Computergestützte
Experimente und Signalauswertung

PHY.W04UB/UNT.038UB

Übungseinheit #1

Arduino

1 Ablauf

Eine Person pro PC. Login-Daten werden am Anfang der Übung bekannt gegeben.

Es wird ein File ‚**CES_Arduino_Vorgabe**' vorgegeben welches die komplette Initialisierung des Arduino, Konfiguration der I/O Pins und externer Hardware übernimmt. Zum Testen des Arduino laden sie die Vorgabe durch starten der Arduino DIE, und laden sie das File in den Arduino, es müsste bei korrekter Initialisierung ein ‚*#partytime*', im Serial Monitor (Tools→Serial Monitor) erscheinen.

2 Arduino Übungsaufbau & Beschaltung

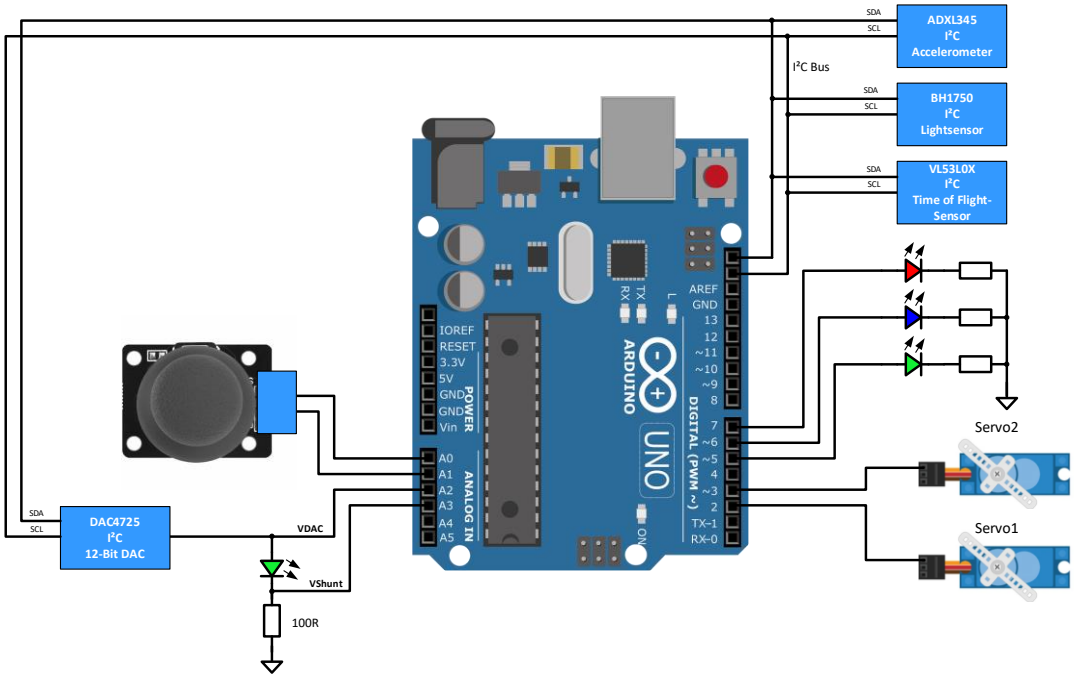


Abbildung 1 - Beschaltung Arduino

Pin-Name	Funktion	Aktor/Sensor
A0	Analoger Eingang	Joystick–X-Achse
A1	Analoger Eingang	Joystick–Y-Achse
A2	Analoger Eingang	DAC-Spannung LED
A3	Analoger Eingang	Shunt-Widerstand LED
D2	Digitaler Ausgang	Servo Ansteuerung
D3	Digitaler Ausgang	Servo Ansteuerung
D5	Digitaler Ausgang	RGB LED
D6	Digitaler Ausgang	RGB LED
D6	Digitaler Ausgang	RGB LED
A4	Digitaler Ein/Ausgang	I^2C – SDA
A5	Digitaler Ausgang	I^2C – SCL

