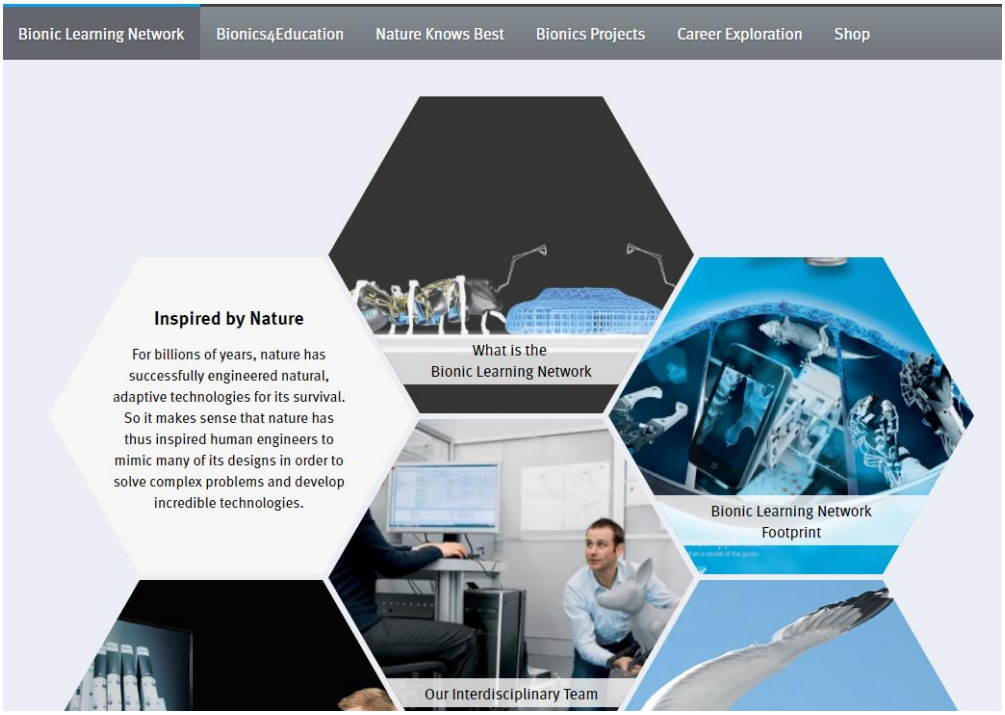
Weiterführende Entwicklungen



Online Plattform Inhalt

Datum
03.04.2018

Ersteller
tswa

Version
1.3

Festo AG & Co. KG

Postfach
73726 Esslingen
www.festo.com
Telefon 0711 347-
Telefax 0711 347-
@de.festo.com
Ruiter Straße 82
73734 Esslingen

Vorabinformation

In dieser Serie der weiterführenden Entwicklungen für den Bionik-Baukasten, ist es zu jedem Zeitpunkt möglich, auf den Urzustand der Software zu wechseln.
Dazu wird das Arduino Projekt von der Bionics4Education Webseite heruntergeladen und auf den Mikrocontroller gespielt. Wie ein Programm mittels Arduino Software auf den im Bionik-Baukasten mitgelieferten Mikrocontroller aktiviert werden kann, wird in Kapitel 2 erklärt.

1 Grundlagen

Um einen Roboter mit Leben zu füllen, benötigt dieser verschiedene programmierbare Komponenten. Diese Komponenten können Sensorwerte auslesen oder Motoren ansteuern und dienen als “Gehirn“ des Roboters. In unserem Fall werden dafür Mikrocontroller eingesetzt, welche über einen programmierbaren Prozessor und verschiedene Ein- bzw. Ausgänge verfügen.
Für die Programmierung eines Mikrocontrollers wird ein Computer mit entsprechender Software benötigt. Hier wird die Logik durch eine Programmiersprache aufgebaut und auf den Mikrocontroller geladen. Folgend werden die verschiedenen Komponenten näher betrachtet um eine Basis für weiterführende Entwicklungen zu schaffen.

1. 1 Programmieren mit C / C++

Der folgende Abschnitt dient dazu ein Grundwissen über die Programmiersprache C++ und die weiterführenden Entwicklungen mit Arduino aufzubauen.

