

# IoT-Innovationswerkstatt



lotwerkstatt.umwelt-campus.de



Umwelt-Campus  
Birkenfeld

H O C H  
S C H U L E  
T R I E R

<b>IoT-Werkstatt.....</b>	<b>2</b>
Entwicklungsumgebung .....	4
Arduino-Plattform .....	4
IoT-Entwicklungskit (ESP-8266) .....	4
Grafische Programmieroberfläche Ardublock .....	6
Internet / Intranet im schulischen Umfeld.....	8
<b>Blaupausen für eigene Projekte .....</b>	<b>10</b>
Hallo Arduino - Hallo IoT-Kit .....	13
Die digitale Flaschenpost – Hallo Server .....	19
Unser Forschungsschiff – Hallo IoT .....	22
Forschungsschiff informiert sich – Hallo M2M .....	27
Forschungsschiff greift ein - Hallo Control.....	28
Wearables – Forschungsschiff kleidet sich .....	34
Das Forschungsschiff hört aufs Wort.....	36
Die Flaschenpost wird flexibel – Hallo Interface .....	39
<b>Anhang.....</b>	<b>44</b>



Der Text dieses Artikels steht unter der [Creative Commons Namensnennung-Weitergabe unter gleichen Bedingungen 3.0 international](https://creativecommons.org/licenses/by-sa/4.0/)

### IoT-Werkstatt

Klaus-Uwe Gollmer, Guido Burger

Umwelt-Campus Birkenfeld der Hochschule Trier, Expertengruppe IoT des Nationalen Digital Gipfels

Die digitale Transformation wird unsere Wirtschaft und Gesellschaft in den nächsten Jahrzehnten grundlegend verändern. Der selbstverständliche Umgang mit Sensoren und Kommunikationsmodulen, aber auch deren Programmierung bis hin zur Cloud-Anwendung ist eine Voraussetzung für neue Anwendungsideen und Geschäftsmodelle.

Das dazu notwendige **algorithmische Denken** sollte deshalb zukünftig schon in der Ausbildung geübt werden. Hierzu stellen wir mit der **IoT-Werkstatt**<sup>1</sup> beispielhafte Ideen und ihre technische Umsetzung vor. Fokussiert auf ein schnelles Erfolgserlebnis bieten die kleinen Programme einen Kristallisationspunkt für die Beschäftigung mit dem Thema und der kreativen Umsetzung eigener Ideen.

Im Wettbewerb mit mehreren Teams (z. B. bei einem Hackathon) macht es noch viel mehr Spaß: es entstehen konkurrierende Ideen, aber vielleicht auch ganz neue Lösungen, da man mit anderen Teams zusammenarbeiten kann. Durch die vorbereiteten Beispiele gelingt der Einstieg kinderleicht. Ein für speziell für die IoT-Innovationswerkstatt entwickeltes IoT-Board in Verbindung mit einer angepassten grafischen Programmieroberfläche, sowie den Grove Interfacekomponenten erleichtern die praktische Umsetzung auch ohne Programmierkenntnisse. Eine Cloud-Lösung auf der Basis des Raspberry Pi stellt lokale Server-Dienste zur Verfügung. Für erste Experimente mit der IoT-Werkstatt ist kein Internetzugang notwendig.

Durch einfache Modifikation der Beispiele dieser Blaupause entstehen daraus komplexere Lösungen für die kleinen Probleme des Alltags:

---

<sup>1</sup> [http://deutschland-intelligent-ernetzt.org/app/uploads/2017/06/Positionspapier\\_IoT-Werkstatt\\_EG\\_M2M.pdf](http://deutschland-intelligent-ernetzt.org/app/uploads/2017/06/Positionspapier_IoT-Werkstatt_EG_M2M.pdf)

## Das Internet der Dinge anfassbar machen

- **Umwelt:** Realisiere ein Citizen-Science-Projekt zur Messung von Feinstaub, Luftqualität in der Stadt oder erfasse das Mikroklima in der Landwirtschaft. Sorge dafür, dass die Bewohner von A-Bach und B-Tal in Zukunft rechtzeitig vor einem folgenschweren Unwetter gewarnt werden. Entwerfe einen geeigneten Umweltsensor (z. B. zur Messung der Bodenfeuchtigkeit am Berghang, des Pegels des nächsten Flusses usw.) und verknüpfe die Sensordaten mit dem Wetterbericht.
- **Smart-City:** Löse Logistikprobleme z. B. smarte Mülleimer, die von sich aus die Müllabfuhr rufen, Laternen, die freie Parkplätze melden, oder Briefkästen, die den Einwurf zeitgenau aufs Smartphone posten. Reduziere so das Verkehrsaufkommen und die Umweltbelastung durch Schadstoffe.
- **Smart Home:** Integriere eigene Komponenten ins intelligente Haus, optimiere die Kellerlüftung oder nutze Alexa, um die Kaffeemaschine zu steuern.
- **Tourismus:** Messe die Nutzungsfrequenz von Wander- und Radwegen, beobachte die Verweilzeiten an Aussichtspunkten oder Mitmachstationen.

Anhand dieser exemplarischen Beispiele fallen einem sicher viele weitere nutzbringende Anwendungsideen ein, die wir gemeinsam umsetzen.

Das System ist als **digitale Flaschenpost** konzipiert und trägt damit als Metapher die Nachricht des einfachen Internetzugangs für die Dinge dieser Welt in die Politik, an die Schulen und an die Hochschulen. In einem zweiten Schritt schicken wir die Flasche als **Forschungsschiff** auf die Reise, um unsere Umwelt zu erkunden und neue digitale Welten zu erobern. So verbindet sich das Internet mit der analogen Welt in der wir leben.

## Entwicklungsumgebung

Die folgende Kurzeinführung soll den Zugang zu Werkzeugen, Technologie und Kreativitätsmethoden erleichtern. Zuerst wird die Arduino-Entwicklungsumgebung und die Hardware vorgestellt, um anschließend anhand einfacher Beispiele in die Welt des **Internets der Dinge** (IoT) einzudringen.

### Arduino-Plattform

Mit der Open-Source Arduino-Entwicklungsumgebung steht eine in vielen Projekten bewährte Plattformtechnologie zur freien Verfügung. Softwareseitig bildet der GNU-C-Compiler den Kern einer einfach zu bedienenden Entwicklungsoberfläche.<sup>2</sup>

### IoT-Entwicklungskit (ESP-8266)

Hardwareseitig sind unzählige kompatible Entwicklungsboards verschiedener Hersteller für wenig Geld verfügbar. Seit kurzem existiert mit dem ESP-8266 eine Plattform<sup>3</sup>, die sich aufgrund des niedrigen Preises und der WLAN-Konnektivität ideal für Ausbildungszwecke eignet. Kommerziell erhältliche Entwicklungsboards, wie NodeMCU<sup>4</sup>, enthalten alle für die Programmentwicklung und Ausführung notwendigen Komponenten inklusive USB und WLAN. Je nach Anwendungsfall werden die benötigten Ein- /Ausgabekomponenten, wie LED, Taster, Sensoren und Aktoren manuell hinzugefügt – ein im schulischen Alltag sehr fehleranfälliges Unterfangen.

Hier reduziert das von uns speziell für das Rapid Prototyping entwickelte IoT-Kit die Verdrahtungsproblematik durch eine Reihe bereits auf dem Board integrierter Komponenten.

---

<sup>2</sup> [arduino.cc/en/Main/Software](http://arduino.cc/en/Main/Software)

<sup>3</sup> [mikrocontroller.net/articles/ESP8266](http://mikrocontroller.net/articles/ESP8266)

<sup>4</sup> [nodemcu.com](http://nodemcu.com)

## Das Internet der Dinge anfassbar machen

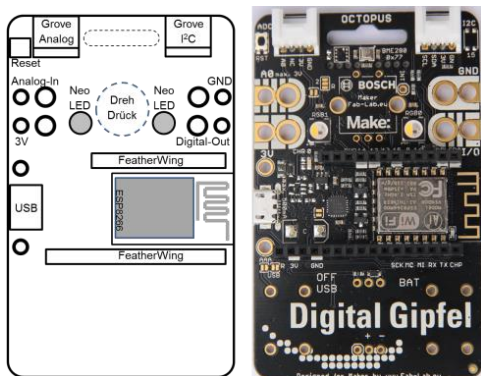


Abb. 1: Das IoT-Experimentier-Kit

Das **Octopus IoT-Experimentier-Kit** legt damit als quelloffenes Projekt (Schaltplan im letzten Kapitel) die Blaupause für die Ausbildung an Schulen und Hochschulen. Produziert und entwickelt in Deutschland folgt es folgenden Prinzipien:

- **Kein Löten:** Erweiterungen werden einfach angesteckt, und sind durch eine Vielzahl von Sensoren und Aktoren flexibel an die eigene Problemstellung anpassbar.
- **Sensorik:** ausgestattet mit einem Bosch-Sensortec Umweltsensor (Barometer, Luftfeuchtigkeit, Temperatur) wird das Internet der Dinge in wenigen Minuten erlebbar.
- **Interaktiv:** durch „Dreh/Drück“ Eingabe und unterschiedliche Anzeigen (farbige NeoPixel, einfarbige LED).
- **Flexible Energieversorgung:** über USB-Schnittstelle oder unter Verwendung von Standard Batterien bzw. Akkus. Das Laden via Solarstrom ist als „hack“ vorgesehen.
- **Kompatibel:** zur Schulausstattung (Bananenstecker, WLAN und USB) und vorhandenen Communities mit vielen Anwendungsbeispielen (Arduino, Adafruit, SeedStudio Grove).

Im Rahmen der Veranstaltungen ist das Board ohne weitere Konfiguration direkt einsetzbar, die Entwicklungsumgebung bereits

vorinstalliert, eine schlüsselfertige Variante für den heimischen Rechner kann per ZIP-File bezogen werden<sup>5</sup>.

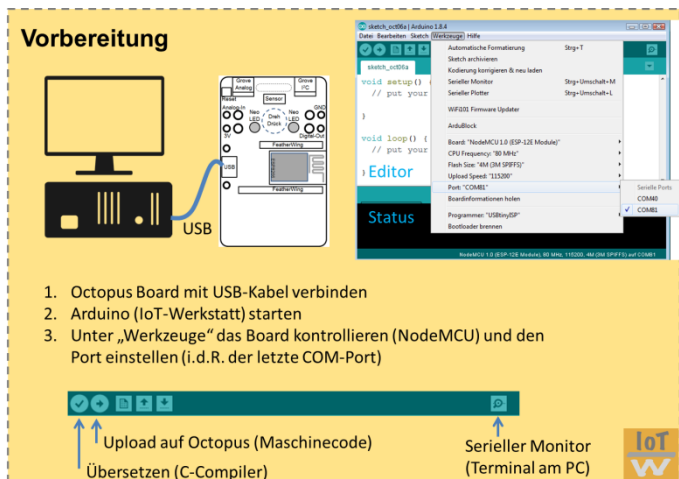


Abb. 2: Vorbereitung Entwicklungsumgebung

## Grafische Programmieroberfläche Ardublock

Insbesondere Programmierneulinge kämpfen ständig mit der Syntax der jeweiligen Programmiersprache. Die Schreibweise einzelner Befehle muss in der Regel mühselig im Handbuch nachgeschlagen werden und die Anordnung von Klammern, Blöcken, Semikolon usw. erscheint auf dem ersten Blick sehr verwirrend. Hier hilft der Einsatz einer grafischen Programmiersprache (wie z. B. Scratch<sup>6</sup>), bei der einzelne Programmblöcke per Mausklick zu einem Programm zusammengepuzzelt werden. Passende Schloss/Schlüssel-Mechanismen sorgen automatisch für ein syntaktisch korrektes Programm. Glücklicherweise gibt es auch für

<sup>5</sup> Nähere Informationen und Link im Dokument Quickstart.pdf

<sup>6</sup> <https://scratch.mit.edu/>

## Das Internet der Dinge anfassbar machen

den Arduino eine entsprechende Erweiterung in Form der Oberfläche Ardublock<sup>7</sup> bzw. der von uns entwickelten IoT-Erweiterung<sup>8</sup>.



