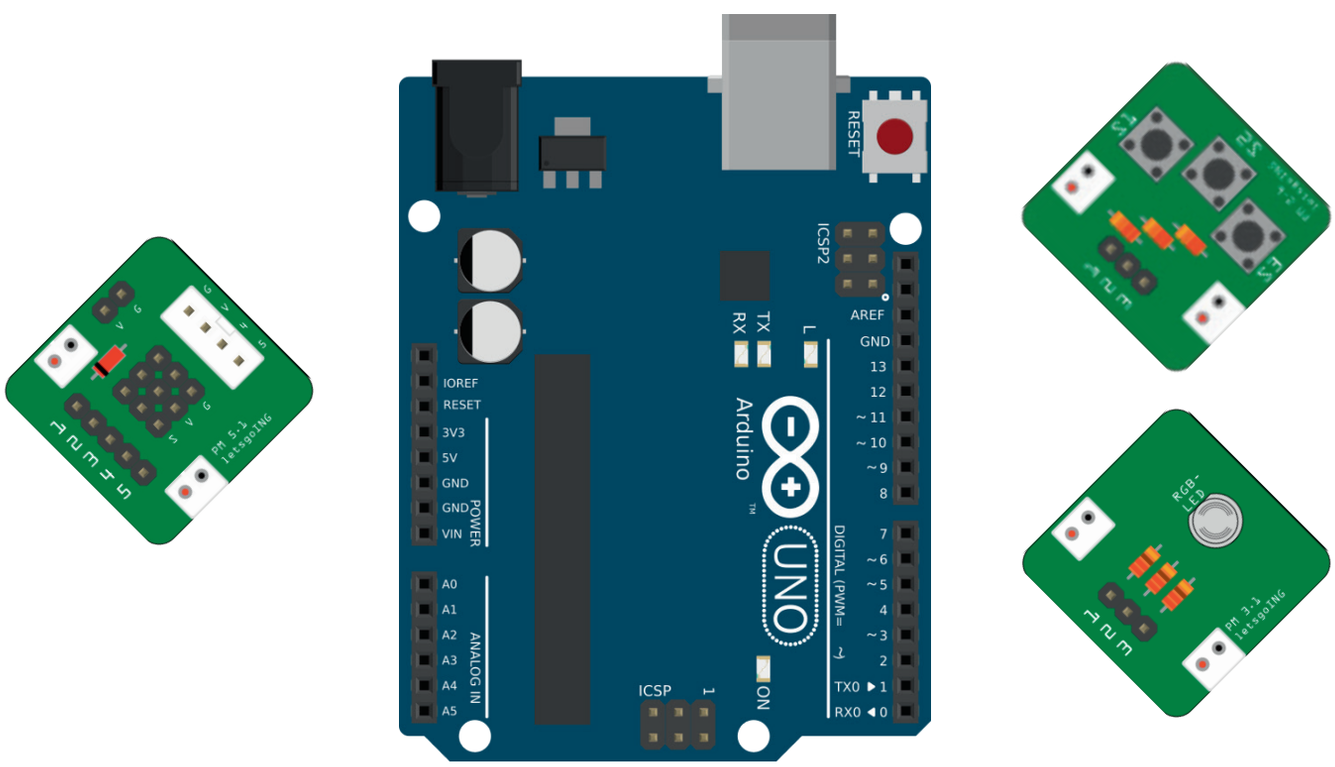
letsgoING

Mechatronik Hochschule Reutlingen

2 - Digitale Signale und Variablen



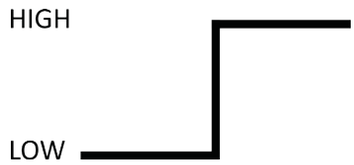
digitale Ein- & Ausgänge • Wiederhole • Falls/Sonst • digitale Variablen

gefördert von der vector Stiftung

2.1 Der digitale Ausgang

Was ist ein digitaler Ausgang?

Ein digitaler Ausgang hat nur zwei Zustände. Er kann entweder AN („true“, „HIGH“ bzw. „1“) oder AUS („false“, „LOW“ bzw. „0“) sein. Die Funktion ist vergleichbar mit einem Lichtschalter, mit dem man eine Lampe an- und ausschalten kann.



Für was brauche ich einen digitalen Ausgang?