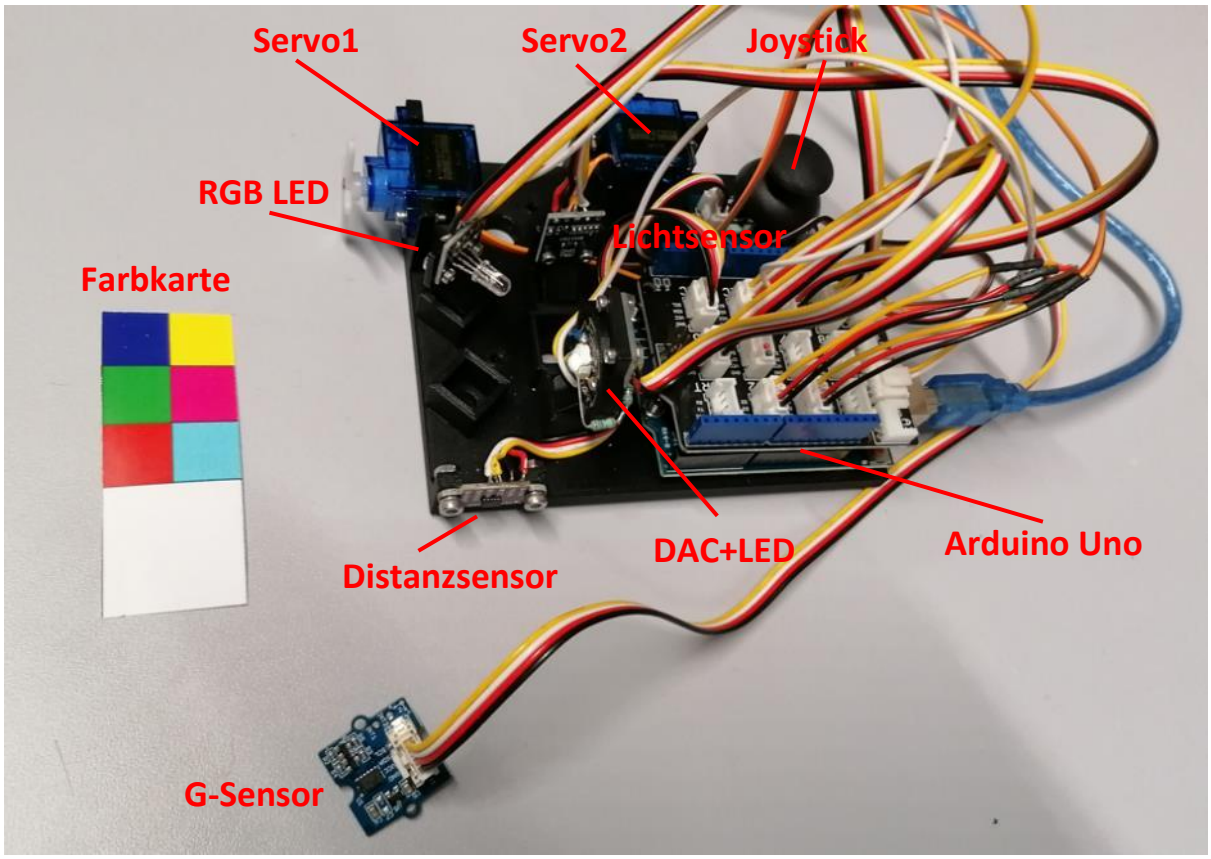


Abbildung 2 - Übungseinheit

2.1 Arduino IDE Einstellungen

- Tools → Board → Arduino Uno
- Tools → Port → COM-Port auswählen
- Tools → Serial Monitor → Baudrate: 57600 (Übereinkunft)

Verwendete Bibliotheken:

- MCP4725 von Rob Tillart – Version 0.3.3
- BH1750 von Christopher Laws – Version 1.3.0
- Adafruit_VL53L0X von Adafruit – Version 1.2.0
- Accelerometer ADXL345 by Seeed Studio – Version 1.0.0

Neuere Versionen können funktionieren, müssen es aber nicht.

2.2 Beschreibung von Aktoren und Sensoren

2.2.1 Joystick

Der Joystick ist mit den analogen Eingängen A0 und A1 verbunden. Das Potentiometer im Joystick ist so verbaut, dass nicht der volle Dynamikbereich des Arduino ADCs ausgereizt wird. Folglich muss der Signalbereich des Joysticks ermittelt werden. Weiters kann der Joystick auch als Drückknopf benutzt werden, dabei wird die Analoge-Eingangspannung des Eingangspins auf +5V gesetzt.

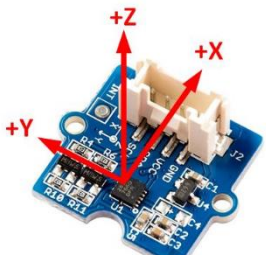
Abbildung 3 - Joystick

2.2.2 Servos

Es sind zwei Servomotoren auf dem Übungsboard verbaut, welche über die Pins D2 und D3 verbunden sind. Es wird empfohlen über die Arduino vorgefertigte ServoLib zu benutzen (Servo.h), aber es kann natürlich auch manuell selber programmiert werden. Welcher Servo mit welchen Digitalpin verbunden ist muss selbstständig bestimmt werden.