Abb. 3: Oberfläche Ardublock zu grafischen Programmierung

Hier existieren neben einfachen Blöcken zur Realisierung von Kontrollstrukturen, Variablen, logischen bzw. mathematischen Operationen auch spezifisch auf das IoT-Kit zugeschnittene Module, die wir im Laufe der Blaupausen kennen lernen werden. Diese grafischen Blöcke besitzen spezifische Andockstellen (Schloss/Schlüssel), die sicherstellen, dass nur kompatible Blöcke zu einem Programm zusammengestellt werden können. Aus der grafischen Darstellung wird über den Menüpunkt „Hochladen auf den Arduino“ direkt der Hochsprachencode (C/C++) in Arduino-Syntax generiert und auf das Board geladen. Eine Übersicht der Code-Bausteine findet sich im Anhang.

<sup>7</sup> <http://blog.ardublock.com/>

<sup>8</sup> <http://div-konferenz.de/events/hackathon/?view=1>

## Internet / Intranet im schulischen Umfeld

Das Internet der Dinge erfordert systembedingt natürlich den Zugriff auf die Infrastruktur eines Netzwerks. In Deutschland verfügen die meisten Haushalte mittlerweile über einen Breitbandzugang mit WLAN-Accesspoint zur drahtlosen Kommunikation, so dass sich IoT-Anwendungen im häuslichen Umfeld problemlos realisieren lassen. Anders sieht die Sache im schulischen Bereich aus. Aus verschiedenen Gründen ist dort flächendeckendes WLAN für alle oft nicht erwünscht. Für solche Fälle bietet die Open-Source-Community handhabbare Ersatzlösungen. Die Expertengruppe empfiehlt dort den Einsatz eines eigenen Cloud-Servers im Intranet. Selbst ein einfacher Raspberry Pi ist heute in der Lage, ein eigenes WLAN-Netz aufzuspannen und unsere IoT-Kits mit der notwendigen Infrastruktur zu versorgen. Sowohl die später noch thematisierte Cloud-Anwendung Thingspeak, als auch ein MQTT-Broker sind als Open-Source-Anwendungen frei verfügbar.

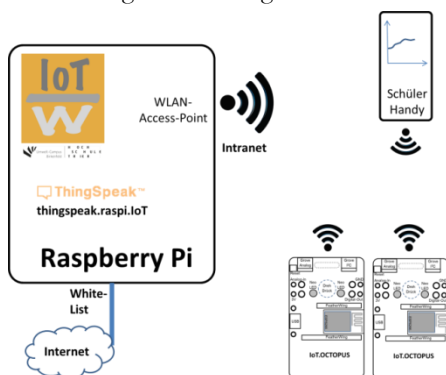


Abb. 4: Ein Raspberry-Pi ersetzt / filtert den Internetzugang

Auch wenn die Schule über eine WLAN-Versorgung verfügt, erscheint die Anwendung einer lokalen Cloud-Lösung durchaus vorteilhaft. Sie ermöglicht die Nutzung der Dienste ohne vorherige Anmeldung bei externen Servern und damit einen sparsamen



## Das Internet der Dinge anfassbar machen

Umgang mit den persönlichen Daten. Alle Messdaten und Anmeldeinformationen verbleiben somit innerhalb der eigenen Infrastruktur. Nachteil ist natürlich, dass diese Dienste dann auch nur im Intranet der Schule abrufbar sind. Sollen einzelne Messdaten auch extern verfügbar abgelegt werden, so können natürlich Schnittstellen zwischen Intranet und Internet konfiguriert werden. Ein entsprechend vorkonfiguriertes Image für den Raspberry steht bereit. Das Image der IoT-Werkstatt bringt dafür einen eigenen DNS-Server mit, der alle Anfragen der Domain .IoT lokal behandelt, andere Verbindungen aber an einen externen DNS-Server weiterleitet. Dabei besteht die technische Möglichkeit, im WLAN nur bestimmte Internet-Dienste zu zulassen.

## Blaupausen für eigene Projekte

Nach Einrichtung der Entwicklungsumgebung können wir jetzt mit der Umsetzung der Beispielprojekte beginnen. Wir starten mit einem einfachen „**Hallo Arduino**“ um uns mit der Bedienung der Entwicklungsplattform vertraut zu machen. Anschließend werden verschiedene Anwendungsszenarien erläutert:

- Im Projekt „**digitale Flaschenpost**“ senden wir eine Nachricht an unseren Flaschenpost-Server – hierdurch lernen wir die Anbindung an das Internet, nutzen eine WLAN Verbindung und eine einfache HTML-Seite zur Eingabe der Nachricht, die Ausgabe der Nachricht nutzt eine Matrixanzeige.

**Ziel:** Wir lernen eine Kommunikationsstrecke aufzubauen, welche wir für unser späteres Forschungsschiff benötigen!

**Mehr:** Sicherheit ist für IoT wichtig, am Hackathon schauen wir bei den anderen Teams vorbei, welche IP Adresse nutzen die? Können wir Nachrichten an deren Flasche senden?

- Im Projekt „**Forschungsschiff**“ senden wir Daten von Sensoren in das Internet und visualisieren diese. Dabei nutzen wir wieder eine WLAN-Verbindung und ein einfaches REST (Representational State Transfer) basiertes Protokoll. **Ziel:** Wir können eine erste Datenauswertung machen: Wie entwickelt sich die Luftfeuchtigkeit? Wovon ist diese abhängig (z. B. Temperatur, Sonnenstand)?

**Mehr:** Welche Sensoren können das Forschungsschiff ergänzen? (Tabelle am Ende des Kapitels), wo könnte das Forschungsschiff ausgesetzt werden? (Tipp: Es muss nicht im Wasser unterwegs sein: Auto, Tasche, Fahrrad, ...).

- Im Projekt „**Forschungsschiff informiert sich**“ holen wir Wetterinformationen für unseren Standort aus dem Internet.

**Ziel:** Technische Systeme kommunizieren untereinander.

**Mehr:** Welche weiteren nützlichen Informationen werden angeboten (Zugverspätungen, Staumeldungen, Aktienkurse). Welche Alternativen zum WLAN gibt es (UMTS, LoRa)?

## Das Internet der Dinge anfassbar machen

- Im Projekt „**Forschungsschiff greift ein**“ schauen wir uns eine komplexere regelungstechnische Anwendung an. Wir nutzen Regeln (Wenn Das Dann Tue Dies) und binden ohne Programmierung unser Smartphone in die Anwendung ein.  
**Ziel:** Nun geht es um den Mehrwert der Idee, wie kann ich diese im Alltag nutzen, wie kann ich diese an meine Bedürfnisse anpassen (was soll passieren, wenn die Luftfeuchtigkeit im Keller zu hoch ist?)  
**Mehr:** Wie kann ich weitere Dienste des Internets einbinden? (Twitter: wenn das Eichhörnchen eine Nuss aus der Futterstation entnommen hat).
- Im vorletzten Projekt „**Wearables**“ verwandeln wir die Flaschenpost in ein tragbares Kleidungsstück.  
**Ziel:** Mobile Verwendung.
- Den Abschluss bildet ein komplexes Szenario unter Verwendung von Amazons Alexa als Spracheingabe.  
**Ziel:** Mehrwert verteilter Systeme erkennen.

Ihr habt sicher ganz viele tolle Ideen ... ran an die Arbeit!

Die folgende Tabelle gibt einen zusammenfassenden Überblick über unsere Blaupausen und zeigt die dazu verwendeten Sensoren und Aktoren. Erweiterungen sind dank Grove<sup>9</sup>-System problemlos möglich. Zusätzliche Sensoren finden sich in Tabellen 3 bis 5 am Ende des Kapitels.

---

<sup>9</sup> <https://www.exp-tech.de/seced-grove-wiki>

## 12 IoT-Innovationswerkstatt Rheinland-Pfalz

**Tab. 1: Welche Interfacekomponenten werden genutzt?**

Blaupause	Internet	LED Dreh/ Drück	BME 280	LED- Matrix	Grove Relais (Digital)	Grove Sensor (Analog)
11.2.1 Hallo IoT-Kit		X	X			
11.2.2 Flaschenpost - Hallo Server	Server			X		
11.2.3 Forschungsschiff Hallo IoT	Client Thingspeak	Neo	X	(X)		
11.2.4. Forschungsschiff Hallo M2M	Client M2M-API yahoo	Neo		(X)		
11.2.5 Forschungsschiff Hallo Control	Client IFTTT	Neo			Pumpe	Feuchte
11.2.6. Wearables		Neo		(X)		Loudness





## Hallo Arduino - Hallo IoT-Kit

Zur Erstellung unseres ersten Programmes starten wir die Arduino-Oberfläche und schauen uns kurz die einzelnen Komponenten an:



Abb. 5: Elemente der Arduino - IDE

Der Editor dient der Programmeingabe in C/C++ Hochsprache und unterstützt den Programmierer durch verschiedene farbliche Hervorhebung von Befehlen und Kommentaren.

Im Bedienfeld besteht die Möglichkeit, das Programm zu speichern , in Maschinencode zu übersetzen (überprüfen)  und schließlich auf die Zielhardware zu übertragen (upload) . Um zur Laufzeit des Programmes interaktive Nutzereingriffe zu ermöglichen, besitzt die Entwicklungsumgebung ein eingebautes Terminal-Programm (hier als Serial Monitor bezeichnet, Ausgaben im Programm mit Serial.print()) .

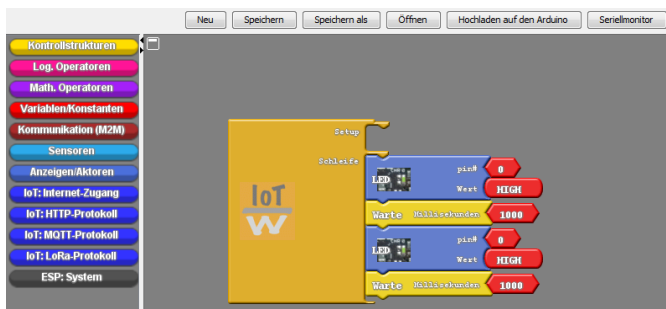
Etwaige Syntax-Fehler, Warnungen oder Probleme beim Hochladen werden im Statusbereich angezeigt.

