Tasten abfragen

Bisher hatten immer nur als Ausgänge geschaltet und hierfür folgende Befehle benutzt:

pinMode(pinNummer, OUTPUT)
digitalWrite(pinNummer, HIGH) oder **digitalWrite(pinNummer, LOW)**

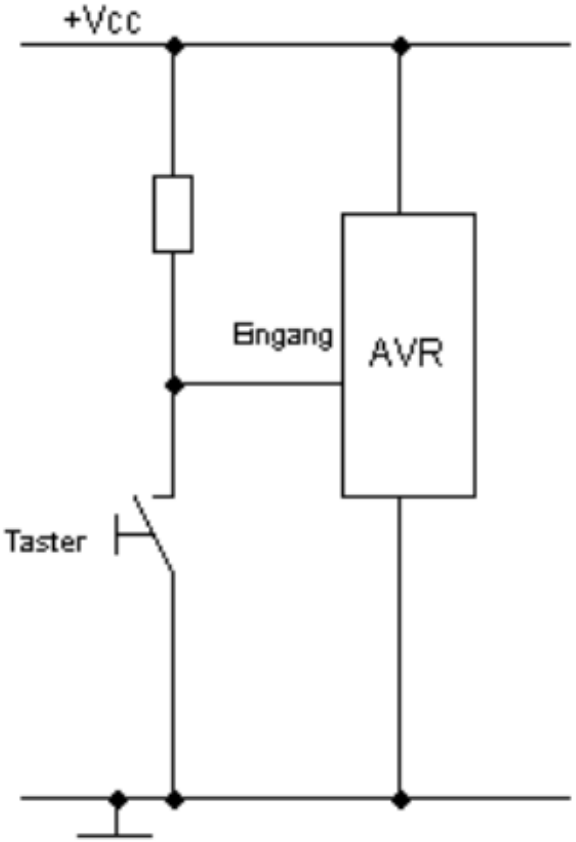
Zum Abfragen von Tasten und Schalter können die Ports aber auch als Eingänge benutzt werden.

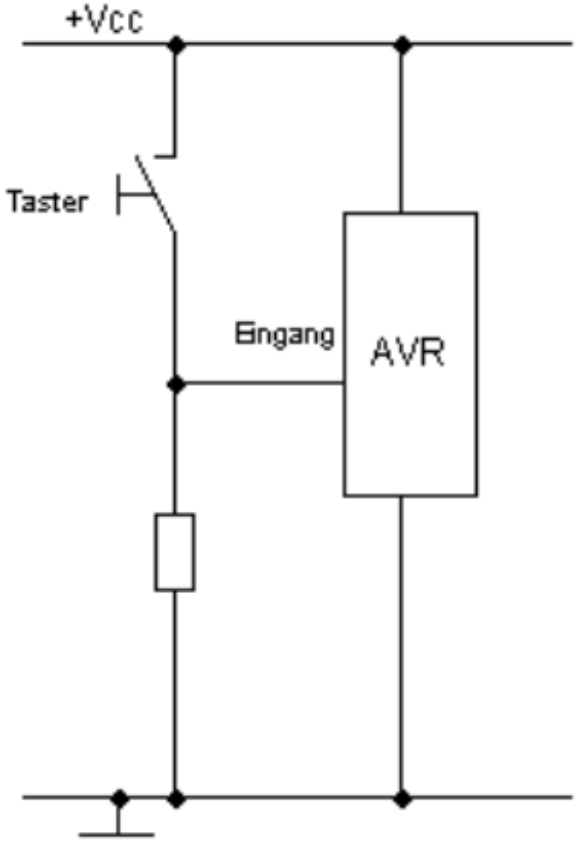
Mit **pinMode(pinNummer, INPUT)** deklarieren wird den Port als Eingang.

Damit die Ports eindeutige Zustände von HIGH oder LOW haben könnte man an die Pins externe Pullup Widerstände von z.B. 10 Kohm anschließen und diese mit + 5V verbinden. Eine Taste zwischen Masse (0V) und dem Pin würde die Spannung kurzschließen und auf 0 V Pegel bringen. Dieser eindeutige Zustand von LOW oder HIGH könnte anschließend ausgewertet werden.

