

Arduino mega 2560 - Eine Anleitung für Schüler

Inhaltsverzeichnis

1 Einführung.....	3
2 Vorbereitungen.....	4
2.1 Installation unter Microsoft Windows.....	4
Windows: Arduino-Software installieren.....	4
Treiber-Installation unter Windows.....	4
Windows: Arduino-IDE konfigurieren.....	4
3 Schaltungen und Algorithmen.....	6
3.1 Erste Schritte.....	6
Aufgaben.....	8
3.2 Hausbeleuchtung.....	9
3.3 Hausbeleuchtung – anders programmiert.....	11
Aufgaben.....	12
3.4 Ein Hauch von Luxus – gedimmte Beleuchtung.....	13
Aufgaben.....	14
3.5 Bewegungsmelder.....	15
Aufgaben.....	17
3.6 Einstellen der Leuchtdauer.....	18
Aufgaben.....	19
zu Aufgabe 1.....	20
3.7 Alarm!.....	21
Aufgaben.....	23
3.8 Abschaltbarer Alarm.....	24
Aufgaben.....	26
3.9 Zerlegen von Problemen – Funktionen.....	26
Aufgaben.....	28
3.10 Beleuchtung? - Aber nur im Dunkeln!.....	28
Aufgaben.....	29
4 Anhang.....	31
4.1 Versuchsaufbau.....	31
4.2 Farbcodes von Widerständen.....	32

Arduino mega 2560 - Eine Anleitung für Schüler

1 Einführung

Viele stellen sich unter „Computer“ auch heute noch den typischen PC mit Bildschirm, Tastatur und Maus vor, vielleicht auch einen Notebook-Rechner. Man vergisst dabei häufig,

dass auch Smartphones nichts anderes sind als kleine Computer.

Wieder andere Computer nehmen wir in unserem Alltag gar nicht wahr, weil sie unsichtbar in anderen technischen Geräten eingebaut sind, etwa in Autos oder Waschmaschinen.

Solche kleinen Steuergeräte, auch Mikrocontroller genannt, werden – wenn man den Medien glauben darf – auch unser tägliches Wohnumfeld in Haus und Garten erobern: Stichwort „Smart Home“.

In diesem Kurs wollen wir schrittweise immer kompliziertere „Smart Home“-Technologien mit dem Mikrocontroller „Arduino mega 2560“ entwickeln, den wir natürlich auch selbst programmieren.

Hierfür brauchen wir den Mikrocontroller selbst (eine kleine Platine, auf der der eigentliche Computerchip sitzt), und verschiedene andere elektronische Bauteile. Außerdem benötigen wir einen PC oder Notebook-Rechner, mit dessen Hilfe wir die eigentliche Programmierung vornehmen.

Ein wirkliches Haus, das wir mit unserer Technik ausstatten können, haben wir im Kurs leider nicht. Als „Ersatz“ haben wir das sehr einfache Modell eines Hauses (auf einem laminierten Papier), damit wir uns besser vorstellen können, wie die von uns entwickelten Programme sich auf Haus und Garten auswirken.

Arduino mega 2560 - Eine Anleitung für Schüler

2 Vorbereitungen

Bevor wir richtig loslegen, müssen wir auf dem PC oder Laptop, den wir verwenden, die Programmierumgebung für den Arduino-Mikrocontroller installieren. Die einzelnen Schritte hängen dabei davon ab, welches Betriebssystem man nutzt. Sollte es Probleme bei der Installation geben, gibt es jede Menge Hilfen und Tipps, Texte und Filme im Internet. Vor der Installation der Software sollte der Arduino-Controller noch nicht an den Rechner angeschlossen sein.

2.1 Installation unter Microsoft Windows

Um mit Ihrem Arduino-Board programmieren zu können, müssen Sie zunächst die Windows-Treiber installieren. Wir zeigen Ihnen, wie es funktioniert.

Windows: Arduino-Software installieren

Mithilfe der Open-Source IDE-Software (integrated development environment) können Sie ihre Sketche auf Ihren Arduino-Microcontroller laden.

- Laden Sie sich die Installationsdatei der Arduino-IDE (<https://www.arduino.cc/en/Main/Software>) herunter und starten Sie die Installation.

Treiber-Installation unter Windows

Schließen Sie Ihr Arduino-Board mit einem USB-Kabel an Ihren Computer an. Windows startet eine Treiber-Installation, die aber meistens fehlschlägt. Ist das der Fall, gehen Sie folgendermaßen vor:

1. Klicken Sie auf den Start-Button und suchen Sie nach dem "Geräte-Manager".
2. Damit Sie Ihr Arduino-Board finden, drücken Sie auf des kleine Dreieck vor dem Punkt "Anschlüsse (COM & LPT)".
3. Klicken Sie mit der rechten Maustaste auf Ihren Arduino und wählen Sie die Option "Treibersoftware aktualisieren".
4. Drücken Sie auf "Auf dem Computer nach Treibersoftware suchen" und navigieren Sie zum Pfad: "C/Programme/Arduino/drivers". Bestätigen Sie die Treiber-Installation und folgen Sie den weiteren Schritten. Der Treiber ist nun erfolgreich installiert.

Windows: Arduino-IDE konfigurieren

Damit Sie Sketche mithilfe der Software auf Ihr Arduino-Board laden können, müssen Sie das richtige Board und den richtigen Port auswählen:

Arduino mega 2560 - Eine Anleitung für Schüler

- Klicken Sie hierzu in der Arduino-IDE auf "Werkzeuge" und wählen Sie unter Board Ihren Arduino-Controller aus.
- Wiederholen Sie diesen Schritt für den Port, den das Arduino-Board verwendet.

Grundlegende Programmier-Kenntnisse helfen Ihnen, Ihre ersten Arduino-Sketches zu schreiben. Im Internet finden Sie zahlreiche Anlaufstellen, um das Programmieren kostenlos online zu lernen. (https://praxistipps.chip.de/kostenlos-online-programmieren-lernen_28831)

Arduino mega 2560 - Eine Anleitung für Schüler

3 Schaltungen und Algorithmen

3.1 Erste Schritte

Wir fangen klein an und probieren uns an der kleinstmöglichen „Schaltung“, die nämlich nur aus dem Arduino selbst besteht.

Wir schließen diese über die USB-Leitung an den Computer an und starten die Arduino-IDE.

In dieser geben wir das folgende Programm ein.

Im Menü Werkzeuge muss die Einstellung Board auf Arduino mega eingestellt werden (s.u.)

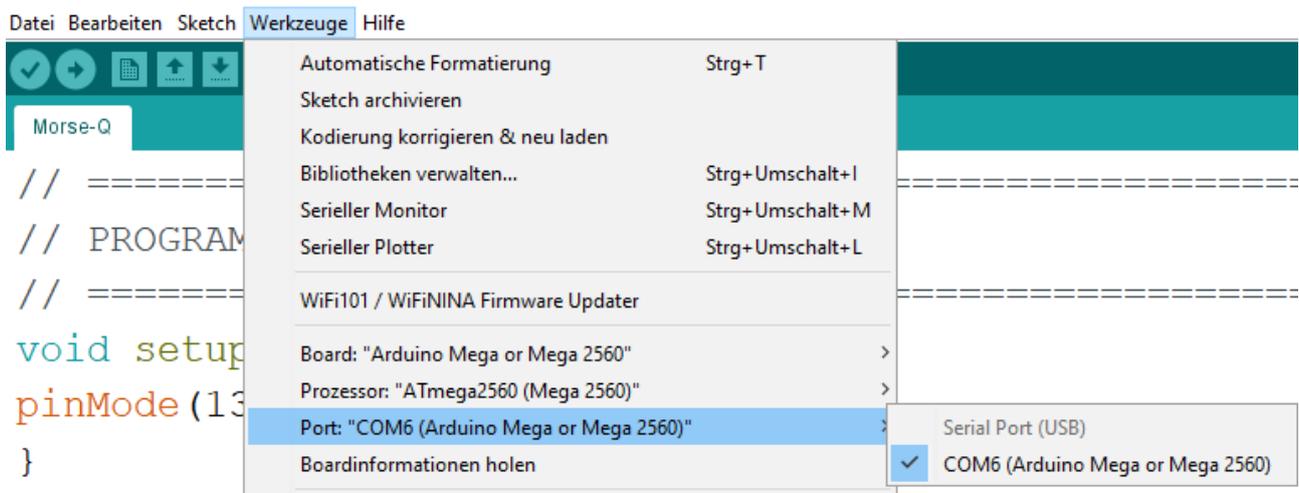
```
// =====  
// PROGRAMM: 01_DigitalOut  
// =====  
void setup() {  
    pinMode(13, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(13, HIGH);  
    delay(1000);  
    digitalWrite(13, LOW);  
    delay(500);  
}
```

Die Zeilen mit „//“ am Anfang sind nur Kommentare, die für das Funktionieren des Programms selbst nicht wichtig sind. Man muss sie nicht abschreiben, wenn man es eilig hat.