Mit einem digitalen Ausgang kannst du Aktoren ein- und ausschalten. Z.B.:
- eine LED ein- und ausschalten
- einen Motor ein- und ausschalten
- ein Relais ein- und ausschalten

Wie setze ich einen digitalen Ausgang?

Um einen digitalen Ausgang zu setzen, nimmst du den Block „digitalWrite“ aus dem dunkelblauen Menü „Output“. Dafür musst du
- den PIN auswählen (2-13 → Pin 0+1 solltest du frei lassen) und
- den Zustand einstellen (HIGH oder LOW)

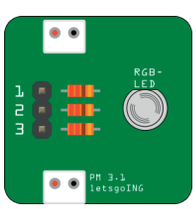
Der Pin bleibt dann solange in dem eingestellten Zustand bis du ihn änderst.


```
pinMode(2, OUTPUT);
digitalWrite(2, HIGH);
```

Das RGB-LED-Modul

Die LED:
Die RGB-LED ist aus drei farbigen LEDs aufgebaut: Rot, Grün und Blau. Durch Ein- und Ausschalten der einzelnen LEDs kann man verschiedene Farben erzeugen. Sind alle drei LEDs an, leuchtet sie weiß.

Das Modul (PM 3.1):
Das Modul wird über das Adaptermodul mit Strom versorgt. Du kannst die einzelnen LEDs über die Pins 1 (blau), 2 (rot) und 3 (grün) ansteuern.

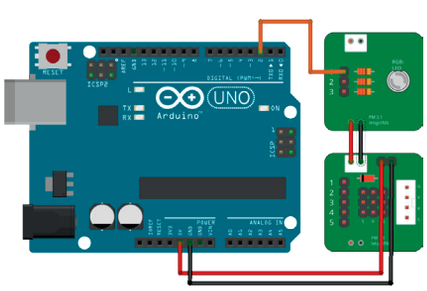


Übungsaufgabe

Übung

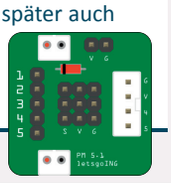
Lade ein Programm auf den Arduino, das den Morse-Code „SOS“ ausgibt. Die Ausgabe soll auf Pin 2 deines Arduinos erfolgen. Du kannst dazu das Programm aus dem Projekt 1.7 verwenden und anpassen.

Schließe das RGB-Modul mit einer Farbe deiner Wahl an den Pin 2 an.
Verwende das Adapter-Modul (s.u.), um das RGB-Modul mit Strom zu versorgen.
Die Module kannst du wie im Bild rechts anschließen.



Das Adapter-Modul

Das Adaptermodul brauchst du, um die anderen Module mit Strom zu versorgen. Dazu verbindest du die Pins V und G mit den Anschlüssen 5V und GND am Arduino. Außerdem kannst du hier später auch noch andere Komponenten anschließen.



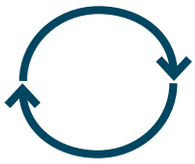
Troubleshooting

Die LED leuchtet nicht
- → Ist die Stromversorgung richtig eingesteckt?
- → Ist das Modul am richtigen Pin eingesteckt? (Vergleiche das Programm mit deiner Schaltung)

Der Arduino wird heiß oder wird am PC nicht mehr gefunden
- → Hast du die Anschlüsse V (5V) und G (GND) vertauscht?

2.2 Kontrollstrukturen: Wiederhole

Was macht ein Wiederhole-Block?



Mit dem „Wiederhole“-Block kannst du, wie der Name schon sagt, Programmteile wiederholen. Dabei kannst du festlegen, wie oft der Teil wiederholt werden soll.
Man nennt einen solchen Block auch „Wiederhole“-Schleife (engl. „for-loop“).

Was kann ich mit einem Wiederhole-Block machen?

Mit dem „Wiederhole“-Block kannst du z.B.
- eine LED für eine bestimmte Zeit an- und ausschalten (blinken lassen)
- eine Messung mehrfach wiederholen
- Eingänge mehrfach überprüfen (z.B. zur Mittelwertbildung)

Die Anzahl der Wiederholungen kannst du entweder fest vorgeben, indem du eine Zahl einträgst oder im Programm verändern (mit Variablen → die werden später erklärt).

