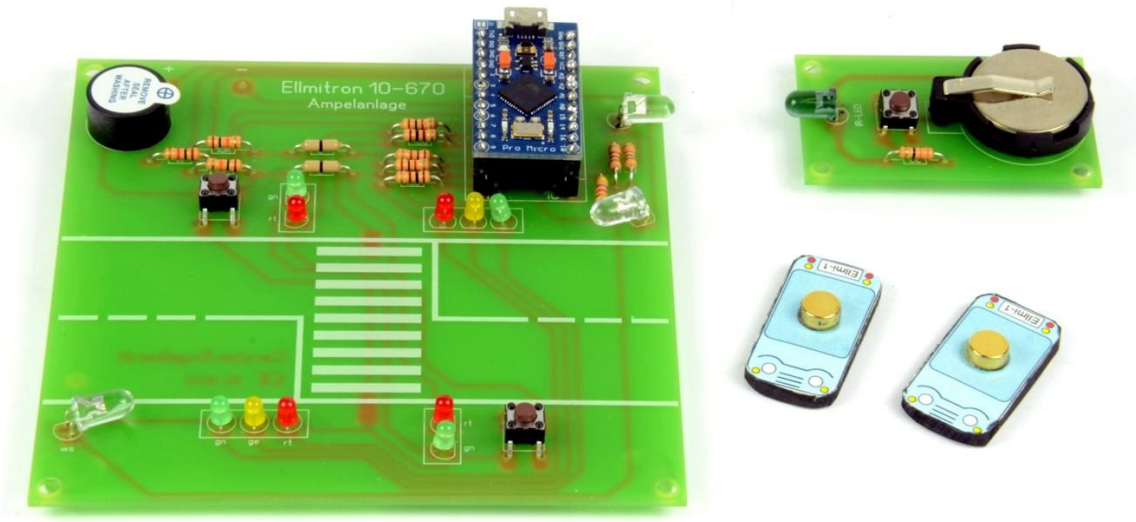
Ampelanlage für Pro-Micro

auch geeignet für Arduino-UNO oder BASIC-Stamp

Ellmitron-Best.Nr.: 10-670 und 10-670-FB



Thema

Es wird eine Ampelanlage hergestellt, die je nach Programmierung mit der Arduino-Software oder Basic-Stamp-Software verschiedene Funktionen erfüllen kann. Der Pro-Micro kann dabei direkt auf die Trägerplatine gesteckt werden, die alle weiteren nötigen Bauteile enthält. Eine lose Verdrahtung entfällt.
Die Verwendung eines Arduino-Uno oder einer BASIC-Stamp ist ebenso möglich, erfordert aber die Verdrahtung mit Verbindungsleitungen (Stecker-Stecker).
In dieser Anleitung werden der Aufbau, die Möglichkeiten und grundlegende Programmierschritte sowie ein paar "Programm-Goodies" besprochen. Der Betrieb erfolgt über USB oder 9-Volt.

Möglichkeiten

Ansteuerung von je zwei LEDs der beiden Fußgängerampeln
Ansteuerung von je drei LEDs der beiden Fahrzeugampeln
Zwei Tasten an den Fußgängerampeln können den Ampelzyklus beeinflussen.
Ein Reedkontakt unter der Kreuzung kann das Überfahren der Kreuzung registrieren. (Hierfür wird ein Mini-Auto mit Magnet mitgeliefert)
Ein Blitzer (2 weiße LEDs) können angesteuert werden, wenn das Auto bei Rot über die Kreuzung fährt. Für das Auto bitte die Vorlage ausschneiden und an der Markierung einen Magnet aufkleben. Es liegt Material für 2 Autos bei.

Mit dem Ergänzungspack 10-670-FB kommen folgende Möglichkeiten hinzu:

Eine einfache Infrarot-Fernbedienung kann die Steuerung beeinflussen (nicht bei Verwendung einer BASIC-Stamp) und weitere, über die Steuerung der Ampel hinausgehende Funktionen ermöglichen (Siehe Testprogramme 6++). Hierbei wird auch ein Summer verwendet und das Morsen als Form drahtloser Kommunikation angesprochen. Interessante und lustige Programme sind hier möglich und bieten genügend Freiraum für Kreativität.

