


| | | |
|---|---------------------------------|---|
|  | IT: Hardwarenahes Programmieren | Name: Rahm Datum: 18.09.2022 1_3_Blink.docx |
| | Programmierung: Blink | 1.3.1 |

Beispiel: „Blink-LED“

Am „Blink-LED“ Beispiel wird der Grundaufbau eines Arduino-C Programms erläutert. Die LED auf dem Proto-Shield soll mit der Frequenz $f = 1 \text{ Hz}$ blinken. Der Quellcode wird in eine Datei mit der Endung *.ino geschrieben. Dies nennt man auch einen Arduino-Sketch, oder kurz Sketch.

Am Anfang des Programmcodes wird eine Variable vom Typ Integer (= Ganzzahl) festgelegt. Dort wird direkt der Wert des GPIO-Pins gespeichert. Grundsätzlich könnte die Anweisung auch folgendermaßen formuliert werden:

```
#define LEDpin 32
```

In der Funktion **setup()** wird mit dem Aufruf von **pinMode()** die Datenrichtung des GPIO festgelegt. Eine LED benötigt einen Ausgang (OUTPUT) des Controllers.

```
const int LEDpin = 32;

void setup()
{
  pinMode(LEDpin,OUTPUT);
}

void loop()
{
  digitalWrite(LEDpin,0);
  delay(500);
  digitalWrite(LEDpin,1);
  delay(500);
}
```

Die Funktion **loop()** wird vom Arduino-Code zyklisch in einer Endlosschleife aufgerufen. Der **digitalWrite()**-Befehl gibt ein 1- oder 0-Signal am GPIO aus. Alternativ könnte man hier auch die global definierten Konstanten **HIGH** und **LOW** einsetzen. In der **delay()**-Funktion wird eine Programmschleife ausgeführt, die insgesamt so viele Millisekunden Ausführungszeit benötigt, wie im Parameter übergeben wird. Also hier 500ms.

Den allgemeinen Aufbau eines Arduino-Sketch zeigt noch einmal das nebenstehende Schema. Selbstverständlich können, wie in allen C-Programmen, auch weitere Funktionen definiert werden.

Globale Definitionen

- Bibliotheken einbinden
- Globale Variablen/Objekte



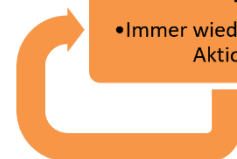
setup()

- Initialisierungen
- Einmalige Aktionen



loop()

- Immer wiederkehrende Aktionen



Arbeitsauftrag

1. Geben Sie den Code im geöffneten Sketch ein und speichern Sie die Datei unter einem sinnvollen Namen im Sketchordner ab.
2. Auf der Funktionsleiste der Arduino-IDE befinden sich wenige Bedien-Schaltflächen:




Verifizieren
= Compilieren

Verifizieren
mit Download
auf die Zielhardware

Serialer Monitor

Verifizieren Sie nun den Code und übertragen (downloaden) Sie ihn anschließend auf den ESP-Controller.

3. Testen Sie die Funktion des Programms.

| | | |
|--|---------------------------------|---|
|  Friedrich-Ebert-Schule Esslingen FES | IT: Hardwarenahes Programmieren | Name: Rahm Datum: 18.09.2022 1_3_Blink.docx |
| | Programmierung: Blink | 1.3.2 |

Syntax der Arduino I/O-Befehle

```
pinMode( GPIO, mode)
  GPIO:  0 ... 37
  mode:  OUTPUT, INPUT, INPUT_PULLUP

digitalWrite( GPIO, wert )
  wert:  LOW  = 0
         HIGH = 1

wert = digitalRead( GPIO )
wert:  LOW/HIGH
```

Beispiel: Blinken mit Taster Freigabe und seriellem Debugging

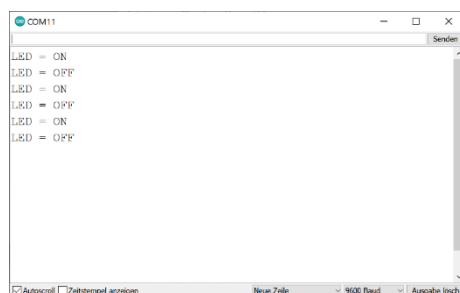
Die LED soll nun erst blinken, wenn der Taster betätigt ist. Zusätzlich erfolgt eine Status-Ausgabe über den seriellen Monitor.

Arbeitsauftrag

1. Erweitern Sie den „Blink“-Sketch wie dargestellt und übertragen Sie das Programm auf die Zielhardware.

Achtung: Achten Sie bei Schlüsselwörtern, Namen und Variablen immer auf Groß-/Kleinschreibung.

2. Testen und Analysieren Sie den Programm-Code. Öffnen Sie dazu auch den seriellen Monitor.



```
const int LEDpin = 32;
const int TasterPin = 18;

void setup()
{
  Serial.begin(9600);
  pinMode(LEDpin,OUTPUT);
  pinMode(TasterPin,INPUT_PULLUP);
}

void loop()
{
  while(digitalRead(TasterPin)==0)
  {
    digitalWrite(LEDpin,0);
    Serial.println("LED = ON");
    delay(500);
    digitalWrite(LEDpin,1);
    Serial.println("LED = OFF");
    delay(500);
  }
}
```