Wie verwende ich einen Wiederhole-Block?


```
for (int a=1; a<= ( 5 ); ++a ){

}
```

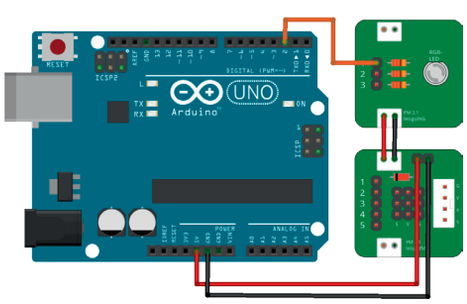
Den „Wiederhole“-Block findest du in dem gelben Menü „Steuerung“.
Der Block hat zwei Bereiche.
- Im ersten legst du die Anzahl der Wiederholungen fest (Mal).
- In den zweiten kannst du das Programm einfügen, welches wiederholt werden soll (Befehle)

Die Blöcke die du unter „Befehle“ einbaust, werden also X-mal (hier 5 mal) wiederholt.

Übungsaufgabe

Schreibe ein Programm, das eine LED drei mal blinken lässt (500 ms) und anschließend eine längere Pause (2 s) macht. Verwende dafür eine „Wiederhole“-Schleife.

Die Ausgabe soll auf Pin 2 deines Arduinos erfolgen. Schließe das RGB-Modul mit einer Farbe deiner Wahl an den Pin 2 an. Die Module kannst du wie im Bild rechts anschließen.



Troubleshooting: Häufige Fehler

Man erkennt die Wiederholungen nicht (z.B. beim Blinken).
→ Wie du weißt, arbeitet der Arduino sehr schnell.
Wenn dann in der Schleife keine Wartezeiten sind, kann man häufig nichts mehr erkennen.
→ Wird der Ausgang in der Schleife ein und ausgeschaltet?
→ Sind zwischen den Schleifen Pausenzeiten?

Die LED leuchtet nicht
→ Ist die Stromversorgung richtig eingesteckt?
→ Ist das Modul am richtigen Pin eingesteckt?
(Vergleiche das Programm mit deinem Aufbau)

Der Arduino wird heiß oder wird am PC nicht mehr gefunden
→ Hast du die Anschlüsse V (5V) und G (GND) am Adaptermodul (Arduino) vertauscht?

2.3 Übungsprojekt

Projektbeschreibung

Die Projektbeschreibung ist identisch mit der aus Kapitel 1.7, wird dieses Mal aber effektiver umgesetzt:
Eine der einfachsten Arten über lange Strecken zu kommunizieren ist das Morsen.
Dabei werden Buchstaben durch bestimmte Impulsfolgen dargestellt. Früher wurden so über Telegrafen-Leitungen zum ersten mal über lange Strecken Informationen ausgetauscht.
Heutzutage werden Morsezeichen nur noch in sehr seltenen Fällen eingesetzt. Wenn man z.B. weit ab der Handy-Netze in Gefahr kommt, kann man über das internationale Notrufzeichen „SOS“ Hilfe rufen. Das besondere an Morsezeichen ist, dass man sie sowohl elektrisch, optisch als auch akustisch übertragen kann.

Programmiere nun einen einfachen Algorithmus, der das internationale Notrufzeichen SOS optisch darstellt und die ganze Zeit wiederholt.

Aufgabenstellung

Schreibe ein Programm, welches die Zeichenfolge „SOS“ auf einer Farbe der RGB-LED ausgeben kann.

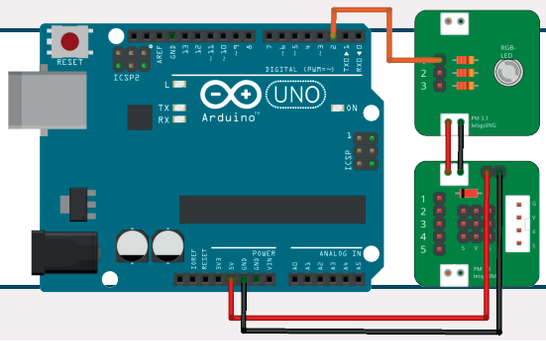
- Das Zeichen „S“ besteht aus drei kurzen Blink-Impulsen (z.B. 300 ms)
- Das Zeichen „O“ besteht aus drei langen Blink-Impulsen (z.B. 900 ms)
- Die Pausen sind immer gleich lang wie die kurzen Impulse (z.B. 300ms)
- Nach jedem Wort kommt eine längere Pause (z.B. 1200 ms)

Verwende dieses Mal für jeden Buchstaben eine „Wiederhole“-Schleife.

