Kabelsalat

Kurzbeschreibung

Innerhalb des Projektes „Kabelsalat“ wollen wir der Öffenlichkeit das Prinzip der automatisierten Hydroponik in einem abgeschlossenen System näherzubringen, wobei wir selbst ein hydroponisches System in ein Gewächshaus integrieren.

Github: https://github.com/j-h-f/kabelsalat-lab

Webseite: https://www.kabelsalat-lab.de/

Darstellung bisheriger Arbeit

Arbeitsaufteilung

Hydroponik und Gewächshaus

- Planung und Bau des hydroponischen Systems
- Planung und Bau des Gewächshauses
- Videodreh und -schnitt
- Erstellung eines Tutorials für den Bau

Sensorik

- Bewässerung (Pumpe) **Jan-Hendrik**
- Feuchtigkeitsmessung durch Sensoren **Inken**
- Belichtung durch LED Stripes
- Belüftung des Gewächshauses
- Luftstrom an den Pflanzen
- Electrical Conductivity des Wassers (optional: PH-Wert)
- optional: Wasserstand
- Temperaturmessung- Luft und Wasser **Inken**

Wissenschaftskommunikation

- Pflege dieses Wikis
- Erstellung und Pflege einer Website

Bisherige Arbeit

Hydroponik und Gewächshaus

- Bearbeitung einer Teströhre und Start des Baus eines Prototypen

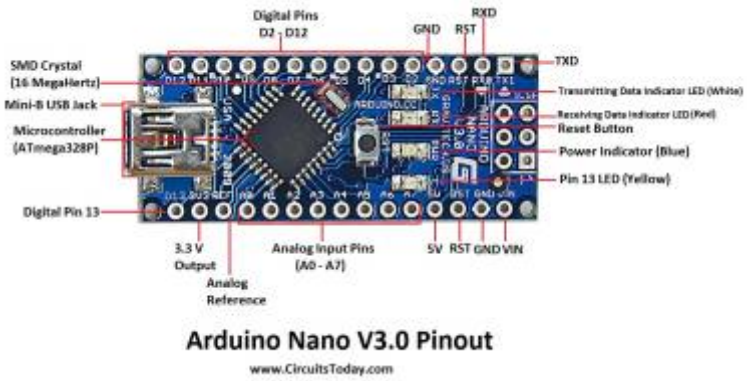
Sensorik

In diesem Abschnitt wird der technische Aufbau des Gewächshausen vor allem im Hinblick auf die verwendete Sensorik beschrieben.

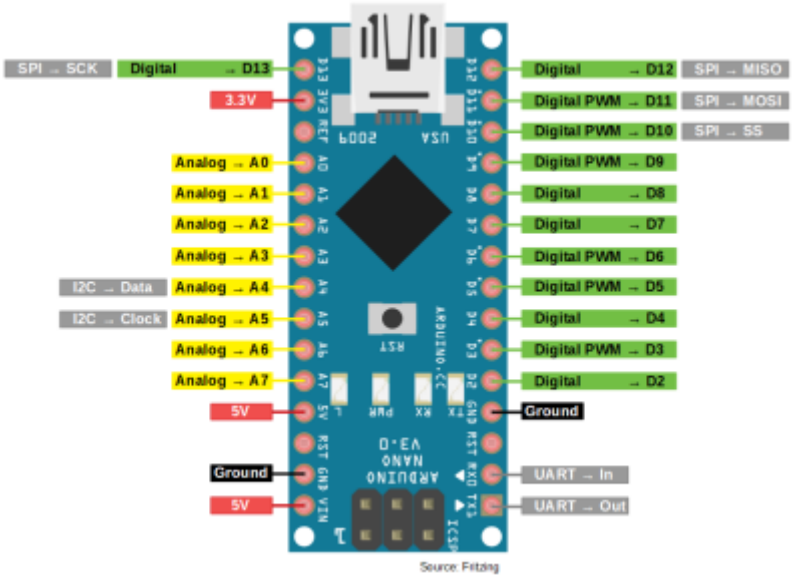
Verwendete Bauteile

- Arduino Nano
- ESP01-S
- Relais-Modul
- Temperatursensor BME/BMP280
- Reich Tauchpumpe
- LED-Streifen für Pflanzen

