEXT_ALIGN_CENTER);        // Alinear texto al centro
  display.setFont(ArialMT_Plain_16);                  // Tamaño de fuente 16 píxeles
  display.drawString(63, 20, "OLED");                 // Posición 63,20  Texto: "OLED"

  display.setTextAlignment(TEXT_ALIGN_RIGHT);         // Alinear texto a la derecha
  display.setFont(ArialMT_Plain_16);                  // Tamaño de fuente 16 píxeles
  display.drawString(127, 40, "TEST");                // Posición 127,40  Texto: "TEST"

  display.display();                                  // Exposición en el display

  delay(1000);
}
```

Si desea probar un tamaño de fuente diferente o incluso un tipo distinto, el programador de la librería OLED ofrece un editor de fuentes en línea: http://oleddisplay.squix.ch/#/home

Después de haber definido el texto con la posición, alineación, tamaño y contenido, puede mostrarlo en el OLED con el comando display.display();

Y ahora puede escribir diferentes textos en los diferentes tamaños y posiciones que desee!

6.5 Entradas analógicas

6.5.1 Convertidor A/D - conceptos básicos

La conversión A/D significa conversión analógico-digital. El objetivo es medir valores analógicos, como por ejemplo una tensión desconocida, y convertirlos en un valor digital. El ordenador puede seguir procesando este valor. Un ordenador normal no puede procesar valores analógicos. Aquí se realizan dos pasos importantes, una cuantificación de la amplitud, por ejemplo de la tensión, y una cuantificación del tiempo con un curso cronológicamente variable del valor analógico.

¿Qué significa esto?

La cuantificación en amplitud es fácil de entender. Asumiendo que una tensión analógica puede tomar cualquier valor entre 0 y 5V. Así que 2,3 V o 2,31 V o 2,315 V... y así sucesivamente. Por lo tanto, surge la pregunta, ¿cómo quiero medir exactamente y con qué precisión, es decir, en cuántos pasos quiero resolver mi señal? Finalmente, los valores numéricos deben estar preparados para ser procesados en un sistema de cálculo digital.

Ejemplo:

Queremos digitalizar un rango de tensión de 0 a 5V en 6 pasos. ¿Qué valor digital se asigna entonces para 2.1V? El diagrama siguiente muestra la asignación en los puntos rojos. El valor 2.1 está más cerca de 2 que de 3, por lo que se seleccionará el valor digital 2. Con un valor de 2,5, puede asignar 3 como valor digital si redondea la cifra 2,5 comercialmente.

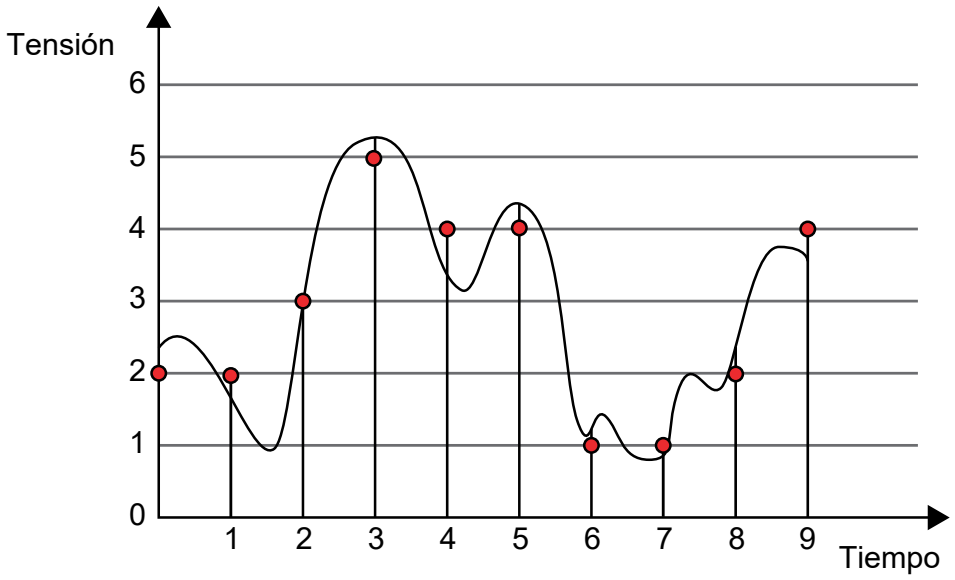


Fig. 37: Escanear señal analógica

En la figura anterior, registramos el curso de tiempo de una señal de voltaje (en el eje x vemos el tiempo en segundos y en el eje y el voltaje en voltios). Cuando la señal se convierte en una secuencia numérica digital, se muestrea un valor de medición (líneas verticales) por segundo y a continuación se realiza el redondeo a un dígito. El resultado es la siguiente serie de mediciones:

Tiempo	0s	1s	2s	3s	4s	5s	6s	7s	8s	9s
Voltaje	2V	2V	3V	5V	4V	4V	1V	1V	2V	4V

Hay dos efectos interesantes:

1. Perdemos información en amplitud, ya que en nuestro ejemplo se asume que la unidad de tensión detectable más pequeña es 1V. Este efecto también se denomina error de cuantización, que en nuestro caso puede ser de hasta 0.5 V de desviación del valor real en dirección positiva o negativa. Con una resolución más alta de la conversión, también obtendríamos más información sobre la señal original.
2. También perdemos información relevante para el tiempo. En el rango entre 1 y 2 segundos, vemos una caída de tensión de hasta 1V, que no es visible en la serie de medición. Los expertos en la materia notarán ahora que el llamado teorema de muestreo de Nyquist-Shannon ha sido vulnerado. Esto significa que la frecuencia de muestreo de una señal periódica debe ser al menos el doble de alta que su componente de frecuencia máxima (también llamado sobremuestreo). Este criterio se incumple en el caso de los osciladores superiores e inferiores más cortos.

Si ahora conecta los puntos rojos, obtendrá la señal "visible" para el ordenador, que se escaneó en una franja de tiempo de 1 segundo. La curva original ya no se puede reconstruir con precisión. Sin embargo, cuanto mayor sea el número de puntos de muestreo, mejor podrá reconstruirse la señal (véase el teorema de muestreo).

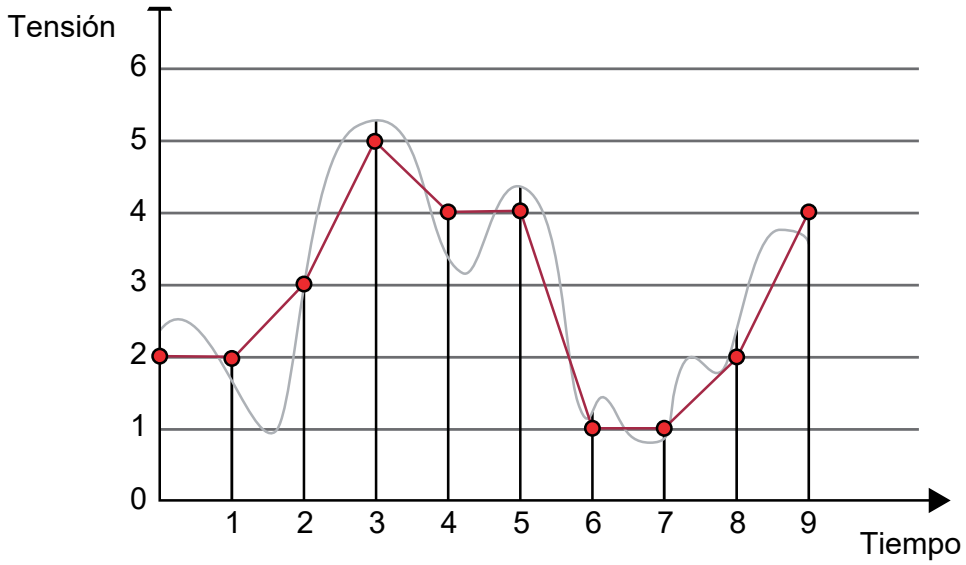


Fig. 38 : Reconstrucción de la señal de muestreo

La resolución de la amplitud se determina por el número de pasos discretos asignados a un valor numérico. En nuestro ejemplo, distinguimos 6 niveles de tensión. A modo de comparación, un convertidor A/D de 10 bits ya proporciona 1024 etapas, que por lo general están codificadas en binario (véase también el apartado 6.5.2.5 en la página 38). En el brick IdC podemos elegir entre un convertidor A/D de 10 bits y uno de 18 bits.

Conversión A/D de 10 y 18 bits en comparación

	Convertidor A/D de 10 bits	Convertidor A/D de 18 bits
Número de etapas (resolución)	$2^{10} = 1024$	$2^{18} = 262144$
Rango de medición y valor digital correspondiente (decimal)	0…9,99 V $0…1023_{(10)}$	0… 19,999924 V $0…262143_{(10)}$
Nivel de tensión mínimo (llamado "Least Significant Bit" (LSB))	10 V / 1024 = 0,0098 V (= 9,8 mV)	20 V / 262144 = 0,000076 V (= 76 µV)
Cuantización	±4,9 mV	±38 µV
Frecuencia de muestreo máx.	200 Muestras/segundo (= 200 S/s)	3,75 Muestras/segundo (= 3,75 S/s)
Tipo de convertidor.	Convertidor-SAR	Convertidor-Delta-Sigma (convertidor-ΔΣ)

Para convertir las señales analógicas en valores codificados binarios y digitales, existen diferentes métodos de conversión que irían más allá del alcance de este manual. Como parte de nuestra propia investigación nos gustaría mencionar sólo los procedimientos más importantes:

- **Convertidores de integración (método de conteo)**
  Más lento que el método de pesaje, sin interferencias, bajos costes de hardware, realización: convertidor de una, dos y cuatro pendientes, ejemplo de aplicación: multímetro.
- **Transductores de realimentación (método de pesaje)**
  Buen compromiso entre velocidad y costes de hardware, realización: Convertidor delta-Sigma (ΔΣ-convertidor), ejemplo de aplicación: convertidor A/D de 1 bit en tecnología de audio. Convertidor SAR (Successive Approximation Register), ejemplo de aplicación: tecnología de medición de resolución muy alta.
- **Convertidor en paralelo (Convertidor Flash y Pipeline)**
  Realización muy rápida, muy cara: Convertidor de flash y de pipeline, Ejemplo: ingeniería de radar.

6.5.2 Los convertidores A/D en el Brick IdC

6.5.2.1 El convertidor A/D de 10 bits

El ESP8266 ofrece un convertidor analógico-digital (A/D) integrado con una resolución de 10 bits, que corresponde a una resolución del valor de tensión de 210 = 1024 pasos. El rango de tensión de entrada del propio convertidor es de 0V a 1V. Para una mejor realización, se conectó un divisor de tensión de 10 a 1 (10: 1) en la entrada de brick "ADC 10 bit", de forma que se puede medir directamente una tensión de 0V a 10V. Esto le permite construir un voltímetro simple con el cual puede medir voltajes DC de 0V a 10V.

6.5.2.2 El convertidor A/D de 18 bits

Para otros experimentos se instaló adicionalmente un convertidor A/D muy preciso del tipo MCP3421 en el brick IdC, que tiene una resolución muy alta de 18 bits (218 = 262,144 pasos). Este módulo convertidor se conecta a través del bus I²C y ofrece un rango de tensión de entrada de 0V a 2V con un convertidor de tensión de 10 a 1 corriente arriba. Esto da como resultado un rango de tensión de entrada de 0V a 20V, es decir, se puede aplicar una tensión de entrada máx. de 20V en la entrada del brick "ADC 18 bit".

6.5.2.3 El divisor de tensión

En ambas entradas del convertidor de tensión "ADC 10 bit" y "ADC 18 bit", el brick IdC tiene un divisor de tensión con un ratio de división de 10:1:

Asegúrese de que no haya más de 10V en la entrada "ADC 10 bit" y nunca más de 20V en la entrada "ADC 18 bit". De lo contrario, el convertidor A/D puede dañarse!

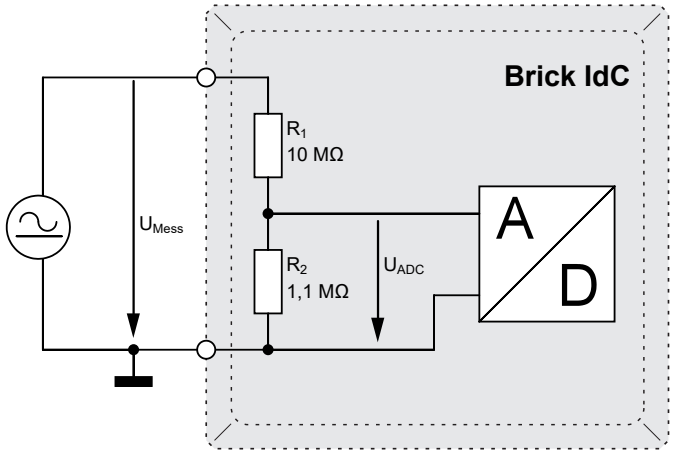


Fig. 39: Divisor de tensión (10.1)

El ratio de división se calcula de la siguiente manera: $\frac{U_{Mess}}{U_{ADC}} = \frac{R_1 + R_2}{R_1} = \frac{11{,}1\ \text{MΩ}}{1{,}1\ \text{MΩ}} = 10{,}09 \approx 10$

6.5.2.4 Consejo práctico: factor de corrección

Tenga en cuenta que pueden producirse imprecisiones en el resultado de la medición debido a las tolerancias de los divisores de resistencia interna en las entradas "ADC 10 bit" y "ADC 18 bit". Esta desviación puede ser compensada por un factor de corrección. Para determinar el factor de corrección, por ejemplo, utilice un multímetro preciso y ajuste la tensión UMess. A continuación, calcule el factor de corrección a partir del ratio entre la tensión medida con el multímetro y la tensión mostrada en el brick IdC sin corrección! En el programa de ejemplo, hemos preparado el código apropiado para esto (Quitar las dos barras oblicuas (comentario) sólo después de haber hecho la medición sin corrección). Para una mayor claridad, el factor de corrección se define como una constante al principio del programa (véase la siguiente sección del programa para el convertidor A/D de 10 bits).

$$Factor\ de\ corrección = U_{Mess,\ Multimeter} / U_{Display\ Brick}$$

Sección de programa para el factor de corrección (vea el ejemplo_6.5.3.ino)