Zusatzaufgabe:
Lasse dir jeden Buchstaben in einer anderen Farbe ausgeben.

Versuchsaufbau

Du brauchst für dieses Übungsprojekt deinen Arduino, das Adapter-Modul und das RGB-Modul.
Verwende das Adapter-Modul um das RGB-Modul mit Strom zu versorgen.
Die Module kannst du wie im Bild rechts anschließen.
→ Der Aufbau ist identisch mit den Übungsaufgaben 2.1 und 2.2.



Troubleshooting

LED geht nicht an / LED geht nicht aus

→ hast du im Programm denselben Pin verwendet wie am Arduino?
→ wird die LED an- und ausgeschaltet (HIGH / LOW)?
→ sind die Pausen an der richtigen Stelle? (immer nach dem „digitalWrite“ Befehl)
→ sind die Pausen lang genug? (der Mensch erkennt mit dem Auge nur Änderungen die länger als 20ms sind)

Programm wird nicht hochgeladen

→ sind alle Einstellungen in der Arduino-Oberfläche richtig? (siehe Kapitel 1.3 Arduino IDE)
→ ist der Arduino eingesteckt?

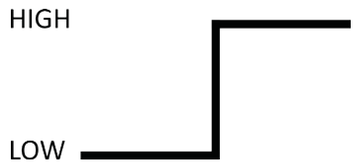
Wissensfragen

1. Nenne die Zustände die ein digitaler Ausgang einnehmen kann. (Liste auch alle alternativen Bezeichnungen auf)
2. Für was wird das Adaptermodul benötigt?
3. Auf was musst du beim Adaptermodul achten, wenn du es an deinen Arduino anschließt?
4. Welchen Vorteil bietet eine „wiederhole“-Schleife gegenüber der direkten Programmierung.

2.4 Der digitale Eingang

Was ist ein digitaler Eingang?

Ein digitaler Eingang kann nur zwei Zustände erkennen. Er erkennt entweder AN („true“, „HIGH“ bzw. „1“) oder AUS („false“, „LOW“ bzw. „0“).



Für was brauche ich einen digitalen Eingang?

Immer wenn du etwas überprüfen willst, das nur zwei Zustände haben kann, nimmst du einen digitalen Eingang:

- ist das Licht an oder aus
- ist der Taster gedrückt oder nicht
- ist die Lichtschranke unterbrochen oder nicht

Wie lese ich einen digitalen Eingang ein?

Um einen digitalen Eingang zu lesen, nimmst du den Block „digitalRead“ aus dem hellblauen Menü „Input“. Dazu musst du

- den einzulesenden PIN auswählen (2-13 → Pin 0+1 solltest du frei lassen) und
- den „digitalRead“-Block an einen geeigneten Block andocken (Form beachten)

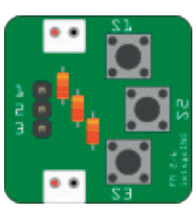
Der Block gibt dann den Wert des Eingangs (HIGH/LOW) an den angedockten Block (z.B. einen Vergleicher) weiter.


```
pinMode(2, INPUT);
digitalRead(2);
```

Das Taster-Modul

Die Taster:
Auf dem Taster-Modul befinden sich drei Taster. Jeder kann einzeln an den Arduino angeschlossen werden. Wird ein Taster am Arduino eingelesen, gibt er ungedrückt „LOW“ und gedrückt „HIGH“ zurück.

Das Modul (PM 2.6):
Das Modul wird über das Adaptermodul mit Strom versorgt. Du kannst die einzelnen Taster über die Pins 1 (Taster S1), 2 (S2) und 3 (S3) einlesen.