Arduino Nano



"Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices." [https://en.wikipedia.org/wiki/Arduino]
Der Arduino ist also ein Mikrocontroller, der nach Belieben programmiert werden kann und dann als Hauptsteuereinheit diverser Projekte genutzt werden kann.
Bei einem Mikrocontroller handelt es sich, vereinfacht gesagt, um einen winzigen Computer, bestehend aus einem Prozessor, einem Arbeitsspeicher (RAM) und ggf. weitere Peripheriebauteile, wie bspw. externer Speicher oder I/O-Schnittstellen (USB, I2C, SPI…)
Der große Unterschied zwischen einem richtigen PC und einem Mikrocontroller liegt darin, dass der Mikrocontroller auf einen bestimmen Anwendungsfall zugeschnitten und entwickelt ist. Dies bringt den Vorteil, dass kleine Projekte, wie bspw. unser Hydroponisches Gewächshaus sehr günstig "zum leben erweckt werden können". Das liegt daran, dass die Mikrocontroller aufgrund ihrer beschränkten Rechenleistung und Speicherkapazität sehr günstig sind, aber dennoch ausreichend Funktionalität bieten, um ein etwas komplexeres Projekt zu realisieren.



ESP01-S

Der ESP01-S ist auch ein Mikrocontroller, der auch, ähnlich wie der Arduino, programmiert werden kann, um Steuerungsaufgaben zu übernehmen. Der Unterschied zwischen einem ESP01-S und einem Arduino liegt darin, dass der ESP01-S Mikrocontroller auf dem sogenannten ESP8266 Mikroprozessor basiert. Der Vorteil an dem ESP8266 Mikroprozessor ist, dass dieser einen WLAN-Chip verbaut hat, mit dem der ESP01-S eine Verbindung zum Internet aufbauen kann. Der Nachteil am ESP01-S Mikrocontroller ist. Dass dieser nur 2 I/O Pins besitzt. Das bedeutet, dass man nur über 2 Signalleitungen mit anderen Geräten kommunizieren kann. Somit eignet sich der ESP01-S meist nur für sehr kleine Projekte, wie z.B. ein Thermometer, welches über WLAN ausgelesen werden kann. Alternativ kann man den ESP01-S auch als WLAN-Modul mit einem Arduino verbinden.

Relais Modul

Das Relais-Modul wird benötigt, damit der Arduino Höhere Spannungen und Höhere Ströme mit einem sehr kleinen Steuerungssignal steuern kann. Wenn beispielsweise eine Handelsübliche LED-

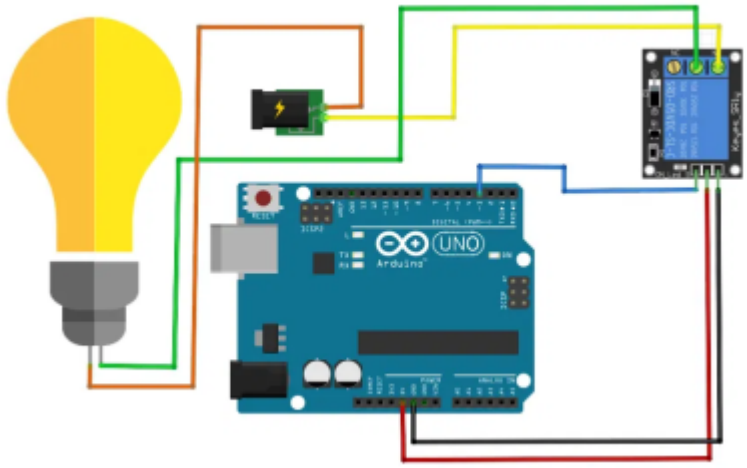
Glühlampe mit einer GU4-Fassung steuern möchte, fällt auf, dass die LED eine Spanung von 12 Volt braucht. Der Arduino hingegen hat aber nur eine Ausgangsspannung von 5V an den GPIO Pins und einen Ausgangsstrom von maximal 40mA (wobei max. 20mA empfohlen sind). Das reicht nicht aus, um die LED, oder Andere Geräte zu steuern, die entweder eine höhere Spannung oder eien größeren Strom brauchen zu betreiben. In diesem Fall schafft unser Relais-Modul abhilfe. Dieses ist im Grunde eine mechanischer Schalter, der durch den Arduino gesteuert werden kann. Der Arduino wird an die 3 (oder mehr) sogenannten „Pin-Header“ des Relais-Moduls angeschlossen. Das Relais-Modul verfügt über mindestens 3 dieser Pin-Header Anschlüsse, die Anzahl ist dabei abhängig, wie viele einzelne Relais auf dem Modul Verbaut sind. Die Bezeichnungen der Anschlüsse lauten wie folgt:

- VCC → ist die Versorgungsspannung und wird an den 5V Pin des Arduinos angeschlossen.
- GND → ist der Ground bzw. Minus Anschluss und wird mit dem GND Anschluss des Arduinos verbunden
- IN1 bis INX → sind die Signal Anschlüsse um die einzelnen Relais zu steuern. Wenn auf dem Modul 4 Relais verbaut sind, dann gibt es die Anschlüsse IN1, IN2, IN3, IN4. Wenn nur 1 Relais Verbaut ist, dann gibt es nur IN1 usw.