Bevor das Programm für den mega in dessen Sprache übersetzt und auf ihn übertragen werden kann, müssen noch einige Einstellungen vorgenommen werden.

Im Menü Werkzeuge muss die Einstellung Board auf Arduino mega 2560 eingestellt werden, der Prozessor auf ATmega2560. Die Einstellung für Port hängt vom verwendeten Rechner ab.

Arduino mega 2560 - Eine Anleitung für Schüler



Unter Microsoft Windows sollte auf jeden Fall COMx ausgewählt sein, wobei x für irgendeine Zahl steht. Meist ist es die letzte der im Auswahlménüs genannten Ports, es kann jedoch auch anders sein; im Notfall muss man alle einmal ausprobieren.

Sind die Einstellungen richtig gewählt, kann man zunächst testen, ob der geschriebene Computer-Code „grammatisch“ stimmig ist. Dafür drückt man die runde Taste mit dem Haken. Ist der Code korrekt, erscheint im Meldungsfenster „Kompilieren abgeschlossen“. War der Code nicht korrekt oder ein anderer Fehler ist aufgetreten, erscheint in dem schwarzen Feld unter dem Meldungsfenster ein entsprechender Hinweis.

Nun kann man versuchen, den übersetzten Code auf den Mikrocontroller hochzuladen. Hierfür drückt man die runde Taste mit dem Pfeil nach rechts. War das Hochladen erfolgreich, erfolgt im Meldungsfenster die Meldung „Hochladen abgeschlossen“. Ist ein Fehler aufgetreten, erscheint in dem schwarzen Feld unter dem Meldungsfenster ein entsprechender Hinweis.

Nach erfolgreichem Hochladen sollte eine der eingebauten LEDs des Arduino blinken, eine Sekunde an, eine halbe Sekunde aus usw.

Man sieht, dass das Programm in zwei Teile zerfällt:

- Einen Teil, der sich Setup nennt, und in dem festgelegt wird, was direkt nach dem Hochladen bzw. nach dem Einschalten der Versorgungsspannung geschehen soll.
- Einen Teil, der sich Loop (engl. für Schleife), in dem festgelegt wird, was nach dem Setup immer und immer wieder passieren soll.

Arduino mega 2560 - Eine Anleitung für Schüler

Betrachten wir die Zeilen des Programms nun einzeln:

<code>void setup() {</code>	Hier beginnt die Setup-Routine
<code>pinMode(13, OUTPUT);</code>	//Hier wird der Pin D13 als Ausgang festgelegt. (interne LED)
<code>}</code>	//Hier endet die Setup-Routine

<code>void loop() {</code>	//Hier beginnt die Loop-Routine
<code>digitalWrite(13, HIGH);</code>	//Der Pin D13 wird auf die Spannung +5V gesetzt, //so dass durch die LED Strom fließen kann.
<code>delay(1000);</code>	//Der Controller wartet 1000 Millisekunden, //also eine Sekunde lang.
<code>digitalWrite(13, LOW);</code>	//Der Pin D13 wird auf Spannung 0V gesetzt, //so dass durch die LED kein Strom fließen kann.
<code>delay(500);</code>	//Der Controller wartet 500 Millisekunden, //also eine halbe Sekunde lang.
<code>}</code>	//Hier endet die Loop-Routine

Aufgaben

1. Andere Zeiteinstellungen

Verändere die Zeitangaben so, dass die Lampe immer abwechselnd 5 Sekunden lang leuchtet und dann 2 Sekunden lang dunkel bleibt.

2. Morsen

Programmiere den Arduino so, dass er eine Folge von „Q“s morst. Das Morsezeichen für „Q“ ist lang-lang-kurz-lang (– – . –).

Zwischen den einzelnen „Q“s soll jeweils eine Pause von 1 Sekunde sein.

Überlege Dir sinnvolle Längeneinstellungen für „kurz“ und „lang“, sowie die Pausen dazwischen.

Arduino mega 2560 - Eine Anleitung für Schüler

3.2 Hausbeleuchtung

Bisher haben wir nur die interne Leuchtdiode (LED) verwendet. Wir wollen jedoch nun den Garten beleuchten, müssen also eine externe LED verwenden.

Exkurs: Kurze LED-Kunde

(<https://www.inf-schule.de/vernetzung/raspi/grundlagen/led>)

Bei der LED ist wichtig, dass man die Kathode(-) und die Anode(+) unterscheiden kann. Das ist relativ leicht, wenn man folgendes weiß:

Die Seite der LED, auf der die Kathode ist, hat das kürzere "Beinchen". Auf dieser Seite ist auch das LED-Gehäuse abgeflacht und im Gegenlicht erkennst du ein "Fähnchen" innerhalb der LED.

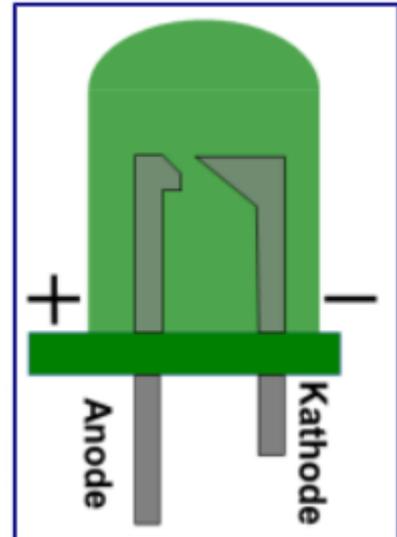


Schaubild 1: LED schematisch

Die Kathode muss an den Minus-Pol, also an die Erdung (GND) angeschlossen werden. Die Anode muss an den Plus-Pol angeschlossen werden. Vertauschst du die Anschlüsse der LED, kann folgendes passieren: Wenn du Glück hast, leuchtet sie einfach nicht. Wenn du Pech hast und die Spannung zu hoch war, ist sie zerstört.

Diese darf nur über einen Widerstand betrieben werden, der in unserem Fall einen Widerstandswert von 330Ω hat 2 . (orange – orange – Schwarz – schwarz – braun). Wir verbinden den Pin D5 mit dem Widerstand und diesen mit dem langen Pin der Leuchtdiode. Den kurzen Pin der Leuchtdiode verbinden wir mit der durchgängigen blauen Linie unten. Dies wiederum verbinden wir mit dem Pin GND des mega.

Tipp: ggf. die Kabel (Kupplungsuchse) um 90° drehen, wenn sie auf den „Füßchen“ der externen Geräte nicht halten.

Wir wollen unsere Software so schreiben, dass die Beleuchtung zwischen drei Zuständen wechselt.