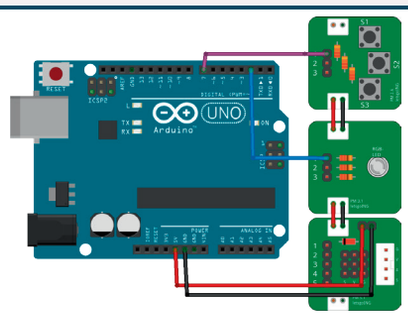


Übungsaufgabe

Übung

Schreibe ein Programm das den Wert eines Tasters einliest und den gelesenen Wert direkt auf einen digitalen Ausgang schreibt. Schließe eine LED an diesen Ausgang an.

Schließe das RGB-Modul mit einer Farbe deiner Wahl an den Pin 2 an.
Schließe das Tastermodul mit einem Taster an Pin 7 an.
Verwende das Adapter-Modul um die anderen Module mit Strom zu versorgen.
Die Module kannst du wie im Bild rechts anschließen.



Troubleshooting

LED geht nicht an / LED geht nicht aus
→ hast du im Programm denselben Pin verwendet wie am Arduino?

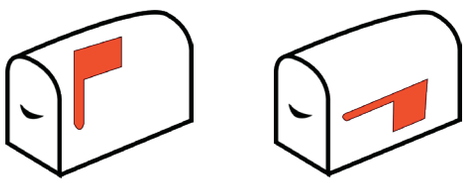
2.5 Kontrollstrukturen: falls/sonst

Was macht ein falls/sonst-Block?

Mit dem „falls/sonst“-Block kannst du Verzweigungen/Entscheidungen in dein Programm einbauen. Der Block überprüft eine Bedingung und führt dann den entsprechenden Teil aus.
Der Block führt also **entweder** den ersten **oder** den zweiten Teill aus.

Was kann ich mit einem falls/sonst-Block machen?

Mit dem „falls/sonst“-Block kannst du Entscheidungen treffen
→ Falls ein Wert größer ist als X, mache A, sonst mache B
→ Falls Taster gedrückt schalte LED an, sonst aus



Beispiel „amerikanischer Briefkasten“:
Der Postbote setzt am Briefkasten eine kleine rote Flagge, wenn er neue Post eingeworfen hat. Der Hausbewohner muss also

testen: ist Flagge gesetzt?
dann: gehe zum Postkasten
sonst: bleibe im Haus

Wie verwende ich einen falls/sonst-Block ?

Den „falls/sonst“-Block findest du in dem gelben Menü „Steuerung“. Wenn du ihn verwenden willst, brauchst du

- eine digitale Prüfbedingung (teste)
- einen Programmteil der ausgeführt wird, wenn Bedingung „true“ ist (dann)
- einen Programmteil der ausgeführt wird, wenn Bedingung „false“ ist (sonst)

Brauchst du den „sonst“-Teil nicht, kannst du auch den „falls“-Block verwenden. Dieser verhält sich wie ein falls/sonst“-Block bei dem der „sonst“-Teil leer ist.

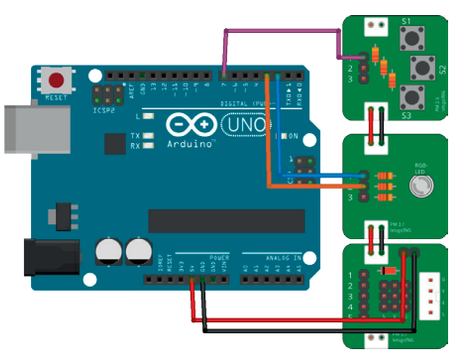

```
if (  ){
}
else{
}
```

Übungsaufgabe

Übung

Schreibe ein Programm, bei dem die LED in einer Farbe leuchtet (z.B. blau), wenn kein Taster gedrückt ist. Wird der Taster gedrückt , soll die LED in einer anderen Farbe leuchten (z.B. rot).

Schließe das RGB-Modul mit zwei Farben deiner Wahl an den Pin 2 und 3 an.
Schließe das Tastermodul mit einem Taster an Pin 7 an.
Verwende das Adapter-Modul um die anderen Module mit Strom zu versorgen.
Die Module kannst du wie im Bild rechts anschließen.



Troubleshooting

Die LED geht nicht an / geht nicht aus
→ hast du im Programm denselben Pin verwendet wie am Modul?

Wenn der Taster gedrückt wird, leuchtet die LED in Mischfarben (z.B. violett)
→ schaltest du die Ausgänge für deine LEDs immer an und aus?