Anfänger bevorzugen sicher die grafische Programmierung mit Ardublock und rufen unter „Werkzeuge“->“Ardublock“ die entsprechende Oberfläche auf. Wir erstellen dort ein Ardublock-Programm, indem wir die entsprechenden Programmblöcke aus

dem Menü auf der linken Seite auswählen und in den bereits vorhandenen Programmrumpf einfügen.

Die Einführung einer neuen Programmierumgebung erfolgt üblicherweise anhand eines einfachen Beispiels: Das Kultprogramm „Hallo-Welt“, welches einen kleinen Begrüßungstext ausgibt. Mangels eigenem Bildschirm beschränkt sich die Begrüßung bei unserem IoT-Kit auf ein einfaches Blinken der integrierten Leuchtdiode (LED). Diese Diode ist bei unserem Board mit dem Beinchen (Pin) GPIO0 (General Purpose I/O) des ESP-8266 verbunden.

Die `digitalWrite()` Funktion zur Ansteuerung der Leuchtdiode finden wir auf der linken Seite im Menü „Anzeigen/Aktoren“, die Warteanweisungen im Menü „Kontrollstrukturen“.



**Abb. 6:** Ardublock Programm Blink.abp

Nach Fertigstellung wird das erzeugte Programm mittels „Hochladen auf den Arduino“ auf das IoT-Kit geladen:

Damit wird der von Ardublock generierte C-Code in den Editor des Arduino kopiert und das Arduino-Programm sofort vom C-Compiler in Maschinencode übersetzt. Dieser Maschinencode wird anschließend über die USB-Verbindung auf das IoT-Kit geladen. Nach kurzer Wartezeit befindet sich das Programm im Hauptspeicher des IoT-Kits und wird wie von Geisterhand sofort ausgeführt – die LED blinkt.

Hurra, wir haben unser IoT-Kit erfolgreich programmiert!

Zur Verdeutlichung des Programmaufbaus schauen wir uns das von Ardublock generierte Arduino-Programm etwas genauer an:

```
1: void setup(){ // Initialisierung, wird nur einmalig ausgeführt
2:   pinMode(0,OUTPUT); // GPIO0 ist Ausgang
3: }
4:
5: void loop() { // Kontinuierliche Wiederholung
6:   digitalWrite(0,HIGH); // LED an
7:   delay(1000); // 1000 ms warten
8:   digitalWrite(0,LOW); // LED aus
9:   delay(1000); // 1000 ms warten
10: }
```

Ein typisches Arduino-Programm (auch Sketch genannt) besteht aus zwei Funktionen. Im Unterprogramm `setup()` (Zeilen 1-3) werden alle Initialisierungsaufgaben einmalig ausgeführt. Alle im Unterprogramm `loop()` (Zeilen 5-10) stehenden Befehle werden dagegen zyklisch wiederholt. Der Programmblock eines Unterprogrammes wird immer mit geschweiften Klammern `{}` umschlossen. Kommentare werden mit `//` eingeleitet. Die in der ersten Spalte angegebenen Programmzeilen dienen nur der vereinfachten Referenzierung im Beschreibungstext und sind nicht Bestandteil des eigentlichen Programmcodes.

Da eine Spannungsausgabe auf einem Pin natürlich die Umgebung (hier in Form der angeschlossenen LED) aktiv beeinflusst und ggf. Schaden anrichten kann, muss dieser Pin im `setup()` erst explizit als Ausgang frei geschaltet werden (Zeile 3 des Programmes).

In unserem Beispiel erfolgt die abwechselnde Ausgabe eines hohen Spannungspegels (HIGH, 3.3 V, Zeile 6) und eines niedrigen Spannungspegels (LOW, 0V, Zeile 8) an dem Pin mit der Nummer 0, d. h. dem Pin, an dem die LED angeschlossen ist.

Unser Mikrocontroller ist in der Lage, diese Befehle mehrere tausend Male in der Sekunde zu wiederholen. So schnell kann das

Auge nicht folgen, weshalb unser Programm künstlich durch Wartezeiten (delay) verlangsamt wird (Zeilen 7 und 9).

Natürlich können wir statt der grafischen Ardublock-Programmierung auch gleich den C-Code in den Editor eingeben oder wir laden den C-Code über „Datei“ -> „Öffnen vom Datenträger“ in den Editor.

Zur erneuten Ausführung übertragen wir unser Programm in der Arduino-GUI mittels Hochladen  an die Zielhardware.

Ardublock ist dann überflüssig - vorausgesetzt, wir sind mit der Programmierung in C/C++ vertraut. Auch können wir mit Ardublock einen Rumpf erzeugen und diesen im Editor erweitern. So bietet unser Arduino-System größtmögliche Flexibilität sowohl für Anfänger, als auch für Fortgeschrittene.

Übrigens: Ein heruntergeladenes Programm ist auch nach Ausschalten und Wiederinbetriebnahme des Kits vorhanden. Erst ein erneutes Hochladen ändert die Programmausführung.

## Die Komponenten des IoT-Kits

Das Kit besitzt neben der bereits bekannten LED an Pin 0 einen Dreh/Drückknopf (bei Ardublock zu finden im Menü „Sensoren“) und zwei **RGB-NeoPixel** (farbige Leuchtdioden, im Menü „Aktoren“) sowie verschiedene Umweltsensoren.

Der folgende Algorithmus erkennt das Drücken des Bedienknopfes und lässt die rechte LED weiss aufleuchten. Ist der Knopf nicht gedrückt, so wird die LED ausgeschaltet.

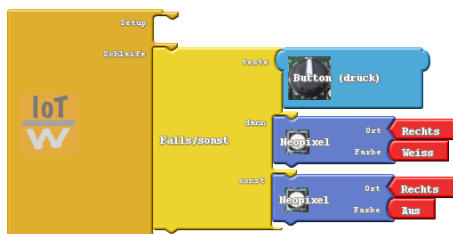


Abb. 7: Das Programm macht aus dem Octopus eine Taschenlampe



## Das Internet der Dinge anfassbar machen

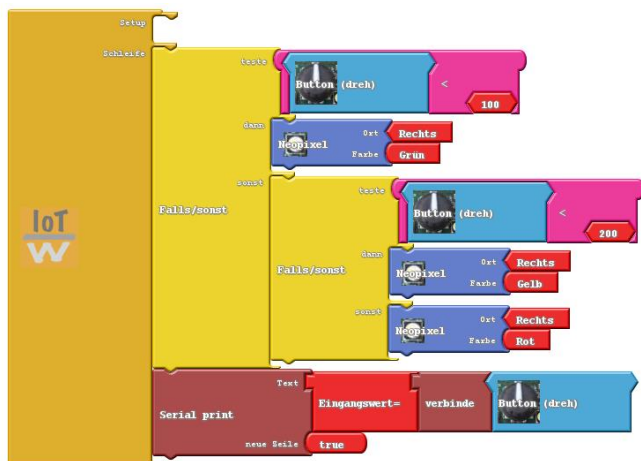
Auch hier wieder ein kurzer Blick hinter die Kulissen des generierten C-Programmes:

```
1: #include <Adafruit_NeoPixel.h>
2: Adafruit_NeoPixel
3: pixels = Adafruit_NeoPixel(2,13,NEO_GRBW + NEO_KHZ800);
4:
5: void setup(){ // Einmalige Initialisierung
6:   pixels.begin();//----- Initialisierung Neopixel
7:   delay(1);
8: }
9:
10: void loop() { // Kontinuierliche Wiederholung
11:   if (digitalRead(2)==LOW) {
12:     pixels.setPixelColor(0,30,30,30);
13:     pixels.show();
14:   } else {
15:     pixels.setPixelColor(0,0,0,0);
16:     pixels.show();
17:   }
18: }
19:
```

Zur Ansteuerung der NeoPixel wird ein spezielles Protokoll verwendet, dieses wird über eine Bibliothek bereitgestellt (include Anweisung Zeile 1).

Der **Dreh/Drückknopf** ist mit drei digitalen Eingangspins verbunden. Ein Pin (GPIO2) liegt immer am High-Pegel, es sei denn, wir drücken auf den Knopf. Dann schaltet dieser Pin gegen Masse (Low). Abfragen lässt sich der aktuelle Signalzustand einfach über einen digitalRead()-Befehl (Zeile 11).

Komplizierter ist die Sache mit der Drehfunktion, aber Dank des bereitgestellten Ardublock-Moduls lässt sich diese letztendlich genauso einfach bedienen, wie die Drückfunktion.



**Abb. 8: Ampelfarben in Abhängigkeit der Drehbewegung**

Das obige Programm liest die Stellung des Drehkopfes ein. In einer geschachtelten Fallunterscheidung wird das rechte NeoPixel in Abhängigkeit der Drehposition auf eine andere Farbe gesetzt. Der Drehknopf beginnt bei Programmstart mit Position Null. Eine Rechtsdrehung erhöht, eine Linksdrehung dekrementiert die vom Sensor zurückgelieferte Positionsangabe. Ist der Wert kleiner 100, so leuchtet unser NeoPixel grün, bei einem Wert zwischen 100 und kleiner 200 gelb, darüber rot. Die letzte Zeile gibt uns zur Kontrolle die Positionsangabe auf der seriellen Schnittstelle aus. Diese Ausgabe lässt sich über den Menüpunkt „Seriellmonitor“ aktivieren.

Hurra, die Ein-/Ausgabemöglichkeiten sind klar!<sup>10</sup>

<sup>10</sup> Übrigens: Mit dem Rotary-Encoder lassen sich z. B. die Drehgeschwindigkeiten von Motoren ermitteln, oder die Position eines drehbaren Roboterarmes bestimmen.

## Die digitale Flaschenpost – Hallo Server

Natürlich lassen sich auch mehrere LEDs zu einer Matrix zusammenschalten. Bei unserem Kit geschieht dies durch Nutzung eines externen Erweiterungsboards: Das CharlieWing Matrix-Display<sup>11</sup>. Dieses wird auf die FeatherWing Steckleisten unseres Kits gesetzt und dient der erweiterten Visualisierung.

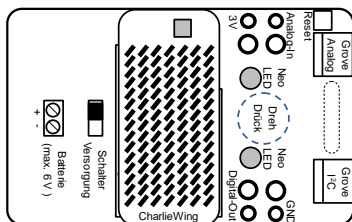


Abb. 9: IoT-Kit mit LED-Matrix

Durch Verwendung der entsprechenden Bibliotheksfunktionen gelingt so die Textausgabe des „Hallo IoT-Kit“ mit wenigen Programmzeilen:



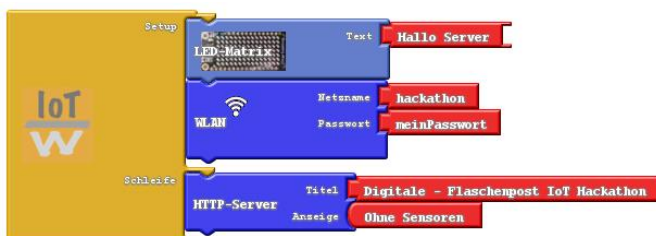
Abb. 10: Textausgabe über LED-Matrix

Allerdings ist die Ausgabe eines festen Textes nach einiger Zeit recht langweilig und wir wünschen uns mehr Flexibilität.