In den Abendstunden zwischen 18:00 und 0:00 Uhr soll der Garten hell erleuchtet sein, denn

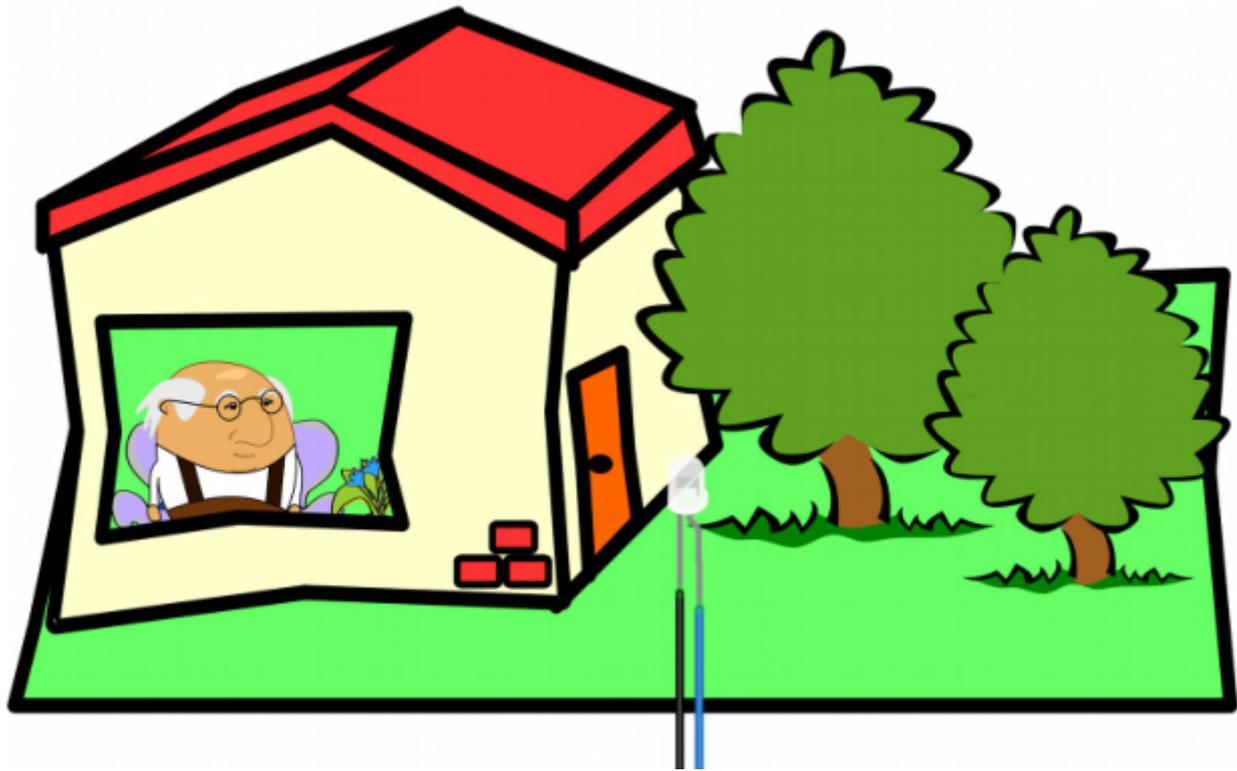
unser Auftraggeber feiert abends gerne Gartenpartys. Zwischen 0:00 und 6:00 Uhr stört ihn

das grelle Licht beim Schlafen, deshalb soll hier das Licht zwar nicht ausgeschaltet, aber gedimmt sein (als Abschreckung für mögliche Einbrecher). Zwischen 6:00 und 18:00 Uhr soll

das Licht ganz ausgeschaltet sein, da es zumeist hell ist.

Damit wir nicht stundenlang warten müssen, reduzieren wir in unserem Modell die Stunden auf Sekunden: 6 Sekunden helles Licht, 6 Sekunden gedimmtes Licht, 12 Sekunden kein Licht.

Arduino mega 2560 - Eine Anleitung für Schüler



```
// =====  
// PROGRAMM: 02_Beleuchtung  
// =====  
void setup() {  
    pinMode(5, OUTPUT);  
}  
  
void loop() {  
    analogWrite(5, 255);  
    delay(6000);  
    analogWrite(5, 32);  
    delay(6000);  
    analogWrite(5,0);  
    delay(12000);  
}
```

Als Befehl neu hinzugekommen ist nur der Befehl
`analogWrite(Pin, Wert);`

Die Pin-Angabe gibt an, welcher Pin gerade angesteuert wird. Im Beispiel der PIN D5.
Die Wert-Angabe muss einen Wert zwischen 0 und 255 haben. Je nach Wert leuchtet die
Leuchtdiode verschieden hell ; 0 bedeutet: ganz dunkel; 255: ganz hell.

Arduino mega 2560 - Eine Anleitung für Schüler

3.3 Hausbeleuchtung – anders programmiert

Betrachtet man das Programm aus dem vorigen Abschnitt, so können wir es wahrscheinlich ganz gut verstehen. Nehmen wir aber an, jemand, der nicht soviel Erfahrung hat wie wir, versucht das Programm zu verändern. Beispielsweise könnte er die Beleuchtungslängen ändern wollen, weil die Gartenpartys doch häufig bis 1:00 Uhr nachts dauern.

Für eine solche Änderung muss man aber wissen, was die Zahlen bedeuten und man darf sich nicht irren. Wäre es nicht viel besser, man wüsste, welche Bedeutung die einzelnen Zahlen haben und man müsste sie sich nicht im Programmtext suchen?

Das folgende Programm tut genau das gleiche wie das vorhergehende Programm, ist aber deutlich übersichtlicher (obwohl es länger ist).

```
// =====  
// PROGRAMM: 03_Beleuchtung2  
// =====  
int ledPin = 5;  
int hell = 255;  
int gedimmt = 32;  
int dunkel = 0;  
int abendlaenge = 6000;  
int nachtlaenge = 6000;  
int taglaenge = 24000-abendlaenge-nachtlaenge;  
  
void setup() {  
    pinMode(ledPin, OUTPUT);  
}  
  
void loop() {  
    analogWrite(ledPin, hell);  
    delay(abendlaenge);  
    analogWrite(ledPin, gedimmt);  
    delay(nachtlaenge);  
    analogWrite(ledPin, dunkel);  
    delay(taglaenge);  
}
```

Die Zeilen, die mit dem Schlüsselwort `int` beginnen, bedeuten, dass eine so genannte Variable mit einem bestimmten Namen existieren soll und möglicherweise auch, welchen Wert sie haben soll. `int` steht dabei für `integer`, das englische Wort für „ganze Zahl“. `int ledPin = 5` bedeutet: Es soll ab hier eine ganzzahlige Variable `ledPin` existieren, und sie soll den Wert 5 haben.

Durch die Verwendung von „sprechenden“ Variablennamen wird das Programm zwar länger, aber deutlich lesbarer. Auch genügt es für kleinere Änderungen, die Variablenwerte am Anfang zu ändern, man muss nicht die entsprechenden Programmzeilen im ganzen Programm suchen. Statt der Zahlen selbst stehen nun die Variablennamen in den einzelnen Befehlen.

Arduino mega 2560 - Eine Anleitung für Schüler

Aufgaben

1. Sinnvolles Rechnen-Lassen

Beschreibe, was die Programmzeile

```
int taglaenge = 24000-abendlaenge-nachtlaeenge;
```

bedeutet.

Begründe, weshalb diese Zeile sinnvoller ist als einfach

```
int taglaenge = 12000;
```

zu schreiben.

2. Beleuchtung für Frühaufsteher

Damit Frühaufsteher auch eine Beleuchtung haben, soll auch morgens in der Dämmerung die volle Beleuchtung eingeschaltet sein.

Verändere das Programm so, dass hier auch eine entsprechende neue Variable eingeführt wird.

Bedenke dabei: Der Tag hat trotzdem nur 24 Stunden!

Arduino mega 2560 - Eine Anleitung für Schüler

3.4 Ein Hauch von Luxus – gedimmte Beleuchtung

Wenn abends um 18:00 Uhr die Beleuchtung plötzlich angeschaltet wird, wirkt das nicht besonders luxuriös. Einen viel edleren Eindruck bekommt man, wenn das Licht langsam hochgedimmt wird.