Statt externe Widerstände zu benutzen kann man auch auch interne Pullup Widerstände auf die Pins zu schalten. Dies erfolgt mit dem Befehl

digitalWrite(inPin, HIGH)





Pullup mit High - Pegel

digitalWrite(pinNummer, HIGH)

Pullup gegen Low ist auch möglich.

digitalWrite(pinNummer, LOW)

Der Zustand (**LOW** oder **HIGH**) kann nun eingelesen werden mit:

```
int val = 0;
val = digitalRead(pinNummer);
```

Es wird zuvor eine Variable deklariert, die daß Ergebnis der Abfrage speichert. Danach wird der Status abgefragt und der Variable zugewiesen.

Das Ergebnis kann nur die beiden Zustände **0** (LOW) oder **1** (HIGH) annehmen.

Anschließend kann man mit einer **if .. then** Abfrage den Wert auswerten und entsprechende Reaktionen auslösen:

```
if (val == 0)
{
// Taste wurde gedrückt, wenn val den Wert 0 enthält - schalte LED ein
 digitalWrite(ledPin1, HIGH);
}
else
{
// Taste wurde nicht gedrückt, wenn val den Wert 1 enthält - schalte LED aus
digitalWrite(ledPin1, HIGH);
}
```

Achtung: Bei Vergleichsoperationen ist == erforderlich !

If Abfragen können folgende Form haben:

```
if (x > 120) digitalWrite(LEDpin, HIGH);

if (x > 120)
digitalWrite(LEDpin, HIGH);

if (x > 120){ digitalWrite(LEDpin, HIGH); }

if (x > 120)
{
  digitalWrite(LEDpin1, HIGH);
  digitalWrite(LEDpin2, HIGH);
}
```

x == y (x ist gleich y)
x != y (x ist nicht gleich y)
x < y (x ist kleiner als y)
x > y (x ist größer als y)
x <= y (x ist kleiner oder gleich y)
x >= y (x ist größer oder gleich y)

Durch Hinzufügen von **else** werden Befehle ausgeführt, wenn die Bedingung nicht wahr ist:

```
if (x > 120)
{
digitalWrite(LEDpin1, HIGH); // x > 120
}
else
{
digitalWrite(LEDpin1, HIGH);  // x < 120
}
```

Praktisches Beispiel - Taster wird abgefragt und LED eingeschaltet

Bitte LED mit Pin13 = PB5 und Taster mit Pin8 = PB0 verbinden

```
// Input und Output

int ledPin1 = 8;                    // LED Pin8 - PB0
int inPin = 13;                     // Taster Pin13 - PB5
int val = 0;                        // Variable für Eingangswert

void setup()
{
  pinMode(ledPin1, OUTPUT);         // Setze Pin8 als Ausgang
  pinMode(inPin, INPUT);            // Setze Pin13 als Eingang
  digitalWrite(inPin, HIGH);        // Pull Up Eingang auf High
 }

void loop()
{
  val = digitalRead(inPin);         // Lese Eingangs - Pin

if (val == 0)                       // Schalte LED
{
   digitalWrite(ledPin1, HIGH);
}
else
{
   digitalWrite(ledPin1, LOW);
}
   }
```

Beim letzten Beispiel wurde die LED durch Tastendruck eingeschaltet.

Es wäre aber auch möglich, daß beim Tastendruck die LED eingeschaltet und erst beim nächsten Tastendruck wieder ausgeschaltet wird (nächstes Beispiel).

Hierbei wird bei jedem Tastendruck der Inhalt einer Hilfsvariable um 1 erhöht.

einaus = einaus + 1;

Sobald die Variable den Wert **2** erreicht hat, wird diese wieder auf **0** zurück gesetzt.

if (einaus == 2) einaus = 0;

Die Variable kann somit nur die Zustände **0** und **1** erreichen. Dieser Wert wird dann in einer weiteren **if – Abfrage** ausgewertet.

