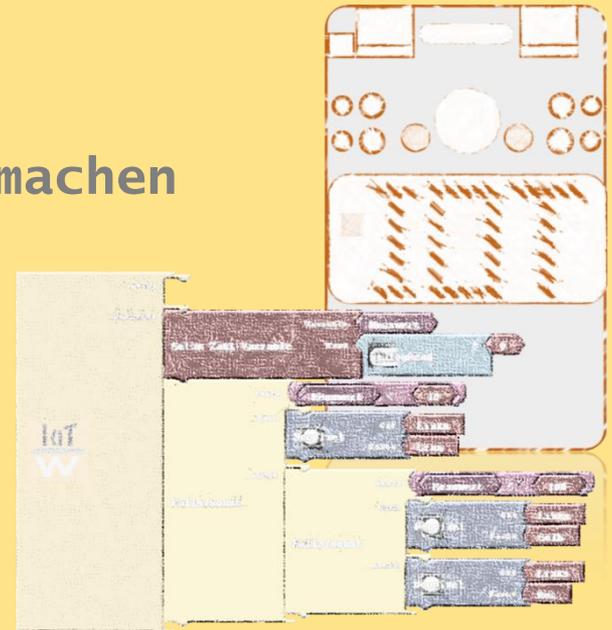


# IoT-Werkstatt

Das Internet der Dinge anfassbar machen



## Algorithmisches Denken



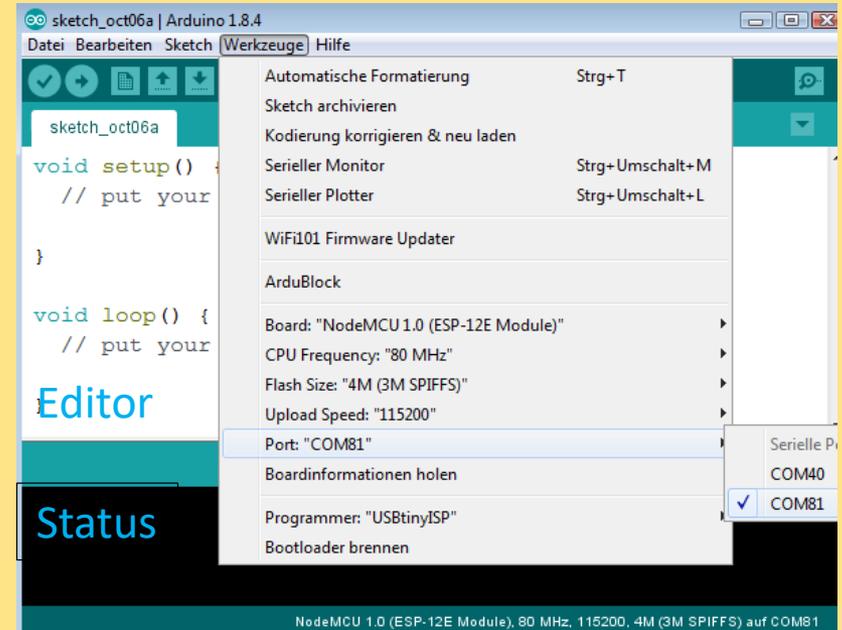
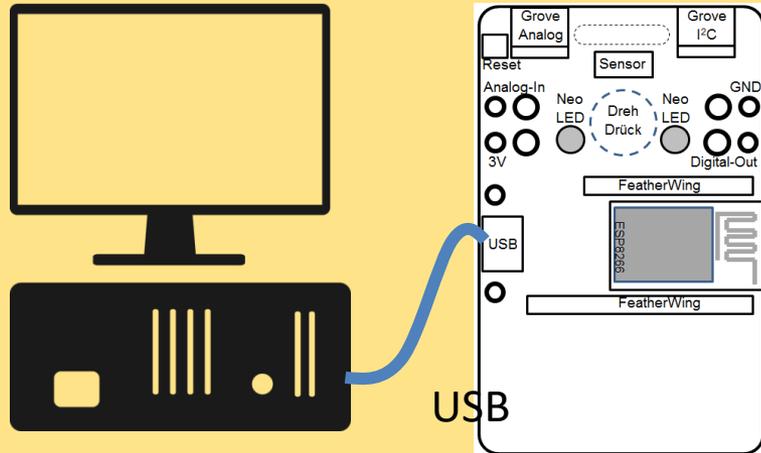
Umwelt-Campus  
Birkenfeld