Auch das können wir mit einer kleinen Erweiterung des Programms gut erreichen:

```
// =====  
// PROGRAMM: 04_Dimmer  
// =====  
int ledPin = 5;  
int hell = 255;  
int gedimmt = 32;  
int dunkel = 0;  
int abendlaenge = 6000;  
int nachtlaenge = 6000;  
int taglaenge = 24000-abendlaenge-nachtlaenge;  
int wert;  
  
void setup() {  
    pinMode(ledPin, OUTPUT);  
}  
  
void loop() {  
    for (wert=dunkel; wert<=hell; wert++){  
        analogWrite(ledPin, wert);  
        delay(20);  
    }  
    delay(abendlaenge);  
    analogWrite(ledPin, gedimmt);  
    delay(nachtlaenge);  
    analogWrite(ledPin, dunkel);  
    delay(taglaenge);  
}
```

Wir betrachten die neu hinzugekommenen Zeilen:

<pre>for (wert=dunkel; wert<=hell; wert++){</pre>	<pre>//Die Variable wert wird auf jeden Wert zwischen dunkel und hell gesetzt, bei dunkel //angefangen. Für jede solche Belegung der Variablen Wert wird der Programmteil //zwischen den geschweiften Klammern einmal ausgeführt 4 .</pre>
<pre> analogWrite(ledPin, wert);</pre>	<pre>//Die Helligkeit wird auf den entsprechenden Wert wert gesetzt.</pre>

Arduino mega 2560 - Eine Anleitung für Schüler

<code>delay(20);</code>	<code>//Zwischen den einzelnen Wertänderungen muss ein wenig Zeit vergehen //(hier 20ms), damit das Dimmen auch langsam genug erfolgt.</code>
<code>}</code>	<code>//Hier endet das wiederholte Programmstück. //Man spricht bei solchen Wiederholungen auch von Schleifen. Bei dieser Art von //Schleife wird eine Variable (hier: wert) von einem Startwert nach oben gezählt. //Man spricht deshalb bei den for-Schleifen auch von Zählschleifen.</code>

Aufgaben

1. Unterschiedlich schnelles Dimmen

Bisher wurden fast alle wichtigen Größen über Variablen festgelegt, allerdings noch nicht, wie rasch bzw. wie langsam das Heller-Werden des Lichts geschehen soll. Erfinde hierfür eine neue Variable.

2. Luxus bei allen Übergängen

Erweitere das Programm so, dass alle Übergänge fließend geschehen.

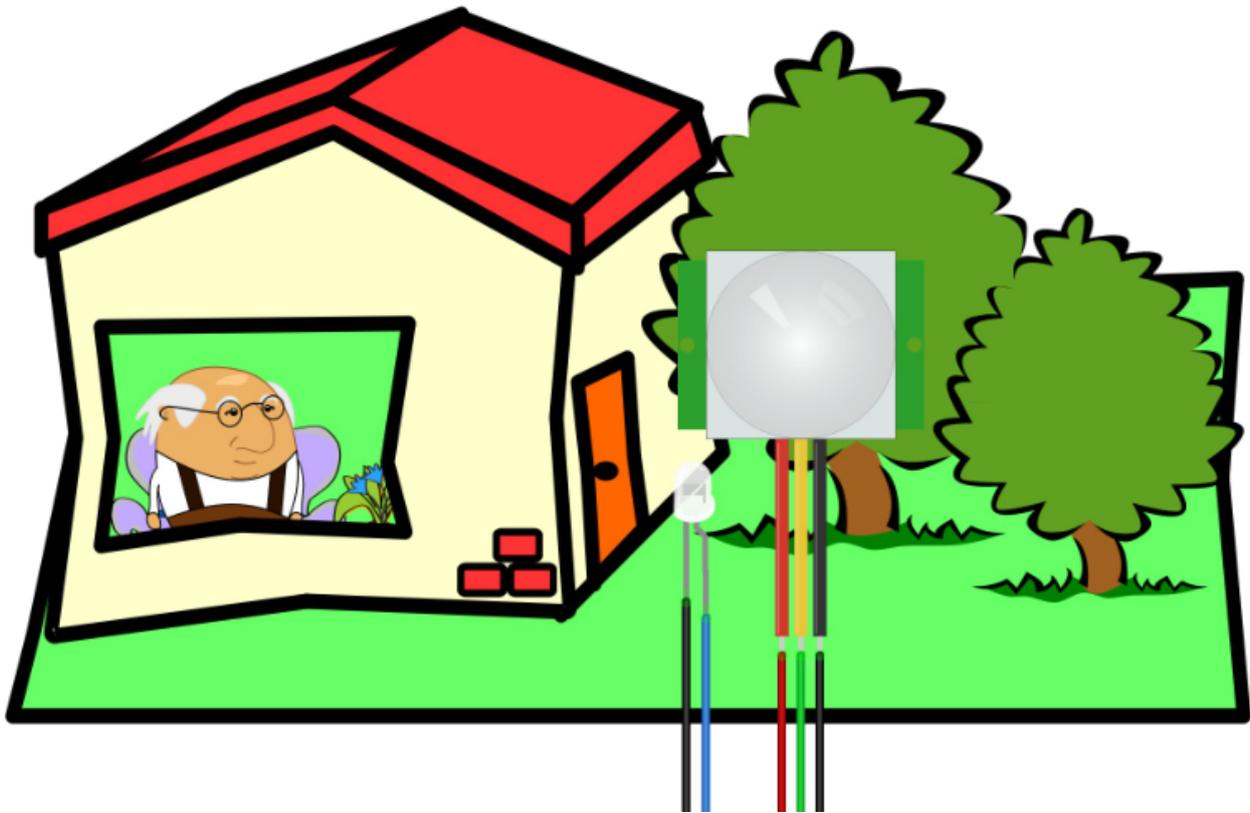
3. Genauigkeit

Schaut man sich das Beispielprogramm genau an, so erkennt man, dass hier in Wirklichkeit ein Tag etwas mehr als 24 Sekunden lang ist. Die „eingebaute Uhr“ verstellt sich also an jedem simulierten Tag.

Ändere das Programm so ab, dass wieder ein genauer 24-Sekunden-Rhythmus vorliegt.

Arduino mega 2560 - Eine Anleitung für Schüler

3.5 Bewegungsmelder



Bisher wurde die Beleuchtungsanlage nur durch die Zeit gesteuert, es gab keine Sensoren, die Umwelteinflüsse aufnehmen konnten, um die Beleuchtung davon abhängig zu machen.

Das ändert sich nun: wir schließen zusätzlich zur Leuchtdiode einen Bewegungsmelder (einen so genannten Pyroelektrischen Sensor, kurz PIR-Sensor) an unseren mega an.

Der Bewegungsmelder benötigt eine Spannungsversorgung. Hierfür verbinden wir den Pin 5V mit der roten, horizontalen Buchsenleiste und verbinden mit dieser auch den Eingang Vcc des Bewegungsmelders. Den Anschluss GND des Bewegungsmelders verbinden wir mit der blauen horizontalen Buchsenleiste, der bereits mit GND des mega verbunden war. Den Ausgang des Bewegungsmelders verbinden wir mit Pin D6.

Arduino mega 2560 - Eine Anleitung für Schüler

Vorsicht:

Die Reihenfolge der Anschlüsse beim PIR-Sensor muss nicht unbedingt der Abbildung entsprechen. Manchmal ist die Beschriftung der Anschlüsse direkt an der Unterseite des PIR-Sensors ablesbar, manchmal muss man aber auch die Kunststoffkappe abziehen.

Informationen zum Bewegungssensor:

Details: <https://draeger-it.blog/arduino-lektion-15-pyroelektrischer-infrarot-motion-sensor-pir/?cn-reloaded=1>

Aufbau und Funktion: <https://www.elv.de/elektronikwissen/pir-sensor-aufbau-und-funktion.html>

Nun zum entsprechenden Programm

```
// =====  
// PROGRAMM: 05_DigitalIn  
// =====  
int ledPin = 5;  
int pirPin = 6;  
int hell = 255;  
int gedimmt = 32;  
int dunkel = 0;  
int leuchtdauer = 5000;  
  
void setup() {  
    pinMode(ledPin, OUTPUT);  
    pinMode(pirPin, INPUT);  
}  
  
void loop() {  
    if (digitalRead(pirPin)==HIGH){  
        analogWrite(ledPin,hell);  
        delay(leuchtdauer);  
    }  
    else {  
        analogWrite(ledPin,dunkel);  
        delay(100);  
    }  
}
```

Am Programmcode sieht man: Wir müssen einen neuen Pin als Eingang festlegen (Integer-Variable pirPin bzw. pinMode(pirPin, INPUT);) .

Der Zustand des Sensors wird mit dem Befehl digitalRead ausgelesen. Dieser hat den Wert HIGH, wenn sich in den letzten Sekunden etwas vor dem Sensor bewegt hat, sonst den Wert LOW.

Arduino mega 2560 - Eine Anleitung für Schüler

Wir betrachten die wichtigsten Befehle im Einzelnen:

<code>if (digitalRead(pirPin)==HIGH){</code>	Wenn der Bewegungsmelder eine Bewegung festgestellt hat, dann ...
<code> analogWrite(ledPin,hell);</code>	schalte die Beleuchtung an
<code> delay(leuchtdauer);</code>	warte die Zeit leuchtdauer ab.
<code>}</code>	Ende des „dann“-Teils
<code>else {</code>	... ansonsten ...
<code> analogWrite(ledPin,dunkel);</code>	schalte die Beleuchtung ab
<code> delay(100);</code>	warte 0,1 Sekunden.
<code>}</code>	Ende des „ansonsten“-Teils.