Arbeitsschritte

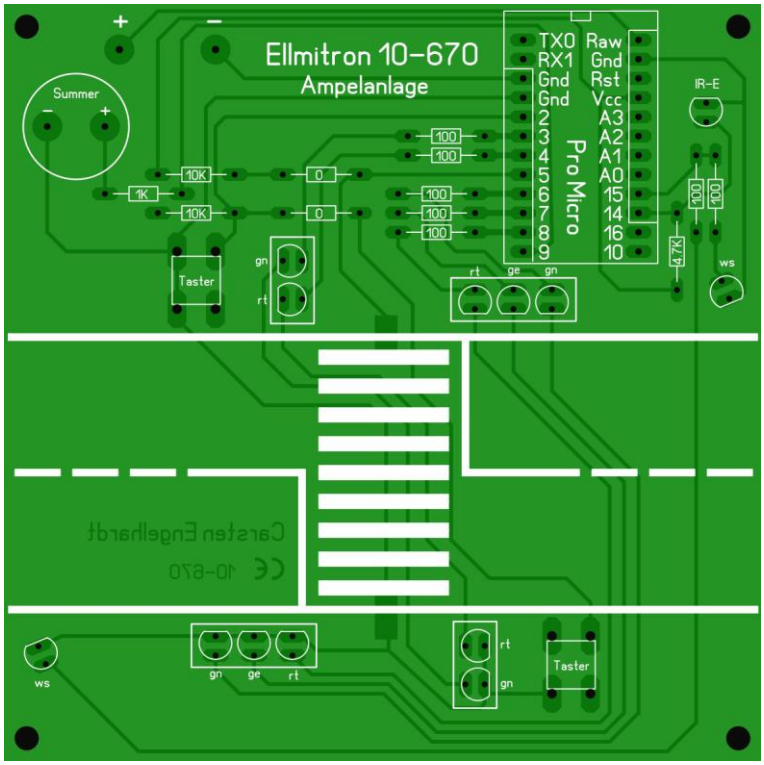
Folgende Arbeiten sind zur Herstellung und zum Betrieb der kompletten Ampelanlage nötig und werden in dieser Anleitung beschrieben (alternative Hinweise sind *kursiv* gedruckt):

- Aufbau und Verlöten der Platine (In der Fertigversion nicht nötig)
- Funktionsprüfung der Platine mit aufgestecktem Pro-Micro oder angeschlossenem Arduino-Uno bzw. BASIC-Stamp
- Programmierung des Pro-Micro, Arduino oder der BASIC-Stamp
  (In dieser Anleitung wird auf die BASIC-Stamp nicht weiter eingegangen

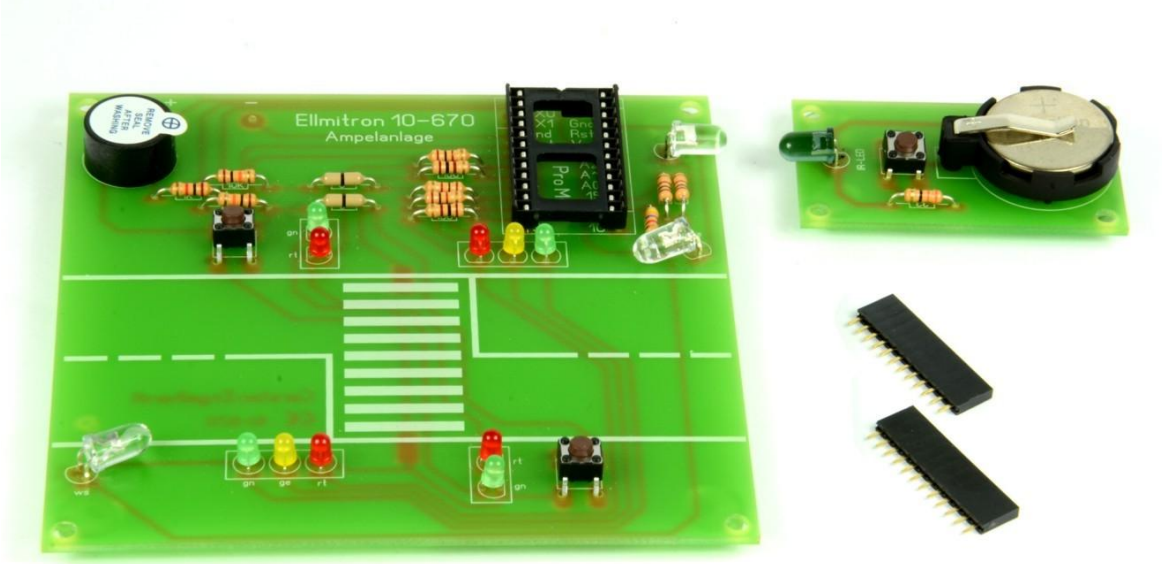
Aufbau und Verlöten der Platine

Widerstände und Taster einlöten, dann die LEDs. Die LEDs der vier Ampeln direkt in die Platine löten. Beachten Sie die LED-Farbe und die Polung. Die abgeflachte Seite im Bestückungsdruck entspricht dem kurzen Anschluss der LED (-). Beim IR-Empfänger ist dies umgekehrt (langer Anschluss = abgeflachte Seite) Die weißen LEDs, der IR-Empfänger und die IR-Sende-Diode werden zuvor abgewinkelt und horizontal wie auf der Abb. eingelötet. Wenn später nur ein Pro-Micro verwendet werden soll, wird der IC-Sockel 24pol. eingelötet. Soll ein Arduino oder eine BASIC-Stamp verwendet werden, bieten sich Buchsenleisten *(43-1010)* an, da man dann die Anschlussbezeichnungen gut lesen kann und die Kontaktierung mit den Verbindungsleitungen *(24-212 Stecker-Stecker) optimal ist*.

