Cheatsheet Arduino Serial

DI(FH) Andreas Pötscher, HTL Litec

Initialisierung

1. `Serial.begin(Baudrate)`:
   - **Beschreibung**: Initialisiert die serielle Kommunikation mit der angegebenen Baudrate (Bits pro Sekunde).
   - **Parameter**: `Baudrate` (z.B. 9600, 115200).
   - **Beispiel**: `Serial.begin(9600);`

Senden

2. `Serial.print(Daten)`:
   - **Beschreibung**: Sendet Daten über die serielle Schnittstelle ohne Zeilenumbruch. Kann verschiedene Datentypen wie Strings, Zahlen usw. senden.
   - **Parameter**: `Daten` (kann ein String, eine Zahl oder ein anderes Datentyp sein).
   - **Beispiel**: `Serial.print("Hallo, Welt!");` oder `Serial.print(123);`
3. `Serial.println(Daten)`:
   - **Beschreibung**: Sendet Daten über die serielle Schnittstelle mit einem Zeilenumbruch. Kann verschiedene Datentypen wie Strings, Zahlen usw. senden.
   - **Parameter**: `Daten` (kann ein String, eine Zahl oder ein anderes Datentyp sein).
   - **Beispiel**: `Serial.println("Hallo, Welt!");` oder `Serial.println(123);`
4. `Serial.write(Daten)`:
   - **Beschreibung**: Sendet Daten als Rohbytes über die serielle Schnittstelle.
   - **Parameter**: `Daten` (kann ein einzelnes Byte oder ein Array von Bytes sein).
   - **Beispiel**: `Serial.write(65);` // Sendet das ASCII-Zeichen 'A'

Empfangen

5. `Serial.available()`:
   - **Beschreibung**: Gibt die Anzahl der Bytes zurück, die im seriellen Puffer verfügbar sind.
   - **Rückgabewert**: Anzahl der verfügbaren Bytes.
   - **Beispiel**: `int availableBytes = Serial.available();`
6. `Serial.read()`:
   - **Beschreibung**: Liest ein einzelnes Byte aus dem seriellen Puffer.

- **Rückgabewert**: Das gelesene Byte.
- **Beispiel**:

```
void loop() {
  if (Serial.available() > 0) {
    char incomingByte = Serial.read();
    Serial.print("I received: ");
    Serial.println(incomingByte);
  }
}
```

7. `Serial.readString()`:

- **Beschreibung**: Liest eine Zeichenkette aus dem seriellen Puffer bis zu einer bestimmten Zeitüberschreitung (Timeout). Standardwert für das Timeout ist eine Sekunde. Das bedeutet die Funktion wartet eine immer eine Sekunde (oder andere Timeout Einstllung) und gibt dann alle bis dahin empfangenen Daten als String zurück.
- **Rückgabewert**: Die gelesene Zeichenkette.
- **Beispiel**:

```
void loop() {
  if (Serial.available() > 0) {
    //wartet 1 sek (timeout)
    String incomingString = Serial.readString();
    Serial.print("I received: ");
    Serial.println(incomingString);
  }
}
```

8. `Serial.readBytes(Puffer, Länge)`:

- **Beschreibung**: Liest eine bestimmte Anzahl von Bytes in einen Puffer bis zu einer bestimmten Zeitüberschreitung (Timeout). Standardwert für das Timeout ist eine Sekunde. Die Funktion wartet entweder die Timeout Zeit oder bis der Puffer voll ist und gibt dann den Puffer zurück.

- **Parameter**: `Puffer` (ein Array, in das die Bytes gelesen werden), `Länge` (die Anzahl der zu lesenden Bytes).
- **Rückgabewert**: Die Anzahl der tatsächlich gelesenen Bytes.
- **Beispiel**:

```
void loop() {
  if (Serial.available() > 0) {
    char buffer[10];
    //wartet bis der Puffer voll ist oder max 1sek (timeout)
    int bytesRead = Serial.readBytes(buffer, 10);
    Serial.print("I received: ");
    for (int i = 0; i < bytesRead; i++) {
      Serial.print(buffer[i]);
    }
    Serial.println();
  }
}
```

9. `Serial.flush()`:
    - **Beschreibung**: Wartet, bis alle ausgehenden seriellen Daten gesendet wurden.
    - **Beispiel**: `Serial.flush();`
10. `Serial.setTimeout(ms)`:
    - **Beschreibung**: Setzt die Zeitüberschreitung für das Lesen von Daten in Millisekunden. Wird die Funktion nicht aufgerufen ist die Standardeinstellung für das Timeout 1sek.
    - **Parameter**: `ms` (Zeitüberschreitung in Millisekunden).
    - **Beispiel**: `Serial.setTimeout(1000);` // Setzt die Zeitüberschreitung auf 1 Sekunde