Jedes auf dem Relais-Modul verbaute, einzelne Relais verfügt zusätzlich über je 3 Anschlüsse, die mit sogenannen Schraubterminals versehen sind. Diese Anschlüsse dienen dazu die Versorgungsspannung für die zu Schaltenden Geräte enweder an oder aus zu schalten. Die Anschlüsse am Relais haben folgende bezeichnungen:

- COM/CO/C → ist der Hauptanschluss des Relais. An diesen sollte der Plus-Pol der Versorgungsspannung angeschlossen werden.
- NO → ist der „Normally Open“ Anschluss. Das bedeutet, dass dieser Anschluss „geöffnet“ ist, wenn keine Schaltspannung anliegt und somit kein Strom fließen kann. Das Relais schaltet also ab, wenn keine Schaltspannung anliegt.
- NC → ist der „Normally Closed“ Anschluss. Das bedeutet, dass dieser Anschluss „geschlossen“ ist, wenn keine Schaltspannung anliegt und somit Strom fließen kann. Das Relais Schaltet das angeschlossene Gerät als an, wenn keine Schaltspannung anliegt.

Möchte man nun eine Lampe mit einem Solchen Relais schalten, sieht der Aufbau wie folg aus:
<dtable>



Relais-Pin	Arduino Pin
VCC	5V
GND	GND
IN1	D3

Der Anschluss des Relais an den Arduino ist also relativ simpel und benötigt an sich nur 3 Kabel. beim Anschluss der Lampe an das Relais muss man jedoch ein wenig aufpassen. Erstens sollte man genauestens auf die maximalen Werte achten, die ein solches Relais schalten kann. Zweitens ist es nicht ratsam eine Lampe, die mit Netzspannung funktioniert (also direkt 220V aus der Steckdose) denn **bei unsachgemäßem Arbeiten mit Netzspannung besteht Lebensgefahr!**. Es ist Ratsam sich ein Fertiges Netzteil zu besorgen, welches in die Steckdose gesteckt wird, und dann die Netzspannung in 12V Gleichspannung umwandelt. Nichtsdestotrotz ist hier immer noch Vorsicht geboten. Um die 12V aus dem Netzteil auf ein Breadboard zu bekommen bietet es sich an, eine passende Buchse für den Stecker des Netzteiles zu verwenden. Nun kann der Plus-Anschluss der Buchse mit dem COM-Anschluss des Relais verbunden werden, der Minus-Anschluss wird mit dem Minus-Pul der Lampe verbunden. Um nun den Stromkreis zu vervollständigen muss der Plus-Pol der Lampe mit dem NO-Anschluss des relais verbunden werden. Somit ist der Aufbau der Hardware komplett und wenn nun ein Signal von Pin D3 zum Relais Modul gesedet wird, sollte die Lampe leuchten. Der Code um das zu realisieren ist denkbar einfach:

```
int IN_PIN = 3;
void setup() {
  pinMode(IN_PIN, OUTPUT);
  digitalWrite(IN_PIN, LOW);
  Serial.begin(9600);
}
void loop() {
   digitalWrite(IN_PIN, HIGH);
   Serial.println("Lamp is on");
   delay(2500);
   digitalWrite(IN_PIN, LOW);
   Serial.println("Lamp is of");
   delay(2500);
}
```

Der Codeblock sorgt jetzt dafür, dass das Relais alle 2,5 Sekunden (2500 Millisekunden) invertiert wird. Beudeutet, wenn das Relay an ist, dann wird es ausgeschaltet und anders herum. In der ersten Zeile wird als erstes eine Variable definiert, die den Pin, an dem der In-Anschluss des Relais angeschlossen ist speichert, definiert. In dem Beispiel ist das Pin 3 des Arduinos. In der Setup Funktion wird durch den Aufruf von

```
pinMode(IN_PIN, OUTPUT);
```