Die Bestückung der weißen LEDs und des IR-Empfängers bzw. der IR-Sendediode auf der Sendeplatine kann auch etwas erhöht erfolgen und die LEDs können in ein kleines Gehäuse eingebaut werden (Fieser Blitzer). In diesem Fall wurde die LED in einen geschwärzten Holzstab 8X8mm eingebaut.



Fertig bestückte Platinen (Ampelanlage und IR-Sender)



So sollte die fertige Platine Ampelsteuerung und die Platine IR-Sender aussehen. (Abb. incl. Erweiterungen)
Hier wurde ein IC-Sockel für den Pro-Micro eingesetzt. Für Arduino-UNO oder BASIC-Stamp bitte die Buchsenleisten verwenden.

Funktionsprüfung

Vor der Funktionsprüfung der selbst verlöteten Platine schauen Sie bitte noch einmal, ob es auf der Platine kalte Lötstellen oder ungewünschte Kurzschlüsse gibt (evtl. Lupe verwenden).
Der Pro-Micro ist nach dem Aufstecken mit allen Ampelkomponenten verbunden und Sie können die Testprogramme ausprobieren. Wenn der Pro-Micro mit dem Computer verbunden ist, öffnen Sie die Arduino-Oberfläche und achten Sie darauf, dass unter "Werkzeuge_Board" der Arduino-Leonardo ausgewählt ist.
Anschließend schauen Sie, dass unter "Werkzeuge_Port" derjenige Port ausgewählt ist, der den Arduino-Leonardo erkannt hat.

Den Arduino-Uno verbinden Sie mit Verbindungsleitungen (optimal sind hier die Busleitungen 24-212 Stecker-Stecker) wie folgt:

Alle 3 Gnd-Anschlüsse der Platine mit je einem Gnd-Anschluss des Arduino-Uno,
Vcc mit 5V, Pin 2 bis 9 mit Pin 2 bis 9, Pin 14 und 15 der Platine mit Pin 12 und 13 des Arduino-Uno. (*In den Test-Programmen müssen die Pin-Nummern entsprechend geändert werden. 14 wird zu 12, 15 wird zu 13*)

Wenn der Arduino-Uno mit dem Computer verbunden ist, öffnen Sie die Arduino-Oberfläche und achten Sie darauf, dass unter "Werkzeuge_Board" der Arduino-Uno ausgewählt ist.
Anschließend schauen Sie, dass unter "Werkzeuge_Port" derjenige Port ausgewählt ist, der den Arduino-Uno erkannt hat.

Die BASIC-Stamp verbinden Sie mit Verbindungsleitungen (optimal sind hier die Busleitungen 24-212 Stecker-Stecker). Bitte beachten Sie, dass nur entweder die Erweiterung IR-Fernbedienung oder der Blitzer verwendet werden kann, da die BASIC-Stamp nur über 8 Ein-/Ausgänge verfügt.

Programmieren der Ampelanlage mit dem Arduino

Für den Einstieg und die grundlegende Programmierung des Arduino gibt es gute Literatur, z.B. Arduino-Kompendium Best.Nr. 05-032, Arduino-Handbuch für Einsteiger Best.Nr.05-030 oder Roboter bauen mit Arduino Best.Nr.05-031, sowie die umfangreiche Referenz, die man in der Hilfe der Programmieroberfläche aufrufen kann und die in den allermeisten Fällen die schnellste und beste Hilfe bietet.

Testprogramme

Die Testprogramme können per Copy-Paste in die Arduino-Oberfläche kopiert und dort auf den Arduino übertragen werden. Programme für die BASIC-Stamp stehen momentan nicht zur Verfügung, werden aber evtl. künftig auf unserer Website zum Download bereitstehen oder in diese Anleitung aufgenommen.

Alle Testprogramme können beliebig abgeändert werden. Nach dem Hochladen sind die Änderungen sofort sichtbar.

Hier sollen nur die Programmelemente angesprochen werden, die sich auf die Ampelsteuerung beziehen. Zunächst ist es wichtig, im Einzelnen zu wissen, welches Bauteil mit welchem Pin der Buchsenleiste auf der Platine verbunden ist und welche Möglichkeiten es hat.