Abbildung 4 - Servo

2.2.3 Accelerometer – ADXL345



Die Accelerometer Platine, auf welchem der Accelerometerchip verbaut ist, ist mit dem Arduino über I^2C verbunden. Das Accelerometer (AKA G-Sensor) kann Beuschleunigung, und damit natürlich auch Erdbeschleunigung im Ruhezustand über die Neigung bestimmen.

Abbildung 5 - Accelerometer

2.2.4 RGB LED

Abbildung 6 - RGB LED

Die RGB LED wird als ‚Common-Cathode' ausgeführt und ist mit den Digital-Pins D5, D6 und D7 verbunden. Welcher Pin mit welcher LED verbunden ist, muss selbstständig bestimmt werden. Vorwiderstände sind bereits auf der RGB-LED Platine integriert. Wenn eine LED eingeschalten werden soll, so muss der entsprechende Pin auf HIGH geschalten werden, und der Pin muss auch als OUTPUT konfiguriert werden damit die LED leuchtet. Pin D7 verfügt über keinen PWM-Generator.

2.2.5 Lichtsensor – BH1750

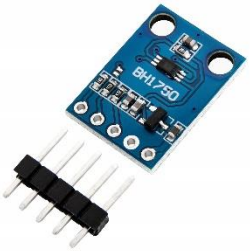


Abbildung 8 - Lichtsensor

Der BH1750 ist ein Lichtsensor-Chip, welcher vier Komponenten integriert hat: Fotodiode, Integrator zum vermessen des Fotodiodenstroms, ADC und I²C Schnittstelle. Aufgrund dieses hohen Grades an Integration muss der Sensor nur noch konfiguriert und die Lichtwerte ausgelesen werden. Üblicherweise haben Silizium-Fotodioden eine spektrale Empfindlichkeit von ca. 300-1100nm. Beim BH1750 sind allerdings integrierte Filter vor der Fotodiode verbaut, und es kann von einer photopischen Spektralen Empfindlichkeit (400-700nm) gesprochen werden. Es sollte noch angemerkt werden, dass in der Übung die Einheit des Treibers in ‚Lux' übernommen wird, auch wenn diese natürlich nicht dem CIE1931 Tristimulus übereinstimmt. Als Mess-Zeit für eine Messung können ca. 180ms angenommen werden.

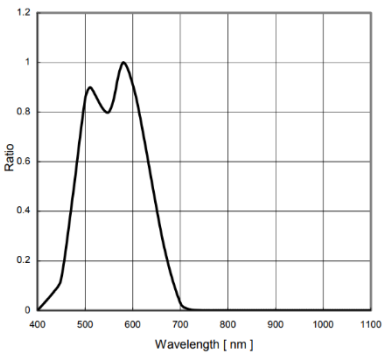


Abbildung 7 - Spektrale Empfindlichkeit

2.2.6 Distanzsensor – VL53L0X

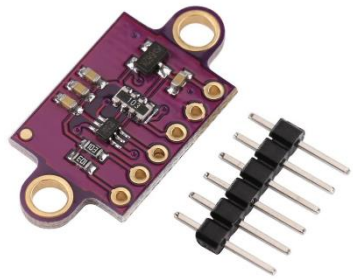


Abbildung 9 - Distanzsensor

Der Distanzsensor ist ein TOF / Time-Of-Flight Sensor. Dabei wird ein sehr kurzer Lichtpuls (einige Pikosekunden bei 940nm) ausgesendet, welcher sich mit Lichtgeschwindigkeit ausbreitet. Dieser Lichtpuls wird vom Objekt teilweise zurückreflektiert/-gestreut in Richtung des TOF-sensors. Aus der Laufzeit des Lichtes wird die Distanz ermittelt, welche vom Chip in Millimeter ausgegeben wird. Der Sensor hat eine effektive Reichweite von 20mm bis ca. 1000mm und kann mittels I²C Interface programmiert werden.

3 Mögliche Übungen

Es werden während der Übungseinheit mehrere Übungen angeboten, die Studierenden können entweder bei einem ‚**Crashkurs für Arduino**' die verschiedenen Sensoren und Aktoren kennenlernen, aber auch selbständig eines der unten aufgeführten Übungsbeispiele Probieren. Auf Anfrage kann auch der Samplecode für jedes der unten angeführten Übungsbeispiele zur Verfügung gestellt werden. Mitarbeit in der Übungseinheit wird nicht benotet.

3.1 Crashkurs für Arduino

In diesem Beispiel wird über den PC-Beamer im Übungsraum einfache Codebeispiele vorprogrammiert und es können alle Studierenden diese Codes nachprogrammieren. Es werden mehrere Sensoren/Aktoren in einfachen Kombinationen gezeigt sowie einfache Beispiele der Datenevaluierung zeigt.