Man beachte die beiden Gleichheitszeichen in der if-Anweisung.

Aufgaben

1. Unterschiedliche Verzögerungszeiten

Begründe, weshalb es sinnvoll ist, dass im else-Teil die Delay-Anweisung nur 100ms beträgt, und nicht etwa die Leuchtdauer.

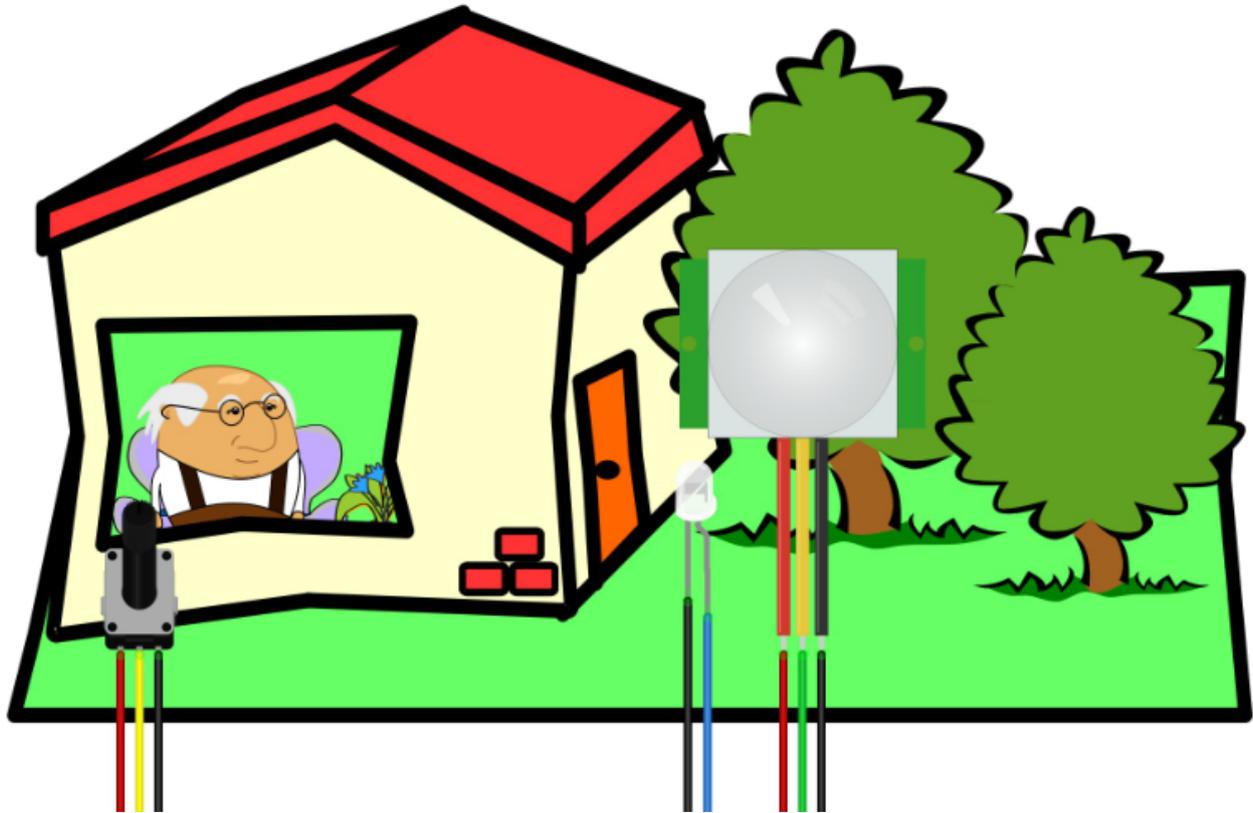
2. Wieder ein bisschen Luxus

Es erscheint sinnvoll, dass direkt nach einer beobachteten Bewegung das Licht mit voller Helligkeit scheint. Beim Wieder-Abschalten würde ein „Herunterdimmen“ jedoch wieder einen luxuriöseren Eindruck machen.

Ändere das Programm entsprechend.

Arduino mega 2560 - Eine Anleitung für Schüler

3.6 Einstellen der Leuchtdauer



Bisher war die Leuchtdauer, wenn eine Bewegung festgestellt wurde, durch das Programm starr festgelegt. Das soll sich nun ändern: die Hausbewohner sollen die Leuchtdauer an einem Drehknopf (genaugenommen: Drehwiderstand) selbst einstellen können.

Ein Drehregler hat drei Anschlüsse. Außen wird + und – angeschlossen. Von dem mittleren Pin geht ein Kabel zu einem analogen Eingangspin am Mikrocontroller-Board. Wenn man den Drehregler dreht, dann gibt der mittlere Pin eine Spannung zwischen 0 und 5 Volt aus.

Drehregler ganz links: 0 V und Drehregler ganz rechts: 5V, bzw. Seitenverkehrt, je nach Verkabelung.

Das erreichen wir mit der folgenden Schaltung:

Der Drehwiderstand ist – wie der Bewegungsmelder auch – mit 5V und GND verbunden. Am mittleren Pol des Widerstands liegt nun – je nach Stellung – eine Spannung zwischen 0V und 5V an. Dieser Pol ist mit Pin A1 verbunden und die Stellung des Drehwiderstands kann mit Software ausgelesen werden.

Das Programm sieht so aus:

Arduino mega 2560 - Eine Anleitung für Schüler

```
// =====  
// PROGRAMM: 06_AnalogIn  
// =====  
int ledPin = 5;  
int pirPin = 6;  
int potiPin = 1;  
int hell = 255;  
int gedimmt = 32;  
int dunkel = 0;  
int leuchtdauer;  
  
void setup() {  
    pinMode(ledPin, OUTPUT);  
    pinMode(pirPin, INPUT);  
}  
  
void loop() {  
    leuchtdauer = analogRead(potiPin)*10;  
    if (digitalRead(pirPin)==HIGH){  
        analogWrite(ledPin,hell);  
        delay(leuchtdauer);  
    }  
    else {  
        analogWrite(ledPin,dunkel);  
        delay(100);  
    }  
}
```

Es hat sich eigentlich nur eine Sache geändert. Statt die Leuchtdauer direkt auf den Wert 5000 festzulegen, wird dieser Wert bei jedem Durchlauf der Schleife neu berechnet:

```
leuchtdauer = analogRead(potiPin)*10
```

Der Befehl `analogRead(...)` 5 liest den Wert des in Klammern angegebenen Analogpins aus.

Für 0V Spannung beträgt dieser Wert 0, für 5V Spannung beträgt dieser Wert 1023.

Entsprechend wird hier die Leuchtdauer auf einen Wert zwischen 0 und 10230 eingestellt.

In der Praxis zeigt sich, dass die Leuchtdauer trotzdem mindestens etwa 3 Sekunden

beträgt. Das liegt daran, dass der PIR-Sensor ein etwa 3 Sekunden langes Signal ausgibt, so dass die Lampe bei kürzeren Einstellungen immer wieder neu angeschaltet wird.

Aufgaben

1. Mehrere Einstellmöglichkeiten

Erweitere die Schaltung um einen weiteren Drehwiderstand und ändere die Software so ab, dass man nicht nur die Leuchtdauer, sondern auch die Helligkeit der Beleuchtung einstellen kann.

Tipp: Der Wert einer Variable (z.B. helligkeit) kann maximal 255 betragen. Da

Arduino mega 2560 - Eine Anleitung für Schüler

potiHell max. 1023 liefert, kann man dies z.B. wie unten umsetzen:
helligkeit = int(analogRead(potiHell)/4);

2. Mehr Licht!

Selbst bei maximaler Helligkeit der einen LED ist den Bewohnern die Beleuchtung nicht stark genug.

Erweitere die Schaltung um eine weitere LED an Ausgang D9. Ab einem bestimmten Wert des Helligkeits-Drehwiderstands aus Aufgabe 1 soll die zweite LED langsam zugeschaltet werden.