- Fußgängerampel, grün (D-Pin 3)
  Beide Fußgängerampeln sind parallel geschaltet. Funktion: Ein/Aus
- Fußgängerampel, rot (D-Pin 4)
  Beide Fußgängerampeln sind parallel geschaltet. Funktion: Ein/Aus
- Fahrzeugampel, grün (D-Pin 6)
  Beide Fahrzeugampeln sind parallel geschaltet. Funktion: Ein/Aus
- Fahrzeugampel, gelb (D-Pin 7)
  Beide Fahrzeugampeln sind parallel geschaltet. Funktion: Ein/Aus
- Fahrzeugampel, rot (D-Pin 8)
  Beide Fahrzeugampeln sind parallel geschaltet. Funktion: Ein/Aus
- Taster Fußgänger, (D-Pin 2),
  Beide Taster für Fußgänger sind parallel geschaltet. Funktion: Ein/Aus
- Reedkontakt, (D-Pin 5), nur im Ergänzungspack Blitzer (10-670-BL) enthalten.
  Funktion: Ein/Aus (wird ausgelöst, wenn das M-Fahrzeug über die Kreuzung fährt)
- Blitzer, weiß (Pin D-15), nur im Ergänzungspack Blitzer (10-670-BL) enthalten.
  Beide Blitzer sind parallel geschaltet. Funktion: Ein/Aus
- Summer (Pin D-9), nur im Ergänzungspack Fernbedienung (10-670-FB) enthalten
  Der Summer kann für die Signalbestätigung der IR-Fernbedienung oder für Ampelsignale (Z.B. für Blinde) verwendet werden.
  Funktion: Ein/Aus
  Der Summer wird über einen Vorwiderstand betrieben und liefert ein leises Signal. Wer es laut möchte, kann den Vorwiderstand durch eine Drahtbrücke ersetzen.
- IR-Empfangstransistor (Pin D-14), nur im Ergänzungspack Fernbedienung (10-670-FB) enthalten
  Funktion: Ein/Aus

Nachfolgend finden Sie kleine Programmbeispiele für die oben genannten Funktionen. Rot ist der eigentliche Programmiertext, Blau die Erklärung, der Rest ist vorgegeben. Die Beschreibung ist kursiv.

1. LED ein / aus / Gelbes Blinklicht

Im folgenden Programm wird die gelbe Fahrzeugampel (D-Pin 7) programmiert. Sie wird mit dem Befehl digitalWrite(x,y) angesteuert. x ist dabei die Pin-Nummer (hier 7) und y ist der gewünschte Ausgabewert. 0=aus, 1=ein. In diesem Programm wird die LED für 0,5 Sekunden ein- und wieder ausgeschaltet. Da das Programm ständig wiederholt wird, entsteht ein Blinklicht, wie es z.B. verwendet wird, wenn keine zyklische Ampelsteuerung nötig ist.

Versuchen Sie auch andere Pins anzusteuern (welcher Pin welche LED ansteuert, finden Sie oben) oder die Blinkgeschwindigkeit zu ändern.

```
void setup() {
  // put your setup code here, to run once:
pinMode(7,OUTPUT); // Macht Pin 7 zu einem Output
}

void loop() {
  // put your main code here, to run repeatedly:
digitalWrite(7,1); // Schaltet Pin 7 ein
delay(500); // Pause für 0,5 Sekunden (500msec)
digitalWrite(7,0); // Schaltet Pin 7 aus
delay(500); // Pause für 0,5 Sekunden (500msec)
}
```

2. Ampelsteuerung mit ständiger Wiederholung

Im folgenden Programm wird die ganze Ampelanlage gesteuert. Das Programm wiederholt sich ständig und entspricht einer Ampelsteuerung, bei der keine Fußgängertaste nötig ist. Der Ablauf kann frei verändert werden.

```
void setup() {
pinMode(3,OUTPUT); // Macht Pin 3 zu einem Output F grün
pinMode(4,OUTPUT); // Macht Pin 4 zu einem Output F rot
pinMode(6,OUTPUT); // Macht Pin 6 zu einem Output A grün
pinMode(7,OUTPUT); // Macht Pin 7 zu einem Output A gelb
pinMode(8,OUTPUT); // Macht Pin 8 zu einem Output A rot
}
void loop() {
digitalWrite(4,1); digitalWrite(6,1); // Schaltet Pin 4 + 6 ein
delay(4000); // Pause für 4 Sekunden (4000msec)
digitalWrite(6,0); digitalWrite(7,1); // Schaltet Pin 6 aus + Pin 7 ein
delay(1000); // Pause für 1 Sekunde (1000msec)
digitalWrite(7,0); digitalWrite(8,1); // Schaltet Pin 7 aus + Pin 8 ein
delay(2000); // Pause für 2 Sekunden (2000msec)
digitalWrite(3,1); digitalWrite(4,0); // Schaltet Pin 3 ein + Pin 4 aus
delay(4000); // Pause für 4 Sekunden (4000msec)
digitalWrite(3,0); digitalWrite(4,1); // Schaltet Pin 3 aus + Pin 4 ein
delay(2000); // Pause für 2 Sekunden (2000msec)
digitalWrite(7,1); // Schaltet Pin 7 ein
delay(1000); // Pause für 1 Sekunde (1000msec)
digitalWrite(7,0); digitalWrite(8,0); // Schaltet Pin 7 aus + Pin 8 aus
}
```