Durch **delay(1000)** wird eine Verzögerung von 1 Sekunde eingefügt.

// Einschalten - Ausschalten

```
int ledPin1 = 13;                       // LED verbunden mit Pin13 – PB5
int inPin = 8;                          // Taster verbunden mit Pin8 – PB0
int val = 0;                            // Variable für Eingangswert
int einaus = 0;                         // Status Ausgang

void setup()
{
  pinMode(ledPin1, OUTPUT);             // setze Pin8 als Ausgang
  pinMode(inPin, INPUT);                // setze Pin13 als Eingang
  digitalWrite(inPin, HIGH);            // Pull Up Eingang auf High Level
 }

void loop()
{
   val = digitalRead(inPin);            // Lese Eingangs - Pin

 if (val == 0)
 {
    delay(1000);                        // Warte 1 Sekunde
    einaus = einaus +1;

  if (einaus == 2) einaus = 0;          // einaus darf nur 0 oder 1 enthalten
  }

 if (einaus == 1)                       // Schalte LED wenn einaus = 1
{
    digitalWrite(ledPin1, HIGH);
}
else
{
   digitalWrite(ledPin1, LOW);
}
   }
```

Serielle Schnittstelle(1)

Nachdem wir nun die Arduino Input - Funktion kennengelernt haben, benötigen wir nun eine Möglichkeit eingegebenen Daten anzeigen zu können. Im einfachsten Fall geht dies über LED‘s. Eine etwas komfortablere Variante wäre die Nutzung der seriellen Schnittstelle (**RS232 TTL**). Der Arduino verfügt hierfür über die Anschlüsse

TX Pin 0

RX Pin 1

Zum Aktivieren der Schnittstelle fügen wir unter Setup den folgen Befehl ein.

```
void setup
{
Serial.begin(9600);
}
```

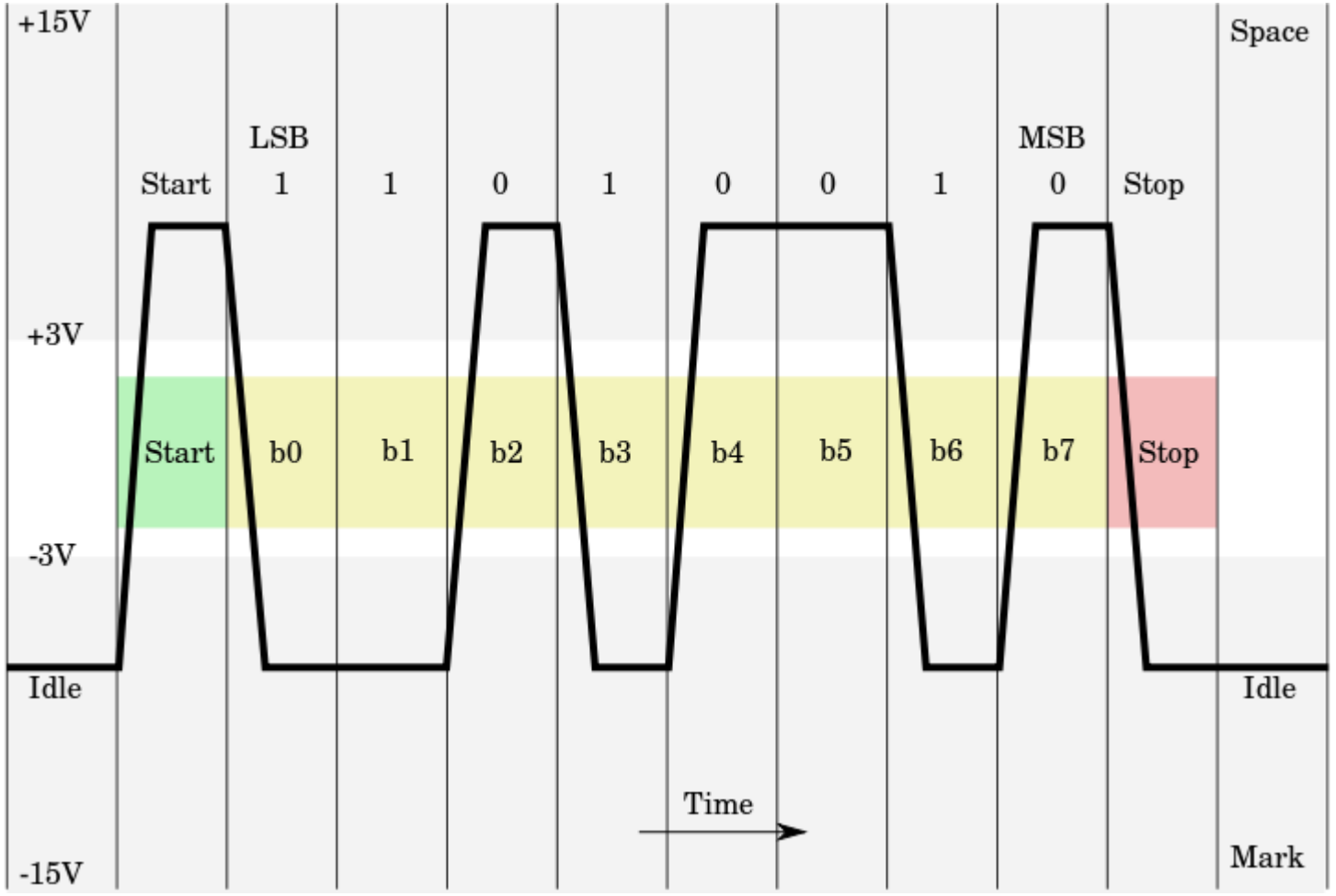
Als Baudrate sind möglich: 300, 600, 1200, 2400, 4800, **9600**, 14400, 19200, 28800, 31250, 38400, 57600, and 115200