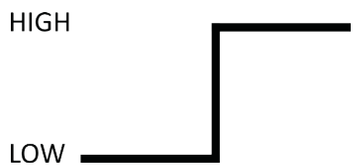
Die Farbe der LED ändert sich nur einmal und bleibt dann gleich
→ schaltest du die Ausgänge für deine LEDs an und aus?

2.6 Die digitale Variable

Was ist eine digitale Variable?

Eine digitale Variable ist ein Platzhalter, der während der Laufzeit eines Programms den Wert HIGH oder LOW einnehmen kann. „Digital“ bedeutet, wie bei den digitalen Ein- und Ausgängen, dass es genau zwei Zustände gibt:
- AN („true“, „HIGH“ bzw. „1“) und
- AUS („false“, „LOW“ bzw. „0“)

Du kannst den digitalen Variablen beliebige Namen geben. Mit sinnvolle Namen („LED_ROT“ statt „xyz“) versteht man ein Programm viel besser (→ Fehlersuche).



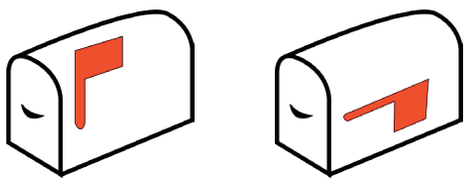
Für was brauche ich eine digitale Variable?

Eine digitale Variable brauchst du immer dann, wenn du einen digitalen Wert zwischenspeichern willst, um ihn später weiterzuverarbeiten (z.B. um ihn zu vergleichen) oder an einer anderen Stelle im Programm auszuwerten (z.B. mit falls/sonst).

Beispiel „amerikanischer Briefkasten“ (aus Kapitel 2.5):
Der Postbote setzt am Briefkasten eine kleine rote Flagge, wenn er neue Post eingeworfen hat. Die Flagge bleibt oben und speichert so die Eingabe vom Briefträger. Der Hausbewohner muss also nur zum Postkasten, wenn die Flagge gesetzt ist.

→ Die rote Flagge ist eine digitale Variable, die
- von dem Postbosten auf „true“ gesetzt wird und
- von dem Hausbewohner gelesen wird (und auf „false“ zurückgesetzt)

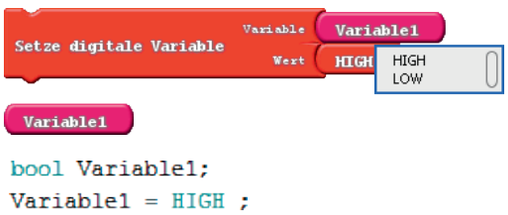
der darauf hin entscheidet, welche Aktion er macht („falls/sonst“-Entscheidung)



Wie verwende ich eine digitale Variable?

Wenn du einer digitalen Variable einen Wert zuweisen willst, brauchst du den Block „setze digitale Variable“ aus dem roten Menü „Variablen/Konstanten“. Dort kannst du
- den Namen der Variablen angeben (violett)
- den Wert der Variablen festlegen (rot)
  oder z.B. auf den Zustand eines digitalen Eingangs setzen

Willst du den Wert der Variable lesen (oder vergleichen), kannst du den violetten Block klonen (rechte Maustaste) oder aus dem selben Menü holen. Wichtig ist dabei, dass
- die Namen identisch sind
- die Namen nicht nur aus einer Zahl bestehen und
- die Namen keine Leerzeichen oder Sonderzeichen enthalten




```
bool Variable1;
Variable1 = HIGH ;
```

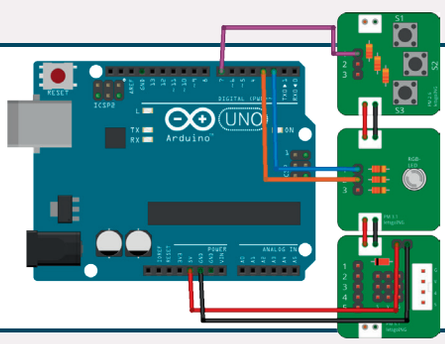
Übungsaufgabe

Übung

Schreibe ein Programm das den Wert einer digitalen Variable invertiert*, wenn ein Taster gedrückt wird. Der Wert der Variable soll auf einen digitalen Ausgang geschrieben werden. Schließe eine LED an diesen Ausgang an.