3. Ampelsteuerung mit Fußgängertaste

Im folgenden Programm wird die Fußgängertaste verwendet um den Ampelzyklus zu starten. Zu Beginn ist die Fahrzeugampel grün und die Fußgängerampel rot. Erst durch Drücken einer der beiden Taster startet der Zyklus. Der Ablauf kann frei verändert werden.

```
void setup() {
pinMode(3,OUTPUT); // Macht Pin 3 zu einem Output F grün
pinMode(4,OUTPUT); // Macht Pin 4 zu einem Output F rot
pinMode(6,OUTPUT); // Macht Pin 6 zu einem Output A grün
pinMode(7,OUTPUT); // Macht Pin 7 zu einem Output A gelb
pinMode(8,OUTPUT); // Macht Pin 8 zu einem Output A rot
pinMode(2,INPUT); // Macht Pin 2 zu einem Input Fußgängertaste
}

void loop() {
if (digitalRead(2)==LOW) { // Tastenabfrage
delay(3000); // Pause für 3 Sekunden (3000msec)
digitalWrite(6,0); digitalWrite(7,1); // Schaltet Pin 6 aus + Pin 7 ein
delay(1000); // Pause für 1 Sekunde (1000msec)
digitalWrite(7,0); digitalWrite(8,1); // Schaltet Pin 7 aus + Pin 8 ein
delay(2000); // Pause für 2 Sekunden (2000msec)
digitalWrite(3,1); digitalWrite(4,0); // Schaltet Pin 3 ein + Pin 4 aus
delay(4000); // Pause für 4 Sekunden (4000msec)
digitalWrite(3,0); digitalWrite(4,1); // Schaltet Pin 3 aus + Pin 4 ein
delay(2000); // Pause für 2 Sekunden (2000msec)
digitalWrite(7,1); // Schaltet Pin 7 ein
delay(1000); // Pause für 1 Sekunde (1000msec)
digitalWrite(7,0); digitalWrite(8,0); // Schaltet Pin 7 aus + Pin 8 aus
}
digitalWrite(4,1); digitalWrite(6,1); // Schaltet Pin 4 ein + Pin 6 ein
delay(20); // Pause für 0,02 Sekunde (20msec)
}
```

4. Ampelsteuerung mit Blitzer

Im folgenden Programm wird die Fahrzeugampel überwacht. Ein Reedkontakt unter der Straße schließt, wenn das Magnetauto darüber fährt. Geschieht dies, wenn die Fahrzeugampel rot ist, werden die beiden weißen LEDs angesteuert (Blitzer).
Der Ablauf kann frei verändert werden.

```
void setup() {
pinMode(3,OUTPUT); // Macht Pin 3 zu einem Output F grün
pinMode(4,OUTPUT); // Macht Pin 4 zu einem Output F rot
pinMode(6,OUTPUT); // Macht Pin 6 zu einem Output A grün
pinMode(7,OUTPUT); // Macht Pin 7 zu einem Output A gelb
pinMode(8,OUTPUT); // Macht Pin 8 zu einem Output A rot
pinMode(15,OUTPUT); // Macht Pin 15 zu einem Output Blitzer
pinMode(2,INPUT); // Macht Pin 2 zu einem Input Fußgängertaste
pinMode(5,INPUT); // Macht Pin 5 zu einem Input Reedkontakt
}

void loop() {
if (digitalRead(2)==LOW) { // Abfrage Taster
delay(3000); // Pause für 3 Sekunden (3000msec)
digitalWrite(6,0); digitalWrite(7,1); // Schaltet Pin 6 aus + Pin 7 ein
delay(1000); // Pause für 1 Sekunde (1000msec)
digitalWrite(7,0); digitalWrite(8,1); // Schaltet Pin 7 aus + Pin 8 ein
        for (int i=0; i<=100; i++){  // Zählschleife (Erklärung siehe unten)
                if (digitalRead(5)==LOW) { // Abfrage Reedkontakt
                digitalWrite(15,1); // Schaltet Pin 15 ein
                delay(50); // Pause für 0,05 Sekunden (50msec)
                digitalWrite(15,0); // Schaltet Pin 15 aus
```

```
                delay(300); // Pause für 0,3 Sekunden (300msec)
                digitalWrite(15,1); // Schaltet Pin 15 ein
                delay(50); // Pause für 0,05 Sekunden (50msec)
                digitalWrite(15,0); // Schaltet Pin 15 aus
                }
        delay(20); // Pause für 0,02 Sekunden (20msec)
        }
digitalWrite(3,1); digitalWrite(4,0); // Schaltet Pin 3 ein + Pin 4 aus

        for (int i=0; i<=200; i++){  // Zählschleife (Erklärung siehe unten)
                if (digitalRead(5)==LOW) { // Abfrage Reedkontakt
                digitalWrite(15,1); // Schaltet Pin 15 ein
                delay(50); // Pause für 0,05 Sekunden (50msec)
                digitalWrite(15,0); // Schaltet Pin 15 aus
                delay(300); // Pause für 0,3 Sekunden (300msec)
                digitalWrite(15,1); // Schaltet Pin 15 ein
                delay(50); // Pause für 0,05 Sekunden (50msec)
                digitalWrite(15,0); // Schaltet Pin 15 aus
                }
        delay(20); // Pause für 0,02 Sekunden (20msec)
        }
digitalWrite(3,0); digitalWrite(4,1); // Schaltet Pin 3 aus + Pin 4 ein
        for (int i=0; i<=100; i++){  // Zählschleife (Erklärung siehe unten)
                if (digitalRead(5)==LOW) { // Abfrage Reedkontakt
                digitalWrite(15,1); // Schaltet Pin 15 ein
                delay(50); // Pause für 0,05 Sekunden (50msec)
                digitalWrite(15,0); // Schaltet Pin 15 aus
                delay(300); // Pause für 0,3 Sekunden (300msec)
                digitalWrite(15,1); // Schaltet Pin 15 ein
                delay(50); // Pause für 0,05 Sekunden (50msec)
                digitalWrite(15,0); // Schaltet Pin 15 aus
                }
        delay(20); // Pause für 0,02 Sekunden (20msec)
        }
digitalWrite(7,1); // Schaltet Pin 7 ein
delay(1000); // Pause für 1 Sekunde (1000msec)
digitalWrite(7,0); digitalWrite(8,0); // Schaltet Pin 7 aus + Pin 8 aus
}
digitalWrite(4,1); digitalWrite(6,1); // Schaltet Pin 4 ein + Pin 6 ein
delay(20); // Pause für 0,02 Sekunde (20msec)
}
```