Der Arduino Mega verfügt über 4 Serial Ports:
Serial1 Pins 1 (RX) und 0 (TX), **Serial2** Pins 19 (RX) und 18 (TX),
Serial3 Pins 17 (RX) und 16 (TX), **Serial4** Pins 15 (RX) und 14 (TX)

Zu beachten ist Arduino RS232 Schnittstelle mit 5 V Pegeln arbeiten - PC Schnittstellen dagegen mit +/- 12 V.

Serielle Schnittstelle(2)

Datenformat der RS232 Schnittstelle



Serielle Schnittstelle(3)

Daten werden über die serielle Schnittstelle gesendet mit:

```
void setup
{
Serial.begin(9600);

Serial.println(144);                    // Als ASCII-encoded decimal
Serial.println(„Hello World“);          // als String
```

optional

```
Serial.println(Value, DEC);             // print as an ASCII-encoded decimal
Serial.println(Value, HEX);             // print as an ASCII-encoded hexadecimal
Serial.println(Value, OCT);             // print as an ASCII-encoded octal
Serial.println(Value, BIN);             // print as an ASCII-encoded binary
}
```

An jede Zeile wird ein Return - Zeichen angehängt.

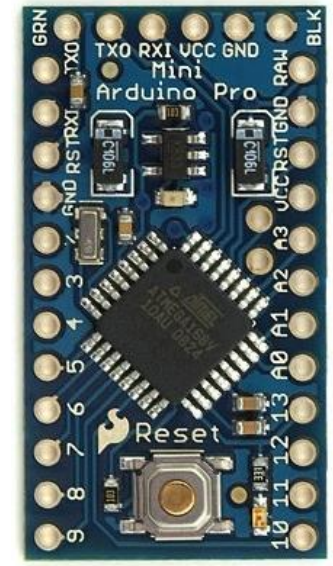
Mit **Serial.print(value)** werden die Zeichen hintereinander ohne Return gesendet.