3.2 Drohne –Steuerung und Regelung

3.2.1 Drohnensteuerung: Joystick + Servo

Aktor: 2x Servos, Sensor: Joystick

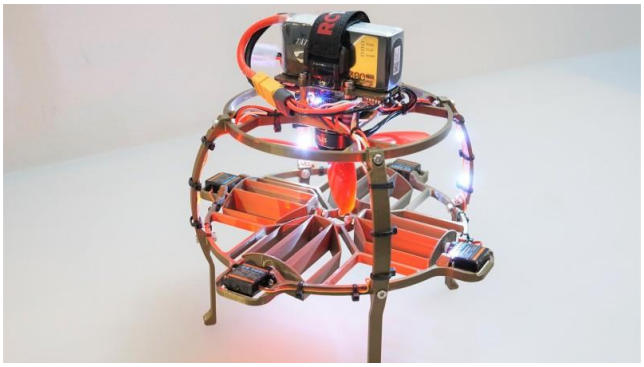


Abbildung 10 - Beispieldrohne - Quelle: Hackaday.com

Ziel dieser Übung ist es eine ‚Drohnensteuerung' zu realisieren. Die Drohne kann sich dabei vorgestellt werden als Drohne mit nur einem Antriebsmotor, bei welchem die Servos die Richtung des Luftstroms steuern. Der Drehmomenten-Ausgleich soll in diesem Beispiel nicht berücksichtigt werden. Dazu soll die Bewegung des Joysticks die Servomotoren ansteuern. Wenn Sie z.B. den Joystick nach Links bewegen, so soll der Links/Rechts Servo ebenfalls nach Links drehen, und wenn sie den Joystick nach vorne bewegen so soll der vor/zurück Servo ebenfalls sich nach vorne bewegen.

Der Joystick ist mit den Arduino Analog-Pins A0 und A1 verbunden. Der Arduino hat eine 5V Referenz, sowie einen dynamischen Bereich von 10Bit (0-1023 Digits möglich). Führen sie eine Analogmessung des Joysticks durch. Ermitteln sie durch auslesen (analogRead) sowie Ausgabe (UART/debugString) welche Werte der Joystick tatsächlich hat. Dazu empfiehlt es sich entweder die Hauptschleife langsamer zu machen oder die Ausgabe nur in vielfachen der Hauptschleife durchzuführen. Geben sie die Messwerte als Dezimalwert aus. Und dokumentieren Sie diese. Aus den dokumentierten minimal/maximal Messwerten können sie nun eine Umrechnung auf Servo-Winkel (0-180°) durchführen.

Dazu können sie die Berechnung entweder manuell durchführen (y=k*x+d) oder sie können den Arduino eigenen „map"-Befehl verwenden. Bei der Mittelposition des Joysticks soll der Servo die 90°, beim Minimalwert z.B. 0° und beim Maximalwert 180° Stellung einnehmen. Die Servo.write Funktion

akzeptiert dezimale Werte von 0 bis 180 welche Äquivalent zum Winkel sind. Um eine schnelle Aktualisierung der Joystickwerte zu erhalten wird eine Loop-Delayzeit von ~20ms empfohlen.

3.2.2 Drohnenstabilisierung: Accelerometer + Servo

Aktor: 2x Servos, Sensor: Accelerometer (G-Sensor) – Der G-Sensor Repräsentiert die Drohne.

Ziel dieser Übung ist es eine ‚Drohnenstabilisierung' zu realisieren. Als Sensor dient nun ein G-Sensor, welcher die (Erd-)Beschleunigung messen kann. Die zwei Servos agieren als Aktoren, welche einem Kippen der Drohne entgegenwirken sollen: Sobald die Drohne, welche in dieser Simulation ausschließlich durch die Neigung des G-Sensors repräsentiert wird, in eine Richtung wegkippt, soll der Servo gegensteuern.

Dazu sollen die Beschleunigungsmesswerte vom Accelerometer ausgelesen werden, welche mit 10 Bit - +/-2g vorgegeben sind. In der Standardkonfiguration, welche in dieser Übung verwendet wird, ist das 10.te Bit das Vorzeichen, die Messwerte lassen sich daher durch 256.0 dividieren, um vielfache der Erdbeschleunigung anzuzeigen. Beachten sie auch, dass der G-Sensor nicht kalibriert ist. Sie können durch evaluieren des Sensors eine Kalibrierung selber durchführen, dabei sollte jede Achse bei Auflegen auf dem Tisch einen Wert von ‚1.0' erreichen.

Um die Stabilisierung zu realisieren untersuchen Sie die Messwerte des G-Sensors um eine geeignete mathematische Funktion zu finden. Beachten Sie, dass abhängig von ihrem Model eine Achse eventuell gar keine Information enthalten könnte. Auch sollte beachtet werden, dass mathematisch inkorrekte Modelle (z.B. lineare Vereinfachungen) durchaus funktionieren können und in der Praxis auch regelmäßig in Verwendung sind (Quelle: Arduino.cc).