Tipp: Der Wert einer Variable kann leicht mit einem festen Wert verglichen werden; hier etwa die Variable mit dem Wert 30 verglichen:

```
if (meine_variable > 30) {
```

```
...
```

```
}
```

```
else {
```

```
...
```

```
}
```

zu Aufgabe 1

Das Programm sieht dannso aus:

```
// =====  
// PROGRAMM: 06_1_AnalogIn  
// =====  
int ledPin = 5;  
int pirPin = 6;  
int potiPin = 1;  
int potiHell = 2;  
int hell = 255;  
int gedimmt = 32;  
int dunkel = 0;  
int leuchtdauer;  
int helligkeit;  
  
void setup() {  
    pinMode(ledPin, OUTPUT);  
    pinMode(pirPin, INPUT);  
    Serial.begin(9600);  
}  
  
void loop() {  
    leuchtdauer = analogRead(potiPin)*10;  
    helligkeit = int(analogRead(potiHell)/4);  
    Serial.print("potiPIN= ");  
    Serial.println(analogRead(potiPin));  
    Serial.print("Helligkeit= ");  
    Serial.println(analogRead(helligkeit));
```

Arduino mega 2560 - Eine Anleitung für Schüler

```
Serial.print("Bewegungssensor = ");
Serial.println(digitalRead(pirPin));
if (digitalRead(pirPin)==HIGH){
    analogWrite(ledPin,helligkeit);
    delay(leuchtdauer);
}
else {
    analogWrite(ledPin,dunkel);
    delay(100);
}
}
```

3.7 Alarm!

In der Nachbarschaft ist in der letzten Zeit häufiger eingebrochen worden. Unsere Hausbewohner möchten ihre Hausbeleuchtung auch als Alarmanlage verwenden können, jedoch dabei nur die Software ändern.

Die Idee: Ist der Drehwiderstand über einem bestimmten Wert eingestellt, soll die Hausbeleuchtung nicht nur einfach leuchten, sondern 20 mal blinken. Die Hoffnung: Durch das auffällige Signal werden potentielle Einbrecher abgeschreckt und die Nachbarschaft wird

aufmerksam. Wir betrachten das zugehörige Programm:

```
// =====
// PROGRAMM: 07_AnalogIn
// =====
int ledPin = 5;
int pirPin = 6;
int potiPin = 1;
int hell = 255;
int gedimmt = 32;
int dunkel = 0;
int leuchtdauer;
int zaehler;

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(pirPin, INPUT);
}

void loop() {
    leuchtdauer = analogRead(potiPin)*10;
    if (digitalRead(pirPin)==HIGH){
        if (leuchtdauer>9000) {
            for (zaehler=1; zaehler<=20; zaehler++){
```

Arduino mega 2560 - Eine Anleitung für Schüler

```
        analogWrite(ledPin,hell);
        delay(50);
        analogWrite(ledPin,dunkel);
        delay(50);
    }
}
analogWrite(ledPin,hell);
delay(leuchtdauer);
}
else {
    analogWrite(ledPin,dunkel);
    delay(100);
}
}
```

Neu ist hier die Zeile

```
if (leuchtdauer>9000) {
```

Hier wird geprüft, ob die Leuchtdauer einen Wert von mehr als 9000 hat. Die wichtigsten Vergleichsoperatoren sind:

>	links größer als rechts
>=	links größer oder gleich als rechts
<	links kleiner als rechts
<=	links kleiner oder gleich als rechts
==	links gleich rechts
!=	links ungleich rechts

Arduino mega 2560 - Eine Anleitung für Schüler

Aufgaben

1. Kontroll-Lampe

Erweitere die Schaltung um ein rotes Kontroll-Lämpchen, das anzeigt, ob der Alarm scharf-geschaltet ist oder nicht.

2. Komplett-Abschaltung

Die kombinierte Beleuchtungs-/Alarmanlage soll durch eine Stellung des Drehwiderstands auch abschaltbar sein. Liegt der Wert der Beleuchtungsdauer unter 1000ms, soll die Lampe gar nicht erst anschalten.

3. Alarmblinken mit einstellbarer Länge

Mit einem zweiten Drehwiderstand soll die Länge des Alarmblinkens einstellbar sein.

4. Akustisches Warnsignal

Viele Alarmanlagen begnügen sich nicht mit einem Lichtsignal, sondern geben einen Alarmton von sich. Zusätzlich zur Leuchtdiode können wir (z.B. an Pin 10 oder 11) einen Lautsprecher anschließen, wie die Leuchtdioden auch ebenfalls über einen 330Ω-Widerstand.

Im Programm kann man nun festlegen, auf welchem Pin, wie lange und wie hoch ein Ton ausgegeben wird:

```
tone(10, 440, 200);
```

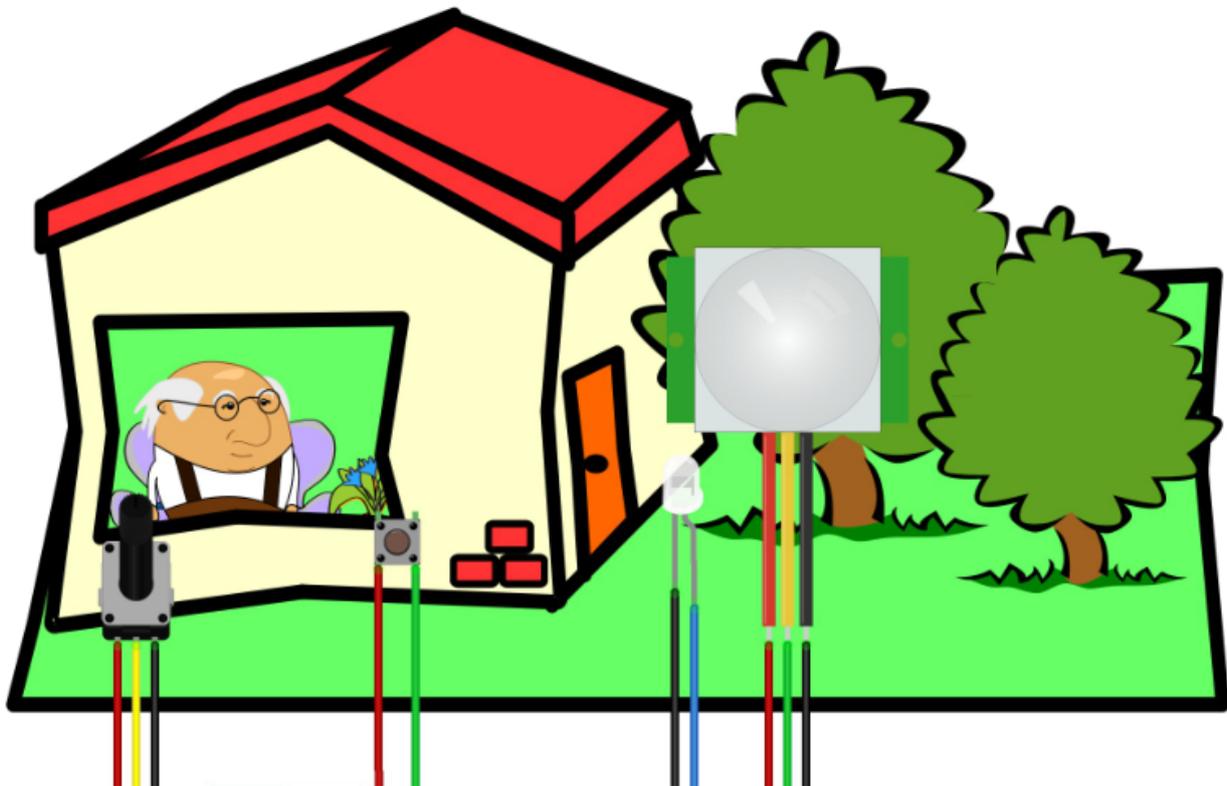
gibt auf Pin 10 einen Ton der Tonhöhe 440 Hertz für 200 Millisekunden aus. Wenn nach dem Ton eine Pause von 100 Millisekunden stattfinden soll, muss man

```
delay(300);
```

eingeben, also die 100 Millisekunden addiert zur Tonlänge von 200 Millisekunden.

Arduino mega 2560 - Eine Anleitung für Schüler

3.8 Abschaltbarer Alarm



Mittlerweile sind unseren Hausbewohnern Zweifel gekommen, ob 20-maliges Blinken ausreicht, um Eindringlinge abzuschrecken. Stattdessen soll die Alarmanlage nun solange blinken, bis man auf einen im Haus versteckten Taster drückt.

