der Arduino Mikrocontroller

Der Arduino hat verschiedene Pins an denen Elektronikbauteile angeschlossen werden können. Am **5V**-Pin liegen konstant 5V an und der **GND**-Pin kann als Erdung genutzt werden. Mit den **analog**-Pins kann eine Spannung gemessen werden und die **digital**-Pins wirst du nutzen um eine Spannung ein- und auszuschalten.

Mit Hilfe des Steckbrettes kannst du deine Schaltungen aufbauen. Die Löcher in der Mitte sind jeweils horizontal miteinander verbunden und die Löcher an den Seiten vertikal. Verbinde den 5V-Pin mit einem Loch auf einer der beiden roten Vertikalen und einen GND-Pin mit der Vertikalen daneben.

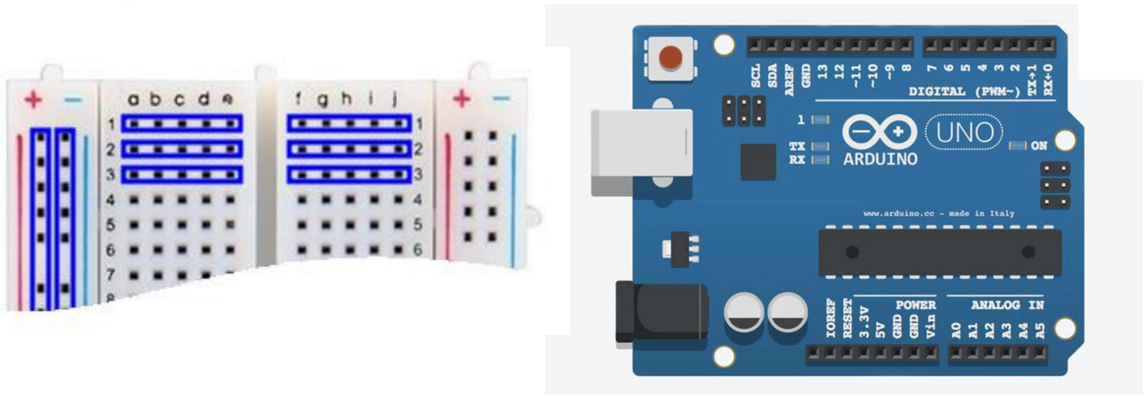

Figure 1: ArduinoBoard

Nutzliche Informationen findest du unter https://www.arduino.cc/reference/de/ und natürlich auf google.de (:

0. Vorbereitung

- Verbinde den Arduino mit dem Computer und starte die Arduino-IDE.
- Wähle unter Tools->Port den Port mit der Beschreibung “Arduino/Genuino Uno” aus.
- Die setup-Funktion wird beim Start einmal ausgeführt. Initialisiere hier die Ausgabe mit `Serial.begin(9600);`.
- Die loop-Funktion wird wiederholt aufgerufen. Hier kannst du mit `Serial.print("Hallo ");` Text ausgeben. Der Befehl `Serial.println("WO");` führt zusätzlich einen Zeilenumbruch durch.
- Übertrage dein Programm auf das Board (“Pfeil nach rechts” Symbol links oben).
- Öffne die Ausgabe mit dem Lupen-Symbol rechts oben.