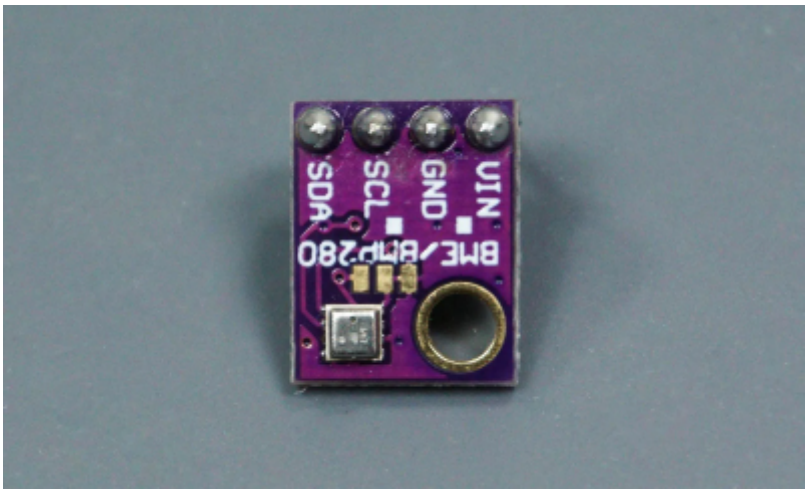
der Pin „IN_PIN“ als OUTPUT definiert. Somit weiß der Arduino, dass dieser Pin „HIGH“ (An) oder „LOW“ (Aus) geschaltet werden kann. (Als gegenstück gibt es auch noch die Definition als INPUT, wodurch der Arduino dann misst, ob der Pin „HIGH“ oder „LOW“ ist.) In der Zeile darunter wird der Initiale Zustand des IN_PINs auf „Low“ (also aus) gestzt, indem die Funktion

```
digitalWrite(IN_PIN, LOW);
```

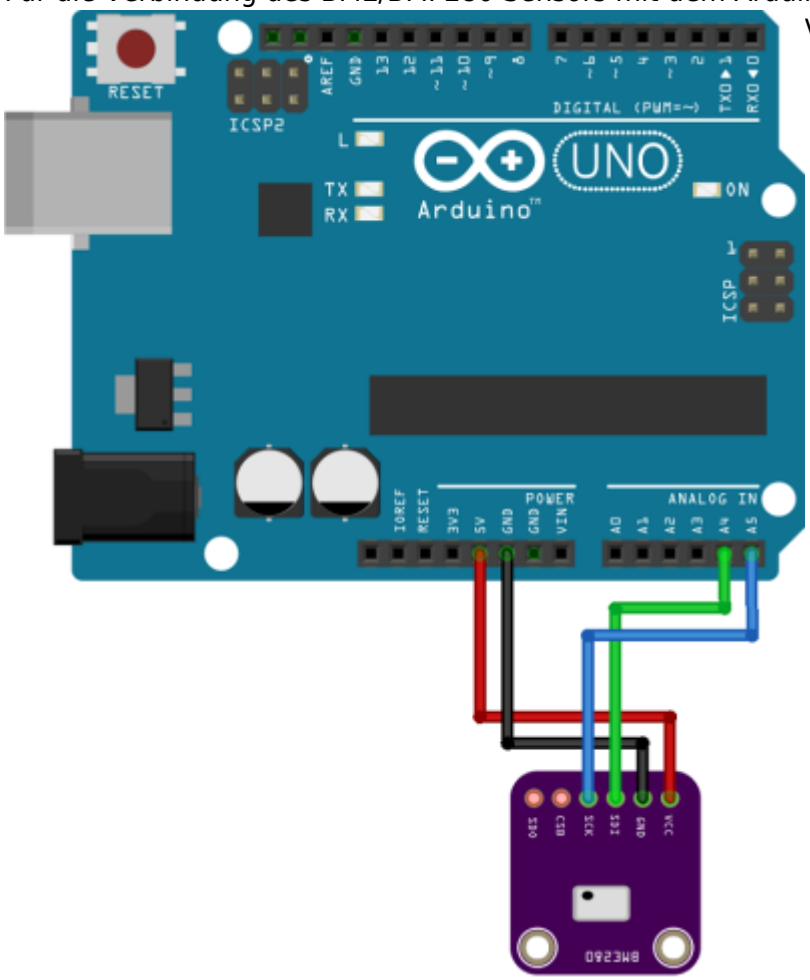
aufgerufen wird. „digitalWrite(pin, status)“ wird immer Verwenden, um einen Pin, der im Code als „OUTPUT“ definiert wurde auf den Zustand <status> zu setzen. Das kann entweder „HIGH“ oder „LOW“ sein, was an dem Pin dann einen 5V Pegel bzw. einen 0V Pegel hervorbringt. Anschließend wird in der Loop Funktion, die ja immer und immer wieder ausgeführt wird, der Input Pin alle 2,5 Sekunden auf „HIGH“ bzw. „LOW“ gesetzt, was dazu führt, dass die am Relais angeschlossene Lampe

im 2,5 Sekunden Takt blinkt.