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

Klaus-Uwe Gollmer  
und Guido Burger



# Vorbereitung



1. Octopus Board mit USB-Kabel verbinden
2. Arduino (IoT-Werkstatt) starten
3. Unter „Werkzeuge“ das Board kontrollieren (NodeMCU) und den Port einstellen (i.d.R. der letzte COM-Port)



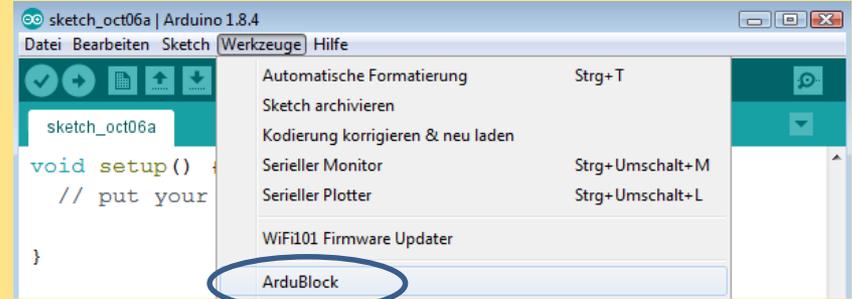
↑  
↑  
Upload auf Octopus (Maschinecode)  
Übersetzen (C-Compiler)

↑  
Serieller Monitor  
(Terminal am PC)