1. 1. 1 Programmaufbau

Ein Arduino Programm besitzt immer den gleichen Aufbau und besteht aus mindestens zwei Funktionen. In Bild 1 ist ein beispielhaftes Programm dargestellt. Es beschreibt ausschließlich den Grundaufbau und beinhaltet keine Funktionalität (Bild 1).

```
1  void setup()
2  {
3  }
4
5  void loop()
6  {
7  }
```

Bild 1

Die erste Funktion in Zeile 1 "void setup()" beinhaltet alles, was zu Beginn des Programms einmalig ausgeführt werden soll. Die auszuführenden Aktionen werden zwischen die geschweiften Klammern "{ … }" geschrieben. Die Klammern begrenzen den Aktionsraum einer Funktion, dies nennt sich “Anweisungsblock“. In Zeile 5 ist die Funktion "void loop()" definiert. Diese wird in einer endlosen Schleife wiederholt aufgerufen (Bild 2).

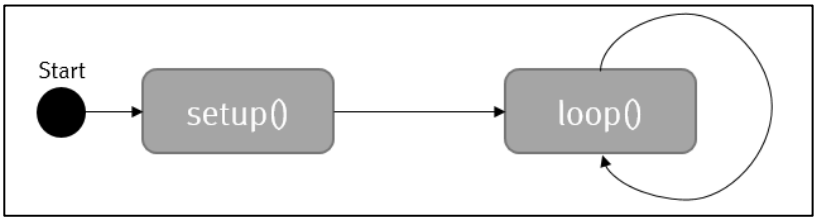


Bild 2

1. 1. 2 Datentypen

Bei C / C++ ist es notwendig für jede Variable einen Datentyp zu definieren. Es gibt noch weitere, für unsere Zwecke erstmal nicht relevante Datentypen.

Datentyp	Bedeutung	Beispiel
int	Ganze Zahlen	42
float	Fließkommazahlen	42.12
char	Zeichen / Character	A
array	Mehrere Werte	[0,42,21]

1. 1. 3 Operatoren

Operatoren können entweder mathematisch oder logisch sein. Die Grundlegendsten Operatoren sind folgende:

Operator	Beispiel	Erklärung
=	a = 2	Der Variablen 2 wird der Wert "2" zugewiesen
+	a = 2 + 3	Mathematische Addition
-	a = 3 – 2	Mathematische Subtraktion
*	a = 3 * 2	Mathematische Multiplikation
/	a = 3 / 2	Mathematische Division
++	a++	Zählt zu der Variablen 1 hinzu (Addition +1)
--	a--	Zieht von der Variablen 1 ab. (Subtraktion -1)
==	a == b	Überprüft auf Gleichheit
!=	a != b	Überprüft auf Ungleichheit
<	a < b	Überprüft ob a kleiner b ist
>	a > b	Überprüft ob a größer b ist
<=	a <= b	Überprüft ob a kleiner oder gleich b ist
>=	a >= b	Überprüft ob a größer oder gleich b ist

1. 1. 4 Variablen

Eine Variable dient dazu, Werte abzuspeichern und diese im späteren Verlauf des Programms wiederzuverwenden. Wollen wir eine Variable einbauen, welche in allen Funktionen sichtbar ist, muss diese außerhalb einer Funktion definiert werden. Dadurch ist diese global sichtbar. Bild 3 veranschaulicht ein solches Programm.
In Zeile 1 wird eine globale Variable "counter" vom Datentyp "int" definiert. Diese ist im gesamten Programm verfügbar und kann von jeder Funktion verändert werden. Im Beispiel wird die Variable in der "setup" Funktion mit dem Wert 0 initialisiert. Lässt man sich den Wert jeder Iteration im Serial Monitor der Arduino IDE ausgeben, ist ersichtlich, dass sich dieser immer um 1 erhöht.

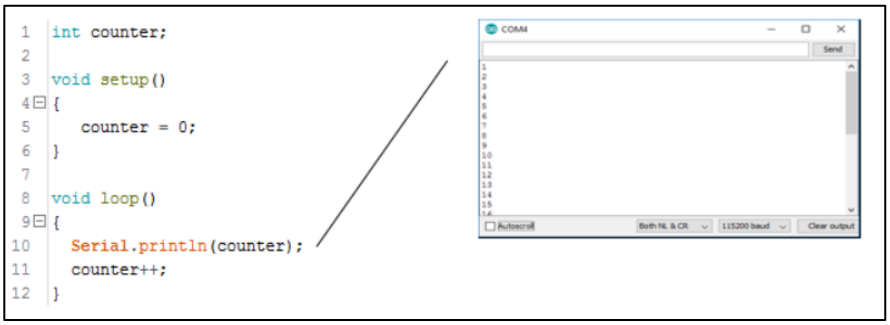


Bild 3

Soll eine Variable nur innerhalb der Funktion sichtbar sein, muss diese in den Anweisungsblock ("{ … }") der Funktion eingebettet werden.