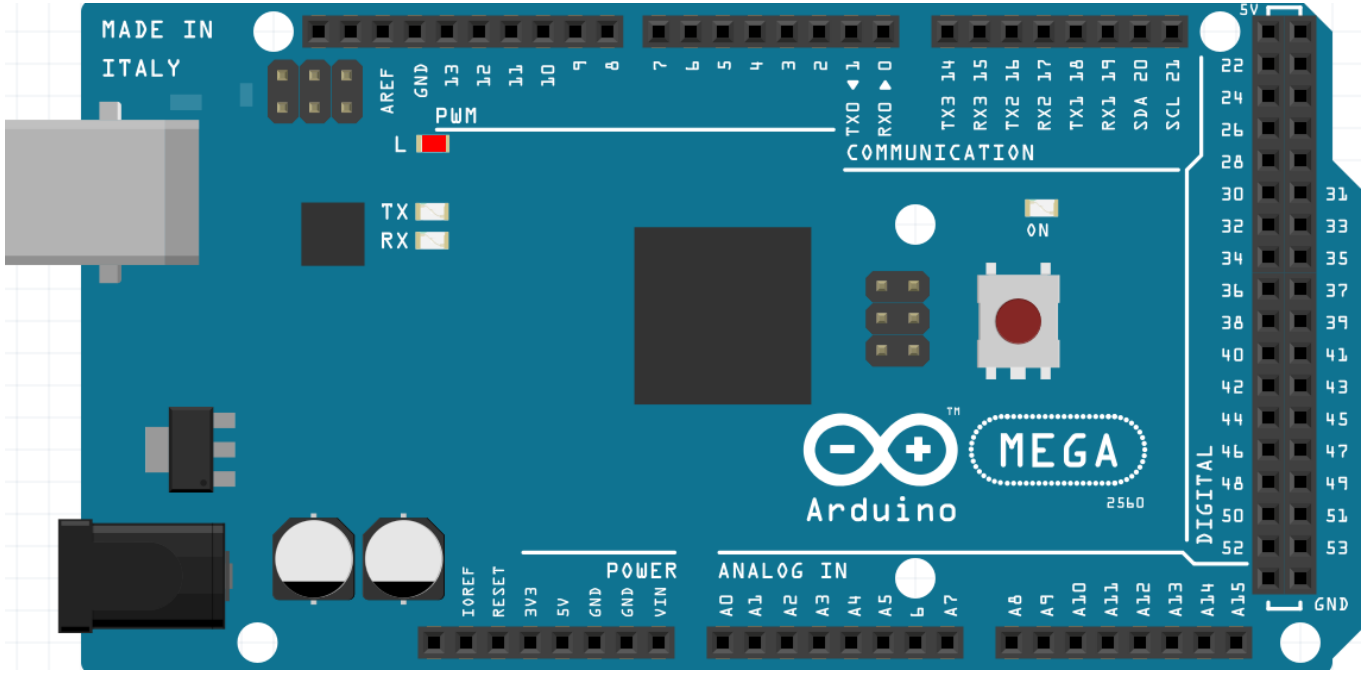
Serielle Schnittstelle(4)

Der Arduino Mega verfügt über eine USB - Schnittstelle und einen virtuellen Com Port. Diese ist intern mit der seriellen Schnittstelle an Pin 0 (TX) und Pin 1 (RX) verbunden. Ein Arduino Mini Pro hat keine USB Schnittstelle – hier ist ein externer Adapter erforderlich.

RS232

USB
Virtueller Com Port





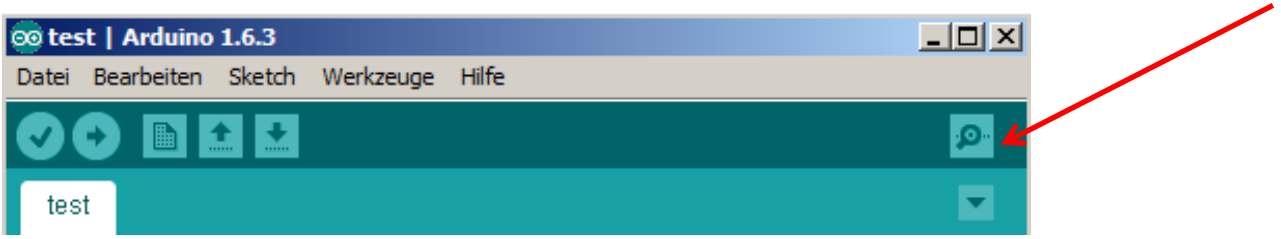
Serielle Schnittstelle(5)