```
//El procedimiento es el mismo para ADC de 10 y 18 bits.
...
//defina el factor de corrección para ADC de 10 bits
const float Correction _ 10bit = 1.046;
...
//Calcular el factor de corrección (determinado con la medición del multímetro)
UMess _ 10bit = UMess _ 10bit * Correction _ 10bit;
...
```

Dado que cada resistencia tiene diferentes tolerancias de fabricación, el factor de corrección para los convertidores A/D también debe determinarse por separado.

6.5.2.5 Codificación binaria

Un convertidor A/D convierte un valor medido analógico en la entrada en un valor codificado binario en la salida. También hablamos de una palabra binaria, cuyo ancho, es decir, el número de caracteres binarios individuales se especifica en bits (escritos en minúsculas como una unidad, digamos: 1 bit). En nuestro caso, tenemos una palabra binaria de 10 o 18 bits dependiendo de la resolución del convertidor A/D. El bit menos significativo se denomina a menudo "Least Significant Bit" (LSB) y el bit más significativo "Most Significant Bit" (MSB).

Número de bits	Número de estados codificables	número binario (Número de estados codificables - 1)	Número decimal	Hexadecimal
		Bit más alto (Most Significant Bit (MSB)) … Valence … Bit menos significtivo (Least Significant Bit (LSB))		
		2^{17} … 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 Convertidor A/D de 10 bits

Abra el sketch en el IDE de Arduino: Beispiel_6.5.3.ino. Se deben incluir los siguientes archivos de cabecera: SSD1306Wire.h. Si no tiene instaladas las librerías apropiadas, vaya al capítulo. 5.2.1 en la página 16 y recupérelo.

En este ejemplo, medimos una tensión a través de una resistencia variable (potenciómetro, a menudo llamado Poti).

El valor de resistencia del potenciómetro varía entre 0 Ω y 10 kΩ dependiendo de la posición.

La caída de tensión en el potenciómetro es lineal al valor de resistencia. Cuando el potenciómetro está en el sentido de las agujas del reloj, el valor de la resistencia entre la masa y el contacto de deslizamiento conectado a la entrada analógica (ADC 10 bit) del bricks IdC es de aproximadamente 10 kΩ. Si ahora giramos el potenciómetro en sentido contrario a las agujas del reloj hasta el tope, entonces hay 0V (masa) en la entrada del convertidor A/D. En la posición central del potenciómetro, se aplica aproximadamente la mitad de la tensión de alimentación,es decir, aproximadamente 4.5V.

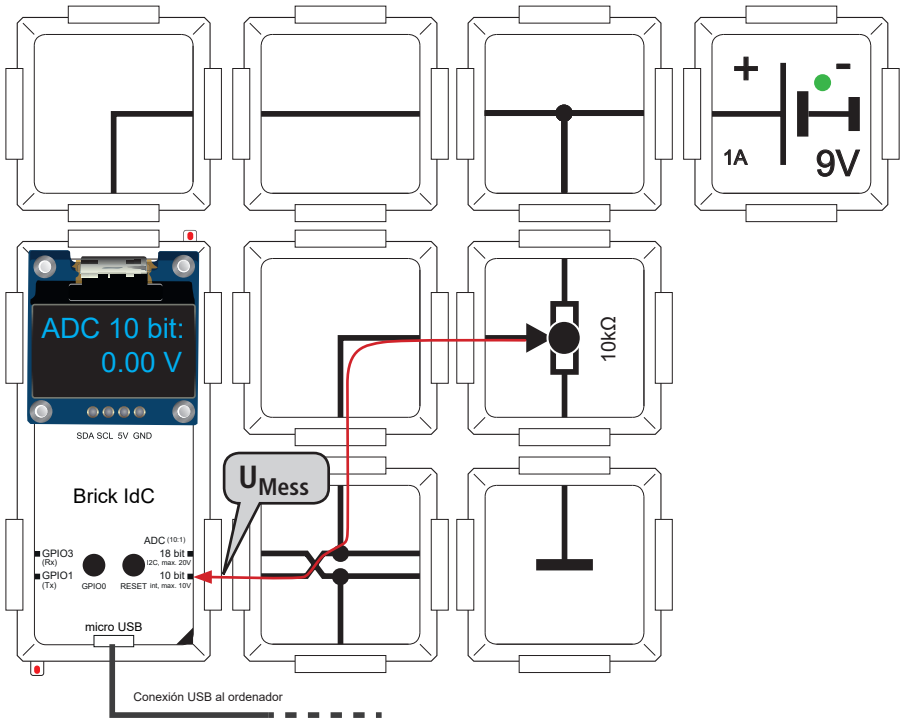


Fig. 40: Circuito de brick convertidor A/D de 10 bits

La tensión medida UMess se visualiza en la pantalla OLED y se actualiza cada segundo. Además, el valor bruto digital y el valor de tensión calculado se muestran en el monitor serie de su ordenador. Para abrir el monitor serie, simplemente haga clic en el icono de lupa en el IDE de Arduino (ver capítulo 5.2.3 en la página 20). Para convertir el valor digital del convertidor A/D al valor de la tensión analógica se puede utilizar la siguiente fórmula (véase también el programa de ejemplo Beispiel_6.5.3.ino).

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

Ajustando 1023 para analogIN, se obtiene el valor máximo del rango de tensión de entrada. En nuestro caso es de 9.99 V.

➔ Vea la página siguiente para los detalles del programa.

Extracto del programa

```
void loop(void)
{
// Lee el voltaje como valor crudo: 0 = 0V bis 1023 = 1V-1LSB
    analogIN = analogRead(adc_esp8266);
// Convertir a Volt con analog IN / resolución (* 1 Volt), * 10 a causa de 10: 1
divisor en el ADC
    UMess = ((float)analogIN/1024)*10.0;
// Convertir el valor de tensión en string para salida OLED
    String UMess_str = String(UMess, 2); // 2 decimales

    display.clear();       //eliminar 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();     // salida a OLED
  delay(1000);
```

Utilizando el comando analogIN = analogRead (adc_esp8266); lee el valor crudo digital del convertidor interno A/D del ESP8266. El rango de valores del convertidor de 10 bits es de 0 a 1023 (decimal). En nuestro ejemplo, el valor 0 corresponde a la tensión 0V y el valor bruto 1023 corresponde al valor máximo del rango de tensión de entrada.

Por razones sintácticas, primero debemos convertir la variable analógica ININg, que devuelve el valor crudo del convertidor A/D, en la variable flotante UMESS para convertirla en el valor de tensión correcto. Para la salida, necesitamos la variable de cadena UMess_str.

6.5.4 Convertidor A/D de 18 bits

Abra el sketch en el IDE de Arduino: Example_6.5.4.ino. Tiene que incluir los siguientes archivos de cabecera: SSD1306Wire.h y MCP3421.h. Si aún no ha instalado las librerías correspondientes, vaya al capítulo 5.2.1 en la página 16 y hágalo.

En este ejemplo queremos controlar el convertidor A/D de 18 bits MCP3421 de Microchip, que puede utilizarse por ejemplo, para la medición precisa de la temperatura con termopares o sensores de temperatura de resistencia (RTD) y comparar los valores medidos del convertidor interno de 10 bits con los del convertidor de 18 bits controlado a través del bus I²C.

La única diferencia entre el circuito de bricks para este ejercicio y el ejercicio anterior es el brick en la parte inferior, en el centro. De este modo, se aplica la misma tensión UMess a ambas entradas analógicas, que pueden ser variadas a través del potenciómetro. Esto facilita la comparación directa de los dos convertidores A/D. La resolución 256 veces mejor del convertidor A/D de 18 bits en comparación con el convertidor A/D de 10 bits, se puede ver claramente en el resultado de la medición.

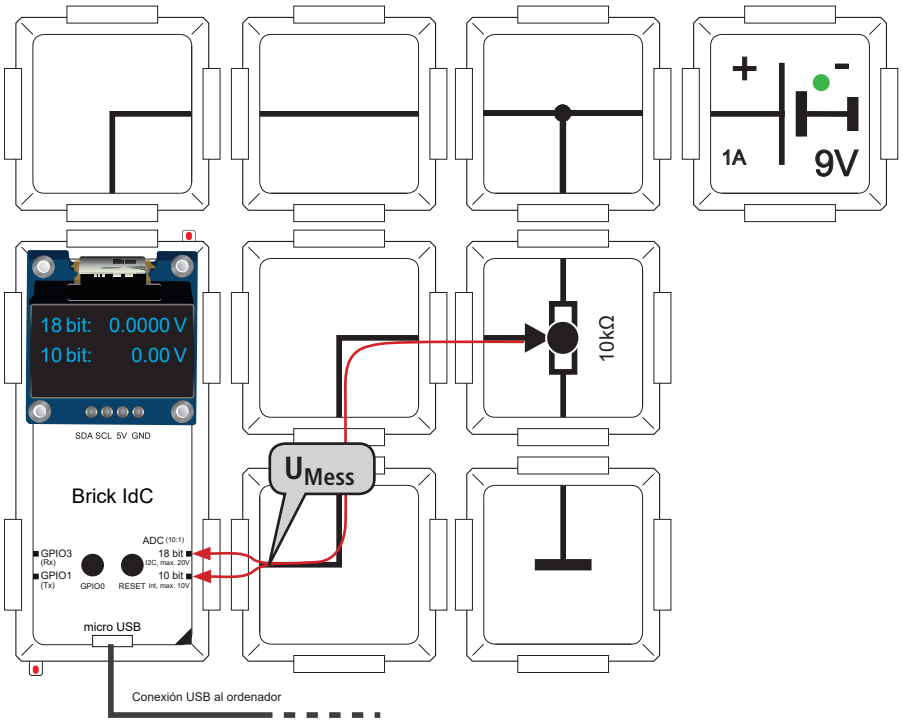


Fig. 41: Circuito de brick convertidor A/D de 18 bits

Como ya se ha mencionado en el capítulo anterior, las tolerancias de los componentes juegan un papel importante. En primer lugar, la tolerancia de resistencia estándar del 10% de los dos divisores de tensión 10:1 en las entradas de los convertidores A/D tiene un efecto importante (véase el capítulo "6.5.2.3 El divisor de tensión" en la página 37).

➔ Para los detalles del programa, véase la página siguiente.

Extracto del programa