Bei den Umrechnungen der Messdaten kann nötig sein den Variablentyp zu wechseln, z.B. von Integer(16Bit) auf Float(32Bit) oder umgekehrt. Bei falscher Anwendung in der Umrechnung kann es daher zu Datenverlust kommen. Es wird empfohlen nach jeder Umrechnung die Daten wieder per serieller Schnittestelle(UART) auszugeben, um sicherzustellen das keine falsche Berechnung durchgeführt wurde.

Nach einer erfolgreichen Implementierung werden sie feststellen, dass die Servomotoren etwas unruhig sind, dies ist durch die hohe Standardabweichung des Sensors zu erklären, als auch eine unruhige Führung. Eine Möglichkeit die Stabilisierung sanfter zu realisieren ist durch einen Einsatz eines ‚Moving-Average-Filter' (Gleitender Mittelwert). Die Filterfunktion wird übrigens stark von der Loop-Frequenz bzw. Periodendauer beeinflusst, mit der die Funktion aufgerufen wird. Der Moving Average Filter bildet den Mittelwert von N vorhergehenden Messwerten.

$$avg = \frac{1}{N}\sum_{k=0}^{N} \mathrm{a(k)}$$

Dies kann leicht realisiert werden z.B. über eine globale Variable, welche als Array definiert wird. Eine weitere globalen Variable wird benötigt, um zu zählen wo der Index des letzten neuen Wertes sich gerade befindet. Tipp: 25ms Delay und N=10 Elemente im Filter könnten (abhängig vom gewählten mathematischen Modell) sehr gut funktionieren.

3.3 Geschwindigkeitsmessung – Mentos + Cola Experiment

Aktor: Menschliche Hand, Sensor: Distanzsensor.

In diesem Experiment soll der TOF-Distanzsensor verwendet werden, um das ‚Mentos + Cola Experiment' zu simulieren. Als Mentos+Cola-Fontäne wird ihre Hand benutzt welche vom Sensor wegbewegt werden kann, um eine Geschwindigkeitsmessung und sogar Beschleunigung zu realisieren. Dazu eignet sich übrigens die „Stopuhr" Methode, welche im Vorlesungsteil besprochen wurde. Als Distanzsensor wird ein VL53L0X verwendet.

Versuchen Sie zuerst in der Hauptschleife den Sensor anzusprechen und die Messwerte mit der UART Schnittstelle darzustellen. Versuchen Sie nun eine ‚Stoppuhr' zu programmieren, um die Messzeit des Chips zu bestimmen (zb.: mittels millis() Befehl) um dadurch die möglichen Messintervalle zu ermitteln. Wenn die Distanzmessung und auch Zeitmessung funktioniert kann mit der Geschwindigkeitsmessung gestartet werden. Dazu werden zwei Distanzen benötigt, z.B.: 200mm und 300mm entfernt vom Sensor. Um die Geschwindigkeit zu bestimmen muss die Zeit gemessen werden, welche die Hand zwischen diesen beiden Punkten benötigt.

Eine Möglichkeit die Distanzmessung zu realisieren ist eine kleine ‚Statemachine' zu programmieren:

- Unter 50mm → Statemachine auf 0 setzen, Zeitmessung auf Reset.
- Wenn aktuell gemessene Distanz > 200mm **UND** vorhergehend gemessene Distanz < 200mm **UND** aktueller Statemachine Wert auf 0 → Statemachine auf 1 setzen und ‚Stoppuhr Starten'.
- Wenn aktuell gemessene Distanz > 300mm **UND** vorhergehend gemessene Distanz < 300mm **UND** aktueller Statemachine Wert auf 1 → Statemachine auf 2 setzen, ‚Stoppuhr Stoppen', Auswertung der Geschwindigkeit in [m/s] und mittels UART ausgeben.

Achten sie beim Programmieren darauf, dass nicht in einer Messung mehrere Statemachine Zustände gleichzeitig aktiviert werden können. Die Messung kann dadurch Resetiert werden, dass man mit der Hand wieder unter 50mm nahe an den Sensor geht. Sie können anschließend auch probieren eine Beschleunigungsmessung zur realisieren, dazu muss die Statemachine natürlich um einen dritten Messpunkt erweitert werden, z.B. bei 400mm.