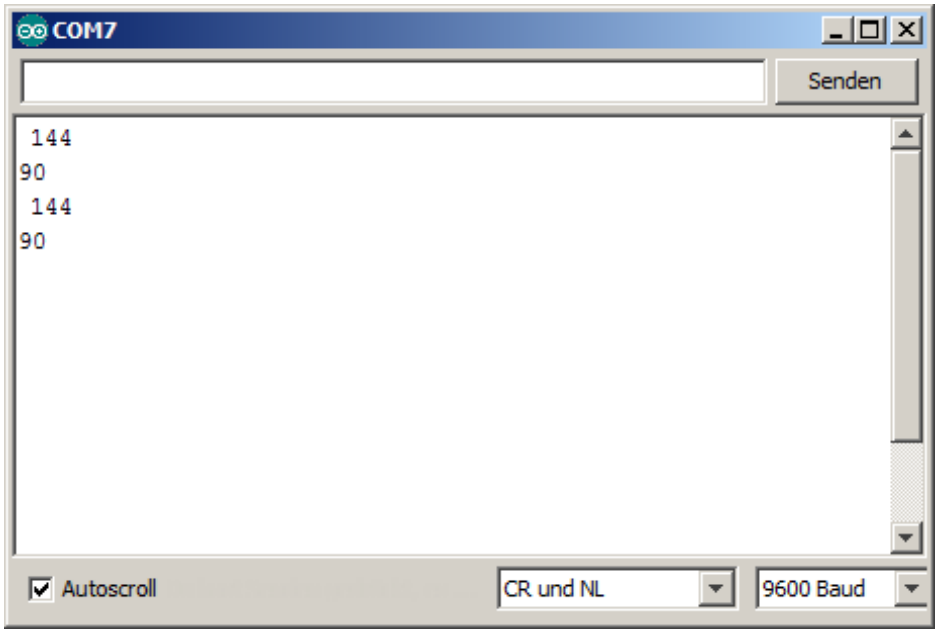
Mit dem Befehl **Serial.read()** können Daten über die serielle Schnittstelle eingelesen werden.

```
int incomingByte = 0;          // für eingehende serielle Daten

void setup() {
    Serial.begin(9600);        // Öffne Seriellen Port mit 9600 bps
}

void loop() {

    // Sende nur wenn Daten empfangen wurden:
    if (Serial.available() > 0) {
        // Lese die eingehenden Bytes:
        incomingByte = Serial.read();

        // Sende die empfangenen Daten zurück:
        Serial.print(„Ich habe folgendes empfangen: ");
        Serial.println(incomingByte, DEC);
    }
}
```

Serielle Schnittstelle(6)

Zum Anzeigen der Daten brauchen wir ein Terminal - Programm. Dieses ist in die Arduino - IDE bereits integriert (Button ganz rechts).





Serielle Schnittstelle(7)

Nun könnten wir Informationen über Input – Werte über die serielle Schnittstelle anzeigen.

```
// Input / Output mit Serial

int ledPin1 = 13;                       // LED Pin13 - PB5
int inPin = 8;                          // Taster Pin8 - PB0
int val = 0;                            // Variable für Eingangswert

void setup()
{
  pinMode(ledPin1, OUTPUT);             // setze Pin13 als Ausgang
  pinMode(inPin, INPUT);                // setze Pin8 als Eingang
  digitalWrite(inPin, HIGH);            // Pull Up Eingang auf High
  Serial.begin(9600);                   // Oeffne Serial mit 9600
 }

void loop()
{
  val = digitalRead(inPin);             // Lese Eingangs - Pin

if (val == 0)                           // Schalte LED
{
   digitalWrite(ledPin1, HIGH);
   Serial.println("Taste ein ... ");
}
else
{
   digitalWrite(ledPin1, LOW);
   Serial.println("Taste aus ... ");
}
  }
```