```
void setup(void)
{
…
// Init MCP3421: dirección I2C, modo de 18 bits, no ganancia
    MCP.init(0x68,3,0);
…
}
void loop(void)
{
…
// Lee el voltaje como valores dobles de MCP3421, rango de entrada: 0 to 2.048V
    analogIN _ 18bit=MCP.getDouble();
    UMess _ 18bit = analogIN _ 18bit*10.0; // * 10 a causa de 10: 1 divisor
// La siguiente línea para determinar el factor de corrección utilizando
// Por favor, mida la medida del multímetro!
UMess _ 18bit = UMess _ 18bit * Corrección _ 18bit; // Factor de Corrección
// Convertir el valor de tensión en cadena para la salida OLED
    String UMess _ 18bit _ str = String(UMess _ 18bit, 4);
…
  display.display();     // Salida de ambos valores de tensión a OLED
  delay(1000);
}
```

Utilizamos la libreria MCP3421.h para simplificar la programación del MCP3421. El convertidor A/D se inicializa primero con MCP.init(0x68,3,0); La función MCP.get-Double() devuelve el valor de tensión como un valor doble, es decir, un número de coma flotante.

Dado que el MCP3421 explora un rango de tensión de entrada de 0 a 2,048 V, el valor debe multiplicarse por el factor 10. A continuación, introducimos un factor de corrección, como el descrito en el capítulo 6.5.2.4 en la página 38, que es definido por el flotador como una constante al principio del sketch Para la salida en pantalla OLED o monitor de serie tenemos que convertir los números de coma flotante UMess_10bit y UMess_18bit en las cadenas correspondientes. Los valores se actualizan aproximadamente cada segundo.

6.6 Ejemplos de IdC

Controle su brick IdC a través de su smartphone o cualquier otro dispositivo compatible con WiFi y descubra nuevas posibilidades. Programe una pequeña pagina web y cree su propio centro de control.

Aprenderá...

...cómo sincronizar la fecha y la hora a través de Internet (ejemplo 6.6.2)

...cómo importar los datos del sensor de temperatura y humedad (ejemplo 6.6.3)

...cómo solicitar el tipo de cambio actual del dólar desde Internet (ejemplo 6.6.4)

Más tarde (ver ejemplo 6.6.5) aprenderá a programar una pequeña página web como base para su propio centro de medición IP. El brick adaptador universal de sensores (ALL-BRICK-0649) le permite conectar fácilmente numerosos sensores. Finalmente, en el ejemplo 6.6.6, aprenderá cómo encender y apagar los LED a través de Internet. Esta es una característica básica que se necesita en diversas aplicaciones de la domótica.

Debido a la sencilla integración en red del brick IdC, existen muchas posibilidades nuevas.

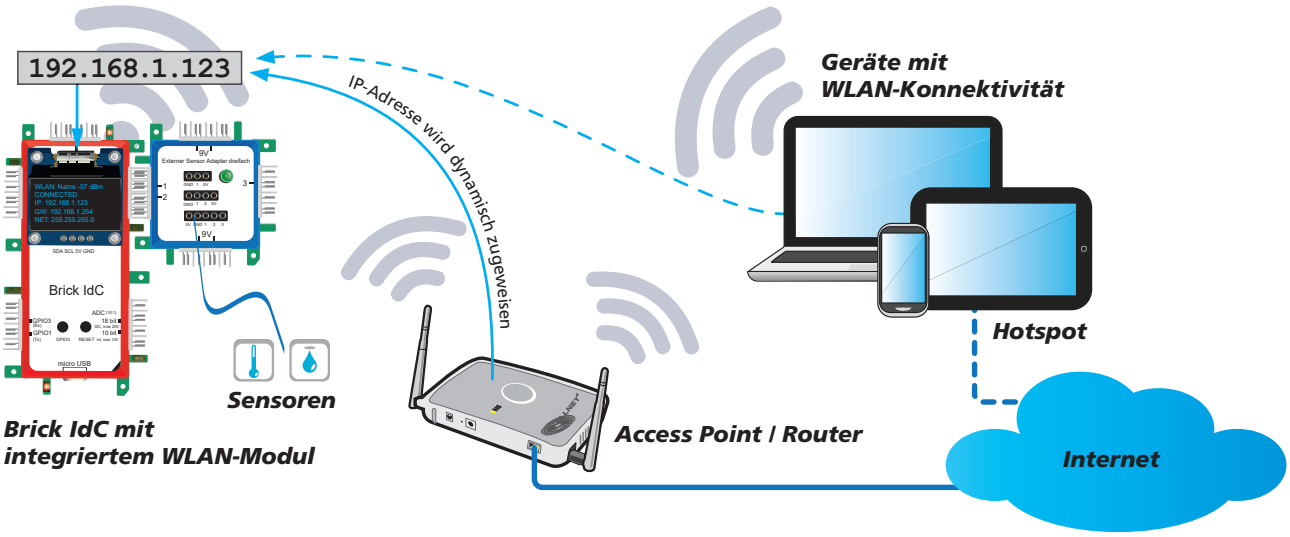


Figura 42: El brick IdC en la red

La gran ventaja del módulo ESP-12-F - instalado en el brick IdC - con respecto al Arduino es su capacidad de red gracias al módulo WiFi integrado. El microcontrolador ESP 8266 puede integrarse fácilmente en una red WiFi existente. La dirección IP se asigna normalmente de forma dinámica. El hotspot puede ser el router de su red local o su smartphone.

Todos los ejemplos de IdC necesitan una conexión a Internet, es por eso que verá el siguiente símbolo:

En los ejemplos 6.6.5 y 6.6.6 se puede acceder al servidor web integrado en el brick IdC a través de su red. Teóricamente también a través de Internet, si puede configurar su red para ello. Las palabras clave aquí son redireccionamiento de puertos y cortafuegos. Los routers estándar normalmente ofrecen la posibilidad de reenviar un acceso específico desde Internet a un puerto interno. Este puerto interno es en realidad el brick IdC con su dirección IP local (el llamado reenvío de puertos). Es necesario que su cortafuegos permita este tipo de acceso desde el "exterior". Encontrará más información sobre las configuraciones relevantes de su router en la documentación o en Internet.

6.6.1 Configuración del brick IdC como cliente WiFi

 Abra el sketch en el IDE de Arduino: Beispiel_6.6.1.ino. Debe incluir los siguientes archivos de cabecera: ESP8266WiFi.h y SSD1306Wire.h. Si aún no ha instalado las librerías, vuelva al capítulo 5.2.1 en la página 16 e instálelas.

Si desea comunicarse con su brick IdC a través de WiFi, primero debe conectarlo a su red local. Necesitarás el nombre WiFi (también llamado SSID) de tu punto de acceso o router y la contraseña correspondiente. Puesto que no hay posibilidad de introducir caracteres en el brick, tenemos que almacenar los datos de acceso en nuestro sketch:

Encuentre las líneas verdes al principio del archivo de sketch. Reemplace mi_nombre_de_wifi por el nombre (SSID) de su WiFi y en lugar de mi_contraseña_de_wifi, introduzca la contraseña correspondiente (las comillas deben permanecer en blanco).

Extracto del programa

```
//Introduzca aquí el nombre del wifi (SSID) :
const char* ssid = "mein_wlan_name";
//Introduzca aquí su contraseña de WiFi :
const char* password = "mein_wlan_passwort";
...
void setup() {
...
}
```

En la función void loop(), puedes ver cómo las siguientes cadenas son ensambladas y y preparadas para el despliegue línea por línea con el comando display.drawString(x,y, "String"). Las siguientes funciones se utilizan para determinar los valores de los parámetros.

`wlan_oled` (Nombre y potencia de WiFi), función `WiFi.SSID()` y `WiFi.RSSI()`.

- `state` (estado de conexión), Función `WiFi.state`
- `ip_oled` (dirección IP del brick IdC), Función `WiFi.localIP()[x]`
- `gw_oled` (gateway-IP), función `WiFi.gatewayIP()[x]`
- `mask_oled` (máscara de subred), función `WiFi.subnetMask()[x]`

La salida se realiza finalmente con el comando `display.display();`

Paralelamente, la información de la conexión de red también se muestra en el monitor serie de su ordenador. Para abrir el monitor serial, simplemente haga clic en el icono de la lupa en la esquina superior derecha del IDE de Arduino (ver Cap.5.2.3 en la página 20).

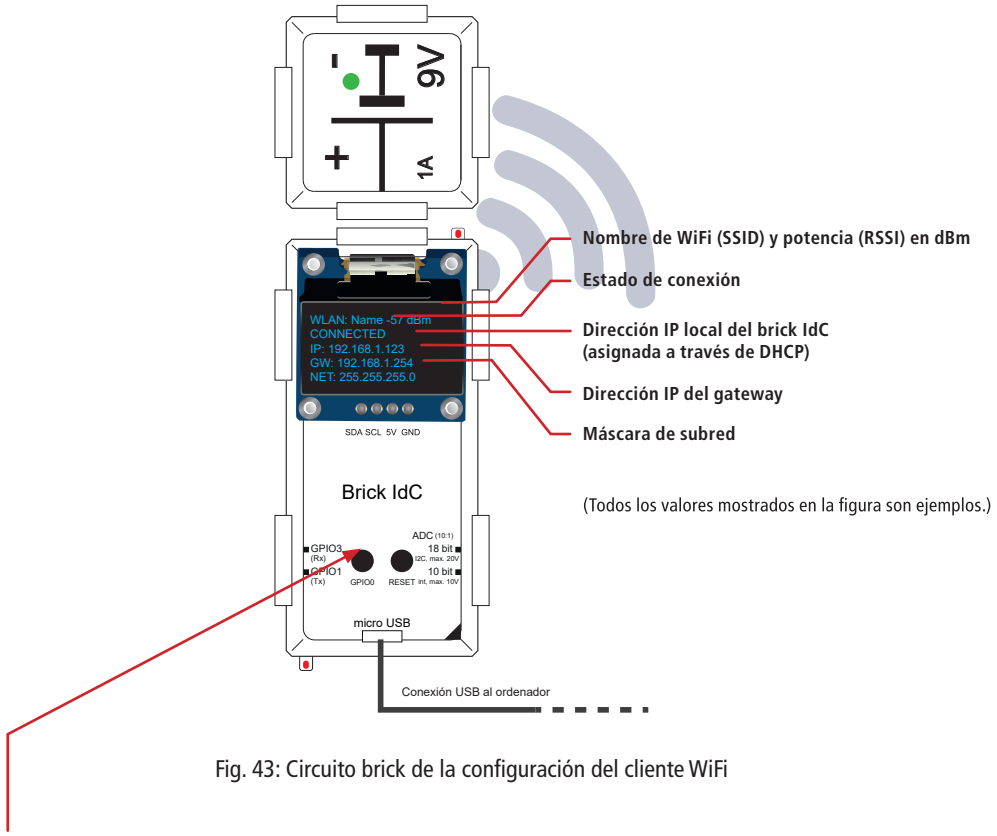
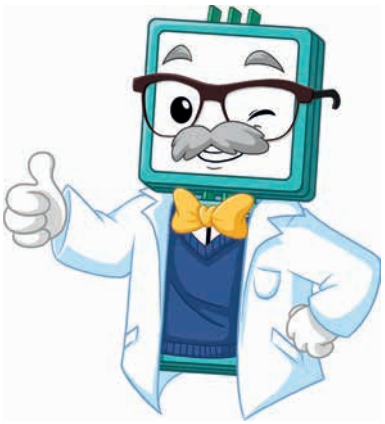


Fig. 43: Circuito brick de la configuración del cliente WiFi

La configuración WiFi también se puede visualizar pulsando el botón GPIO0 (figura 43)!



6.6.2 Tiempo desde Internet

 Abre el siguiente sketch en el IDE de Arduino: Beispiel_6.6.2.ino. Debe integrar los siguientes archivos de cabecera: ESP8266WiFi.h, SSD1306Wire.h, TimeLib.h y NtpClientLib.h. Si aún no ha instalado las librerías, vuelva al capítulo 5.2.1 en la página 16 e instálelas.

Para este ejemplo, primero debe introducir su nombre WiFi (SSID) y la contraseña correspondiente en el programa. Proceda como se describe en el capítulo 6.6.1.

En Internet existen los llamados servidores de tiempo a los que se puede acceder a través de NTP (Network-Time-Protocol) para determinar la hora y la fecha actuales. El dominio pool.ntp.org permite utilizar a cualquiera un pool de servidores de tiempo de forma gratuita. Para simplificar el procesamiento de la información transferida desde el servidor de horario, utilizaremos funciones predefinidas en las librerías TimeLib.h y NtpClientLib.h de este ejercicio. Como pantalla utilizamos de nuevo nuestra pantalla OLED.

Extracto del programa

```
...
void setup() {
...
//si el WiFi está conectado, conéctese al servidor de hora NTP
{
    NTP.begin("pool.ntp.org", 1, true);  ////conectarse al servidor de tiempo NTP
    NTP.setInterval(63);          //sincronizar cada 63 segundos
    }