Für die Beschleunigungsmessung gilt natürlich:

$$a = \frac{\Delta v}{\Delta t} = \frac{v(t_2) - v(t_1)}{t_2 - t_1}$$

3.4 Farbkartenerkennung - Lichtsensor + RGB

Aktor: RGB LED, Sensor: Lichtsensor - der Lichtsensor ist kein Farbsensor

Hilfsmittel: Farbkarte

Ziel dieser Übung ist es eine RGB Messung ohne RGB Sensor zu realisieren. Das RGB Messsystem soll über den weißen Teil der Farbkarte kalibriert werden. Anschließend soll der Benutzer die Farbkarte benutzen, um eine Farbe messen zu können. Es wird empfohlen die Messung in Reflektion durchzuführen. Für Transmission wäre eine transparente Detektorkarte nötig.

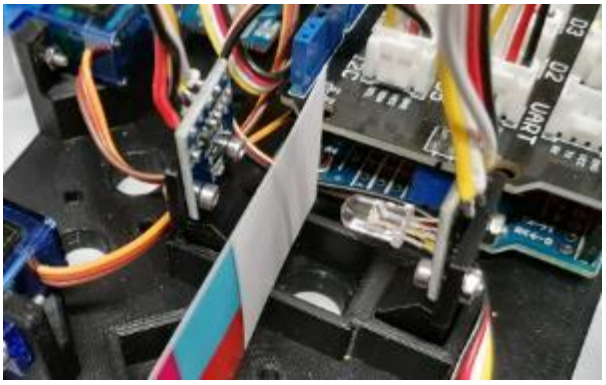


Abbildung 11 - Aufbau Absorptionsmessung

3.4.1 Grundlagen der Farbmessung

Es gibt mehrere Methoden wie man die „Farbe“ messen kann. Eine übliche Methode verwendet eine weiße Lichtquelle (im Spektralbereich von 400-700nm), um das zu messende Objekt zu beleuchten. Aus dem transmittierten oder auch reflektierten Licht lässt sich die Farbinformation z.B., Rot(~650nm), Grün(~520nm) und Blau(~450nm) bestimmen mittels integrierter Filter in einem Sensor. Für diese Methode wird ein Farbsensor benötigt, welcher Spektral getrennt jede Farbe messen kann. Eine andere Methode, welche in dieser Übung verwendet wird, funktioniert ohne Farbsensor. Dabei wird eine Spektral definierte Emissionsquelle verwendet z.B.: eine Rote-, Grüne- und Blaue-LED. Dabei wird jeweils nur eine Farbe einer RGB-LED eingeschalten, und dann die Reflektion oder Transmission gemessen und anschließend zur nächsten Farbe gewechselt. Die Methode mit der RGB LED ist eine sequentielle Methode, da drei separate Intensitätsmessungen hintereinander durchgeführt werden müssen, um eine Farbinformation in RGB zu erhalten. Zusätzlich wird in dieser Übung eine 4te Messung eingebaut, in der keine LED eingeschalten ist, um das Hintergrundlicht (z.B. Raumlicht) zu entfernen. Beachten Sie: Die RGB LED hat unterschiedliche Intensitäten, welche Sie aufeinander abstimmen (Kalibrieren) müssen. Dieses Kalibrieren erfolgt über den weißen Teil der zur Verfügung gestellten Farbkarte.

3.4.2 Sequentielle Farbmessung – Aufgabenstellung

Schreiben sie einen Code zum Messen von reflektierter/absorbierter Farbe. Dabei soll die RGB LED nacheinander einmal Rot, Grün und Blau und anschließend NICHT leuchten. Schreiben Sie dazu ein Unterprogramm ‚setRGB‘, welches die RGB LED umschaltet mit folgenden Variablen:

Eingangsvariable	LED-Zustände	Kommentar
0	R=Aus, G=Aus, B=Aus	Alle LEDs aus
1	R=Ein, G=Aus, B=Aus	Nur Rote LED ein
2	R=Aus, G=Ein, B=Aus	Nur Grüne LED ein
3	R=Aus, G=Aus, B=Ein	Nur Blaue LED ein

Schreiben Sie ein weiteres Unterprogramm ‚MeasureColor', welches z.B. in einer Schleife 4x Messungen durchführt mit ‚kein Leuchten', ‚Rot', 'Grün' und anschließend ‚Blau' leuchten. Dazu müssen sie die LED auf die jeweilige Farbe stellen. Nach jedem Umschalten empfiehlt es sich kurz zu warten (z.B. 10ms). Nach dem warten können Sie den Lichtwert messen (getLuxValue). Subtrahieren Sie nun die Messresultate von den ‚Rot', ‚Grün' und ‚Blau' Messungen mit der ‚kein Leuchten' Messung. Damit ziehen Sie das Hintergrundlicht von den RGB Messungen ab. Vergessen sie nicht nach der Messung die LED wieder auf ‚kein Leuchten' zu stellen. Die Messresultate können entweder in einer globalen Variablen abgespeichert werden oder über Parameterübergabe an die Hauptfunktion zurück übergeben werden.