In Programm 3 werden für die einzelnen Ampelphasen Delays (Pausen) von 1 Sekunde (1000msec) bis 4 Sekunden (4000msec) verwendet. In diesem Zeitraum ist keine Abfrage von Eingängen möglich. Daher wird in Programm 4 (nur, wenn die Fahrzeugampel rot zeigt) mit Zählschleifen gearbeitet, d.h. anstelle einer Pause von 2 Sekunden wird der Reedkontakt 100 mal in Abständen von 20msec abgefragt (=2 Sekunden).

5. Ampel mit Fernbedienung *(mit Ergänzung 10-670-FB)*

Im folgenden Programm soll die einfache Infrarot-Fernbedienung verwendet werden um z.B. die Ampel von einem in den anderen Modus zu schalten. Hier von Programm 2 auf Programm 1. Das Umschalten wird durch den Summer angezeigt. Die Umstellung ist nur möglich, wenn die Fahrzeugampel grün zeigt! Die Variable I wird verwendet, damit das Program weiß, in welchem Modus es sich befindet.
Wie müsste das Programm verändert werden, damit die Umschaltung jederzeit erfolgen kann?

```
void setup() {
```

```
pinMode(3,OUTPUT); // Macht Pin 3 zu einem Output F grün
pinMode(4,OUTPUT); // Macht Pin 4 zu einem Output F rot
pinMode(6,OUTPUT); // Macht Pin 6 zu einem Output A grün
pinMode(7,OUTPUT); // Macht Pin 7 zu einem Output A gelb
pinMode(8,OUTPUT); // Macht Pin 8 zu einem Output A rot
pinMode(9,OUTPUT); // Macht Pin 8 zu einem Output Summer
pinMode(14,INPUT); // Macht Pin 14 zu einem Input IR-Fernbedienung
}
void loop() {
int var=0; // // var=0 -> Normalbetrieb
digitalWrite(4,1); digitalWrite(6,1); // Schaltet Pin 4 + Pin 6 ein
Normalbetrieb: // Beginn Normalbetrieb
while(digitalRead(14)==HIGH && var==0){ // Schleife
for(int i=0; i<100; i++){ // Schleife 100X
        if (digitalRead(14)==LOW && var==0){ // Überprüft FB-Signal
        goto Blinkbetrieb; // Sprung in den Blinkbetrieb
        }
        delay(20); // Pause für 0,02 Sekunden (20msec)
}
digitalWrite(6,0); digitalWrite(7,1); // Schaltet Pin 6 aus + Pin 7 ein
delay(1000); // Pause für 1 Sekunde (1000msec)
digitalWrite(7,0); digitalWrite(8,1); // Schaltet Pin 7 aus + Pin 8 ein
delay(2000); // Pause für 2 Sekunden (2000msec)
digitalWrite(3,1); digitalWrite(4,0); // Schaltet Pin 3 ein + Pin 4 aus
delay(4000); // Pause für 4 Sekunden (4000msec)
digitalWrite(3,0); digitalWrite(4,1); // Schaltet Pin 3 aus + Pin 4 ein
delay(2000); // Pause für 2 Sekunden (2000msec)
digitalWrite(7,1); // Schaltet Pin 7 ein
delay(1000); // Pause für 1 Sekunde (1000msec)
digitalWrite(7,0); digitalWrite(8,0); // Schaltet Pin 7 + 8 aus
digitalWrite(6,1); // Schaltet Pin 6 ein
}
Blinkbetrieb: // Beginn Blinkbetrieb
var=1; // var=1 -> Blinkbetrieb
digitalWrite(9,1); // Schaltet Pin 9 ein Summer
delay(50); // Pause für 0,05 Sekunde (50msec)
digitalWrite(9,0); // Schaltet Pin 9 aus Summer
delay(1000); // Pause für 1 Sekunde (1000msec)
while(digitalRead(14)==HIGH && var==1){ // Schleife
        digitalWrite(4,0); digitalWrite(6,0); // Schaltet Pin 4 + 6 aus
        digitalWrite(7,1);// Schaltet Pin 7 ein
        delay(500); // Pause für 0,5 Sekunden (500msec)
        digitalWrite(7,0);// Schaltet Pin 7 aus
        delay(500); // Pause für 0,5 Sekunden (500msec)
                if (digitalRead(14)==LOW) { // Überprüft FB-Signal
                digitalWrite(9,1); // Schaltet Pin 9 ein
                delay(50); // Pause für 0,05 Sekunden (50msec)
                digitalWrite(9,0); // Schaltet Pin 9 aus
                delay(200); // Pause für 0,2 Sekunden (200msec)
                digitalWrite(9,1); // Schaltet Pin 9 ein
                delay(50); // Pause für 0,05 Sekunden (50msec)
                digitalWrite(9,0); // Schaltet Pin 9 aus
                digitalWrite(4,1); digitalWrite(7,1);
                delay(4000); // Pause für 4 Sekunden (4000msec)
                digitalWrite(6,1); digitalWrite(7,0); // Schaltet Pin 6 ein + Pin 7 aus
                delay(1000); // Pause für 1 Sekunde (1000msec)
```

```
                    var=0; // var=0 -> Normalbetrieb
                    goto Normalbetrieb; // Rücksprung
                    }
            }
}
```