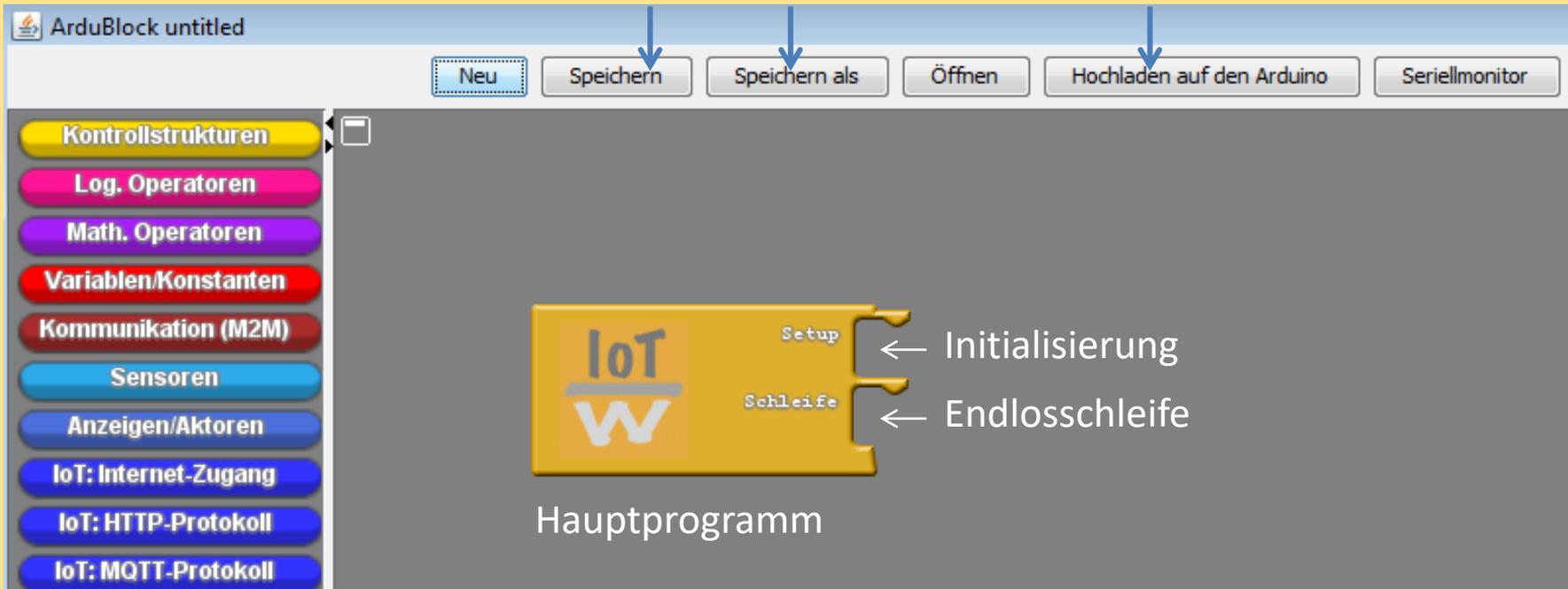
# Ardublock

1. Unter „Werkzeuge“ die grafische Oberfläche Ardublock starten



Speichern des Ardublock Programms

Upload auf den Octopus

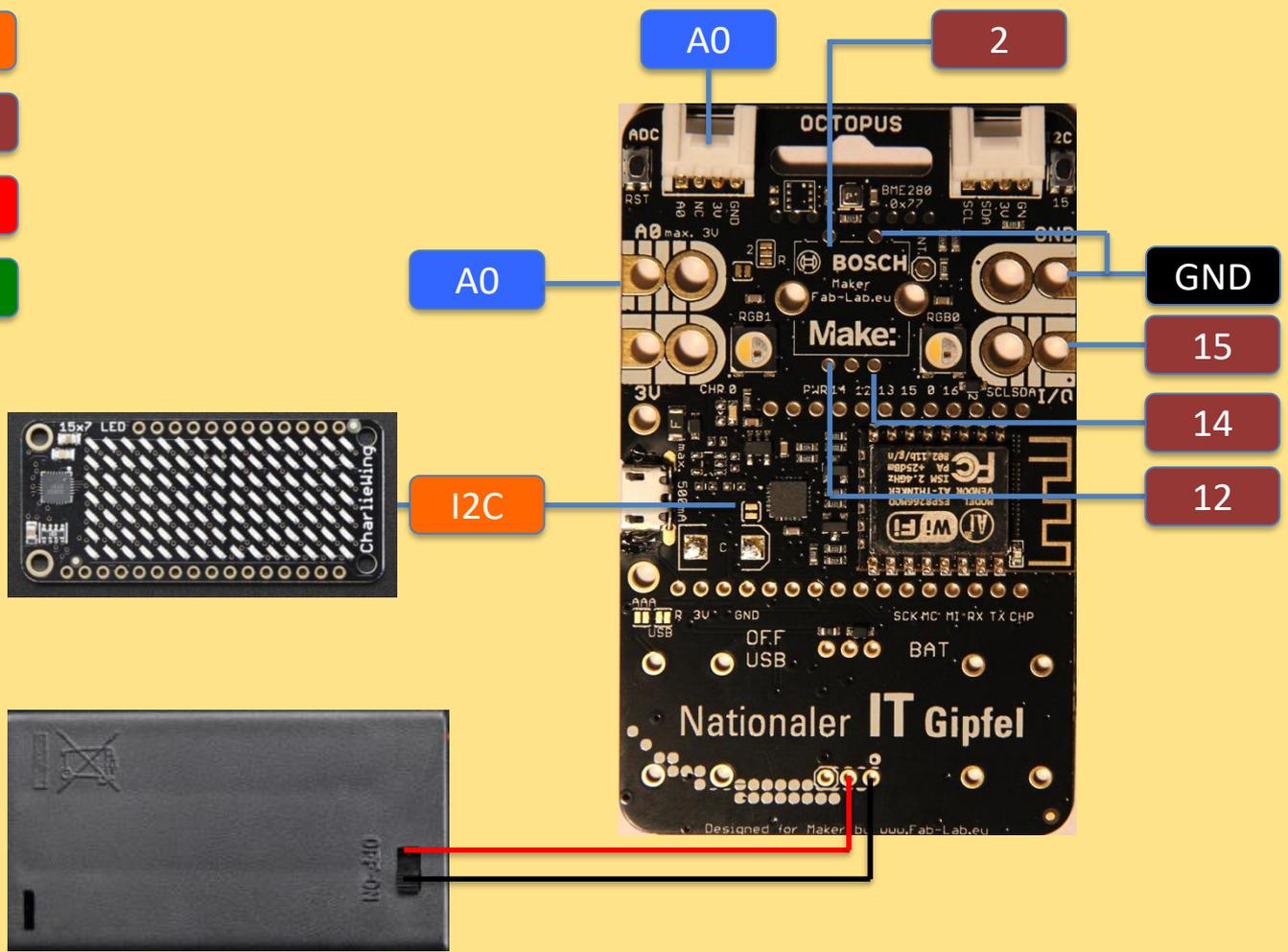


↑  
Werkzeugkasten mit Puzzleteilen



# Hardware

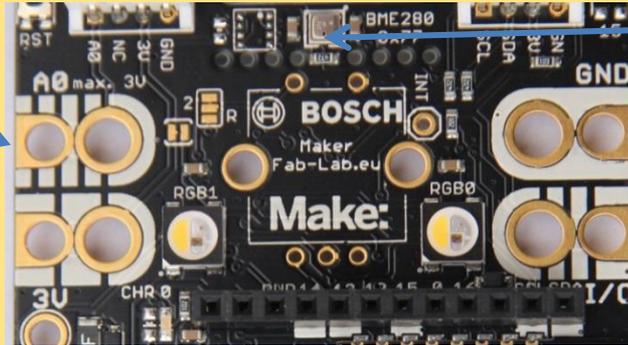
- ADC
- I2C
- I/O
- SPI
- UART



# Sensoren

Der Octopus hat verschiedene Sensoren auf der Platine. Der analoge Spannungseingang kann über die Bananensteckerkontakte