NTP.onNTPSyncEvent([](NTPSyncEvent_t event) {
    ntpEvent = event;
    syncEventTriggered = true;     //conectado al servidor de hora
  });
...
}

void loop() {
...
    String time = NTP.getTimeStr();  //guardar la hora del día en la hora variable
    String date = NTP.getDateStr();  //guardar la fecha en la fecha variable
    display.clear();
    display.setTextAlignment(TEXT_ALIGN_LEFT);
    display.setFont(ArialMT_Plain_24);
    display.drawString(15, 5, time);    //Preparar la salida del tiempo
    display.drawString(5, 35, date);    //preparar la salida de la fecha
    display.display();                 //actualizar pantalla OLED
...
}
```

En cuanto se establece la conexión WiFi, la función NTP.begin("pool.ntp.org", 1, true); establece la conexión con el servidor NTP. Esto puede tardar algunos segundos. El primer parámetro asigna la URL al pool de servidores de tiempo, el segundo es la desviación de nuestra zona horaria al "Universal Time Coordinated" (UTC) de +1 hora y el valor verdadero en el tercer parámetro indica que en nuestra zona horaria cambiamos entre el horario de verano y el de invierno. NTP.set-Interval(63) define el intervalo en segundos en el que la sincronización con el servidor NTP debe actualizarse y syncEventTriggered = true informa de la sincronización correcta con el servidor NTP.

En void loop() la función NTP.getTimeStr() lee la hora y NTP.getDateStr() lee la fecha como una cadena y la prepara para la salida. La salida real a la pantalla OLED se realiza con la función display.display(); La línea display.clear() también es importante en este ejemplo, para borrar la pantalla y no simplemente anularla en cada pasada del bucle.

Paralelamente a la visualización en la pantalla OLED, la salida se envía también al monitor serial de su ordenador. Para abrir el monitor seria, simplemente haga clic en el icono de la lupa en la esquina superior derecha del IDE de Arduino (ver cap. 5.2.3 en la página 20). Además de la hora y la fecha, se muestran otras informaciones como, por ejemplo, el tiempo transcurrido desde la sincronización inicial (tiempo de actividad) y la hora de la sincronización inicial.

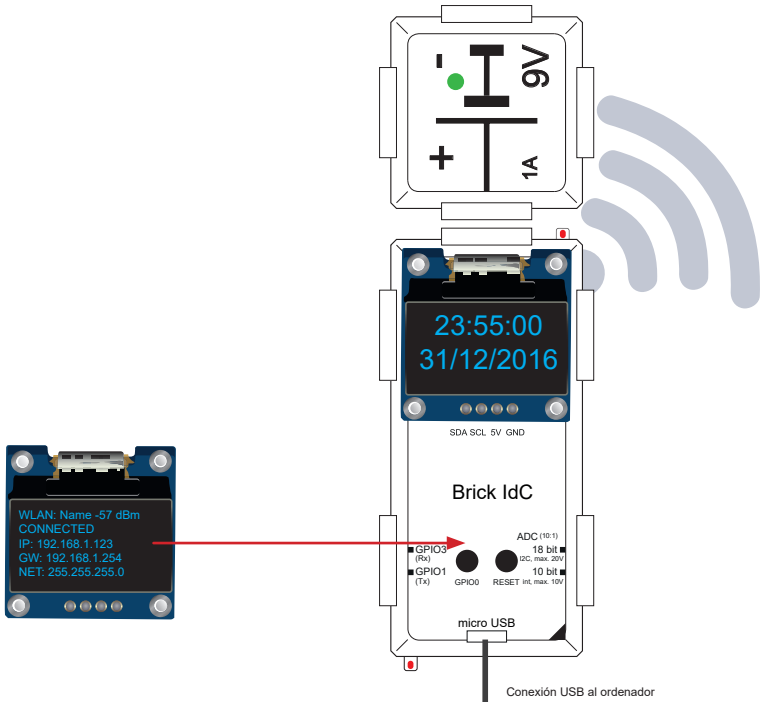
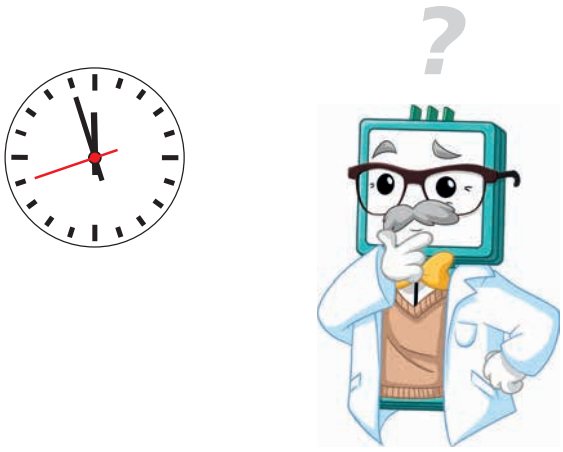


Fig. 44: Circuito brick de la configuración del tiempo desde Internet

PISTA: El estado de conexión del brick IdC siempre se puede visualizar pulsando el botón GPIO0 (ver figura 43)!



6.6.3 Medición de temperatura y humedad

Abra el sketch en el IDE de Arduino: Beispiel_6.6.3.ino. Se deben incluir los siguientes archivos de cabecera: ESP8266WiFi.h, SSD1306Wire.h, TimeLib.h, NtpClientLib.h y DHT.h. Si aún no ha instalado las librerías correspondientes, consulte el capítulo 5.2.1 en la página 16 y hágalo.

Para este ejemplo, también debe introducir primero su nombre WiFi (SSID) y la contraseña correspondiente en el programa de ejemplo. Proceda como se describe en el capítulo 6.6.1.

Para la medición de la temperatura y la humedad, utilizamos el sensor combinado del tipo DHT11, para el cual ya existe una librería (DHT.h) y ejemplos. Además, hay muchos senores diferentes del mundo Arduino, tales como:

- sensores térmicos
- barrera de luz infrarroja
- Detector de movimiento (sensor IR)
- sensor de luz (LDR)
- sensor de gas
- sensor hall
- sensor de impacto
- llave de contacto

El brick adaptador universal para sensores (ALL-BRICK-0649) le permite conectar numerosos sensores estándar muy fácilmente. Para muchos sensores ya existen ejemplos y librerías, de modo que también se pueden integrar sin problemas en proyectos propios.

Primero junte los bricks como se muestra. Preste atención a la conexión correcta del sensor DHT11 suministrado. Inserte el sensor exactamente como se muestra en la fig.45, en el zócalo inferior de 5 polos del brick adaptador del sensor en el extremo izquierdo. La etiqueta "5V" en el sensor debe coincidir con el zócalo con la etiqueta "5V" en el extremo izquierdo. De lo contrario existe peligro de daños irreversibles en el sensor y/o en el brick IdC!

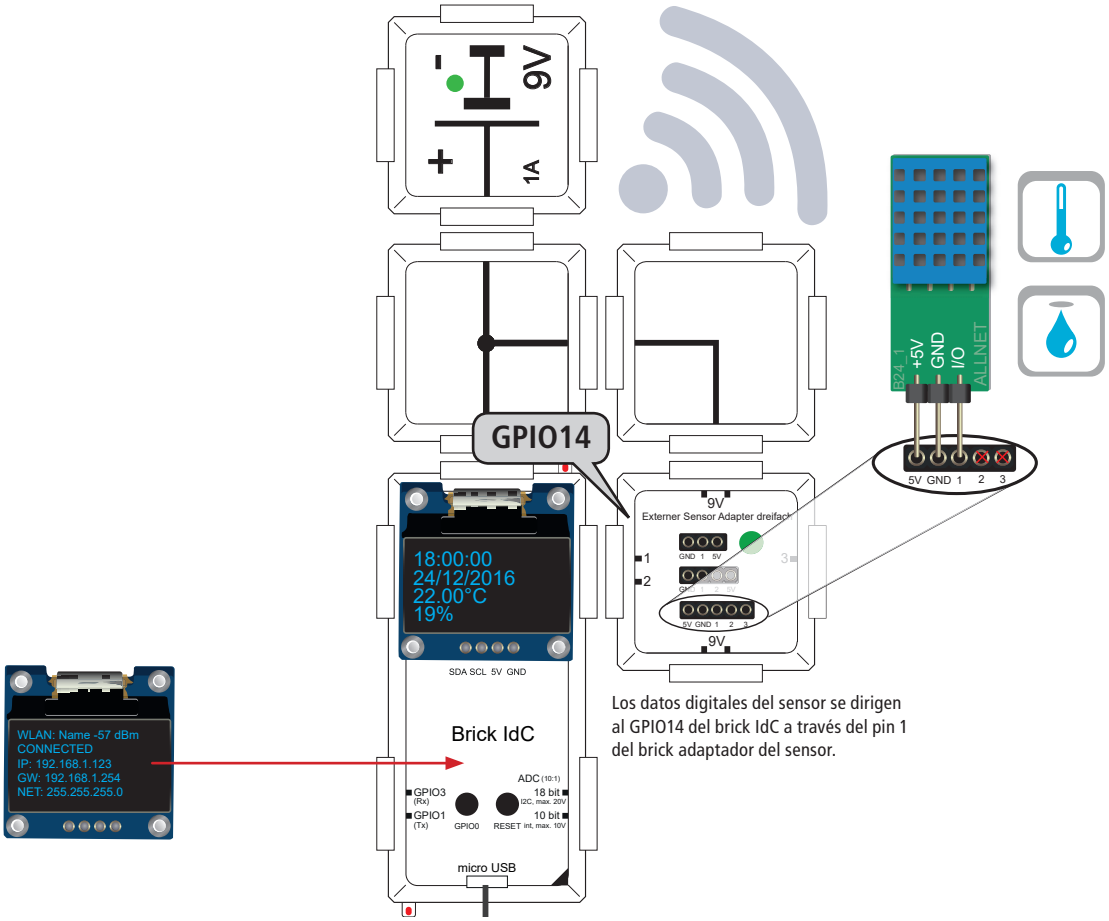


Fig. 45: Circuito brick para medir la temperatura y la humedad

CONSEJO: El estado de conexión del brick IdC se puede visualizar en cualquier momento pulsando la tecla GPIO0 (véase también la Fig. 41)!

Extracto del programa

```
#define DHT_TYPE DHT11     // definir tipo de sensor: DHT11
const int DHT_PIN = 14;  //línea de datos del sensor a GPIO4 del brick IdC
char temp[20];            //definir la variable String para la temperatura
char humi[20];            //definir la variable String para la humedad

DHT dht(DHT_PIN, DHT_TYPE);   //variable de tipo DHT

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

void loop() {
...
 if (counter%1000==0){             //Revise el sensor  una vez por segundo aprox
    float t = dht.readTemperature(); //Leer la temperatura (Celsius)
    float h = dht.readHumidity();     //Leer la humedad

    sprintDouble(temp,t,2);  //Convertir temp con 2 decimales en string
    sprintDouble(humi,h,0);  //Convertir la humedad sin decimales en string
    strcat(temp," °C");      //Añadir °C al string
    strcat(humi," %");       //añadir carácter % a un string

  }
  display.drawString(5, 30, temp);   //Preparar la salida de la temperatura
  display.drawString(5, 45, humi);   //Preparar la salida de humedad
  display.display();                //actualizar la pantalla OLED
...
}
```

Al principio del sketch se definen el tipo de sensor DHT11 y el pin GPIO de la línea de datos (aquí GPIO4). Una variable especial es el tipo DHT, el cual se usa para inicializar el sensor en void setup().

Pero ahora para la medición actual en la sección void loop (). Con las dos llamadas de función de la librería de sensores dht.readTemperature() y dht.readHumidity() se leen la temperatura o humedad del sensor y se almacenan en las dos variables de coma flotante t y h. Con la función de ayuda sprint Double () los números de coma flotante se convierten en strings con el número de decimales deseado y la operación de string strcat () añade la unidad correspondiente.

Además de los valores de temperatura y humedad, la pantalla se complementa con la hora y la fecha, tal y como se muestra en el ejercicio 6.6.2. La salida real de todos los valores en la pantalla OLED es como de costumbre, con la función display.display();

Paralelamente, la salida también se encuentra en el monitor serie de su ordenador. Para abrir el monitor serial, simplemente haga clic en el icono de la lupa en la esquina superior derecha del IDE de Arduino (ver cap. 5.2.3 en la página 20). En este ejemplo, la salida del ejemplo 6.6.2 se ha ampliado para incluir la temperatura y la humedad.

Adaptando la ranura y el sketch, el "Sensor-Adapter-Brick triple" se puede utilizar para conectar numerosos sensores en el mercado.

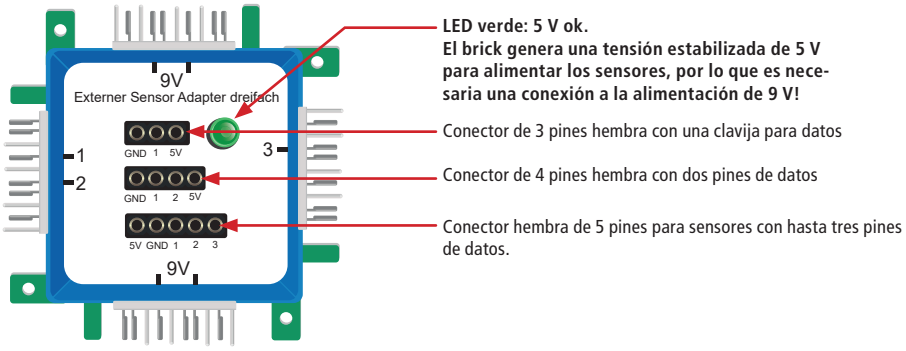


Fig. 46 Conexiones posibles para el "Sensor-Adapter-Brick triple"

Por favor, asegúrese de que sus sensores estén conectados correctamente al brick adaptador de sensores. Si no está seguro, consulte la documentación del sensor en cuestión. De lo contrario, existe peligro de daños irreversibles en el sensor y/o en el brick IdC!

6.6.4 Tipo de cambio del dólar desde Internet

Abra el sketch en el IDE de Arduino: Example_6.6.4.ino. Se deben incluir los siguientes archivos de cabecera: `ESP8266WiFi.h`, `SSD1306Wire.h`, `TimeLib.h`, `NtpClientLib.h` y `CurrencylayerClient.h`. Si aún no ha instalado las librerías correspondientes, vaya al capítulo 5.2.1 en la página 16 y hágalo.

Para este ejemplo también debe introducir primero su nombre WiFi (SSID) y la contraseña correspondiente, en el programa de ejemplo. Proceda como se describe en el capítulo 6.6.1.

Para obtener la cotización del dólar (EUR-USD) de Internet necesita una librería especial "Brick-ESP8266", que puede descargar en http://www.brickrknowledge.de/downloads . Si ya ha instalado todas las bibliotecas en el capítulo 5.2.1, puede continuar con el procedimiento. De lo contrario, primero debe instalar la librería como se describe en el capítulo 5.2.1.2.3 en la página 19.

Extracto del programa

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

if(counter%5000==0){      //Actualizar el tipo de cambio aproximadamente cada 5
segundos
   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();
```