Da wir über ein internetfähiges Kit verfügen wäre es doch toll, wenn wir den auszugebenden Text per Web-Interface vom Handy aus ändern könnten. Dazu werden wir mit wenigen Programmzeilen einen eigenen HTTP-Server implementieren und per Internet-Browser darauf zugreifen.

<sup>11</sup> [learn.adafruit.com/adafruit-15x7-7x15-charlieplex-led-matrix-charliewing-featherwing](https://learn.adafruit.com/adafruit-15x7-7x15-charlieplex-led-matrix-charliewing-featherwing)

Auch hier erleichtert ein entsprechendes Ardublock-Modul die Umsetzung des Vorhabens:



**Abb. 11: Wenige Blöcke realisieren einen Web-Server**

Das Programm begrüßt uns mit einer Startmeldung „Hallo-Server“ und versucht dann, den Zugang zum WLAN herzustellen. Ist dies gelungen, erscheint die vom Accesspoint zugewiesene IP-Adresse in der Anzeige. Melden wir uns mit unserem Handy, oder einem anderen Rechner, am selben WLAN-Accesspoint an, so können wir die Homepage unseres Moduls unter dieser IP-Adresse erreichen.

Eine Homepage besteht normalerweise aus einer Textdatei in einem speziellen Format (Hypertext Markup Language, HTML). Da unser System keine Dateien verwalten kann, erstellt Ardublock stattdessen ein Textfeld mit dem gewünschten Inhalt. Die angegebene Titelzeile erscheint als Überschrift auf der Seite:

```

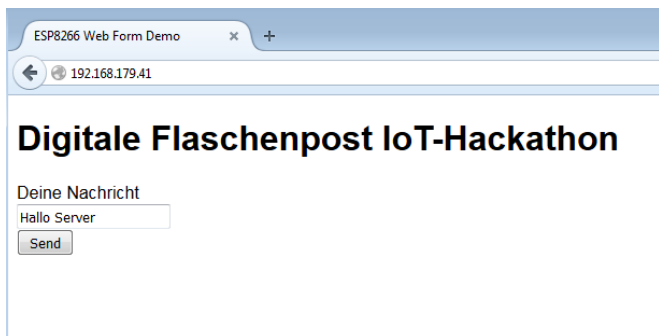
1:  const char INDEX_HTML[] =
2:  " <html>"
3:    "<head>"
4:      "<title>ESP8266 Web Form Demo</title>"
5:    "</head>"
6:    "<body>"
7:      "<h1>Digitale - Flaschenpost IoT Hackathon </h1>"
8:      "<FORM action=\"/\" method=\"post\">"
9:        "<P>"
10:         "Deine Nachricht<br>"
11:         "<INPUT type=\"text\" name=\"message\"><br>"
12:         "<INPUT type=\"submit\" value=\"Send\">"
13:        "</P>"
14:      "</FORM>"
15:    "</body>"
16:  "</html>";

```

## Das Internet der Dinge anfassbar machen

Daneben enthält unsere HTML-Seite mit dem Input-Tag ein im Browser des Clients editierbares Textfeld. Der Inhalt dieses Feldes wird über den Send-Button an unseren Web-Server übermittelt. Über diesen Mechanismus aktualisieren wir den anzuzeigenden Text.

Stolz präsentieren wir die erste eigene Homepage:



**Abb. 12:** Homepage unserer Flaschenpost

In einer Flasche verpackt, kann das Kit so eine digitale Nachricht an den Empfänger senden – ähnlich wie eine Flaschenpost in unserer analogen Welt.

Hurra, wir haben den ersten eigenen Web-Server erstellt!<sup>12</sup>

---

<sup>12</sup> Übrigens: Es gibt keine Sicherheitsfunktionen. Wer unsere IP-Adresse kennt, der kann uns so eine Nachricht schicken! (Tipp: Wir können die Flaschenpost z. B. mit einem geheimen Code vor dem eigentlichen Text schützen!)

## Unser Forschungsschiff – Hallo IoT

Im folgenden Abschnitt wollen wir unsere Flaschenpost zusätzlich über verschiedene Sensoren direkt mit der Außenwelt verbinden (Datenlogger) und als Forschungsschiff auf Reisen schicken. Einfaches Beispiel dafür ist der eingebaute Bosch-Sensor BME280<sup>13</sup>. Dieser ist in der Lage, die Temperatur, die Luftfeuchte und den Luftdruck der Umwelt zu bestimmen. Über eine digitale Schnittstelle, den sogenannten I<sup>2</sup>C-Bus (Inter IC-Bus)<sup>14</sup>, lassen sich die Messwerte vom Mikroprozessor auslesen und anzeigen. Ein spezielles Schnittstellenmodul „verbinde“ wandelt dabei den numerischen Messwert in einen Text, der über die Matrix ausgegeben wird. Ist keine Matrix vorhanden, kann stattdessen auch der serial – Befehl zur Ausgabe auf die serielle Schnittstelle genutzt werden.



Abb. 13: Messung der Umweltinformationen

Zu einem Datenlogger gehört natürlich neben der eigentlichen Messdatenerfassung noch mehr, nämlich die Speicherung der gemessenen Werte. Üblicherweise geschieht dies auf einem USB-Stick oder einer SD-Karte. Da wir ein internetfähiges Kit besitzen, können wir glücklicherweise auf diese externen Speicher verzichten und unsere Daten direkt dem Internet anvertrauen. In diesem Fall spricht der Informatiker von einer Cloud-Lösung, bei

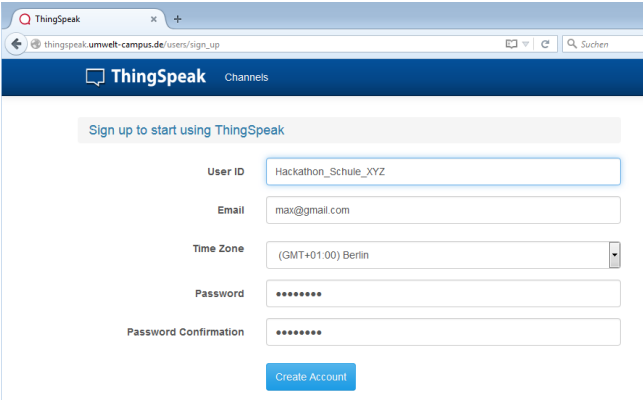
<sup>13</sup> [https://www.bosch-sensortec.com/bst/products/all\\_products/bmp280](https://www.bosch-sensortec.com/bst/products/all_products/bmp280)

<sup>14</sup> <http://www.elektronik-magazin.de/page/der-i2c-bus-was-ist-das-21>

## Das Internet der Dinge anfassbar machen

der ein im Internet verfügbarer Server unsere Daten entgegennimmt und für uns archiviert. Hierzu existieren eine Reihe von Diensten verschiedener Anbieter<sup>15</sup>.

Im Rahmen der Werkstatt verwenden wir die Open-Source Plattform Thingspeak. Zur Nutzung dieses Dienstes müssen wir uns vorher beim Server registrieren. Der offizielle Server ist weltweit unter [thingspeak.com](https://thingspeak.com) erreichbar. Alternativ dazu gibt es eine am Umwelt-Campus gehostete lokale Installation unter der URL [www.thingspeak.umwelt-campus.de](https://www.thingspeak.umwelt-campus.de). Diese ist nur im Campus-Netz erreichbar, alle unsere Daten bleiben lokal. Hierzu müssen wir uns dort zuerst registrieren.



The image shows a web browser window displaying the registration page for Thingspeak. The browser's address bar shows the URL [thingspeak.umwelt-campus.de/users/sign\\_up](https://thingspeak.umwelt-campus.de/users/sign_up). The page header includes the Thingspeak logo and the word "Channels". Below the header, there is a section titled "Sign up to start using Thingspeak". This section contains a registration form with the following fields: "User ID" (containing "Hackathon\_Schule\_XYZ"), "Email" (containing "max@gmail.com"), "Time Zone" (a dropdown menu set to "(GMT+01:00) Berlin"), "Password" (masked with seven dots), and "Password Confirmation" (also masked with seven dots). A blue "Create Account" button is located at the bottom of the form.

**Abb. 14: Registrierung Thingspeak-Cloud (Quelle: thingspeak.com)**

Anschließend können wir unter „Channels“ -> „New Channel“ den ersten Messkanal für den Hackathon konfigurieren. Dort erstellen wir drei Feldeinträge für Temperatur, Feuchte und Luftdruck. Wichtig ist die Reihenfolge, denn ein Feld wird bei der späteren Datenspeicherung durch den Feldindex adressiert. Wer

---

<sup>15</sup> phant.io, thingspeak.com, io.adafruit.com

möchte, kann die Messdaten später auch für andere Nutzer sichtbar machen (Make Public-Flag).

Nach Speichern der Konfiguration steht der Kanal ab sofort zum Zugriff bereit.

The image shows two side-by-side screenshots from the ThingSpeak website. The left screenshot is titled 'New Channel' and contains the following fields: 'Name' (Häckathon Schule egg), 'Description' (Mein erster Messkanal für das Internet der Dinge), 'Field 1' (Temperatur), 'Field 2' (Feuchte), 'Field 3' (Druck), 'Make Public' (checked), 'URL', 'Elevation', 'Show Location' (checked), 'Latitude' (0.0), 'Longitude' (0.0), 'Show Video' (checked), 'Video ID', and 'Show Status' (checked). A green 'Save Channel' button is at the bottom. The right screenshot is titled 'Write API Key' and shows a 'Key' field with the value 'V23DQq...P' and an orange 'Generate New API Key' button.

Abb. 15: Erstellung eines Messkanals (Quelle: thingspeak.com)

Bevor wir aber Daten dort ablegen können, benötigen wir eine Zugangsberechtigung, den sogenannten **API-Key**. Nur wer in Besitz dieses Schlüssels ist, kann Daten schreiben. Sicherheit spielt beim IoT eine große Rolle. Wir notieren uns also unseren Key und setzen ihn später an geeigneter Stelle im Programm ein (oder kopieren ihn mit cut & paste aus der Browseranzeige).

Nach erfolgreicher Konfiguration können die Daten mittels einfacher HTTP- GET Nachricht in der Datenbank des Servers abgelegt werden. Zum Test lässt sich diese GET-Nachricht von jedem Webbrowser aus generieren. Dazu im Browser die URL

`http://api.thingspeak.com/update?api_key=V23DQqyyyyUP&field1=34.5`



## Das Internet der Dinge anfassbar machen

aufrufen, wobei der API-Key an den eigenen Kanal angepasst werden muss. Der Server (host) übernimmt die Daten für die Temperatur (hier 34.5 für das erste Feld) und antwortet mit der laufenden Nummer des aktuellen Eintrags in seiner Datenbank.

Eine solche GET Nachricht läßt sich per entsprechendem Ardublock auch einfach durch unser IoT-Kit absenden.