1. 1. 5 Funktionen

Aus dem Kapitel 1. 1. 1 kennen wir nun die zwei Standard Funktionen "setup" und "loop" eines jeden Arduino Programms. Funktionen können noch anders aufgebaut sein als bereits bekannt. Sie können einen Rückgabewert und / oder einen Übergabeparameter besitzen.

Im Beispiel wollen wir die Addition zweier Werte in eine separate Funktion auslagern, welche das Ergebnis zurückgibt. Dies wird benutzt um einen bestimmten Ablauf nicht mehrfach zu schreiben. In Bild 4 ist eine Funktion "executeAddition" eingefügt, welche zwei Übergabeparameter des Datentyps int entgegennimmt, diese miteinander addiert und das Ergebnis vom Datentyp int zurückgibt. Vergleicht man die Funktion mit den zwei bekannten Funktionen "void setup()" oder "void loop()" fällt auf, dass anstelle von "void" nun "int" steht. Das bedeutet, die Funktion gibt einen Wert vom Typ "int" zurück.
In Zeile 17 wird die Funktion mit der Variablen "counter" und dem Wert 2 aufgerufen. Die in Zeile 3 implementierte Funktion addiert diese zwei Werte miteinander und gibt das Ergebnis zurück, welches in der Variablen "counter" gespeichert wird.

```
1  int counter;
2
3  int executeAddition(int a, int b)
4  {
5    int result = a + b;
6    return result;
7  }
8
9  void setup()
10 {
11   counter = 0;
12 }
13
14 void loop()
15 {
16   Serial.println(counter);
17   counter = executeAddition(counter, 2);
18 }
```

Bild 4

1. 1. 6 Abfragen

Beim Schreiben eines eigenen Programms müssen immer wieder Variablen auf bestimmte Werte überprüft werden. Das wird mit if-Abfragen umgesetzt. Eine solche Abfrage könnte überprüfen ob der Wert der Variablen "counter" größer oder gleich 100 ist und anschließend eine Aktion ausführen. In Bild 5 ist die if-Abfrage dargestellt.

```
1  int counter;
2
3  int executeAddition(int a, int b)
4  {
5    int result = a + b;
6    return result;
7  }
8
9  void setup()
10 {
11   counter = 0;
12 }
13
14 void loop()
15 {
16   Serial.println(counter);
17
18   if (counter >= 100)
19   {
20     counter = 0;
21   }
22   else
23   {
24     counter = executeAddition(counter, 2);
25   }
26 }
```

Bild 5

Zeile 18 überprüft den Wert der Variablen "counter", falls dieser größer oder gleich 100 ist, wird alles innerhalb des Anweisungsblocks der If-Abfrage ausgeführt. In diesem Fall wird die Variable "counter" auf 0 gesetzt. Ist die Bedingung nicht erfüllt, wird ein alternativer Anweisungsblock (else) ausgeführt. In diesem Fall wird die Funktion "berechneAddition()" aufgerufen.

1. 1. 7 Schleifen

Eine Schleife wird genutzt, um eine Aktion mehrfach auszuführen. Mit der for-Schleife können Wiederholungen bis zu einer bestimmten Abbruchbedingung definiert werden.

```
27  ...
28
29  for (int x = 0; i <= 5; i++)
30  {
31    counter++;
32  }
33  ...
```

Bild 6

In Bild 6 ist eine for-Schleife definiert. In den Klammern wird die Initialisierung (int x = 0), die Abbruchbedingung (i <= 5) und die Fortsetzung (i++) übergeben. Beim ersten Durchlauf hat die Variable x den Wert 0 und der counter wird um 1 erhöht. Wird das Ende des Anweisungsblocks der Schleife erreicht, wird die Fortsetzungsanweisung ausgeführt, und i um 1 erhöht. Bis x den Wert 6 erreicht hat, wiederholt sich diese Aktion.