En este ejemplo, obtenemos las cotizaciones actuales del dólar de Internet, para calcular cuántos euros tengo que pagar por un dólar (o viceversa).

$$EUR = USD * tipo\ de\ cambio$$

Para obtener el tipo de cambio, primero llamamos a la función currencylayer.get- LastChannelItem (). Los comandos JSON se utilizan para recuperar el tipo de cambio actual del dólar desde un servidor predefinido. El intervalo de tiempo para la actualización puede ser controlado por el contador en la sentencia if (counter%5000==0) La pantalla está formateada como de costumbre. Sin embargo, la llamada currencylayer. getFieldValue (0) es importante. Sólo en este punto el tipo de cambio del dólar se redondea a 4 decimales disponibles como un string en el sketch.

Además de la tasa de cambio actual, la pantalla también se complementa con la hora y la fecha, como ya se mostró en el ejercicio 6.6.2. La salida real de todos los valores a la pantalla OLED se realiza como de costumbre con la función display.display ().

Paralelamente, la salida también se encuentra en el monitor serie de su ordenador. Para abrir el monitor serial, simplemente haga clic en el icono de lupa del IDE Arduino (ver cap. 5.2.3 en la página 20). Se han añadido algunos mensajes de estado a la salida del ejemplo 6.6.2.

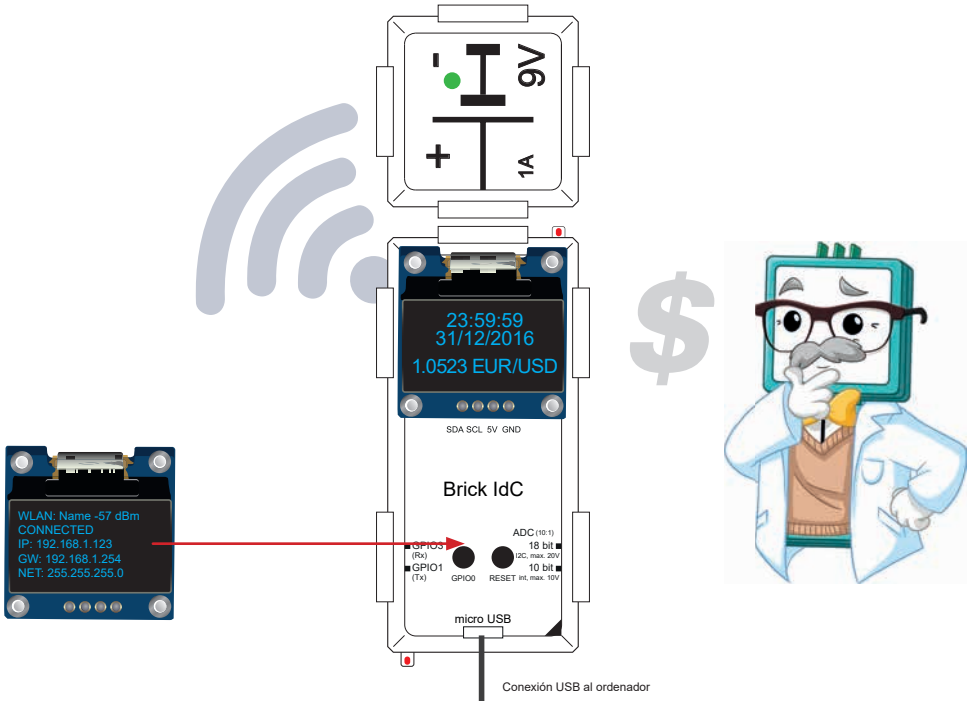


Fig. 47: Circuito brick del tipo de cambio del dólar de Internet

CONSEJO: El estado de conexión del brick IdC siempre se puede visualizar pulsando el botón GPIO0 (ver figura 43)!

6.6.5 Mi primera página web

Abra el sketch en el IDE de Arduino: Example_6.6.5.ino. Se deben incluir los siguientes archivos de cabecera: `ESP8266WiFi.h`, `SSD1306Wire.h`, `TimeLib.h`, `NtpClientLib.h`, `ESP8266WebServer.h` y `DHT.h`. Si aún no ha instalado las librerías correspondientes, consulte el capítulo 5.2.1 en la página 16 y hágalo.

Para este ejemplo, también debe introducir primero su nombre WiFi (SSID) y la contraseña correspondiente en el programa de ejemplo.

En este ejemplo, vamos a construir una página web sencilla que nos dará la hora, la fecha, la temperatura y la humedad. La base de cada sitio web es Hypertext Markup Language, abreviado: HTML. Puede utilizar un editor HTML WYSIWYG para escribir el código HTML de su primer sitio web. Puede descargar varios editores HTML gratuitos, por ejemplo, NVU para Windows, Linux (ver: www.nvu.com) o BlueGriffon para MAC OS X (ver: www.bluegriffon.org).

Los conceptos básicos de HTML son útiles para este ejercicio. En el proyecto web SELFHTML se puede encontrar información básica completa sobre HTML, en: www.selfhtml.org.

El circuito brick y el sketch de este ejercicio se basan en el ejemplo 6.6.3. El estado de conexión del brick IdC se puede visualizar en cualquier momento pulsando la tecla GPIO0 (véase también la Fig. 43)!

En primer lugar, juntar los bricks como se muestra en la fig.48, de lo contrario el sketch no se puede cargar correctamente. Asegúrese de conectar correctamente el sensor DHT11 suministrado. Inserte el sensor exactamente como se muestra en la fig.48, en el zócalo inferior de 5 polos del brick adaptador del sensor, en el extremo izquierdo. La etiqueta "5V" en el sensor debe coincidir con el zócalo con la etiqueta "5V" en el extremo izquierdo. De lo contrario, existe el riesgo de que se produzcan daños irreversibles en el sensor y/o en el brick IdC.

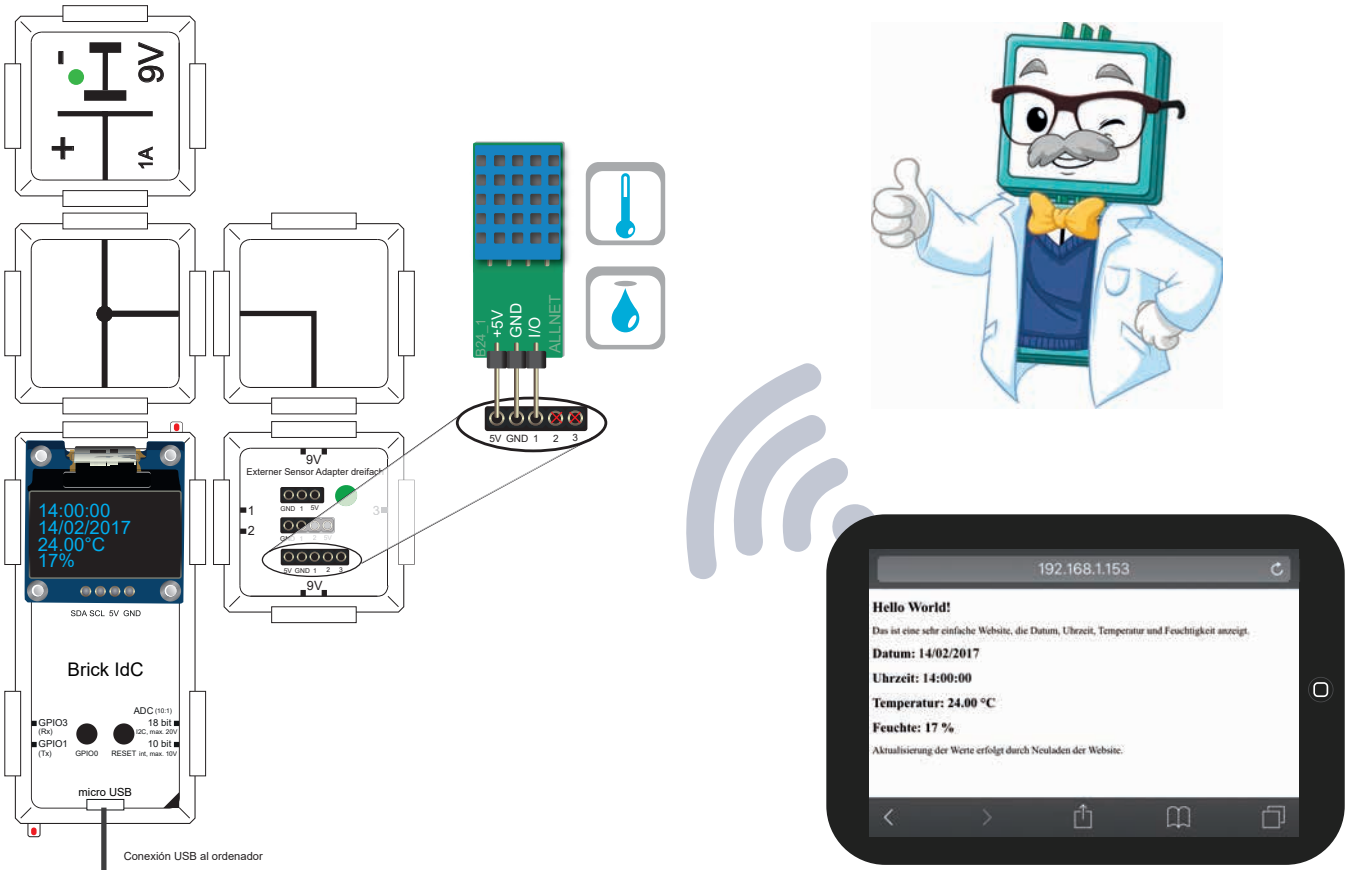


Fig. 48: Circuito brick "Mi primera página web"

Extracto del programa

```
...
ESP8266WebServer server(80); //Inicie el servidor web en el puerto 80
...
void setup() {
...
//cuando un navegador accede directamente al directorio root,
//ejecutar la función handleRoot (ver abajo).
    server.on("/", handleRoot);
    server.begin()    //a partir de ahora, el servidor "escucha" las peticiones HTTP
    Serial.println("HTTP server started");
}

void loop() {

    server.handleClient(); //gestionar la petición HTTP
...
}

//Con la función handleRoot() se suministra el sitio web
//en cuanto llega una solicitud de un navegador
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>";
 //la línea siguiente es importante para que el carácter de grado "°" se muestre
correctamente
  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> Este es un sitio web muy simple de su brick IdC que muestra
fecha, hora, temperatura y humedad.</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> Los valores pueden ser actualizados en cualquier momento recar-
gando el sitio web.</p>";
  content += "</body>";
  content += "</html>";
  server.send(200, "text/html", content);
```

Al principio del sketch, se inicia el servidor web y se asigna el puerto estándar 80 para las peticiones a través de HTTP (p. ej.: Http://www.brickrknowledge.com). En cuanto teclee la dirección local IP de su brick IdC en su navegador con el prefijo http: //, (por ejemplo http://192.168.1.153), accederá al directorio root de su servidor web para que se llame a la función handleRoot (). Como ya habrá leído en www.selfhtml.org, la estructura básica de cada sitio web consiste en las llamadas etiquetas HTML. Para cada elemento, normalmente hay una etiqueta de inicio y una etiqueta de fin (a reconocer por la barra oblicua "/"), que están incrustadas en los corchetes punteados característicos... Por ejemplo Normaler Absatz . En la función handle Root(), el contenido de toda nuestra página web se almacena en la variable string; content. Para que todo esté más

claro, la asignación del código HTML se extiende en varias líneas en el sketch. Esto incluye todas las líneas en el contenido del formulario += "....";

Otra dificultad es que en la programación de sketches, el principio y el final de un string están marcados con comillas en la parte superior ("). Sin embargo, dado que el código HTML también puede contener comillas, debemos utilizar la llamada Escape Sequence \ " para la comilla dentro de un string, para una codificación correcta. El propio string vuelve a empezar y finaliza con una comilla normal.