Hierfür muss unser Schaltplan wieder erweitert werden:

Wir sehen: Der Taster ist einerseits mit 5V verbunden, andererseits mit dem Eingang D4. Außerdem ist D4 – das ist ganz wichtig – über einen Widerstand von 1000 Ω (also 1k Ω) mit GND verbunden. Der Grund: Ansonsten hängt, wenn der Taster nicht gedrückt ist, der Pin D4 „in der Luft“ und man weiß nicht, ob dies dann als HIGH oder LOW gedeutet wird. Auch die Software muss entsprechend geändert werden:

```
// =====  
// PROGRAMM: 08_DigitalIn_R  
// =====  
int ledPin = 5;  
int pirPin = 6;  
int potiPin = 1;  
int tasterPin=4;  
int hell = 255;  
int gedimmt = 32;  
int dunkel = 0;
```

Arduino mega 2560 - Eine Anleitung für Schüler

```
int leuchtdauer;
int zaehler;
bool alarmAn;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(pirPin, INPUT);
  pinMode(tasterPin, INPUT);
}

void loop() {
  leuchtdauer = analogRead(potiPin)*10;
  if (digitalRead(pirPin)==HIGH){
    if (leuchtdauer>9000) {
      alarmAn=true;
      while (alarmAn){
        analogWrite(ledPin,hell);
        delay(50);
        analogWrite(ledPin,dunkel);
        delay(50);
        if (digitalRead(tasterPin)==HIGH){
          alarmAn=false;
        }
      }
    }
    analogWrite(ledPin,hell);
    delay(leuchtdauer);
  }
  else {
    analogWrite(ledPin,dunkel);
    delay(100);
  }
}
```

Wir betrachten den neuen Teil:

alarmAn=true;	Setze die (vorher deklarierte) Variable alarmAn vom Typ bool auf den Wert „wahr“ (engl.: true)
while (alarmAn){	Solange die Variable „wahr“ ist ...
analogWrite(ledPin,hell);	Schalte ein ...
delay(50);	Warte ...
analogWrite(ledPin,dunkel);	schalte wieder aus ...

Arduino mega 2560 - Eine Anleitung für Schüler

delay(50);	Warte ...
if (digitalRead(tasterPin)==HIGH){	Wenn der Taster gedrückt ist ...
alarmAn=false;	... setze die Variable alarmAn wieder auf „falsch“.
}	
}	

Wir haben es diesmal wieder mit einer Wiederholung zu tun. Diesmal jedoch gibt es keine feste Anzahl von Wiederholungen wie bei der for-Schleife, sondern hier wird solange geblinkt, solange eine Bedingung erfüllt ist: der Taster wurde noch nicht gedrückt, die Variable alarmAn ist (noch) true.

Aufgaben

1. An- und Ausschalten mit Tastern

Bislang wurde der Alarm über den Drehwiderstand ein- und ausgeschaltet. Jetzt, wo wir mit Tastern umgehen können, wäre es schön, wenn es einen Taster zum Anschalten und den schon vorhandenen Taster zum Ausschalten des Alarms gäbe. Hier ist es besonders wichtig, eine Kontrollleuchte zu haben, die mitteilt, ob der Alarm scharf ist oder nicht.

2. Ein- und Ausschalten mit einem einzigen Taster

Im Grunde ist eine Lösung mit zwei Tastern wie in Aufgabe 1 Verschwendung von Tastern.

Eigentlich würde es ja reichen, einen einzigen Taster zu haben: Bei einem hinreichend langen Druck des Tasters soll der Alarm scharf-geschaltet werden (Anzeige über die Kontrollleuchte), bei einem weiteren Druck auf den gleichen Taster soll der Alarm wieder ausgeschaltet werden.

3.9 Zerlegen von Problemen – Funktionen

Im Laufe unserer Entwicklung ist die Funktion loop ziemlich groß geworden. Und für einen Leser des Programmtextes mittlerweile auch recht unübersichtlich.

Eine Möglichkeit besteht darin, einzelne Teile des Programms des Programms als selbst definierte Befehle zu beschreiben, so genannte Funktionen.

Für unser Programm aus dem letzten Abschnitt könnte das so aussehen:

```
// =====  
// PROGRAMM: 09_Funktion  
// =====  
int ledPin = 5;  
int pirPin = 6;
```

Arduino mega 2560 - Eine Anleitung für Schüler

```
int potiPin = 1;
int tasterPin=4;
int hell = 255;
int gedimmt = 32;
int dunkel = 0;
int leuchtdauer;
int zaehler;
bool alarmAn;

void daueralarm(){
    alarmAn=true;
    while (alarmAn){
        analogWrite(ledPin,hell);
        delay(50);
        analogWrite(ledPin,dunkel);
        delay(50);
        if (digitalRead(tasterPin)==HIGH){
            alarmAn=false;
        }
    }
}

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(pirPin, INPUT);
    pinMode(tasterPin, INPUT);
}

void loop() {
    leuchtdauer = analogRead(potiPin)*10;
    if (digitalRead(pirPin)==HIGH){
        if (leuchtdauer>9000) {
            daueralarm();
        }
        analogWrite(ledPin,hell);
        delay(leuchtdauer);
    }
    else {
        analogWrite(ledPin,dunkel);
        delay(100);
    }
}
```

Man sieht: Neben den Funktionen setup und loop gibt es jetzt auch die selbstdefinierte Funktion daueralarm. Diese wird zunächst definiert und dann in loop wie ein ganz normaler Befehl aufgerufen. Wichtig: Zwischen den Klammern steht zwar nichts, man muss sie aber trotzdem

Arduino mega 2560 - Eine Anleitung für Schüler

hinschreiben.

Aufgaben

1. Weitere Zerlegung

Wie weit man Programme in einzelne Funktionen zerlegt, ist ein wenig Geschmackssache. Überlege Dir, wie man das Beispielprogramm noch weiter sinnvoll in Einzelteile zerlegen kann. Überlege Dir auch sinnvolle Namen für die entsprechenden Funktionen.

2. Funktionen mit Parameterübergabe

Erkundige Dich im Internet, wie man in C/C++ Werte an Funktionen übergeben kann.

Entwirf dann eine Funktion

`dimmen(int von_wert, int bis_wert, int delay_in_ms)`

und verwende sie sinnvoll in Deinem Programm.

3. Funktionen mit Rückgabewerten

Erkundige Dich im Internet, wie man in C/C++ Funktionen mit Rückgabewerten schreibt.

Überlege Dir, wie man dies in Deinem Programm sinnvoll einsetzen kann.

3.10 Beleuchtung? - Aber nur im Dunkeln!

Wenn unsere Hausbeleuchtungs-Alarmanlage nur auf Beleuchtung gestellt ist, macht es natürlich keinen Sinn, wenn sie auch am hellen Tag leuchtet, wenn sich jemand vor dem Bewegungsmelder bewegt.

Um zu entscheiden, ob es hell oder dunkel ist, brauchen wir einen Fotowiderstand (engl.: Light Dependent Resistor, LDR).

Wie schon der Schalter, wird ein Pol des LDR an 5V gelegt, der andere über einen Widerstand an GND. Dieser Pol des LDR wird dann mit einem der Analogeingänge verbunden, in diesem Fall mit A5.

Das zugehörige Programm entscheidet nun, dass die Beleuchtungsfunktion erst aktiviert wird, wenn der Helligkeitswert eine bestimmte Grenze unterschreitet.

```
// =====  
// PROGRAMM: 10_Funktion  
// =====  
int ledPin = 5;  
int pirPin = 6;  
int potiPin = 1;  
int tasterPin=4;  
int ldrPin=5;  
int hell = 255;
```

Arduino mega 2560 - Eine Anleitung für Schüler

```
int gedimmt = 32;
int dunkel = 0;
int leuchtdauer;
int zaehler;
bool alarmAn;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(pirPin, INPUT);
  pinMode(tasterPin, INPUT);
}

void daueralarm(){
  alarmAn=true;
  while (alarmAn){
    analogWrite(ledPin,hell);
    delay(50);
    analogWrite(ledPin,dunkel);
    delay(50);
    if (digitalRead(tasterPin)==HIGH){
      alarmAn=false;
    }
  }
}

void loop() {
  leuchtdauer = analogRead(potiPin)*10;
  if (digitalRead(pirPin)==HIGH){
    if (leuchtdauer>9000) {
      daueralarm();
    }
    if (analogRead(ldrPin)<90){
      analogWrite(ledPin,hell);
    }
    delay(leuchtdauer);
  }
  else {
    analogWrite(ledPin,dunkel);
    delay(100);
  }
}
```

Aufgaben

1. Einstellbare Helligkeitsschwelle

In obigem Programm ist der aus dem ldrPin ausgelesene Wert für die Helligkeit, ab

Arduino mega 2560 - Eine Anleitung für Schüler

dem die Lampen ggf. angeschaltet werden sollen, mit 90 fest vorgegeben.