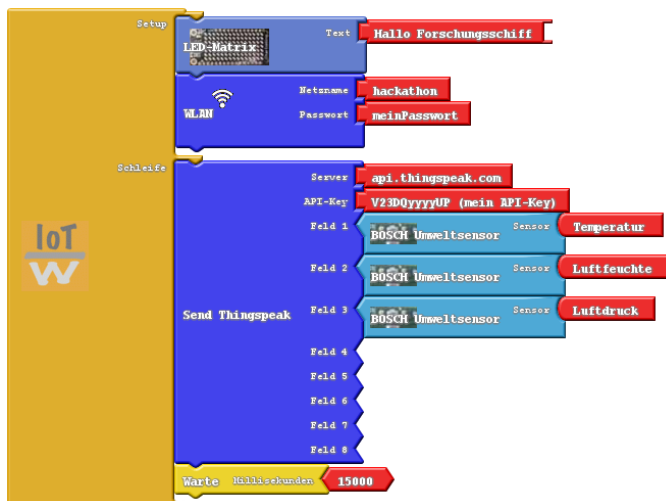


Abb. 16: Cloud-Datenlogger zur Erfassung von Umweltinformationen

Zur individuellen Festlegung des Messkanals muss natürlich der API-Key im Thingspeakblock an unseren persönlichen Key angepasst werden. Der Server Thingspeak.com lässt nur Aktualisierungsraten von mehr als 15 Sekunden zu. Deshalb fügen wir noch einen Wartebefehl ein.

Kontrollieren können wir den Erfolg auf der Thingspeak-Seite im Internet (oder im Falle von thingspeak.umwelt-campus.de nur im Intranet):

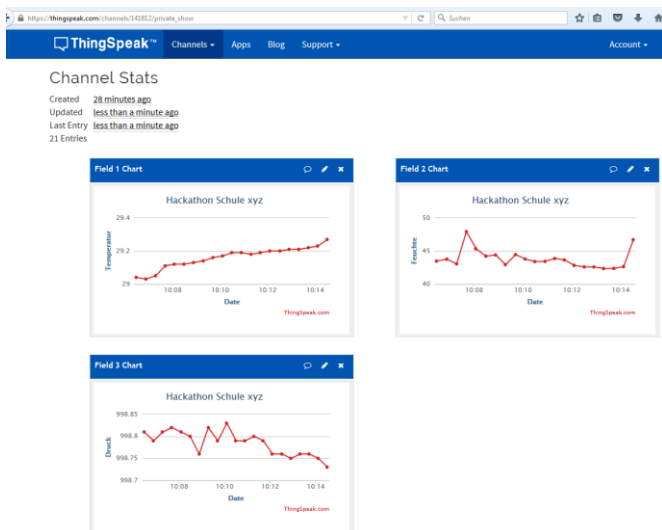


Abb. 17: Visualisierung eines Messkanals (Quelle: thingspeak.com)

Dort sind unsere Messwerte ab sofort weltweit verfügbar. Damit haben wir mit wenig Programmcode einen internetfähigen Datenlogger für Umweltinformationen erstellt.

Hurra, wir haben unsere Flasche in ein Forschungsschiff mit IoT-Anbindung verwandelt!<sup>16</sup>

<sup>16</sup> Übrigens: Die Sensorik lässt sich ganz einfach über den Grove-Connector erweitern. Denkbar sind alle in einer Tabelle am Ende des Kapitels aufgeführten Sensoren.

## Forschungsschiff informiert sich – Hallo M2M

Natürlich kann ein IoT-Device auch selbst Informationen aus dem Internet abfragen. Dazu definieren die Diensterbringer spezifische Schnittstellen (**API, Application Programming Interface**), über die die Informationen bereitgestellt werden können.

Das Prinzip ist ähnlich, wie beim Aufruf normaler Webseiten, nur die Antwort erfolgt in einem maschinenlesbaren Format (z. B. XML, Extensible Markup Language). Als Beispiel verwenden wir die Wettervorhersage von yahoo<sup>17</sup>.

Mit Hilfe des bereits bekannten HTTP-GET können die aktuellen Wetterbedingungen für beliebige geografische Positionen abgerufen werden. Der Ort wird dabei über eine Identifikationsnummer (woeid) festgelegt<sup>18</sup>. Das folgende Programm ermittelt die aktuelle Windgeschwindigkeit für Saarbücken (woeid 690631, ggf. anpassen):

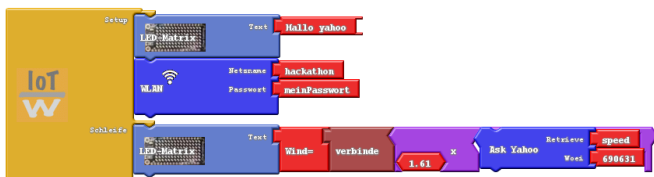


Abb. 18: Wetterabfrage nutzt Cloud-Dienste von yahoo

Yahoo liefert die Daten in Meilen pro Stunde, unsere Ausgabe erfolgt nach Umrechnung in km/h.

Hurra, wir können die Wettervorhersage nutzen!<sup>19</sup>

<sup>17</sup> <https://developer.yahoo.com/weather/>

<sup>18</sup> <http://woeid.rosselliot.co.nz/>

<sup>19</sup> Übrigens: Wenn wir die Windgeschwindigkeit mit einer Fallunterscheidung (if ... then ... else) überprüfen, können wir ab einer bestimmten Windgeschwindigkeit einen Alarm auslösen (z. B. durch Setzen eines digitalen Ausgangs ähnlich wie bei der blinkenden LED). Damit lässt sich eine einfache Überwachung z. B. einer Markise realisieren.

## Forschungsschiff greift ein - Hallo Control

Die bedingte Ausführung von Programmteilen lässt sich prima dazu nutzen, um per Rückkopplung in ein System einzugreifen. Der Ingenieur spricht dann von einer Regelung. Als einfaches Beispiel bauen wir uns gleich eine automatische Bewässerung für die Blumentöpfe auf der Fensterbank. Dazu benötigen wir einen Sensor, welcher die Leitfähigkeit der Erde ermittelt und als Maß für die Feuchtigkeit genutzt werden kann. Ist die Erde zu trocken, steuern wir eine Pumpe an, welche Wasser in den Topf befördert.

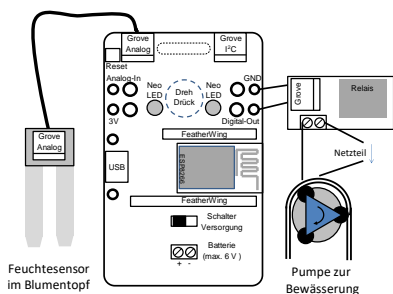


Abb. 19: Das Forschungsschiff als Bewässerungsregler

Mit unserer bisher gesammelten Erfahrung ist das programmtechnisch einfach umzusetzen:

## Das Internet der Dinge anfassbar machen

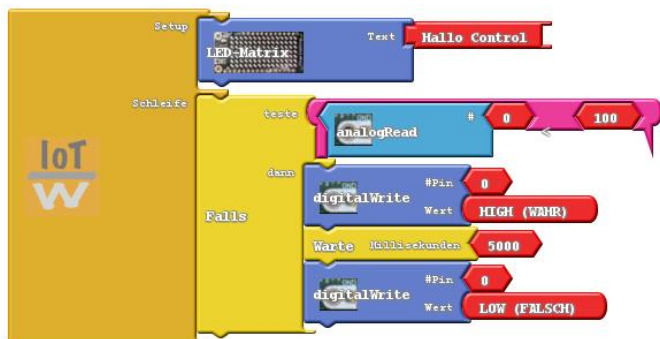


Abb. 20 Ein einfacher Bewässerungsregler

Leider kann der Mikrocontroller nur kleine Lasten wie z. B. eine LED schalten. Hier setzen wir deshalb ein Relais ein. In unserem Fall haben wir das Relais zur Pumpenansteuerung mit dem bereits bekannten I/O-Pin GPIO0 verbunden (Bananensteckerbuchsen I/O auf der rechten Seite). Das Relais wiederum schaltet die Verbindung zwischen Pumpe und Netzteil. **Achtung: Wir verwenden natürlich nur Kleinspannungen – niemals die 230 V aus der Steckdose schalten!**

Innerhalb der Regelschleife lesen wir das vom Feuchtesensor<sup>20</sup> gelieferte Spannungssignal. Dieses wird vom Analog-Digital-Wandler (ADC) des ESP in einen Wert zwischen 0 und 1023 gewandelt. Ist der Wert kleiner als eine Grenze (z. B. 100 für Trockenheit), so wird die Pumpe für kurze Zeit angeschaltet. Die Erde wird feucht, die Leitfähigkeit steigt, der Sensor liefert hoffentlich höhere Werte für die Leitfähigkeit.

Kritisch wird die Anwendung dann, wenn unser Vorratsgefäß leer ist. Dann trocknet die Erde weiter aus und wir erreichen eine untere Alarmgrenze. In diesem Fall wollen wir wieder unsere

<sup>20</sup> [http://seeedstudio.com/wiki/Grove\\_-\\_Moisture\\_Sensor](http://seeedstudio.com/wiki/Grove_-_Moisture_Sensor)

IoT-Fähigkeiten ins Spiel bringen und eine Meldung über Twitter, E-Mail oder SMS generieren.

Dazu bietet sich der Internet-Dienst „If this then that“ (IFTTT)<sup>21</sup> an.

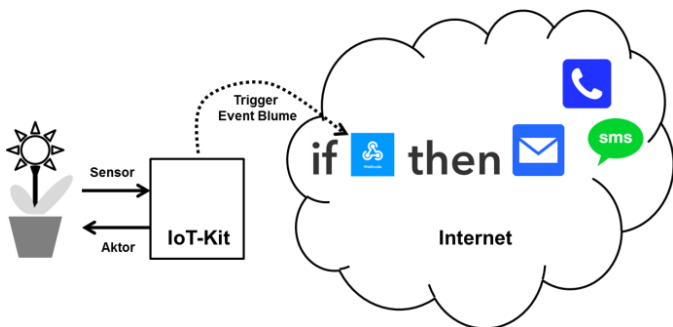


Abb. 21: Informationsfluss IFTTT

Auch hier müssen wir uns natürlich vorher registrieren. Mittels komfortabler Weboberfläche können wir anschließend mit einfachem Knopfdruck ein Rezept (Applet) der Form IF .... then DO.... erstellen (d. h. konfigurieren, nicht programmieren). In unserem Fall ist die Triggerbedingung (IF) ein vom IoT-Kit erzeugter Event „Blume“ auf dem Webhooks-Kanal:

<sup>21</sup> <https://ifttt.com>

## Das Internet der Dinge anfassbar machen

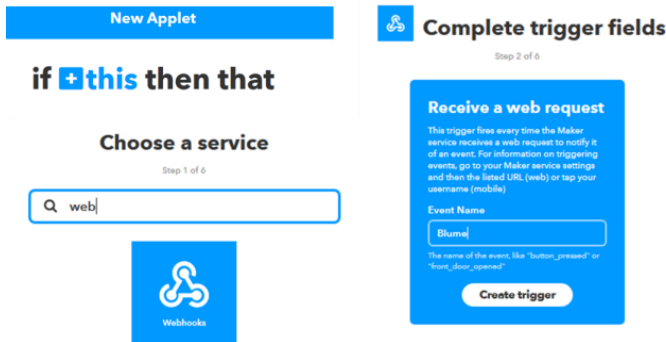


Abb. 22: Ereignis (Event) definieren (Quelle: ifttt.com)

Als Action lassen wir uns eine E-Mail schicken:

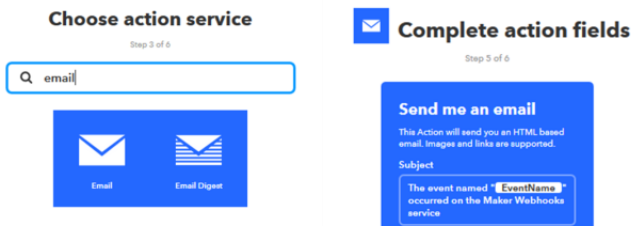


Abb. 23: Action Mailversand (Quelle: ifttt.com)

Damit nicht jeder die Triggerbedingung „Blume“ auslösen darf, müssen wir bei IFTTT noch unsere Zugangsberechtigung in Form eines IFTTT-Keys erfahren.