1. 1. 8 Kommentare

Für eine bessere Lesbarkeit werden Kommentare in den eigenen Code eingebaut. In C++ gibt es zwei Möglichkeiten seinen Code zu kommentieren.
Einzeilige Kommentare werden durch zwei Slashes "//" eingeleitet. Sie können an einer beliebigen Stelle gestartet werden und enden immer am Ende der Zeile.
Mehrzeilige Kommentare werden mit einem Slash und Stern "/*" gestartet und durch einen Stern und Slash beendet "*/". Alles innerhalb dieser Zeichen wird nicht vom Programm ausgeführt. Dies ist sehr hilfreich um mehrere Zeilen Code zu deaktivieren ohne diese löschen zu müssen.

1. 1. 9 Die Entwicklungsumgebung (IDE)

Die Integrierte Entwicklungsumgebung (IDE) ist ein Tool für die Softwareentwicklung. Dazu stellt es verschiedene Bibliotheken und Werkzeuge in einer Oberfläche bereit, welche die Programmierung erleichtern. Um die geschriebene Software für Maschinen lesbar zu machen, muss diese in Maschinencode umgewandelt werden, das übernimmt die IDE.

1. 2 Der Mikrocontroller

Ein Mikrocontroller ist ein programmierbarer Baustein. Er besteht in den meisten Fällen aus einem Prozessor und zugehörigen Komponenten wie Speicher und digitale oder analoge Ein- und Ausgänge.

1. 2. 1 ESP32

Der Mikrocontroller ESP32 (https://www.espressif.com/en/products/hardware/esp32/overview) der Firma Espressif ist ein stromsparender und sehr günstiger Mikrocontroller, welcher speziell für das Internet der Dinge entwickelt wurde. Der Grund dafür ist die Kombination eines WLAN-Moduls mit der 32-Bit-CPU mit bis zu 240 MHz Taktfrequenz, einem Arbeitsspeicher von ~500kB und einem zusätzlich verbauten Flash Speicher mit bis zu 4x16 Mbytes und verschiedenen Ein- und Ausgängen. Das alles ergibt ein perfektes Entwicklerboard für die Maker-Szene.

2 Einrichtung der Entwicklungsumgebung mit Arduino

2. 1 Arduino IDE

2. 1. 1 Was ist die Arduino IDE?

Die Arduino IDE ist eine mögliche Entwicklungsumgebung für Arduino Boards. Sie erlaubt es Software in C/ C++ zu schreiben und auf einen passenden Mikrocontroller zu laden.

2. 1. 2 Installation

In diesem Abschnitt wird die Installation der Arduino IDE auf einem Windows PC erklärt. folgende Voraussetzungen müssen vorhanden sein um eine vollständige Installation und Inbetriebnahme des Mikrocontrollers vorzunehmen:

- Computer mit Internetzugang
- Micro-USB Kabel
- ESP32 Mikrocontroller (Enthalten im Bionik-Baukasten)

Als Erstes wird auf dem Computer die IDE eingerichtet, mit welcher eine Vielzahl von Mikrocontrollern programmiert werden kann. Auf der Arduino Webseite (https://www.arduino.cc/en/Main/Software) findet man die notwendigen Dateien, um die Umgebung auf deinem Rechner zu installieren. Für Windows Computer wird der "Windows Installer" heruntergeladen, die Installation ausgeführt und anschließend die Arduino Software gestartet. Jetzt ist der Nutzer in der Lage verschiedenste Mikrocontroller zu programmieren. Für die Entwicklung mit einem ESP32 müssen noch für das Board relevante Information zu der IDE hinzugefügt werden.
Um mit der Arduino IDE das ESP32 programmieren zu können, müssen folgende Schritte ausgeführt werden. Diese sind zusätzlich zu dieser Erklärung auf Englisch unter dem Link (https://github.com/espressif/arduino-esp32) nachzulesen.

2. 1. 2. 1 Installation Arduino IDE

Laden und installieren Sie die Arduino IDE von der offiziellen Arduino Webseite herunter (https://www.arduino.cc/en/Main/Software).