Por ejemplo, la línea ...

```
content += "<p>\"Este párrafo está entre comillas\"</p>";
```

…es igual al código HTML (UTF-8 codificado):

```
<p>"Este párrafo está entre comillas"</p>
```

Sin la barra invertida precedente en los dos lugares marcados en verde, la primera comilla marcada en verde cerraría el string - en lugar de la cuarta en la línea.

Finalmente, la página web se entrega con el comando server.send(200, "text/html", contenido). El primer parámetro define el código de estado HTTP, que se devuelve cuando la ejecución es exitosa. El código 200 significa OK (la solicitud ha sido procesada con éxito). A continuación, el tipo de contenido entregado se define como "Texto" y el tercer parámetro pasa el string de `content` con el contenido del sitio web.

6.6.6 Cambio a través de la página web

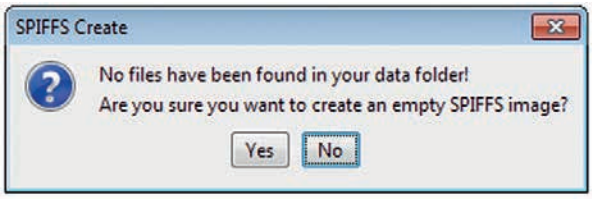
Abra el sketch en el IDE de Arduino: `Beispiel_6.6.6.ino`. Se deben incluir los siguientes archivos de encabezado: `ESP8266WiFi.h`, `SSD1306Wire.h`, `TimeLib.h`, `NtpClientLib.h`, `ESP8266WebServer.h` y `FS.h`. Si aún no tiene las librerías apropiadas instaladas, vaya al capítulo. 5.2.1 en la página 16 y haga esto primero.

Para este ejemplo, también debe introducir primero su nombre WiFi (SSID) y la contraseña correspondiente en el programa de ejemplo.Proceda como se describe en el capítulo 6.6.1.

Para completar los ejemplos de Internet de las cosas, aprenderá cómo activar una acción en el brick IdC o cómo controlar los pines GPIO. Como base para la programación del sitio web, esta vez necesitamos algo más que etiquetas HTML básicas. Para ello, utilizamos el marco de trabajo gratuito de CSS llamado Bootstrap, con el que se pueden crear diseños web con una respuesta relativamente sencilla, que están optimizados para dispositivos móviles (como su smartphone o tablet). Para más información sobre Bootstrap, visite https://www.bootstrapworld.de.

Para empezar con la programación del sketch, todavía tienes que trabajar un poco. Necesitamos tener acceso al sistema de archivos SPIFFS (ESP8266 File System) del brick IdC, para poder copiar archivos allí. Proceda de la siguiente manera:

1. Descarga el cargador de archivos de Arduino ESP8266 desde: https://github.com/esp8266/arduino-esp8266fs-plugin/releases El sketch ha sido probado con la versión 0.2.0 https://github.com/esp8266/arduino-esp8266fs-plugin/releases/download/0.2.0/ESP8266FS-0.2.0.zip
2. Extraiga el archivo ZIP en un directorio de su ordenador. El archivo Esp8266fs.jar que contiene está descomprimido por defecto en el subdirectorio \ ESP8266FS-0.2.0 \ ESP8266FS \ tool.
3. Compruebe si ya existe un directorio "tools" en su ordenador, en la ruta "Documents - Arduino" disponible. Si no es así, crea una nueva carpeta llamada "tools" Allí puedes instalar varias herramientas, que pueden ser operadas desde el IDE de Arduino, incluyendo el cargador de archivos ESP8266 que necesitamos.
4. Crear otra subcarpeta denominada "ESP8266FS" en el directorio "tools".
5. En el directorio "ESP8266FS" crear otra subcarpeta con el nombre "tool" (sin "s" al final).
5. Copiar el fichero esp8266fs.jar (ver punto 2) a la ruta "Documentos -Arduino - tools - ESP8266FS - tool ".
6. Reinicie el IDE de Arduino.
7. La opción "ESP8266 Sketch Data Upload" aparece en el menú "Herramientas".
8. Si tiene un proyecto compatible con el Sistema de Archivo SPIFFS, todos los archivos del subdirectorio "data" de la carpeta del proyecto (ruta estándar: "Documents - Arduino") se convierten a un formato binario y se cargan en la memoria flash SPI del módulo ESP8266.
9. Inicie la carga de archivos en el menú "Herramientas" con la opción "ESP8266 Sketch Data Upload". Si aparece el mensaje "SPIFFS Create" durante la primera carga, confirme con "Yes".



La carga de los archivos puede durar varios minutos!

En este ejemplo, conectaremos el brick doble LED a GPIO14 (LED rojo) y GPIO13 (LED amarillo) como se muestra en la figura. A través de dos botones en nuestra página web, podemos encender y apagar los LED, teóricamente en todo el mundo. La pantalla OLED nos muestra la hora y la fecha adicionales que ya conocemos de los ejercicios anteriores.

Fig. 49: Circuito brick "Cambio a través de la página web"

SUGERENCIA: El estado de conexión del brick IdC puede visualizarse en cualquier momento, pulsando el botón GPI00 (ver figura 43) !

Como ya se ha mencionado anteriormente, en este ejemplo también utilizamos tecnologías web avanzadas como el marco de trabajo bootstrap y varias clases CSS (CSS = Cascaded Style Sheets) y componentes Javascript (Javascript es un lenguaje de scripting de uso común en sitios web). Como resultado, el código es mucho más completo que los ejercicios anteriores, pero se basa en ellos. Por lo tanto, nos centraremos en los elementos esenciales cuando tratemos el sketch. Si tiene preguntas sobre el uso de Bootstrap, CSS y Javascript, le pedimos que busque en Internet o que compre un libro técnico apropiado, ya que esto iría más allá del alcance de este manual de Brick'R'knowledge.

CONSEJO:

Para todos aquellos que ya tengan conocimientos de HTML, Bootstrap, CSS y Javascript y quieran codificar la página web ellos mismos, hemos puesto el código HTML de la función handleRoot() como un comentario "en texto plano" y lo hemos copiado en el sketch. Para editar, puedes descargar varios editores HTML gratuitos como NVU para Windows, Linux o MAC OS X (ver: www.nvu.com) o BlueGriffon (ver: www.bluegriffon.org).

Para que este ejemplo funcione correctamente, necesita habilitar javascript en su navegador. Por lo general, éste es el procedimiento predeterminado.

Extracto del programa

```
...
ESP8266WebServer server(80); //iniciar el servidor web en el puerto 80
...
void setup() {
...
//Si el navegador accede directamente al directorio root,
//ejecutar la función 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);

    // El servidor web maneja las funciones de cambio GPIO
    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();
}
...
//Activar la función de control para GPIO13
 void handleGpio13On() {
  digitalWrite(gpio13, dOn);
  server.send(200, "text/html", gpio13Name+" on");
```

Continúa en la página siguiente...

En la Setup-Routine de nuestros sketches, las numerosas instrucciones server.on (), se utilizan para definir varias manijas que determinan qué funciones se ejecutan en cuanto se llaman ciertos enlaces en el navegador. Por ejemplo, si el brick IdC ha recibido la dirección IP 192.168.1.153 del servidor DHCP, la llamada a la dirección http://192.168.1.153/img/ brklogo.png hará que se ejecute la función handleImgLogo (), que a su vez lee el archivo de imagen brklogo.png desde el sistema de archivos interno (SPIFFS) y lo devuelve al servidor web. Al igual que el archivo de registro, cualquier script Java, CSS o archivo de imagen que esté almacenado en el sistema de archivos interno (SPIFFS) debe tener un gestor de archivos, para que podamos acceder a él. Además, se definen gestores con los que el servidor web responde cuando se recibe una petición del navegador para habilitar o deshabilitar un GPIO. A continuación, el servidor web se inicia con server.begin. Las propias manijas no se definen hasta el final del sketch. En el código de ejemplo anterior hemos seleccionado la función handleGpio13On() para encender el LED en GPIO13.

Extracto del programa (continuación)

```
...
void loop() {

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

//la función handleRoot () se utiliza para entregar la página web
//en cuanto llega una solicitud de un navegador
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
  }
}
```

En la manija de función Root (), - como ya se ha practicado en el ejemplo 6.6.5 - el contenido de nuestra página web, se almacena en el string variable `content` . Una vez más, debe tenerse en cuenta que en la programación de sketch, el inicio y el final de un string deben estar marcados con comillas en la parte superior ("). Debido a que el código HTML también puede contener comillas, dentro del string para la codificación correcta, se utiliza la llamada Escape Sequence \ " para cada comilla que aparece. Para una mejor comprensión, escribimos los fragmentos de código citados en el siguiente texto en "texto plano".

Con el elemento style `<style>body{background-Fig.:url(/img/bg.png;...` se integra la imagen de fondo de la página web y se definen otros parámetros que influyen en su apariencia. En el área del cuerpo se ven los elementos div que sirven para estructurar el sitio web. En un elemento Div, a su vez, se resumen varios elementos como texto, imagen, o formas y se controlan sus propiedades. Por ejemplo, el elemento Div `<div class="col-sm-2"><img class="img-thumbnail" src="/img/brklogo.png">` integra el logotipo.

Los botones para encender y apagar los LED se implementan con un elemento de entrada de tipo "checkbox, el formateo se realiza utilizando elementos de estilo del marco de trabajo bootstrap. Varios elementos de script al final del sketch, especifican la ruta relativa a los archivos javascript, a los que el sitio web necesita acceder. El elemento de script <script type = "text / javascript"> finalmente conduce a una sección más larga con código JavaScript, en la que onSwitchChange consulta el evento de pulsar un botón y llama a la función del mando para encender o apagar el LED.

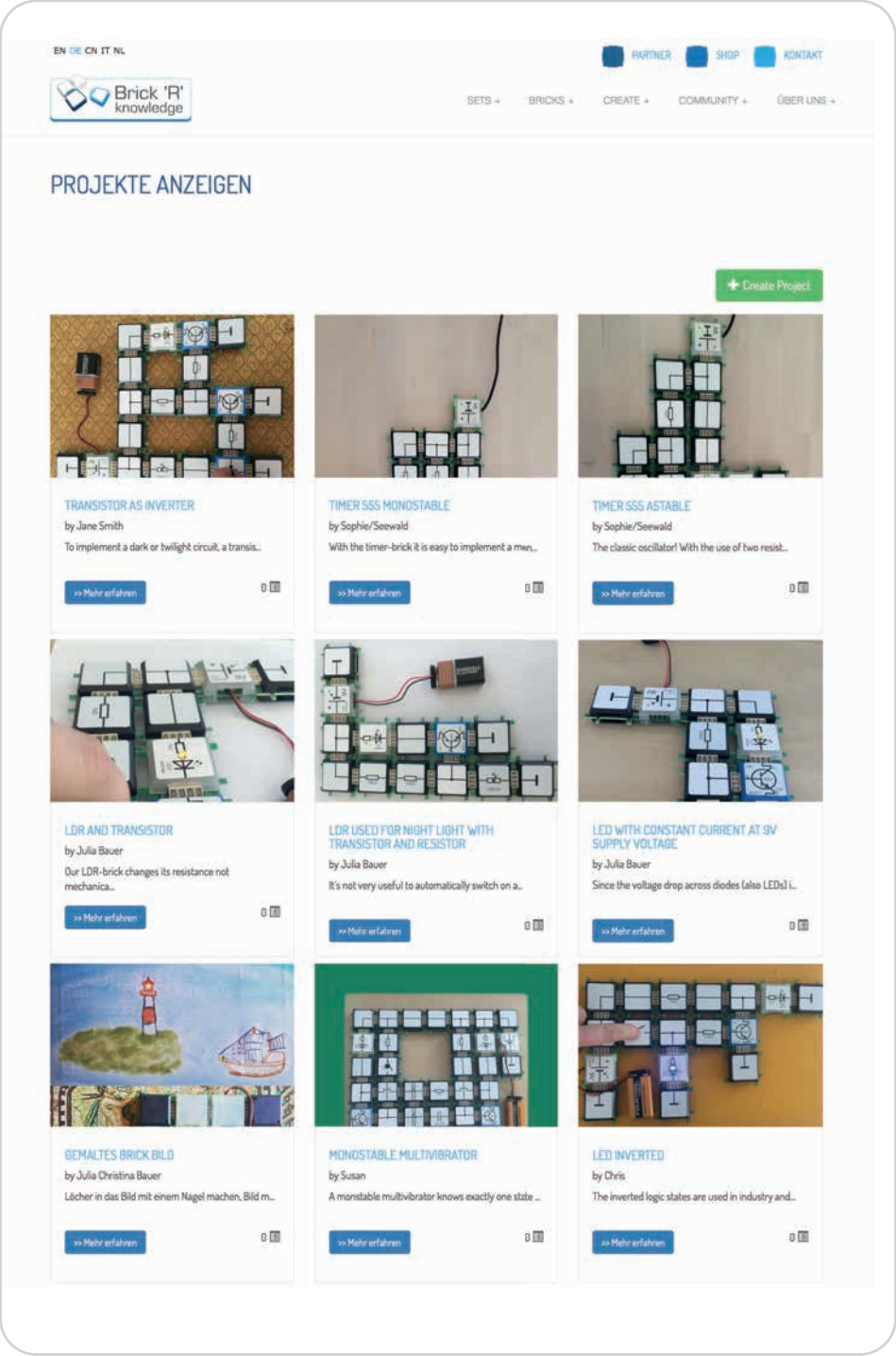
Con el comando server.send (200, "text/html", contenido) el sitio web se entrega dinámicamente al navegador.