***invertieren → umkehren:** Verwende dazu den „NICHT“-Block aus dem rosa Menü „Log. Operatoren“. **TIPP:** Variable1 = NICHT Variable1

Verwende den Aufbau wie in Kapitel 2.4 und 2.5.



Troubleshooting

Die Werte verändern sich nicht / werden nicht übernommen
- → hast du überall exakt denselben Variablennamen verwendet? TIPP: Variablen-Block (violett) klonen
- → hast du immer die richtigen Blöcke verwendet? rot → Werte / violett → Variablen (-namen)
- → der Taster reagiert manchmal mehrfach. → Füge eine kurze Wartepause (200ms) ein

2.7 Übungsprojekt

Projektbeschreibung

Die LED-Technik wird in der Beleuchtungstechnik immer wichtiger. Egal ob zu Hause (z.B. indirekte Beleuchtung im Garten/Wohnzimmer, AmbiLight am Fernseher) oder in der Veranstaltungstechnik (z.B. Bühnenbeleuchtung, Großleinwände) - überall werden immer häufiger RGB-LEDs verwendet.

Will man nicht nur die drei Grundfarben erzeugen, muss man die Farben mischen.
Dazu kann man in der einfachsten Variante 7 Farben erzeugen:
rot, grün, blau, gelb, cyan (türkis), magenta (violett) und weiß

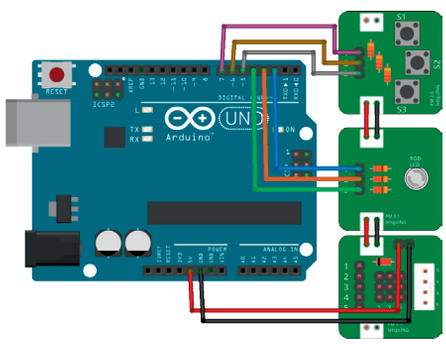
Aufgabenstellung

Schreibe ein Programm mit dem man alle 7 Farben darstellen kann.
- Je ein Taster soll eine LED an- und ausschalten (Taster 1 → rot, Taster 2 → grün,...)
- Die LEDs sollen mit dem ersten Tastendruck ein- und mit dem zweiten ausgeschalten werden (→ siehe Übung Kapitel 2.6)
- Die Farben sollen beliebig miteinander kombiniert werden können

Zusatzaufgabe:
Lasse die jeweiligen LEDs blinken wenn die zugehörige Farbe aktiviert ist.

Versuchsaufbau

Du brauchst für dieses Übungsprojekt deinen Arduino, das Adapter-Modul, das Taster-Modul und das RGB-Modul.
Verwende das Adapter-Modul um das RGB- und das Taster-Modul mit Strom zu versorgen.



Schließe jetzt alle LEDs und alle Taster an.

LED1 (blau) → Pin2	Taster1 → Pin7
LED2 (rot) → Pin3	Taster2 → Pin6
LED3 (grün) → Pin4	Taster3 → Pin5

Troubleshooting

LED geht nicht an / LED geht nicht aus
- → hast du im Programm denselben Pin verwendet wie am Arduino?
- → wird die LED an- und ausgeschaltet (HIGH / LOW)?
- → sind die Pausen an der richtigen Stelle?

Die Werte verändern sich nicht / werden nicht übernommen
- → hast du überall exakt den selben Variablennamen verwendet? TIPP: Variablen-Block (violett) klonen
- → hast du immer die richtigen Blöcke verwendet? rot → Werte / violett → Variablen (-namen)
- → der Taster reagiert manchmal mehrfach. → Füge eine kurze Wartepause (200ms) ein

Programm wird nicht hochgeladen
- → sind alle Einstellungen in der Arduino-Oberfläche richtig? (siehe Kapitel 1.3 Arduino IDE)
- → ist der Arduino eingesteckt?

Wissensfragen

1. Wie kommst du an die Information über den Zustand des Eingangs aus dem „digitalRead“-Block?
2. Was musst du machen, wenn ein Taster an einem digitalen Eingang mehrfach reagiert?
3. Auf was muss man achten wenn man Namen für Variablen erstellt?
4. Welchen Block brauchst du, um einen Wert in eine digitale Variable zu schreiben und welchen um ihn auszulesen?