2. 1. 2. 2 Installation Git

Git ist eine Software zur Versionsverwaltung von Dateien. In diesem Fall wird es genutzt, einen aktuellen Stand der Software, zur Benutzung des ESP32 mit der Arduino Umgebung, zu bekommen.
Dafür wird auf der Git Webseite (https://git-scm.com/download) der Download für ihr Betriebssystem gestartet, in dieser Erklärung wird es für Windows heruntergeladen. Anschließend die Installation mit den Standardeinstellungen durchführen. Nach der Installation gibt es im Windows Kontextmenü (Klick auf Rechter Maustaste im Windows Explorer) zwei neue Einträge.
Der Eintrag "Git GUI here" startet die Oberfläche zum Herunterladen von Git Ordnern.
Mit dem Eintrag "Git Bash here" wird eine Git Kommandozeile geöffnet, in der weitere, für unsere Zwecke nicht notwendige Funktionen, zur Verfügung stehen.

Starten Sie "Git Gui" und klicken Sie auf "Clone Existing Repository". Füllen Sie die Eingabemaske folgendermaßen aus (Bild 7) und klicken anschließend auf "Clone".
Source Location: "https://github.com/espressif/arduino-esp32.git"
Target Directory: "C:/Benutzer/[Dein Nutzername]/Dokumente/Arduino/hardware/espressif/esp32"
Nach dem alle Dateien erfolgreich heruntergeladen wurden, erscheint ein neues Fenster, welches geschlossen werden kann.

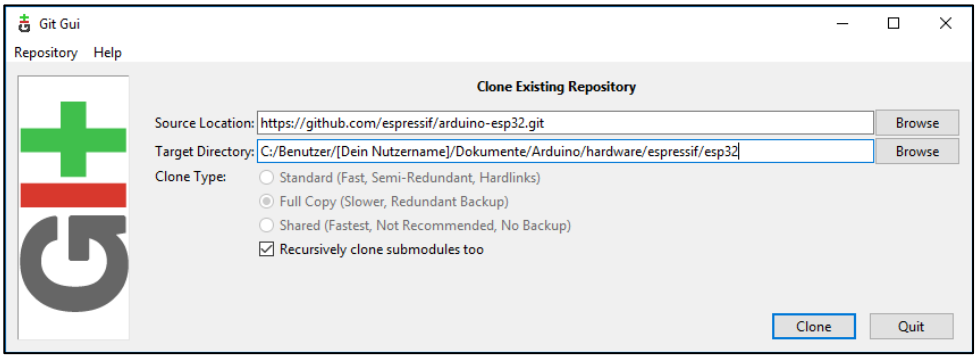


Bild 7

Wechseln Sie im Windows Explorer in den Pfad von "Target Directory" öffnen den Ordner "tools" und starten die "get.exe". Diese lädt weitere notwendige Software für die Entwicklung mit dem ESP32 herunter.
Nun sind alle notwendigen Vorbereitungen abgeschlossen und die Arduino IDE kann gestartet werden.

2. 1. 3 Testen der Entwicklungsumgebung

Für einen vollständigen Funktionstest des ESP32 wird ein Beispielprogramm auf den Mikrocontroller geladen. Dazu wird als Erstes das passende Board ausgewählt. Unter "Werkzeuge -› Board: -› ESP32 Dev Module" auswählen, wie in Bild 8.