7. Comunidad brick

El universo del brick se está expandiendo: encuentra más ideas, experimentos y bricks en ferias, en nuestra página web, en YouTube o en redes sociales. Impulsa tu creatividad!

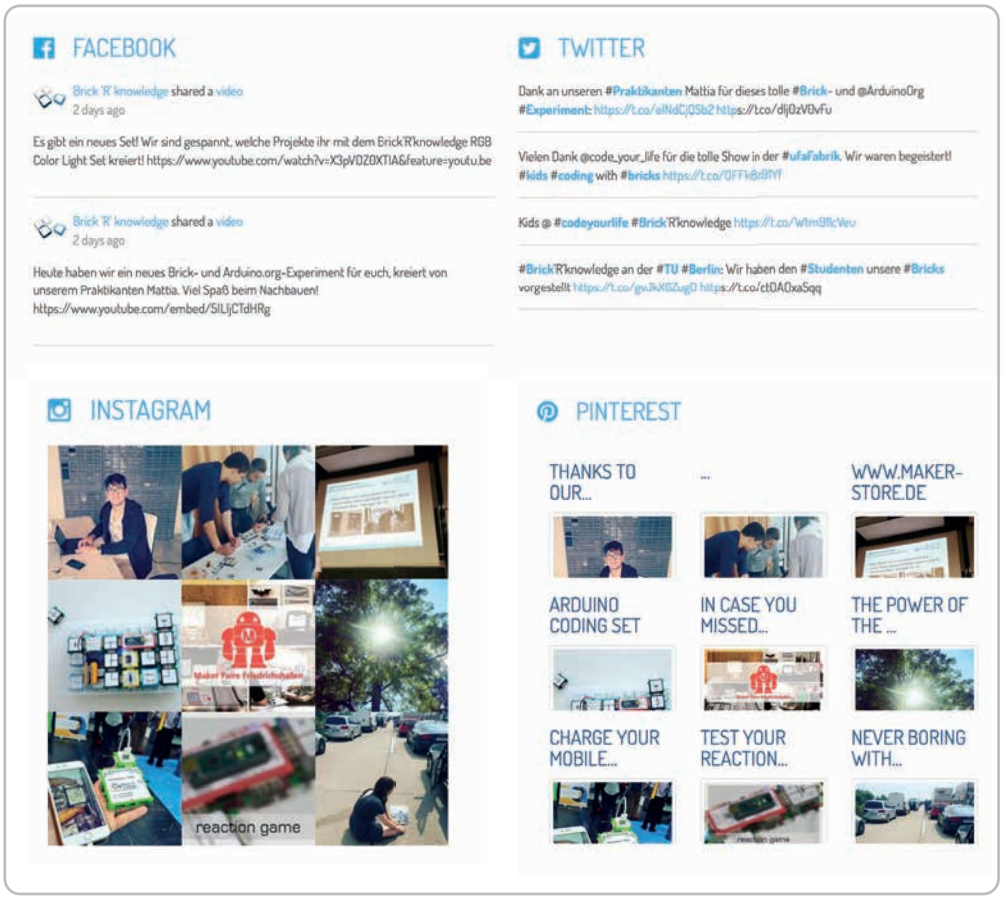
Más proyectos

Haciendo clic en "Crear" puedes probar experimentos de otros usuarios o mostrar tus propios circuitos.



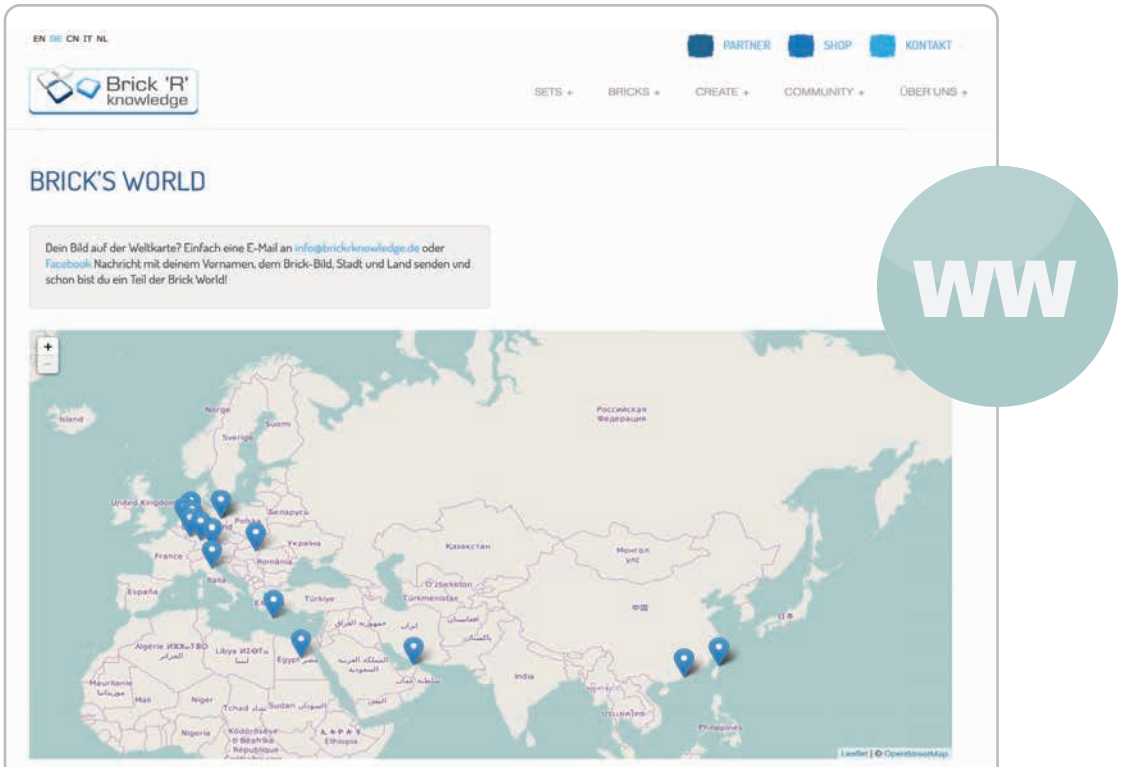
Redes sociales

Haciendo clic en "Comunidad" puedes encontrar todas nuestras redes sociales. Mantente al día!



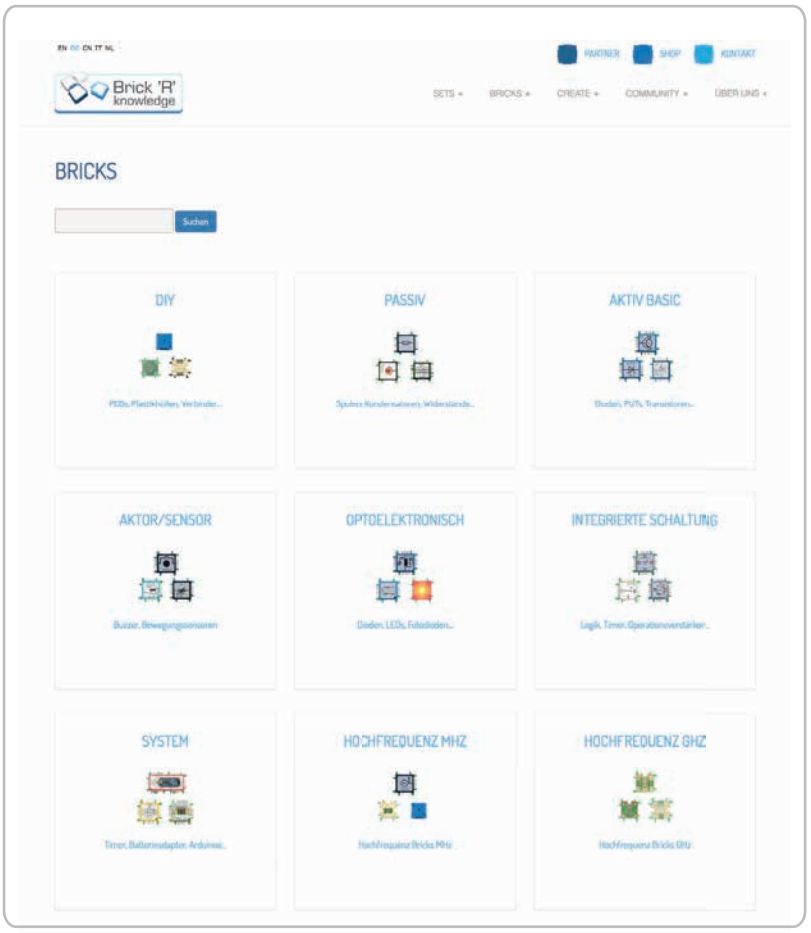
En todo el mundo

En "Comunidad" también puedes saber dónde están nuestros bricks. ¿Tienes una foto de bricks en tu ciudad? Envíenosla y pronto la encontrarás en nuestra página web!



Más bricks!

Haciendo clic en "Bricks" puedes encontrar todos los bricks disponibles con información e ideas para experimentos.



Blog

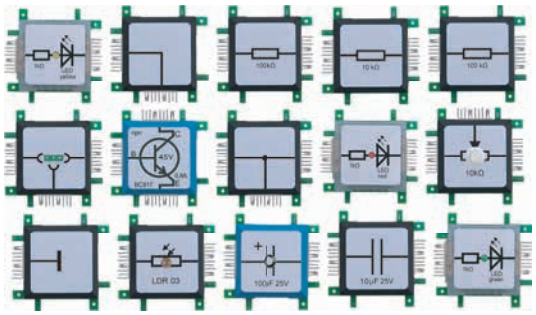
Cada semana subimos un nuevo post en el blog. Puedes leer sobre nuestras experiencias en ferias, nuevos circuitos, divertidas historias e información sobre el mundo de la electrónica.

8. Sets de bricks

Set Básico

ALL-BRICK-0374

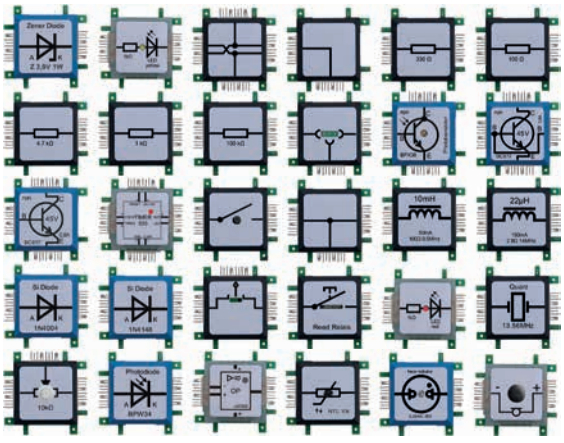
El set básico contiene 19 bricks seleccionados para ofrecer un fácil y rápido comienzo en el mundo de Brick'R'Knowledge, así como la posibilidad de crear numerosos circuitos. El set básico es perfecto para que los niños adquieran sus primeras experiencias con experimentos electrónicos y técnicos.



Set Avanzado

ALL-BRICK-0223

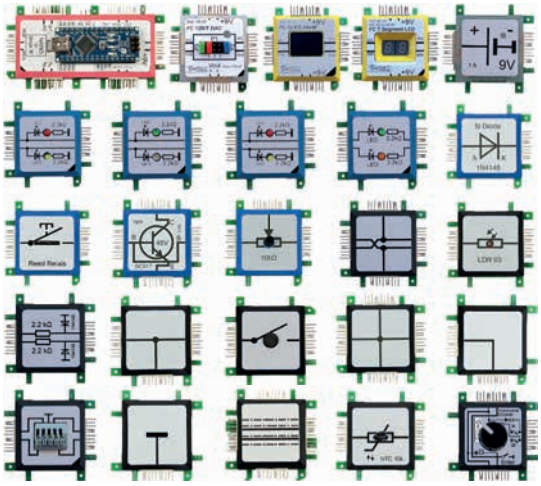
Nuestro set avanzado contiene 111 componentes que te permiten desarrollar soluciones más complejas y complicadas. Gracias al sistema educativo, el conocimiento puede ser recopilado, de manera que no sólo vosotros sino que también nuestra próxima generación pueda beneficiarse de ello. Puedes construir circuitos individuales conectando bricks diferentes juntos. Simple a la vez que complejo, se puede experimentar con temas electrónicos y tecnológicos de una manera totalmente nueva. A través del factor open -source, puedes crear tus propios bricks y desenvolupar tus propias soluciones. El Brick'R'knowledge no se trata solamente sobre electrónica básica, también se pueden realizar experimentos de RF, lo que lo convierte en un sistema único a nivel mundial.