auf der linken Seite gespeist werden. Der Bosch- Sensor BME280/680 liefert Umweltmesswerte.

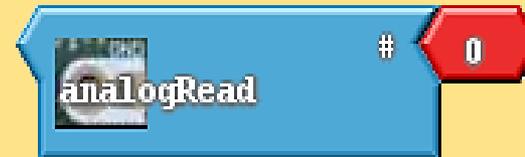
Analog Input



Je nach Version BME280 (oder BME680 (Bestückung dann links daneben))

## Analog Input

Eine Spannung von 0 - 3.3 V entspricht einem ADC-Wert von 0 – 1023 (Integer)



## Bosch BME

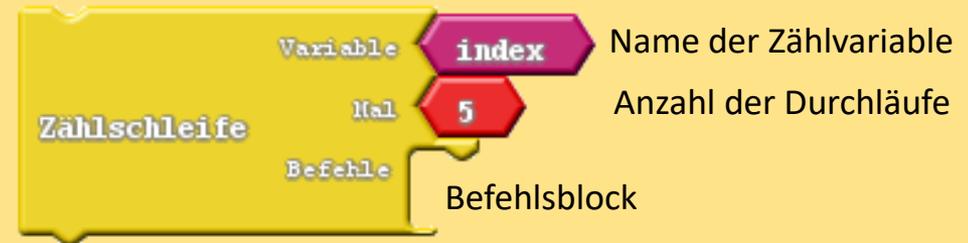
Der Sensor liefert physikalische Größen. Temperatur in Grad Celsius, Druck in hPa, Feuchte in %