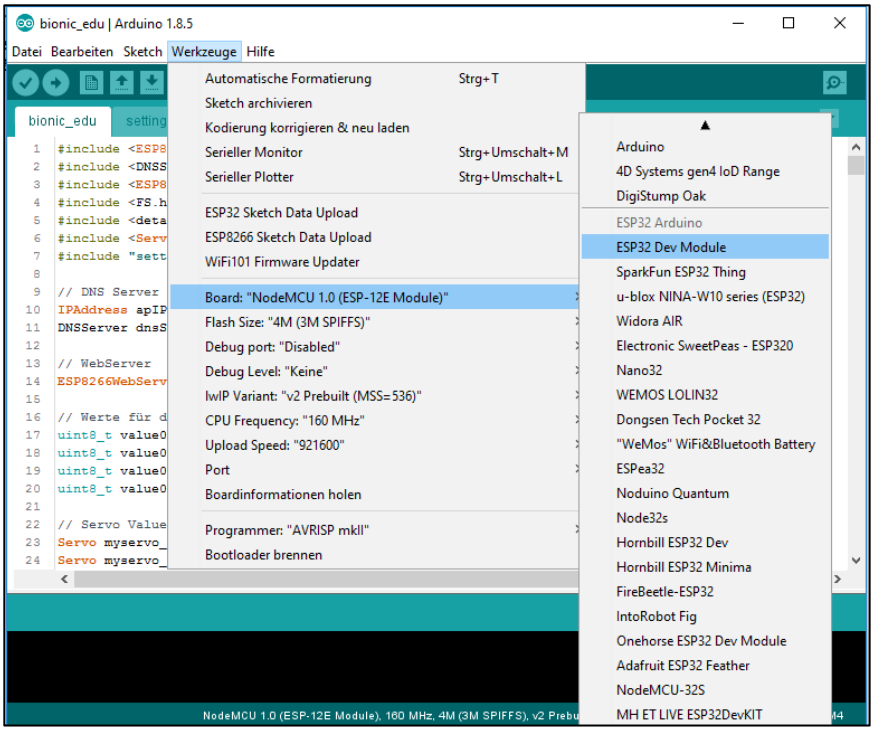


Bild 8

Anschließend unter dem Reiter "Datei -› Beispiele -› 01.Basics -› BareMinimum" auswählen (Bild 9). Es öffnet sich ein neues Fenster mit einer neuen Instanz der Arduino IDE.

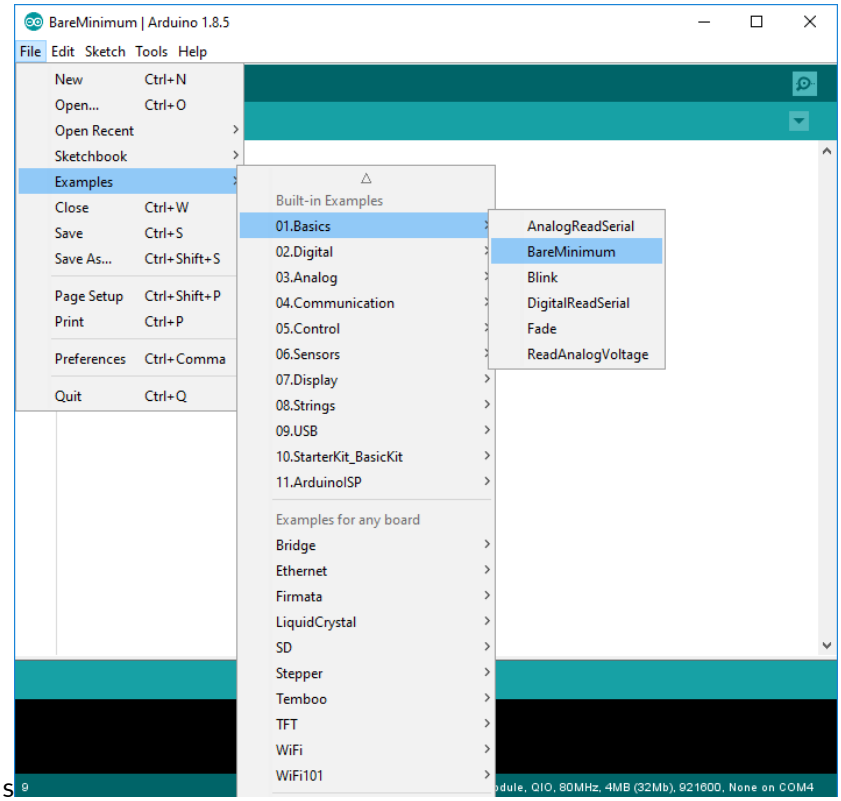


Bild 9

Klickt man unter "Sketch" auf "Upload", wird der Code auf den Mikrocontroller geladen. Es sollten keine Fehlermeldungen im schwarzen Ausgabebereich aufkommen, dann wurde die IDE vollständig eingerichtet.

Im Beispielprogramm "BareMinimum", welches auf das ESP32 geladen wurde, befindet sich kein ausführbarer Code, es dient lediglich zum Testen des Uploads auf den Mikrocontroller. Beim Starten wird die "setup()" Funktion aufgerufen (Zeile 1). Diese beinhaltet alle Initialisierungen für das Programm. In Zeile 6 ist die Funktion "loop()", welche unendlich oft aufgerufen und durchlaufen wird.

Jetzt ist man bereit alle weiteren Entwicklungen, wie das Anschließen einer LED oder verschiedenster Sensoren, vorzunehmen.

Happy Coding