Dies geschieht etwas versteckt über die URL:

[https://ifttt.com/services/maker\\_webhooks/settings](https://ifttt.com/services/maker_webhooks/settings).

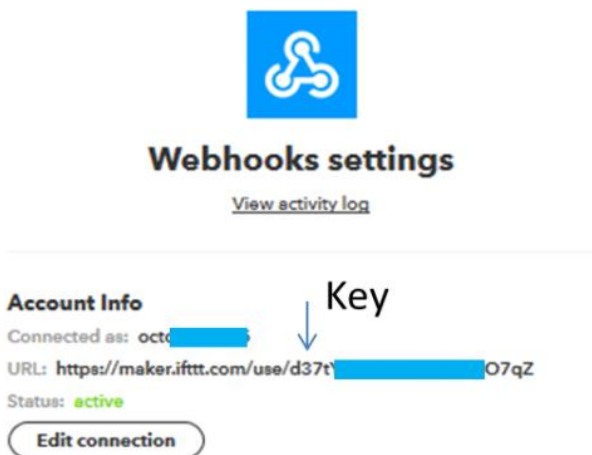


Abb. 24: IFTTT-Key, um uneren Trigger zu autorisieren

Damit können wir uns jetzt im Urlaub beruhigt zurücklehnen. Die Blumen sind versorgt – und sollte mal was schiefgehen, erhalten wir eine mail und können die Nachbarn aktivieren.

Das Ardublock-Programm soll den Trigger nur einmalig auslösen, wir wollen ja nur eine Benachrichtigungsmail erhalten. Dies realisieren wir einfach mit einer digitalen Variable „AlarmScharf“, die wir im Setup scharf stellen und nach der ersten Trigger-Nachricht löschen.



## Das Internet der Dinge anfassbar machen



Abb. 25: Bewässerung mit Cloud-Anbindung

Hurra, wir haben eine Blumenbewässerung realisiert!<sup>22</sup>

<sup>22</sup> Übrigens: So ähnlich lassen sich viele Alltagsaufgaben überwachen. Zum Beispiel die Luftfeuchtigkeit im Keller durch Ansteuerung eines Lüfters, oder die Alarmanlage mit Überprüfung der Fensterkontakte und Ansteuerung einer Sirene.

## Wearables – Forschungsschiff kleidet sich

Die einzeln ansteuerbaren bunten NeoPixel eröffnen viele neue Anwendungen. Als kleine Knöpfe lassen sie sich problemlos in Kleidungsstücke integrieren. Dazu gibt es leitfähiges Nähmaterial, mit denen die Anzeigeelemente geschickt befestigt und mit Energie versorgt werden können. Unser IoT-Kit ist aber an sich schon so kompakt, dass es problemlos am Körper befestigt werden kann. Mit einer mobilen Energieversorgung über Batterien können wir unser „Hallo Server“-Projekt sofort recyceln: Als Namensschild zur Kennzeichnung der Coaches am Hackathon. Fügen wir etwas Sensorik in Form eines Mikrofons hinzu, so haben wir zusätzlich eine mobile Lichtorgel. Je nach Umgebungslautstärke leuchten die NeoPixel in einer anderen Farbe.

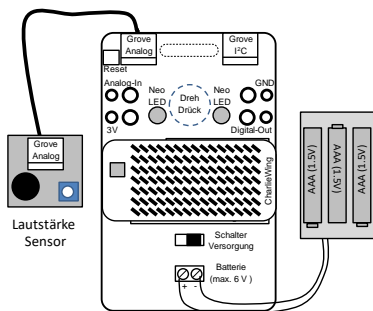


Abb. 26: Tragbares IoT-Kit mit Lärmsensor

Zur Nutzung der Batterie müssen wir den Wahlschalter zur Spannungsversorgung nach rechts stellen. Weitere Hinweise zum Batteriebetrieb finden sich in der C-Code Version der Blaupausen.

Übrigens: Wer es auffällig liebt, kann die CharlieWing-Matrix auch durch ein NeoPixelWing ersetzen und diese statt des einzelnen NeoPixels ansteuern (AmpelNeoWing.abp). So wird unsere Ampel unübersehbar.

## Das Internet der Dinge anfassbar machen

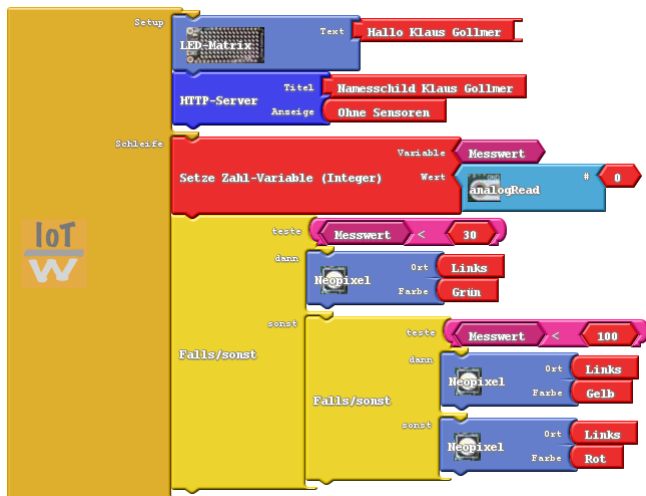


Abb. 27 Name-Tag und Lärmampel

Hurra, unser Kit ist wie ein Kleidungsstück tragbar!

Übrigens: Die Lichtorgel eignet sich auch prima als Lärmampel für die Schule 😊. Wir können die Funktion auch ohne Sensor testen. Einfach mit den Fingern die beiden Bananenbuchsen (ADC und 3 V) auf der linken Seite berühren. Über unseren Hautwiderstand gibt es eine leitende Verbindung zwischen 3 V und dem analogen Eingang. Dies Phänomen können wir zukünftig z. B. prima für einen Touch Sensor verwenden.

## Das Forschungsschiff hört aufs Wort

Ein abschließendes komplexeres Beispiel verdeutlicht nun die enormen Möglichkeiten, die uns ein Zusammenspiel verteilter Systeme im Internet bietet.

Unsere Aufgabe: Wir wollen das IoT-Kit per Alexa-Sprachbefehl fernsteuern und die Farbe eines Neopixels festlegen. Analog dazu könnte Alexa auch einen digitalen Ausgang schalten (und über ein Relais auch das Garagentor öffnen/schließen, die Waschmaschine oder die Kaffeemaschine steuern).

Zentrales Element bildet die bereits aus dem vorhergehenden Kapitel bekannte If...this...then...that Aktivität der IFTTT-Cloud<sup>23</sup>, die im Zusammenspiel mit Alexa<sup>24</sup> und einem MQTT-Broker<sup>25</sup> unser Kit fernsteuert.

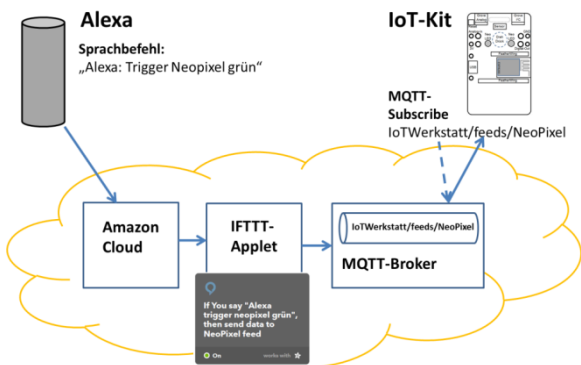


Abb. 28: Ein Beispiel für verteilte Dienste - Alexa steuert die Neopixel

In diesem Fall fällt die Wahl auf den Broker von Adafruit, welcher bereits über fertig vorkonfigurierte IFTTT-Aktionen verfügt

<sup>23</sup> <https://ifttt.com/>

<sup>24</sup> <https://www.amazon.de/>

<sup>25</sup> <https://io.adafruit.com/>

## Das Internet der Dinge anfassbar machen

und die Angelegenheit damit erheblich vereinfacht. Auch Alexa besitzt einen Link zu IFTTT, so dass die Hauptaufgabe in der intelligenten „Verdrahtung“ der Komponenten besteht.

Zur Realisierung müssen wir zuerst einen Account beim Broker-Dienst anlegen. In unserem Fall haben wir den Username „IoT-Werkstatt“ gewählt und einen Feed mit dem Namen „NeoPixel“ angelegt. Im MQTT-Kontext ist dieser Feed später unter der Bezeichnung (Topic) „IoTWerkstatt/feeds/NeoPixel“ sichtbar. Hier wählt jeder Nutzer natürlich später seinen eigenen User-Name und Feed-Bezeichnung.

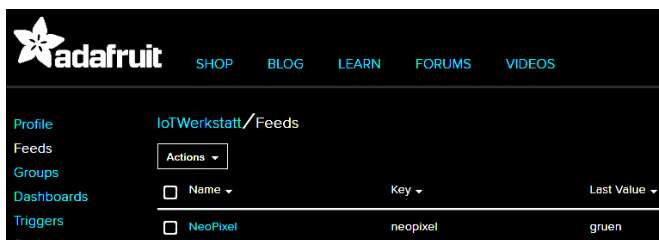


Abb. 29: Konfiguration des MQTT-Brokers

Anschließend loggen wir uns im IFTTT-Dienst ein und verknüpfen dort sowohl unseren Alexa-Account bei Amazon als auch unseren Broker-Account bei Adafruit. Ist dies geschehen, können wir für jede Farbe, die unser Neopixel annehmen soll eine Regel (IFTTT-Applet) generieren.

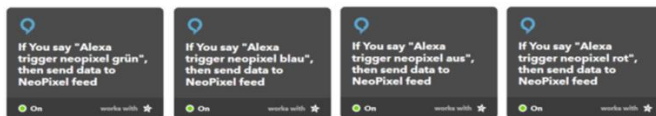


Abb. 30: IFTTT-Applets verknüpfen Amazons Alexa mit der Adafruit-Cloud

Bleibt uns als letzte Aktion nur noch die Realisierung eines MQTT-Clients (Consumer), der das entsprechende MQTT-Topic abonniert, den übertragenen payload auswertet und die Farbe des Neopixels korrekt setzt.