6. Weitere Möglichkeiten mit der Fernbedienung *(mit Ergänzung 10-670-FB)*

Mit der Fernbedienung lassen sich neben der Ampelsteuerung noch weitere Funktionen ausführen. Dazu gehört auch die drahtlose Kommunikation. Ein kleines Programm, das zum drahtlosen Morsen verwendet werden kann finden Sie hier.

```
void setup() {
pinMode(4,OUTPUT); // Macht Pin 4 zu einem Output F rot
pinMode(9,OUTPUT); // Macht Pin 8 zu einem Output Summer
pinMode(14,INPUT); // Macht Pin 14 zu einem Input IR-Fernbedienung
}
void loop() {
if (digitalRead(14)==LOW){ // Überprüft FB-Signal
digitalWrite(9,1); digitalWrite(4,1); // Schaltet Pin 9 + 4 ein
}
else
{
digitalWrite(9,0); digitalWrite(4,0); // Schaltet Pin 9 + 4 aus
}
}
```

7. Morsen mit Anzeige der Zeichen "." und "-" *(mit Ergänzung 10-670-FB)*

Wenn Sie dieses Programm starten und anschließend den Seriellen Monitor öffnen, sehen Sie direkt, welche Signale Sie senden. Kurze Signale werden als Punkt ".", längere Signale als Strich "-" dargestellt. Kürzere Wartezeiten führen zu einem Leerzeichen, längere Wartezeiten zu einem Zeilenvorschub. Die Zeiten können im Programm entsprechend Ihren Wünschen angepasst werden. Versuchen Sie einen Notruf "SOS" = "... --- ..." zu generieren. Das Morsealphabet finden Sie im Anhang. Dieses Programm ist sehr gut zum Üben, bevor man die direkte Ausgabe von Buchstaben, wie im nächsten Programm erzeugt.

```
void setup() {
Serial.begin(9600);
pinMode(4,OUTPUT); // Macht Pin 4 zu einem Output F rot
pinMode(9,OUTPUT); // Macht Pin 8 zu einem Output Summer
pinMode(14,INPUT); // Macht Pin 14 zu einem Input IR-Fernbedienung
}
void loop() {
int beep=0;
int pause=0;
if (digitalRead(14)==LOW){ // Überprüft FB-Signal
digitalWrite(9,1); digitalWrite(4,1); // Schaltet Pin 9 + 4 ein
beep=0;
while(digitalRead(14)==LOW){
beep=beep+1;
delay(10);
}
if (beep<15){
Serial.print(".");
}
else
{
Serial.print("-");
```

```
}
}
else
{
digitalWrite(9,0); digitalWrite(4,0); // Schaltet Pin 9 + 4 aus
while (digitalRead(14)==HIGH){
pause=pause+1;
delay(10);
if (pause==50){
Serial.print(" ");
}
if (pause==120){
Serial.println();
if (pause>120){
exit;
}
}
}
while(digitalRead(14)==HIGH){
delay(10);
}
if (pause>201){
pause=201;
}
}
}
```