### Zählschleife (for)

Wiederholt einen Block genau index Mal  
Index als Schleifenzähler



```
% Matlab  
for index = 1:5  
  % Befehlsblock  
end
```

```
// C-Programm  
for (int index = 1;index<=5;index++) {  
  // Befehlsblock  
}
```

### Bedingte Wiederholung (while)

Testet eine logische Bedingung und  
wiederholt den Block solange diese  
Bedingung erfüllt ist



```
% Matlab  
while Bedingung  
  % Befehlsblock  
end
```

```
// C-Programm  
while (Bedingung) {  
  // Befehlsblock  
}
```



### Bedingte Anweisung (if)

Testet eine logische Bedingung und führt den Block nur aus, wenn diese erfüllt ist (Ausführung nur ein einziges mal)

```
% Matlab
if Bedingung
    % Befehlsblock
end
```

```
// C-Programm
if (Bedingung) {
    // Befehlsblock
}
```



Logischer Ausdruck

### Bedingte Anweisung mit Alternative (if/else)

Testet eine logische Bedingung Und führt Befehlsblock 1 aus, wenn eine diese erfüllt ist , andernfalls den Block 2

```
% Matlab
if Bedingung
    % Befehlsblock 1
else
    % Befehlsblock 2
end
```

```
// C-Programm
if (Bedingung) {
    // Befehlsblock 1
} else {
    // Befehlsblock 2
}
```



Logischer Ausdruck



### Warte (delay)

Stoppt die Programmausführung für eine gewisse Zeitdauer

```
% Matlab  
pause(Sekunden);
```

```
// C-Programm  
delay (Millisekunden);
```



Wartezeit in ms



# Ein- / Ausgabe

Ausgabe von Ergebnissen auf dem Bildschirm erfolgt über print-Befehle.  
Beim Octopus wird auf PC-Seite ein serieller Monitor (Terminal) genutzt.

## Ausgabe von Ergebnissen

Beim Ardublock gibt es nur Textausgaben, ggf. muss der Datentyp über einen Verbinder gewandelt werden (Typecast).

Hinweis: Führende / Endende Leerzeichen lassen sich durch die Zeichenfolge "<space>" erzeugen.

```
% Matlab  
fprintf('Mein auszugebender text');  
fprintf('%f',sin(1));
```



```
// C  
Serial.println("Mein auszugebender Text");  
Serial.println(sin(1.0));
```

## Eingabe über Tastatur

Das Puzzleteil gibt eine Eingabeaufforderung auf dem Terminal aus und erwartet eine Texteingabe an der Tastatur. Diese wird ans Programm zurück-geliefert

```
% Matlab  
input('aa= ', 's')
```



## Vergleichsoperationen

Vergleicht zwei Zahlen und liefert einen Wahrheitswert zurück (wahr / falsch)

```
% Matlab und C  
Zahl1 > 10  
Zahl1 == Zahl2
```



## Logische Verknüpfung

Verknüpft zwei Wahrheitswerte, liefert einen Wahrheitswert zurück



```
% Matlab  
Bedingung1 && Bedingung2  
Bedingung1 || Bedingung2  
~Bedingung1
```

UND  
ODER  
NICHT

```
// C-Programm  
Bedingung1 && Bedingung2  
Bedingung1 || Bedingung2  
!Bedingung1
```

Bedingung1	Bedingung2	UND	ODER
0	0 (falsch)	0	0
0	1 (wahr)	0	1
1	0	0	1
1	1	1	1



# Mathematische Operatoren

Führt eine mathematische Operation aus  
Ergebnis ist wieder eine Zahl.

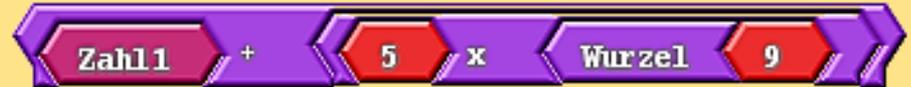
```
% Matlab und C  
Zahl1+1  
sqrt(9)
```



## Komplexe mathematische Ausdrücke

Auch geschachtelte Ausdrücke sind möglich.

```
% Matlab und C  
Zahl1+(5*sqrt(9))
```



# Logische Variablen

Variablen/Konst.