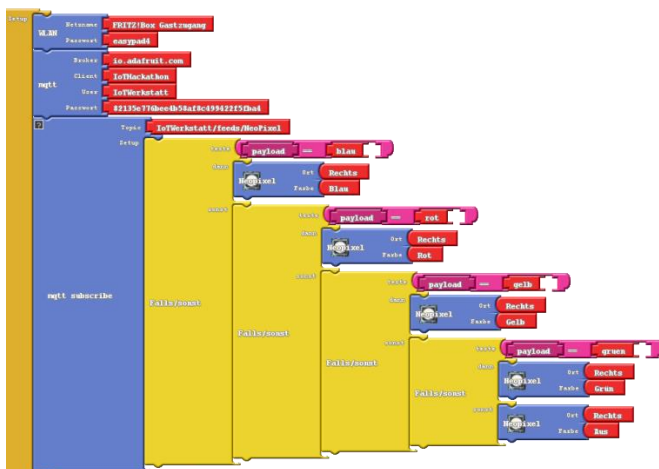


Abb. 31: MQTT-Client zur Steuerung des Neopixels

Der Sprachbefehl „Alexa: Trigger Neopixel rot“ steuert fortan die Farbe unseres Neopixels. Diese Cloud-Lösung funktioniert unabhängig vom Standort der Geräte, Alexa und IoT-Kit müssen sich nicht am selben Ort befinden<sup>26</sup>. Dank intelligenter Verknüpfung von verschiedenen Diensten gelingt die Lösung der komplexen Aufgabe in wenigen Schritten und ganz ohne eine Zeile Hochsprachen-Programmcode zu generieren.

<sup>26</sup> Wer bei der Veranstaltung vor Ort die Alexa nutzen möchte, der sollte bitte sein eigenes Device mitbringen.

### **Die Flaschenpost wird flexibel – Hallo Interface**

Die vorherigen Kapitel haben sicher zum Nachdenken über eigene Projektideen eingeladen. Zur Umsetzung fehlt jetzt noch die richtige Sensorik und Aktorik. Dazu existieren eine Vielzahl an bereits fertig aufgebauten Interfacebausteinen, die über das Grove-Interface an unser IoT-Kit angeschlossen werden. Bevor wir auf kommerzielle Sensoren zurückgreifen, wollen wir einen Licht-Sensor selber bauen. Schließlich verfügt unser IoT-Kit mit dem Analog-Digitalwandler (ADC) über einen universellen Messzugang für elektrische Spannungen.

### **Spannungsteller & LDR**

Schon mit ein paar elektronischen Bauteilen für wenige Cent lässt sich ein einfacher Licht-Detektor bauen. Ein solcher passiver Sensor setzt die Änderung einer physikalischen Größe (Beleuchtungsstärke) in eine Widerstandsänderung um. Neben dem lichtempfindlichen Widerstand (Light Dependent Resistor, LDR) gibt es temperaturabhängige Widerstände (NTC, PTC) oder Dehnungsmesstreifen (DMS), die bei mechanischer Dehnung ihren Widerstandswert verändern.

Am Beispiel eines LDR wollen wir dieses Prinzip zur Messung der Umgebungsbeleuchtung unseres IoT-Kits nutzen. Damit dies gelingt, werden wir die Widerstandsänderung in eine Spannungsänderung transformieren und diese dann über den analogen Eingang messen. Dabei nutzen wir den vom Ohmschen Gesetz bekannten Zusammenhang Spannung  $U$  gleich Widerstand mal Strom ( $U=R \cdot I$ ). Denkbare Einsatzfälle sind z. B. ein Dämmerungsschalter fürs Flurlicht oder eine helligkeitsabhängige Alarmanlage.

Am Grove-Connector des Kits steht zwischen den mit GND und 3V beschrifteten Pins eine Spannung von 3 V zur Verfügung.

Daran schließen wir eine Reihenschaltung aus dem lichtempfindlichen Widerstand  $R_{LDR}$  und einem festen Widerstand  $R_1$  an. Der Gesamtwiderstand einer solchen Reihenschaltung berechnet sich zu  $R = R_1 + R_{LDR}$ . An diesem Gesamtwiderstand liegt die Spannung von 3 Volt an, der Strom  $I$  durch die Widerstände berechnet sich also zu

$$I = \frac{U}{R} = \frac{3V}{R_1 + R_{LDR}}$$

Gemessen wird die Spannung am analogen Eingang, also zwischen dem gelben Kabel und dem schwarzen Kabel als Bezugspotential. Das ist die Spannung am Widerstand  $R_1$ :

$$U_{A0} = U_{R_1} = R_1 \cdot I = \frac{R_1}{R_1 + R_{LDR}} \cdot 3V$$

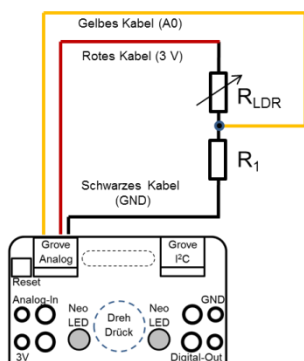


Abb. 32: Spannungsteiler am Grove-Eingang

Diese Schaltung teilt die Spannung 3 V je nach Widerstandswert  $R_{LDR}$  in zwei Teilspannungen, weshalb diese Schaltung auch **Spannungsteiler** genannt wird. Wir messen mit  $U_{A0}$  eine Spannung, die größer wird, wenn  $R_{LDR}$  kleiner wird, also wenn Licht auf unseren LDR fällt.



## Das Internet der Dinge anfassbar machen

Der Widerstand  $R_1$  wird üblicherweise so dimensioniert, dass der Wert in der Nähe des Widerstands  $R_{LDR}$  am Arbeitspunkt liegt. Messen wir bei normaler Beleuchtung z. B. einen Widerstand  $R_{LDR}$  von 3000 Ohm, dann würden wir für  $R_1$  auch 3000 Ohm wählen. Festwiderstände gibt es aber nur in diskreten Abstufungen zu kaufen, hier können wir für  $R_1$  z. B. 2200 Ohm oder 4700 Ohm verwenden.

Tauschen wir den LDR durch einen NTC – Widerstand (Negative Temperature Coefficient), so können wir statt Beleuchtungsstärke damit die Temperatur messen. Anwendung findet diese einfache Schaltung z. B. bei der Temperaturregelung im Kühlschrank.

### **Kommerzielle Sensorik und Aktorik**

Wer nicht selber Hand anlegen möchte, kann auch auf die vielfältigen Komponenten des Grove Systems zurückgreifen. Nachfolgende Tabellen zeigen eine Auswahl an zur Verwendung empfohlener Interfacetechnik:

Tab. 2: Erweiterungsmöglichkeiten Sensorik

Messgröße	Sensor	Informationen
Luftqualität (Erdbeeren bis Pups)	Air Quality Sensor 1.3	<a href="http://www.seecedstudio.com/wiki/Grove_-_Air_Quality_Sensor_v1.3">www.seecedstudio.com/wiki/Grove_-_Air_Quality_Sensor_v1.3</a>
Lagebestimmung (Beschleunigung, Gravitation, Vibration)	9-axis Absolute Orientation Sensor	<a href="http://www.bosch-sensortec.com/bst/products/all_products/bno055">www.bosch-sensortec.com/bst/products/all_products/bno055</a>
Leitfähigkeit (Erd-, Hautfeuchte, Wasserstand)	Feuchtigkeitssensor	<a href="http://seecedstudio.com/wiki/Grove_-_Moisture_Sensor">seecedstudio.com/wiki/Grove_-_Moisture_Sensor</a>
Lautstärke	Loudness Sensor	<a href="http://seecedstudio.com/wiki/Grove_-_Loudness_Sensor">seecedstudio.com/wiki/Grove_-_Loudness_Sensor</a>
Entfernung (5- 80 cm, Parksensor)	Abstandssensor (Sharp)	<a href="http://pololu.com/files/0j85/gp2y0a21yk0f.pdf">pololu.com/files/0j85/gp2y0a21yk0f.pdf</a>
Elektr. Strom	Stromsensor (Wechselstrom)	<a href="http://openenergymonitor.org/emon/buildingblocks/ct-sensors-interface">openenergymonitor.org/emon/buildingblocks/ct-sensors-interface</a>
Elektr. Strom /Spannung	Stromsensor (Gleichstrom)	<a href="http://learn.adafruit.com/adafruit-ina219-current-sensor-breakout">learn.adafruit.com/adafruit-ina219-current-sensor-breakout</a>
UV-Licht (Sonnenbrand)	UV Sensor	<a href="http://seecedstudio.com/wiki/Grove_-_UV_Sensor">seecedstudio.com/wiki/Grove_-_UV_Sensor</a>
Bewegung	PIR Motion	<a href="http://seecedstudio.com/wiki/index.php?title=Twig_-_PIR_Motion_Sensor">seecedstudio.com/wiki/index.php?title=Twig_-_PIR_Motion_Sensor</a>
Feinstaub	Honeywell-HPMA115S0	<a href="https://www.mouser.de/new/honeywell/honeywell-hpma115so-sensor/">https://www.mouser.de/new/honeywell/honeywell-hpma115so-sensor/</a>
CO2	Sensirion scd30	<a href="https://www.sensirion.com/de/umwelt-sensoren/kohlendioxidsensoren-co2/">https://www.sensirion.com/de/umwelt-sensoren/kohlendioxidsensoren-co2/</a>

Tab. 3: Erweiterungsmöglichkeiten Aktorik

## Das Internet der Dinge anfassbar machen

Ausgabe	Aktor	Informationen
Elektr.Verbraucher Pumpe, Motor	Relais	<a href="https://www.seecstudio.com/wiki/Grove_-_Relay">seecstudio.com/wiki/Grove_-_Relay</a>
7x15 Matrix	Charlie- Wing	<a href="https://learn.adafruit.com/adafruit-15x7-7x15-charlieplex-led-matrix-charliewing-featherwing">learn.adafruit.com/adafruit-15x7-7x15-charlieplex-led-matrix-charliewing-featherwing</a>
Motoren	Feather- Motor Shield	<a href="http://www.adafruit.com/products/1438">http://www.adafruit.com/products/1438</a>

Das IoT-Kit besitzt je eine analoge und digitale (I2C) Grove Schnittstelle. Sollen mehr Sensoren/Aktoren Verwendung finden, bieten sich folgender Komponenten an:

**Tab. 4: Erweiterungen Schnittstellen**

Komponenten	Einsatz	Informationen
I2C-ADC	Analog Input	<a href="http://wiki.seec.cc/Grove-I2C_ADC/">http://wiki.seec.cc/Grove-I2C_ADC/</a>
6 Output I/O Expander	Digital	<a href="https://learn.sparkfun.com/tutorials/sx1509-io-expander-breakout-hookup-guide">learn.sparkfun.com/tutorials/sx1509-io-expander-breakout-hookup-guide</a>
I2C-Hub	I2C	<a href="https://www.seecstudio.com/depot/Grove-I2C-Hub-p-851.html">seecstudio.com/depot/Grove-I2C-Hub-p-851.html</a>

## Anhang

Dieses Kapitel richtet sich an alle technisch vorgebildeten Interessierten, die tiefer in die Schaltung und die flexiblen Möglichkeiten unseres IoT-Kits einsteigen möchten.