8. Morsen mit Anzeige der realen Buchstaben *(mit Ergänzung 10-670-FB)*

Wenn Sie dieses Programm starten und anschließend den Seriellen Monitor öffnen, sehen Sie direkt, welche Buchstaben oder Zeichen Sie erzeugt haben. Während in Programm 7 "SOS" als "... --- ..." dargestellt wird, wird in diesem Programm direkt "SOS" angezeigt. Ein Buchstabe entsteht also aus einem bis mehreren Morsezeichen, die kurz aufeinander folgen. Kürzere Wartezeiten führen zu einem neuen Buchstaben, längere Wartezeiten zu einem Leerzeichen und besonders lange Wartezeiten zu einem Zeilenvorschub. Die Zeiten können im Programm entsprechend Ihren Wünschen angepasst werden.
Versuchen Sie einen Notruf "SOS" = "... --- ..." zu generieren. Das Morsealphabet finden Sie im Anhang.

```
void setup() {
Serial.begin(9600);
pinMode(4,OUTPUT); // Macht Pin 4 zu einem Output F rot
pinMode(9,OUTPUT); // Macht Pin 8 zu einem Output Summer
pinMode(14,INPUT); // Macht Pin 14 zu einem Input IR-Fernbedienung
}
void loop() {
int beep=0;
int pause=0;
String Buchstabe = "";
neu:
if (digitalRead(14)==LOW){ // Überprüft FB-Signal
digitalWrite(9,1); digitalWrite(4,1); // Schaltet Pin 9 + 4 ein
beep=0;
while(digitalRead(14)==LOW){
beep=beep+1;
delay(10);
}
if (beep<15){
```

```
//Serial.print(".");
Buchstabe=Buchstabe+".";
}
else
{
//Serial.print("-");
Buchstabe=Buchstabe+"-";
}
}
else
{
digitalWrite(9,0); digitalWrite(4,0); // Schaltet Pin 9 + 4 aus
pause=0;
while (digitalRead(14)==HIGH){
pause=pause+1;
delay(10);
if (pause==50){
        if (Buchstabe="..."){
        Buchstabe="S";
        }
        if (Buchstabe="---"){
        Buchstabe="O";
        }
        if (Buchstabe="-.-.-"){
        Buchstabe="Hallo, wie geht's?";
        }
// An dieser Stelle die weiteren Buchstaben wie im Alphabet in if-Schleifen eingeben.
Serial.print(Buchstabe);
Buchstabe="";
}
if (pause==120){
Serial.print(" ");
}
if (pause==200){
Serial.println();
}
if (pause>200){
exit;
}
}
while(digitalRead(14)==HIGH){
delay(10);
}
if (pause>201){
pause=201;
}
}
goto neu;
}
```

Obiges Programm können Sie abwandeln, indem Sie anstelle von Buchstaben oder Zeichen ganze Wörter oder Sätze erzeugen lassen. Hier können Sie nach Belieben kreativ sein.
Z.B. könnte "..." gleich als "Hilfe" oder wie im obigen Beispiel "-.-.-" als "Hallo, wie geht's?" etc. angezeigt werden.

Über Programme, die Sie gerne teilen möchten, würde ich mich sehr freuen. Sie helfen damit Kolleginnen und Kollegen, vielfältiger mit Mikrocontrollern zu arbeiten.

Ihr Carsten Engelhardt

Morse-Alphabet

Lateinische Buchstaben

Buchstabe	Code
A	·−
B	−···
C	−·−·
D	−··
E	·
F	··−·
G	−−·
H	····
I	··
J	·−−−
K	−·−
L	·−··
M	−−
N	−·
O	−−−
P	·−−·
Q	−−·−
R	·−·
S	···
T	−
U	··−
V	···−
W	·−−
X	−··−
Y	−·−−
Z	−−··

Ziffern

Ziffer	Code
1	·−−−−
2	··−−−
3	···−−
4	····−
5	·····
6	−····
7	−−···
8	−−−··
9	−−−−·
0	−−−−−

Sonder- und Satzzeichen

Zeichen	Code
À, Â	·−−·−
Ä	·−·−
È	·−··−
É	··−··
Ö *(OE)*	−−−·
Ü	··−−
ß *(SZ)*	···−−··
CH	−−−−
Ñ	−−·−−
. *(AAA)*	·−·−·−
, *(MIM)*	−−··−−
: *(OS)*	−−−···
; *(NNN)*	−·−·−·
? *(IMI)*	··−−··
- *(BA)*	−····−
_ *(UK)*	··−−·−
(*(KN)*	−·−−·
) *(KK)*	−·−−·−
' *(JN)*	·−−−−·
= *(BT)*	−···−
+ *(AR)*	·−·−·
/ *(DN)*	−··−·
@ *(AC)*	·−−·−·

Signale

Zeichen	Code
KA *(Spruchanfang)*	−·−·−
BT *(Pause)*	−···−
AR *(Spruchende)*	·−·−·
VE *(verstanden)*	···−·
SK *(Verkehrsende)*	···−·−
SOS *(internationaler (See-)Notruf)*	···−−−···
HH *(Fehler; Irrung; Wiederholung ab letztem vollständigen Wort)*	········