Schreiben sie im Hauptprogramm eine Funktion, welche es Ihnen erlaubt per serieller Schnittstelle ereignisorientiert Befehle zu empfangen. Dabei soll beim Empfangen von einem ‚c' eine Kalibrierung durchgeführt werden. Beim Senden von einem ‚m' soll eine Messung durchgeführt werden. Wenn sie eine Kalibrierung durchführen, so messen sie mit der ‚MeasureColor' Funktion die Farbe und speichern sie die Werte in einem vorher definierten Kalibrier-Array ab. Wenn sie eine Messung durchführen so speichern Sie die Werte ein einem Farb-Array ab. Evaluieren sie anschließend das Farb-Array, indem sie jeden Farb-Array Wert mit seinem Kalibrierwert dividieren (jeweils für Rot, Grün und Blau) und multiplizieren sie diesen Wert anschließend mit 100.0, um die Resultate prozentuell darzustellen. Das verarbeitete Messresultat gibt also den prozentuellen Unterschied in Farbe zurück gegenüber dem Kalibrierten Wert. Schreiben sie eine geeignete Ausgabefunktion, um die RGB Werte bei jeder Messung anzuzeigen, und auch die Kalibrierwerte anzuzeigen. Wenn Sie kein Objekt messen, so sollte sofern Kalibriert, 100% für alle drei Farben zurück gemessen werden. Bestimmen Sie anschließend mit der Farbkarte die prozentuellen RGB Farbänderungswerte und schreiben sie diese mit. Schreiben sie nun eine Auswertefunktion. Dabei soll, wenn sie die Farbkarte vor den Detektor stellen, und wenn sie z.B. das rote Segment vor den Detektor legen, und die Messung starten, ‚ROT' in der Ausgabe erscheinen bei Grün ein ‚Grün' usw…

Profi Modus: Versuchen Sie ob auch Cyan/Magenta/Yelb erkennbar ist.

3.5 LED-Charakterisierung / Ereignisorientierte Programmierung

Aktor: externer DAC (12-Bit), Sensor: interner ADC (10-Bit)

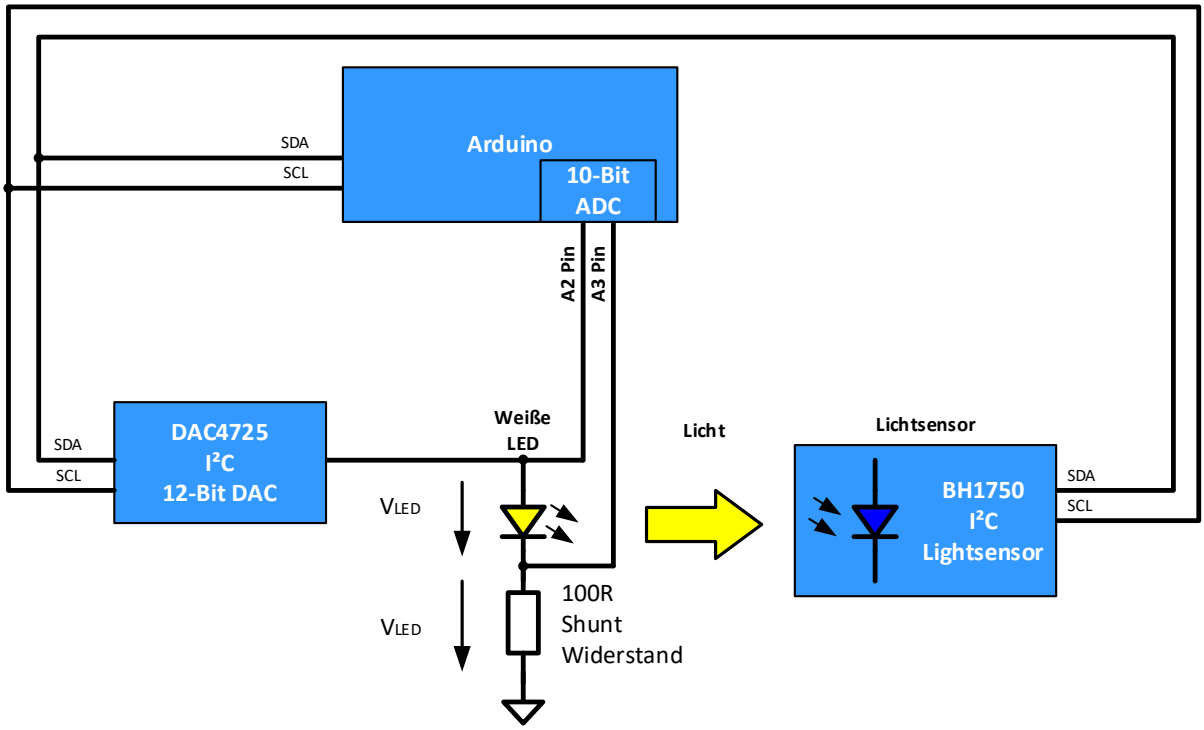


Abbildung 12 - LED Charakterisierungsbeschaltung

Gesuchte Resultate:

- LED-Strom vs. LED-Spannung
- LED-Intensität vs. LED-Strom

Um eine korrekte Intensitätsmessung zu erhalten stecken sie die große LED in einen Steckplatz und stecken sie gegenüber den Lichtsensor BH1750.