Set de codificación de Arduino

ALL-BRICK-0414

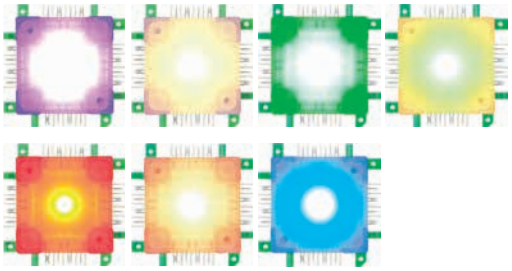
Entra en contacto con la electrónica digital y empieza a entender la programación con Arduino® Nano, que se incluye en el kit. Es nuestro primer set con componentes digitales, como pantallas de 7 segmentos, display OLED, convertidor D/A o bricks I2C , complementario a todos los bricks analógicos. Para empezar con el popular microcontrolador, te ayudamos con varios ejemplos de programación.



Set de luces de 7 colores

ALL-BRICK-0398

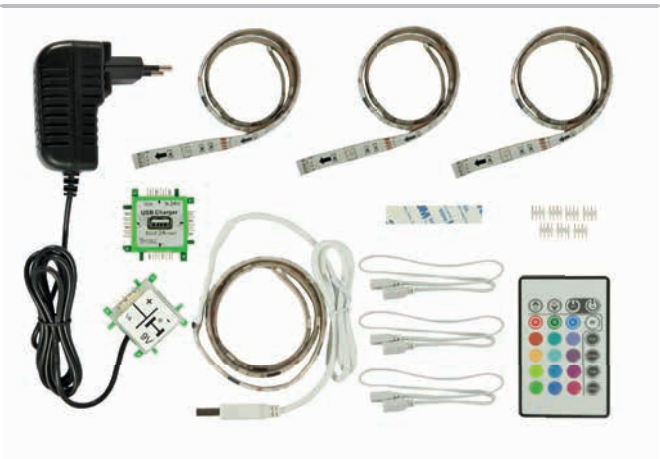
El 7 Color Light Set contiene 28 bricks LED en 7 colores diferentes para crear impresionantes efectos de luz en una arquitectura horizontal y vertical. Los LED rojos, amarillos, azules, naranja, violetas, verdes y blancos cálidos de 1 vatio son perfectos para la iluminación individual o como solución a la iluminación móvil.



Set RGB de luces de colores

ALL-BRICK-0619

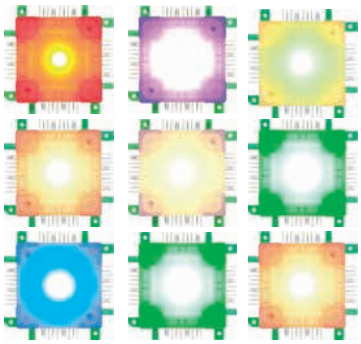
Crea tu show de luces! El set RGB de luces de colores viene con cuatro tiras LED flexibles que contienen un total de 36 LED, los cuales se pueden controlar con el mando a distancia por infrarrojos incluido. Puedes pegar, cortar y conectar las tiras de LED como quieras. El mando a distancia por infrarrojos tiene 16 teclas de colores diferentes y 4 programas de luz.



Set de LED programable

ALL-BRICK-0483

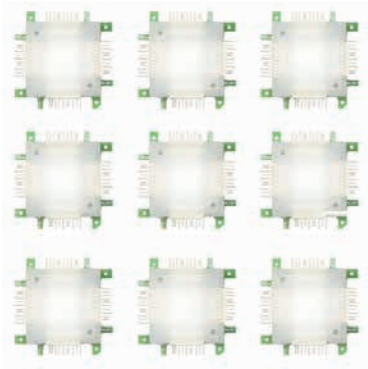
El kit contiene 49 bricks LED RGB programables y controlables, cada uno con dos o tres conectores y un brick de conjunción para la gestión de Arduino y la fuente de alimentación. Además, el conjunto de LED programables de Brick' R' knowledge incluye un brick adaptador de Arduino y un Arduino Nano. Con este set se pueden realizar animaciones de LED coloridas y otras ideas individuales. Y lo mejor de todo: con la realización de diferentes proyectos se puede aprender fácilmente la programación de micro-controladores.



Set del LED de alta potencia

ALL-BRICK-0399

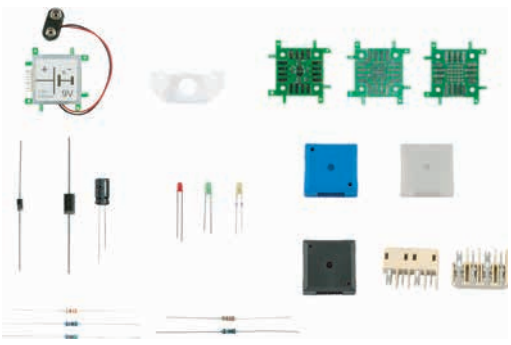
Potenciados por 1 vatio, cada uno de los 50 bricks LED de alta potencia incluidos en el kit irradian toda la zona circundante en blanco brillante. Construye soluciones individuales en cada arquitectura imaginable e inventa luces de noche Brick, lámparas de mesa Brick o cualquier otro iluminante creativo. La fuente de alimentación con 12V 8A soporta la intensa luminosidad para ofrecer una atmósfera elegante y acogedora. El High Power LED Set 50 permite tratar con un diseño de luz moderno y al mismo tiempo aprender sobre electrónica.



Set DIY

ALL-BRICK-0397

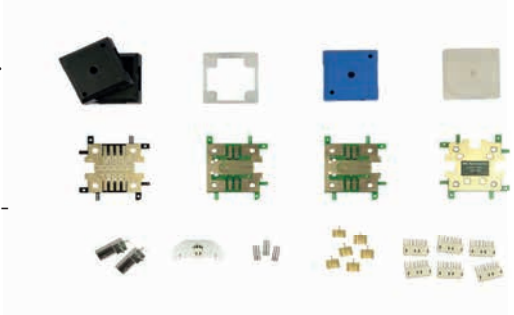
El set de bricolaje va un paso más allá. Los componentes incluidos ofrecen una visión mucho más detallada de la arquitectura del brick y permiten incluso la producción de bricks individuales. El kit de bricolaje ofrece una enorme flexibilidad para la generación del fabricante o para las personas que crean bricks individuales.



Set DIY de MHz

ALL-BRICK-0457

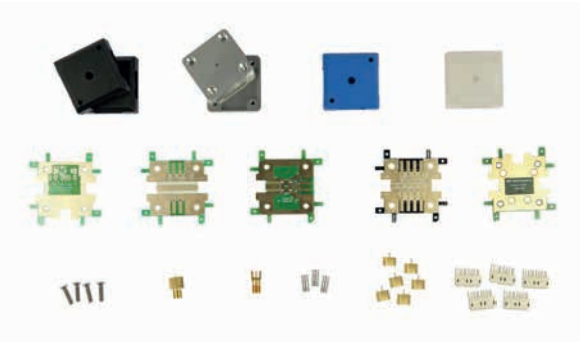
Se pueden crear proyectos individuales desafiantes dentro del rango de frecuencias de MHz con el sistema DIY de MHz. Tres rejillas diferentes y paneles de experimentación, tomas BNC, enchufes P-SMP y los conectores necesarios hacen que el kit sea perfecto para cualquier experimento de alta frecuencia. El set contiene conectores hermafroditas y una plantilla de soldadura para que los conectores SMD desarrollen sus propios bricks u otros componentes para el sistema Brick.



Set DIY de GHz

ALL-BRICK-0458

Realice experimentos avanzados y complicados en el rango de alta frecuencia hasta frecuencias de GHz.
El kit ofrece cuatro placas de circuito impreso diferentes, P-SMP, tomas SMA, conectores P-SMP y conectores hermafroditas específicos del brick. El conjunto DIY de GHz es perfecto para operadores de radio HAM y aficionados a la medición.



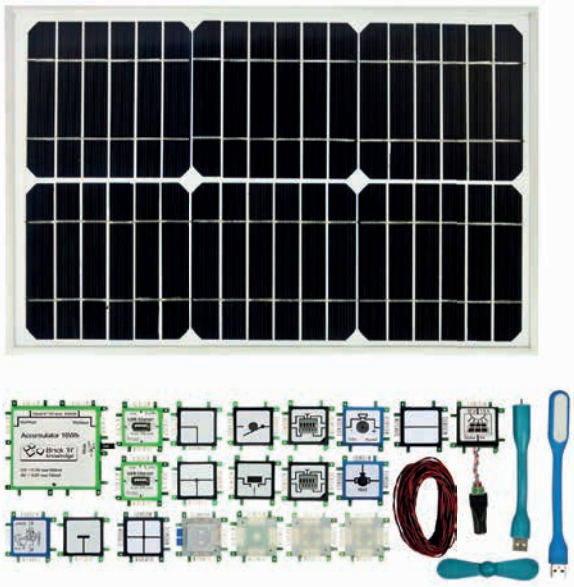
Set solar

ALL-BRICK-0484

El set solar garantiza la diversión de experimentar en familia y acerca a los niños a las energías renovables de una manera lúdica.

- ¿Cómo funciona una célula solar?
- ¿Cómo almacena la energía una batería?
- ¿Cómo construir una luz nocturna con detector de movimiento?

El set solar proporciona respuestas a estas y otras preguntas. Con este set os convertís en miembros oficiales de la generación Maker!



Set de medición 1

ALL-BRICK-0637

El set de medición 1 permite medir la tensión, la corriente y otras magnitudes con instrumentos de medición estándar. El set contiene adaptadores de medición (3x2mm), con abrazadera de cable adicional y adaptadores de medición (4mm) con un punto final amarillo.

Set de medición 2

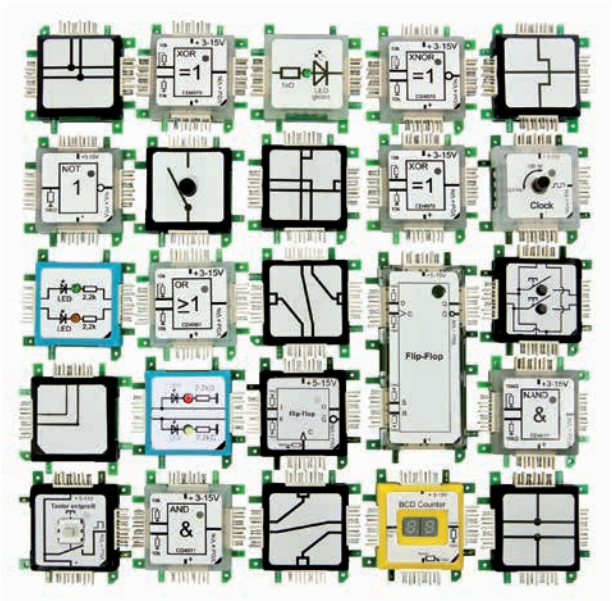
ALL-BRICK-0638

El set de medición 2 permite medir la tensión, la corriente y otras magnitudes con instrumentos de medición estándar. Contiene adaptadores de medición (4mm) con extremo cerrado GND , adaptadores de medición (4mm) rojo en línea y adaptadores de medición (4mm) con extremo abierto GND negro.

Set Lógico

ALL-BRICK-0630

El set lógico es ideal para un inicio rápido en la tecnología de circuitos digitales. Mientras trabajan con el manual, que incluye ejemplos didácticos estructurados de circuitos, los estudiantes aprenden sobre los circuitos digitales más importantes como el sumador, el registro de desplazamiento y el numerador. El set lógico ampliamente equipado proporciona a los profesores una base práctica para la enseñanza diaria. Conectar los bricks y experimentar con ellos es divertido y anima a construir tus propias variantes de circuito. El set lógico abarca desde bricks lógicos fáciles (AND, OR, NAND, NOR, XOR, XNOR, NOT), a una variedad de bricks flip flop (tipo D-, RS- y JK-), a un brick de impulso (alternativamente un interruptor de supresión de rebotes para pulsos individuales) hasta un brick de contador BCD con un display integrado de 7 segmentos. Una amplia gama de bricks LED, interruptores y bricks de cables completan el set.



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