Temperatursensor BME/BMP280



Der BME/BMP280 Tempratursensor ist ein Vollständiges Modul, das die Temperatur, die Luftfeuchtigkeit und den Lufdruck misst und die gemessenen Daten anschließend über eine I²C Schnittstelle ausgibt. Durch die Ausgabe über die I²C Schnittstelle können die Daten sehr einfach mit dem Arduino ausgelesen und verarbeitet werden. Zusätzlich gibt es eine Bibliothek, die es erlaubt mit einem Funktionsaufruf die Daten auszulesen.
Für die Verbindung des BME/BMP280 Sensors mit dem Arduino werden 4 Kabel benötigt. Die Verbidnung sieht wie folgt aus:



BME280 Pin	Arduino Pin
VCC	5V
GND	GND
SCL	A5
SDA	A4

Die Pins SCL und SDA sind die Anschlüsse des I²C Busses. Über diese Pins werden Datenpakete hin und her gesendet, wodurch der im Code des Arduinos die gemessenen Werte des Sensors angefragt werden können.

Um die Daten des BME/BMP280 einfach auszulesen gibt es die Adafruit BME280 Library auf Github. Wenn diese Bibliothek installiert ist, dann sieht der Code wie folgt aus:

```
/*
 * Complete Project Details
https://randomnerdtutorials.com/bme280-sensor-arduino-pressure-temperature-h
umidity/
*/

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // I2C

unsigned long delayTime;

void setup() {
  Serial.begin(9600);
  Serial.println(F("BME280 test"));

  bool status;
```

```
  // default settings
  // (you can also pass in a Wire library object like &Wire2)
  status = bme.begin();
  if (!status) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }

  Serial.println("-- Default Test --");
  delayTime = 1000;

  Serial.println();
}


void loop() {
  printValues();
  delay(delayTime);
}


void printValues() {
  Serial.print("Temperature = ");
  Serial.print(bme.readTemperature());
  Serial.println(" *C");

  Serial.print("Pressure = ");
  Serial.print(bme.readPressure() / 100.0F);
  Serial.println(" hPa");

  Serial.print("Approx. Altitude = ");
  Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
  Serial.println(" m");

  Serial.print("Humidity = ");
  Serial.print(bme.readHumidity());
  Serial.println(" %");

  Serial.println();
}
```

Der Kern dieses Codeblocks liegt in den Zeilen

```
Adafruit_BME280 bme; // I2C
status = bme.begin();
```

mit der ersten Zeile wird eine Variable „bme“ der Klasse „Adafruit_BME280“ Instanziert. Somit sind jetzt über die Variable „bme“ alle benötigten Funktionen Verfügbar. In der zweiten Zeile wird dann dann die Funktion „bme.begin()“ aufgerufen, die eine I²C Verbindung zu dem Sensor aufbaut. Der Rest der Funktionen, die in der Setup Funktion aufgerufen werden sind einfache Konfigurationen, die

nur für dieses Beispiel benötigt werden, eine detailierte Erklärung kann unter https://randomnerdtutorials.com/bme280-sensor-arduino-pressure-temperature-humidity/ gefunden werden. Letztendlich sehr wichtig sind die Funktionen

```
bme.readHumidity()
bme.readTemperature()
```

Diese Funktionen machen im Hintergrund alles nötige, um die Daten, die der BME/BMP280 Sensor misst, als einfache Zahl in den Arduino zu bekommen. In dem konkreten Fall sind das die Werte der Luftfeuchtigkeit in Prozent und die Temperatur in °C.

Tauchpumpe

Die Pumpe wird verwendet, um das Wasser vom Unteren Wassertank zur spitze des Hydroponikturms zu pumpen.

LED-Streifen für Pflanzen

Quellen

- Baras, T: DIY Hydroponic Gardens: How to Design and Build an Inexpensive System for Growing Plants in Water

From:
http://www.labprepare.tu-berlin.de/wiki/ - **Project Sci.Com Wiki**

Permanent link:
http://www.labprepare.tu-berlin.de/wiki/doku.php?id=ws2021:kabelsalat&rev=1619791408

Last update: **2021/04/30 16:03**