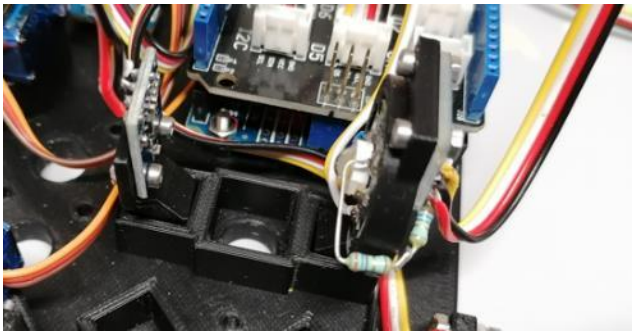


Abbildung 13 - Aufbau LED Charakterisierung

Es soll eine Messschleife geschrieben werden, in welcher der externe DAC zur Spannungsvorgabe mit einem geeigneten Wert inkrementiert wird. Die Messchleife soll ‚Ereignisgesteuert' aktiviert werden. Dabei können sie auswählen, ob sie entweder die Messchleife über das Drücken des Joysticks aktivieren, oder über das Senden eines einzelnen Zeichens über den Serial-Monitor. Wenn Sie mit der Messung starten, schreiben sie über die serielle Schnittstelle „Start Messung!" um anzuzeigen, dass sie eine Messung durchführen. Anschließend schicken sie über die serielle Schnittstelle die Beschriftung für die Messwerte. Achten sie dabei, dass die Worte immer mit einem Strichpunkt ‚;' voneinander getrennt sind:

"DAC;LED_Spannung[V];LED_Strom[mA];LED_Beleuchtungsstärke[Lux]"

In der Messschleife soll der externe DAC über den vollen Wertebereich durchgefahren werden (Schrittweite kann frei gewählt werden). In dieser Messschleife soll zuerst der DAC auf einen Wert gestellt werden. Anschließend soll einige Zeit gewartet werden (z.B. 5ms), um sicher zu sein, dass die Spannung am DAC Ausgang auch sicher anliegt. Anschließend soll die Spannung am DAC als auch die Spannung am Shunt-Widerstand über den Arduino internen ADC gemessen werden. Nach den Analogmessungen soll der Lichtsensor eine Messung durchführen um die Intensität der LED zu messen. Aus den Resultaten der Analogmessungen können sie die LED-Spannung als auch den LED-Strom berechnen (Tipp: Ohmsches Gesetz). Sobald die Berechnung abgeschlossen ist, können Sie die Messresultate über die serielle Schnittstelle ausgeben, dazu wird folgendes Format empfohlen:

DACWert[Digits];LEDSpannung[V];LEDStrom[mA];LEDBeleuchtungsstärke[Lux].

Zwischen den Messresultaten soll immer ein Strichpunkt ‚;' gesetzt werden, und die Resultate sollen alle in einer Zeile stehen. Nach der Durchführung einer erfolgreichen Messung werden sie beobachten, dass die LED nach der Messung noch weiter leuchtet, und sie auch bei der Lichtmessung nicht bei 0 beginnen. Es wird empfohlen die LED nach der Messung mit dem DAC wieder auf 0 zu setzen.

Um die Lichtmessung korrekt durchführen zu können gibt es zwei Möglichkeiten. 1) Die Messung im Dunklen durchführen, was in der Übung nicht möglich ist. 2) Vor der Messung eine „Hintergrundlicht" Messung durchzuführen und dann den Hintergrundlicht-Messwert von jedem Lichtintensitäts-Messpunkt abzuziehen. Achten sie bei der Messung darauf, dass sich die Lichtverhältnisse in der Umgebung des Detektors NICHT ändern, da dies die Beleuchtungsstärkenmessung verfälschen würde.

Nach Abschluss einer Hintergrundlichtkorrektur können sie eine Messung starten und diese z.B. mit Copy+Paste in ein Excel-File laden und entweder als CSV abspeichern, oder auch direkt über Python die Messdaten aufnehmen und weiterverarbeiten.e