Logische Variablen sind Speicherbehälter für Wahrheitswerte.

Die Inhalte können nur zwei Werte annehmen (wahr/falsch bzw. 1/0)

Logische Ausdrücke besitzen ein rundes Puzzlestück

## Deklaration und Initialisierung

Vor der ersten Verwendung muss Speicher reserviert und mit einem Wert initialisiert werden

```
% Matlab  
NameLogischeVariable = 1;
```

```
// C  
bool NameLogischeVariable = true;
```



## Lesender Zugriff

Danach kann auf den Inhalt der Variable lesend zugegriffen werden

```
% Matlab und C  
NameLogischeVariable
```



# Zahlvariablen

Variablen/Konst.

Für Zahlen gibt es verschiedene Datentypen.

Man unterscheidet ganze Zahlen (Integer) und Kommazahlen (float)

Zahlen besitzen ein dreieckiges Puzzlestück

## Deklaration und Initialisierung

Vor der ersten Verwendung muss Speicher reserviert und mit einem Wert initialisiert werden

```
% Matlab  
NameIntegerVariable = 4711;  
NameFloatVariable   = 3.14;
```



```
// C  
int   NameIntegerVariable = 4711;  
float NameFloatVariable   = 3.14;
```

## Lesender Zugriff

Danach kann auf den Inhalt der Variable lesend zugegriffen werden

```
% Matlab und C  
NameZahlVariable = NameZahlVariable+1
```

NameZahlVariable



# Textvariablen

Texte besitzen ein dreieckiges Puzzlestück

## Deklaration und Initialisierung

Vor der ersten Verwendung muss Speicher reserviert und mit einem Wert initialisiert werden

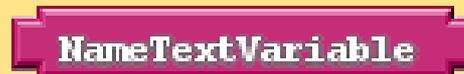
```
% Matlab  
NameTextVariable = 'Mein erster Text';
```

```
// C  
string nameTextVariable = "Mein erster Text";
```



## Lesender Zugriff

Danach kann auf den Inhalt der Variable lesend zugegriffen werden

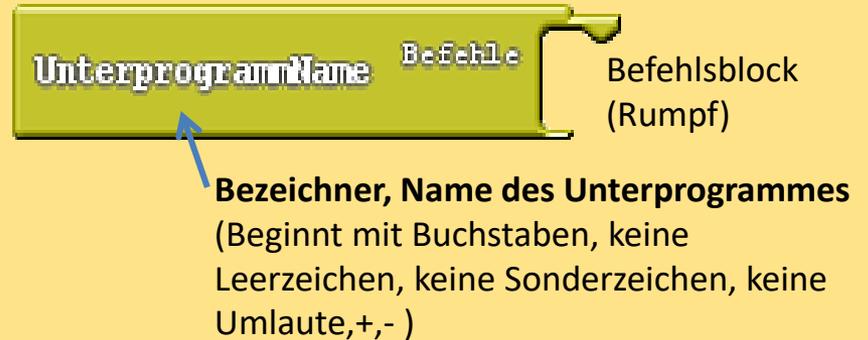


### Unterprogrammdeklaration

Definiert die in einem Unterprogramm zusammengefassten Befehle. Gibt dem Unterprogramm einen eindeutigen Namen

```
% Matlab  
function UnterprogrammName()  
% Unterprogrammrumppf
```

```
// C-Programm  
void UnterprogrammName(void) {  
    // Unterprogrammrumppf  
}
```



### Unterprogrammaufruf

Führt das Unterprogramm aus, d.h. das Programm verzweigt und arbeitet die im Befehlsblock des Unterprogrammes angegebenen Befehle ab. Anschließend beim nächsten Programmbefehl weiter gemacht

```
% Matlab  
UnterprogrammName();
```

```
// C-Programm  
UnterprogrammName();
```