Ändere das Programm so ab, dass die Einschalt-Schwelle über einen weiteren Drehwiderstand fest eingestellt werden kann.

2. Zusätzlicher Lichtschalter

Manchmal möchte man auch dann Licht im Garten haben, wenn sich draußen nichts bewegt.

Mit einem zusätzlichen Taster wollen die Bewohner das Licht im Garten einfach ein- und abschalten können, ganz wie in der guten alten Zeit oder Hausautomation.

3. Weitere Optimierungen

Überlege Dir weitere Optimierungsmöglichkeiten (sowohl in Sachen Hardware als auch in Sachen Software) für das hier vorgestellte Heimautomations-Projekt.

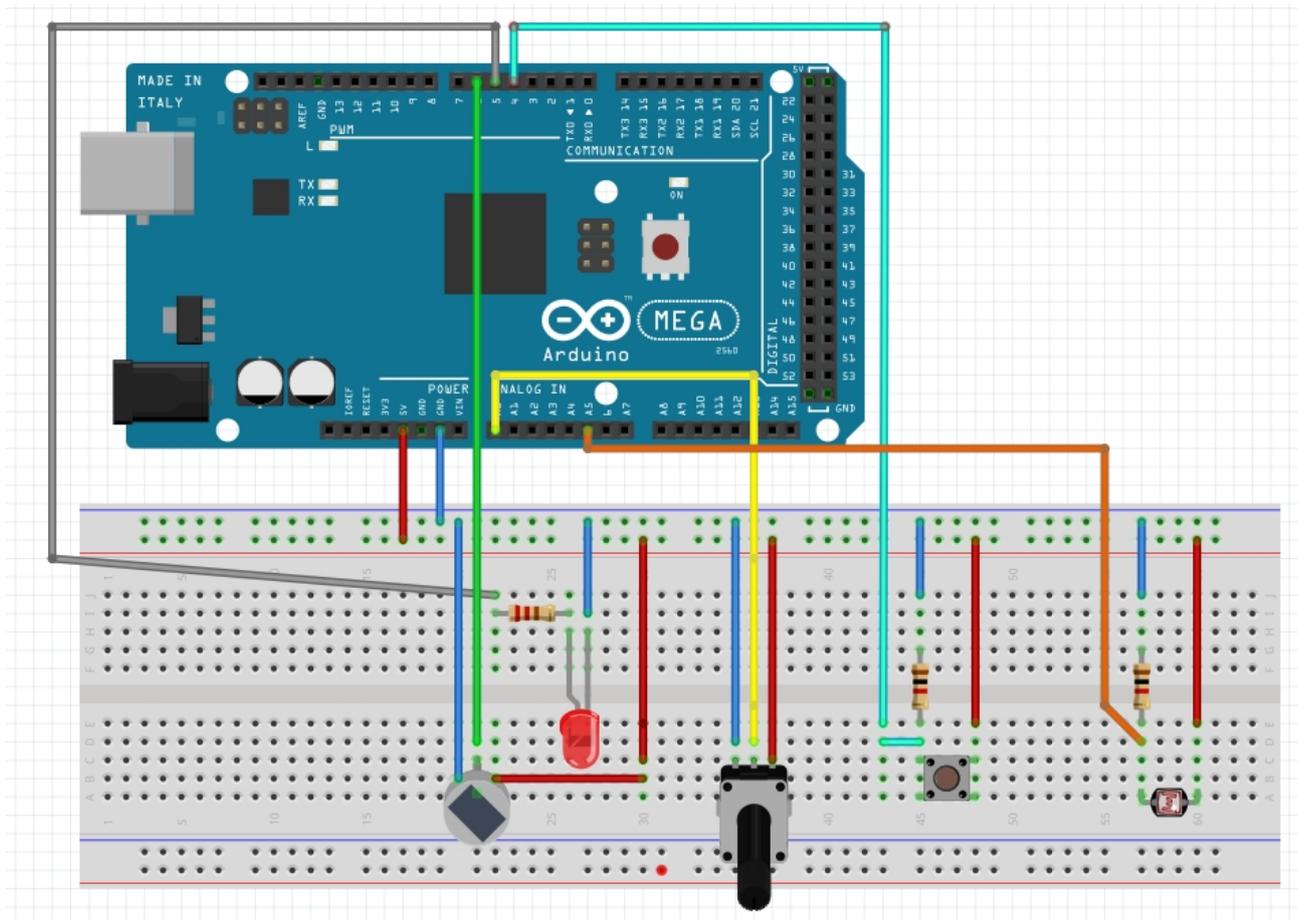
Dokumentiere Deine Ideen auch in Form eines kurzen Textes.

Arduino mega 2560 - Eine Anleitung für Schüler

4 Anhang

4.1 Versuchsaufbau

Die Grafik zeigt den Aufbau der kompletten Versuchsschaltung mit dem Arduino mega, ist aber unter Einbehaltung der Pinbelegung mit jedem anderen Mikrocontroller nutzbar.



Arduino mega 2560 - Eine Anleitung für Schüler

4.2 Farbcodes von Widerständen

Widerstände sind häufig mit farbigen Streifen codiert. Die Bedeutung der Streifen hängt davon ab, ob sich 4, 5 oder mehr Streifen auf dem Widerstand befinden.

Eine Übersicht bieten die folgenden beiden Tabellen:

Widerstände mit 4 Ringen

Farbe		Widerstandswert in Ω			Toleranz
		1. Ring (Zehner)	2. Ring (Einer)	3. Ring (Multiplikator)	4. Ring
„keine“	X	—	—	—	$\pm 20\%$
silber		—	—	$10^{-2} = 0,01$	$\pm 10\%$
gold		—	—	$10^{-1} = 0,1$	$\pm 5\%$
schwarz		—	0	$10^0 = 1$	—
braun		1	1	$10^1 = 10$	$\pm 1\%$
rot		2	2	$10^2 = 100$	$\pm 2\%$
orange		3	3	$10^3 = 1.000$	—
gelb		4	4	$10^4 = 10.000$	—
grün		5	5	$10^5 = 100.000$	$\pm 0,5\%$
blau		6	6	$10^6 = 1.000.000$	$\pm 0,25\%$
violett		7	7	$10^7 = 10.000.000$	$\pm 0,1\%$
grau		8	8	$10^8 = 100.000.000$	$\pm 0,05\%$
weiß		9	9	$10^9 = 1.000.000.000$	—

Arduino mega 2560 - Eine Anleitung für Schüler

Widerstände mit 5 oder 6 Ringen

Farbe	1. Ring (Hunderter)	2. Ring (Zehner)	3. Ring (Einer)	4. Ring (Multiplikator)	5. Ring (Toleranz)	6. Ring (Temp.-Koeffizient)
silber				10^{-2}		
gold				10^{-1}		
schwarz		0	0	10^0		$200 \cdot 10^{-6} \text{ K}^{-1}$
braun	1	1	1	10^1	$\pm 1 \%$	$100 \cdot 10^{-6} \text{ K}^{-1}$
rot	2	2	2	10^2	$\pm 2 \%$	$50 \cdot 10^{-6} \text{ K}^{-1}$
orange	3	3	3	10^3		$15 \cdot 10^{-6} \text{ K}^{-1}$
gelb	4	4	4	10^4		$25 \cdot 10^{-6} \text{ K}^{-1}$
grün	5	5	5	10^5	$\pm 0,5 \%$	
blau	6	6	6	10^6	$\pm 0,25 \%$	$10 \cdot 10^{-6} \text{ K}^{-1}$
violett	7	7	7		$\pm 0,1 \%$	$5 \cdot 10^{-6} \text{ K}^{-1}$
grau	8	8	8		$\pm 0,05 \%$	
weiß	9	9	9			