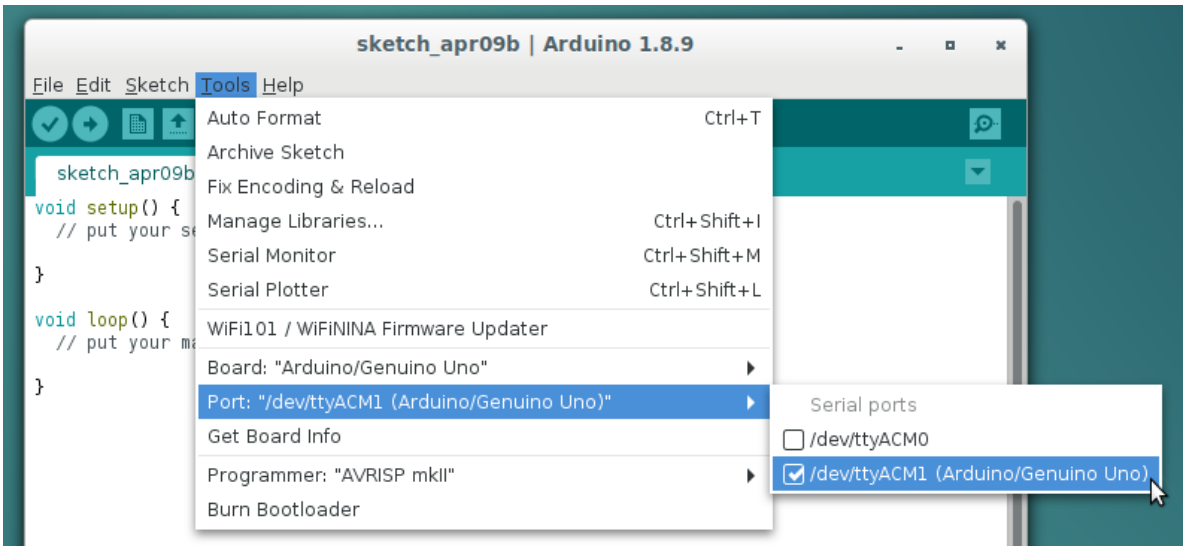

Figure 2: SerialPort

1. LED

- Verbinde das längere Ende der LED mit Pin 13 und das Kürzere mit einem 330 Ohm Widerstand. Das andere Ende des Widerstandes kommt an GND.
- Es ist eine gute Angewohnheit die Belegung der Pins einmalig am Anfang des Programms als Variable festzulegen. Eine globale und konstante Variable des Typs int kannst du wie folgt erstellen:

  ```
  const int LED_PIN = 13;
  ```

- Digitale Pins können als Ein- oder Ausgänge genutzt werden. Lege fest, dass der LED-Pin als Ausgang genutzt wird. Füge hierzu den Befehl `pinMode(LED_PIN, OUTPUT);` zur setup-Funktion hinzu.
- Jetzt kannst du die LED in der loop-Funktion mit `digitalWrite(LED_PIN, HIGH);` einschalten und mit `digitalWrite(LED_PIN, LOW);` auszuschalten. Mit `delay(1000);` wird die Ausführung für 1000 Millisekunden pausiert. Bringe die LED zum Blinken.

2. Temperatur- und Feuchtigkeitssensor DHT11

- Verbinde den DHT11-Sensor mit dem Arduino.
- Am besten nutzt du 5V und GND vom Steckbrett und nicht direkt vom Arduino, wie es im Schaltplan gezeigt wird.
- Informiere dich unter https://github.com/winlinvip/SimpleDHT wie du den Sensor nutzen kannst.

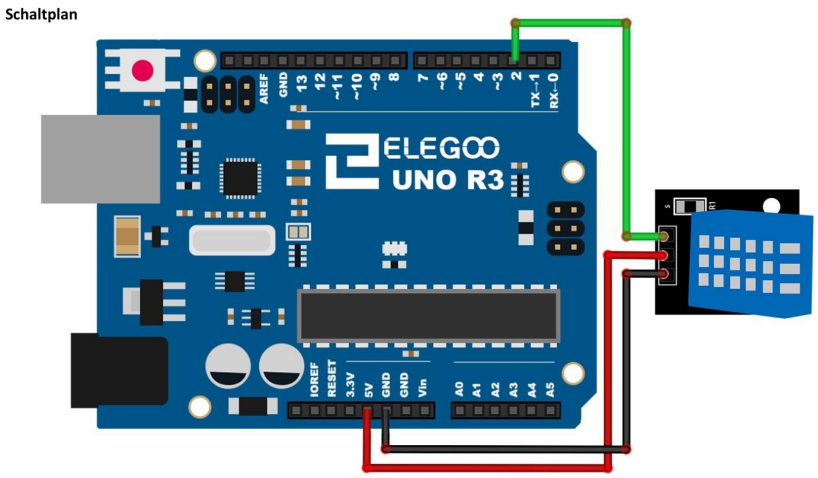

Figure 3: DHT

- Beachte dabei, dass du den Sensor “DHT11” nutzt und du zuerst die Bibliothek SimpleDHT installieren musst.
- Lies die Temperatur und Feuchtigkeit aus und gebe sie aus.
- Aktiviere die LED, falls die Feuchtigkeit mehr als 80% beträgt.

3. Übertragung an den PC

Die Messwerte sollen nun vom Arduino an den PC übertragen werden. Hierzu benötigen wir ein Programm, welches auf dem PC läuft. Mit Hilfe der Processing-IDE und dessen Serial-Bibliothek können die Ausgaben des Arduinos eingelesen und visualisiert werden. Alternativ kannst du auch eine Java-IDE deiner Wahl in Kombination mit der RXTX-Bibliothek verwenden. Dies ist jedoch aufwendiger.

- Öffne das Processing-Template und führe es aus.
- Lese mit Hilfe der Serial-Bibliothek von Processing die Ausgaben des Arduinos ein. siehe https://processing.org/reference/libraries/serial/index.html
- Tip: Serial, bufferUntil, serialEvent, readStringUntil
- Ermittle aus der Ausgabe die Messwerte. (vermutlich nützlich: myString.indexOf(), myString.substring())
- Schreibe die Messwerte in eine Datei.