### Schaltplan:

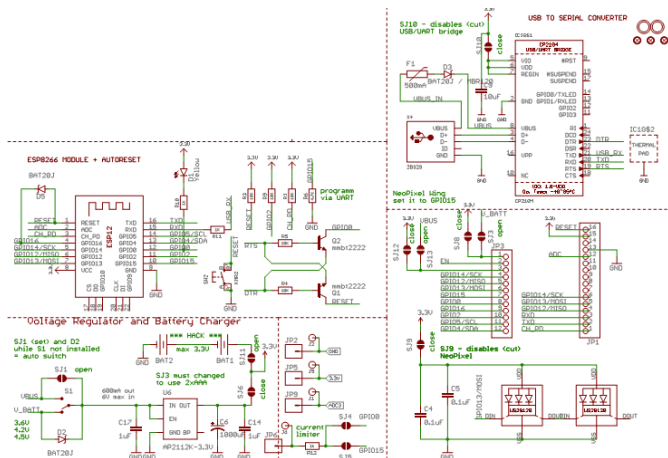


Abb. 33: Schaltplan

### Entwicklungsumgebung

Da es sich um ein sogenanntes „Third Party Board“ handelt, erfolgt die Anbindung an die Entwicklungsumgebung über den „Boardverwalter“. Um das Board auch am häuslichen PC nutzen zu können, muss dort zuerst unter „Datei“ im Untermenü „Voreinstellungen“ die folgende Boardverwalter-URL eingetragen werden:

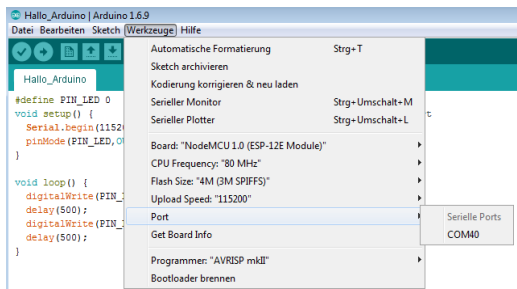
„[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)“.

Anschließend ist im Menü „Werkzeuge“ im Untermenü „Board“ der „Boardverwalter“ auszuwählen und die „esp8266 Toolchain“

## Das Internet der Dinge anfassbar machen

zu installieren. Nach erfolgreicher Installation steht schließlich das Board zur Auswahl zur Verfügung. Dazu ist unter „Werkzeuge“, das kompatible Board „NodeMCU V1.0 (ESP12E Modul)“ anzuwählen. Nach Anschluss des Kits an den USB-Port des PCs sollte der entsprechende Treiber vom Betriebssystem automatisch initialisiert werden. Bei Problemen kann der Treiber manuell nachinstalliert werden<sup>27</sup>

Anschließend unter „Werkzeuge“ -> „Port“ den vom Betriebssystem vergebenen COM-Port eintragen.



**Abb. 34: Einbindung in die Arduino-Entwicklungsumgebung**

Zur Installation von Ardublock ist die Java Datei „ardublock-IoT.jar“ in den Unterordner „\tools\ArduBlockTool\tool\“ des Arduino Sketchbook Verzeichnisses zu kopieren. Das Sketchbook-Verzeichnis lässt sich in der Arduino-GUI unter „Datei“->„Voreinstellungen“ ermitteln. Nach Neustart der Arduino-GUI existiert fortan unter dem Punkt „Werkzeuge“ ein Eintrag „Ardublock“, über den die Oberfläche gestartet werden kann.

Die bereits auf dem Board integrierten Interfacekomponenten benötigen zur einfachen Ansteuerung die Einbindung spezieller Bibliotheken. Damit kann die betreffende Komponente dann im Programmcode sehr einfach über Arduino-Befehle angesprochen werden. Ein sonst übliches intensives Studium der Datenblätter entfällt. Dieses flexible Bibliothekskonzept ist eine der Gründe

<sup>27</sup> <https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx>

für die enorme Verbreitung der Arduino-Plattform. Insbesondere aufwendige Sensorprotokolle sind als Bibliotheken vorhanden und machen deren Nutzung komfortabel.

Auf der Veranstaltung der IoT-Werkstatt und im Zip-File für den heimischen Rechner sind alle benötigten Bibliotheken bereits installiert. Diese können, ähnlich wie beim „Boardmanager“, mit Hilfe eines „Librarymanager“ eingebunden werden (Menü „Sketch“->„Bibliothek einbinden“). Zur Vollständigkeit enthält die Tabelle die im Rahmen unserer Blaupausen eingebundenen Komponenten.

## Das Internet der Dinge anfassbar machen

Interface-komponente	Detail	Library	Lizenz	Name der Autoren
Umweltdaten Bosch BME	Bosch BME280	Sparkfun BME280 V1.0.0 <a href="https://github.com/sparkfun/SparkFun_BME280_Arduino_Library">https://github.com/sparkfun/SparkFun_BME280_Arduino_Library</a>	MIT	Marshall Taylor @ SparkFun Electronics
	Bosch BME680	<a href="https://github.com/adafruit/Adafruit_BME680">https://github.com/adafruit/Adafruit_BME680</a>	BSD	Limor Fried/Ladyada
	Adafruit_Unified_Sensor	<a href="https://github.com/adafruit/Adafruit_Sensor">https://github.com/adafruit/Adafruit_Sensor</a>		
Beschleunigung Bosch BMO055	<a href="https://www.adafruit.com/product/2472">https://www.adafruit.com/product/2472</a>	<a href="https://github.com/adafruit/Adafruit_BNO055">https://github.com/adafruit/Adafruit_BNO055</a>	GNU GPL	Kevin (K'TOWN)
	<a href="http://wiki.seeed.cc/Grove-3-Axis_Digital_Accelerometer-1.5g/">http://wiki.seeed.cc/Grove-3-Axis_Digital_Accelerometer-1.5g/</a>	<a href="https://github.com/Seeed-Studio/Accelerometer_MMA7660">github.com/Seeed-Studio/Accelerometer_MMA7660</a>		
Grove MMA7660	<a href="http://wiki.seeed.cc/Grove-3-Axis_Digital_Accelerometer-1.5g/">Axis_Digital_Accelerometer-1.5g/</a>			Frankie Chu
RGB-LED	Neopixel	<a href="https://github.com/adafruit/Adafruit_NeoPixel">https://github.com/adafruit/Adafruit_NeoPixel</a>	GNU GPL	Phil Burgess. PJRC, Michael Miller
	WS2801	<a href="https://github.com/adafruit/Adafruit-WS2801-Library">https://github.com/adafruit/Adafruit-WS2801-Library</a>	BSD	Limor Fried/Ladyada
Touch	<a href="https://www.adafruit.com/product/1982">https://www.adafruit.com/product/1982</a>	Adafruit_MPR121 <a href="https://github.com/adafruit/Adafruit_MPR121">https://github.com/adafruit/Adafruit_MPR121</a>	BSD	Limor Fried/Ladyada
NFC/RFID	<a href="http://wiki.seeed.cc/Grove-NFC/">http://wiki.seeed.cc/Grove-NFC/</a>	<a href="https://github.com/Seeed-Studio/PN532">https://github.com/Seeed-Studio/PN532</a> <a href="https://github.com/Seeed-Studio/Grove-NFC-libraries-Part">https://github.com/Seeed-Studio/Grove-NFC-libraries-Part</a>	BSD	@Don, @JiapengLi, @awieser Kevin Townsend Don Coleman

Control App	Blynk	<a href="https://github.com/blynkkk/blynk-library/releases/latest">https://github.com/blynkkk/blynk-library/releases/latest</a>	MIT	Volodymyr Shymanskyy
Gassensor Multichannel	<a href="http://wiki.seeed.cc/Grove-Multichannel_Gas_Sensor/">http://wiki.seeed.cc/Grove-Multichannel_Gas_Sensor/</a>	<a href="https://github.com/Seeed-Studio/Mutichannel_Gas_Sensor">https://github.com/Seeed-Studio/Mutichannel_Gas_Sensor</a>	MIT	Grove system
Dreh/Drück Rotary-Encoder	<a href="https://www.pjrc.com/teensy/td_1ibs_Encoder.html">https://www.pjrc.com/teensy/td_1ibs_Encoder.html</a>	Encoder by Paul Stoffregen V1.4.1 <a href="https://github.com/PaulStoffregen/Encoder">https://github.com/PaulStoffregen/Encoder</a>		Paul Stoffregen
LED-Matrix Charlieplex	<a href="https://learn.adafruit.com/adafruit-15x7-7x15-charlieplex-led-matrix-charliewing-featherwing/">learn.adafruit.com/adafruit-15x7-7x15-charlieplex-led-matrix-charliewing-featherwing/</a>	<a href="https://github.com/adafruit/Adafruit_IS31FL3731">https://github.com/adafruit/Adafruit_IS31FL3731</a>		
	Textausgabe	Adafruit GFX V1.1.5 <a href="https://github.com/adafruit/Adafruit-GFX-Library">https://github.com/adafruit/Adafruit-GFX-Library</a>	BSD	
MQTT	<a href="https://pubsub-client.knolleary.net/index.html">https://pubsub-client.knolleary.net/index.html</a>	PubSubClient <a href="https://github.com/knolleary/pubsubclient/releases/tag/v2.6">github.com/knolleary/pubsubclient/releases/tag/v2.6</a>		Nick O'Leary
GSM	Mobilfunk	TinyGSM, 0.1.7, <a href="http://tiny.cc/tiny-g_n_sm-re-adme">tiny.cc/tiny-g_n_sm-re-adme</a> StreamDebugger		Volodymyr Shymanskyy
Serielle Schnittstelle	Software Seriell	<a href="https://github.com/ple-rup/espsoftwareserial">github.com/ple-rup/espsoftwareserial</a>		Peter Lerup
OTA	On Air Update	<a href="https://github.com/esp8266/Arduino/tree/master/libraries/ArduinoOTA">github.com/esp8266/Arduino/tree/master/libraries/ArduinoOTA</a>		



## Das Internet der Dinge anfassbar machen

---

LoRaWAN	FeatherWing (IBM-LMIC)	<a href="https://github.com/matthijskooijman/arduino-lmic">https://github.com/matthijskooijman/arduino-lmic</a>	Thomas Telkamp Matthijs Kooijman
Digital I/O Extender	<a href="https://learn.sparkfun.com/tutorials/sx1509-io-expander-break-out-hookup-guide">https://learn.sparkfun.com/tutorials/sx1509-io-expander-break-out-hookup-guide</a>	<a href="https://github.com/sparkfun/SparkFun_SX1509_Arduino_Library">https://github.com/sparkfun/SparkFun_SX1509_Arduino_Library</a>	Jim Lindblom
I2C-ADC Extender	<a href="http://wiki.seeed.cc/Grove-I2C_ADC/">http://wiki.seeed.cc/Grove-I2C_ADC/</a>		
Motoren	FeatherWing Adafruit Motor-Shield <a href="http://www.adafruit.com/products/1438">http://www.adafruit.com/products/1438</a>	<a href="https://github.com/adafruit/Adafruit_Motor_Shield_V2_Library">https://github.com/adafruit/Adafruit_Motor_Shield_V2_Library</a>	

---