

## IR Signale generieren

### Ziel

Es sollen IR Signale nach dem RC5 Standard gesendet werden. Dazu wird eine IR Diode mit dem Arduino verbunden. Eine Folge von RC5 Nachrichten wird gesendet, indem die Diode entsprechend gepulst wird.

Zur Abnahme wird Ihnen ein kleines System mit einem IR Empfänger und drei LEDs in Rot, Gelb und Grün zur Verfügung gestellt. Bei Empfang einer RC5 Nachricht mit gewechseltem Toggle-Bit wertet der Empfänger die Nachricht aus: Eine Adresse von 1 bezeichnet rote LED, 2 die gelbe LED und 3 die grüne LED. Das Kommando 0 schaltet die LED aus, das Kommando 1 die LED an. Nach einem Reset sind alle LEDs des Empfängers ausgeschaltet. Schreiben Sie ein Programm, welches die in Abbildung 1 gezeigte Sequenz der LEDs per RC5 Code auslöst.

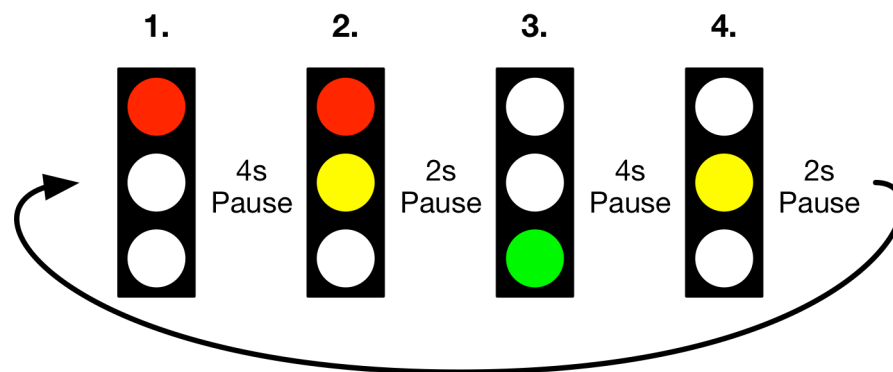


Abbildung 1) Reihenfolge der zu steuernden Lichter

### Vorgehensweise

Die Übung #11 stellt das Gegenstück zur Übung #10 dar. Nun geht es darum, RC5 Codes als 14bit Nachrichten selbst zu generieren, daraus die 28bit Manchester Kodierung zu erzeugen und diese per Software mit 38kHz

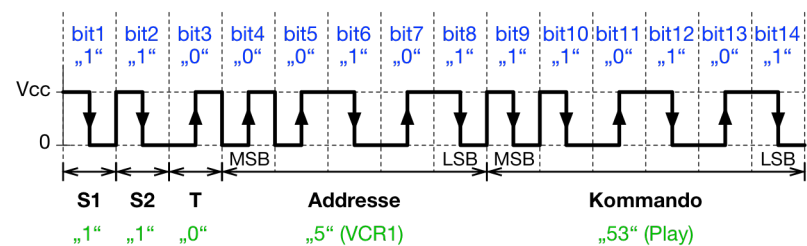


Abbildung 2) Struktur einer RC5 Nachricht

auf der Diode auszugeben (sogenanntes „Bit Banging“). Verwenden Sie zum Anschluss der IR Diode den Pin 9 des Arduino. Schreiben Sie zuerst Code, mit dem Sie die drei benötigten RC5 Nachrichten als 14bit Werte erzeugen können (für ein Beispiel siehe Abbildung 2). Wandeln Sie anschließend die 14bit Nachricht in ein 28bit Manchester-kodiertes Signal um (siehe Abbildung 3). Nun geben Sie das 28bit Signal auf dem Pin 9 aus. Dazu müssen Sie, wie in Abbildung 4 dargestellt, während eines LOW Pegels des Signals eine 38kHz Rechteckschwingung und während eines HIGH Pegels nichts ausgeben.

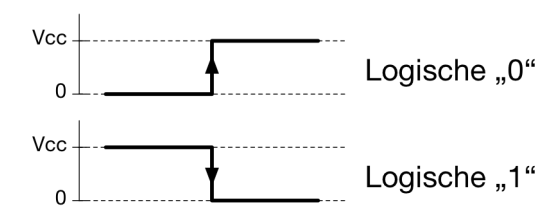


Abbildung 3) Manchester Encoding

Für die Generation der Rechteckschwingung ist das Timing sehr wichtig. Wir verwenden dazu die Hardwareunterstützung zur Pulsweitenmodulation (PWM) des Arduino. Der Pin 9 unterstützt PWM. Wenn Sie auf diesem Pin

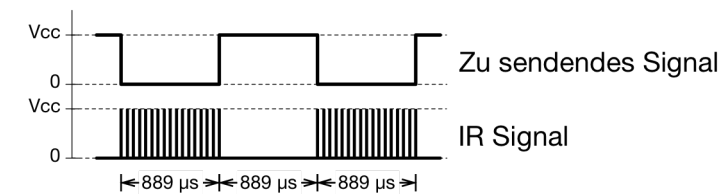


Abbildung 4) Timing des IR Signals

mittels `analogWrite()` den Wert 127 schreiben, erhalten Sie ein Rechtecksignal. Dieses wird normalerweise mit einer Frequenz von etwa 420 Hz generiert. Durch Setzen eines neuen Frequenzteilers kann die PWM Frequenz angepasst werden. Durch die Anweisung

```
TCCR1B &= 0b11111001;
```

direkt nach dem Setzen des `PinMode()` im `setup()` erhalten Sie eine PWM Frequenz von etwa 31kHz<sup>1</sup> – diese ist für das Ampelsystem ok.

### Vorbereitung

Beschäftigen Sie sich mit Manchester Encoding, dem Aufbau des Signals und PWM Signalen. Entwerfen und implementieren Sie einen Algorithmus zur Kodierung des Signals. Entwerfen Sie ein Programm zur Steuerung der Ampel mit korrekter Verzögerung und Reihenfolge der Lichter.

### Achtung

Es stehen nur drei Test-Ampeln zur Verfügung. Bitte beschränken Sie sich in der Zeit, für die Sie das Testsystem brauchen. Hinweis zum Testen der Dioden: IR-Signale sind auf digitalen Kameras ohne IR-Filter als Leuchten erkennbar.

**Es dürfen keine Bibliotheken verwendet werden.**

### Notengebung

4,0 (Anwesend); 3,0 (Die Diode wird angesteuert); 2,3 (+ Das Signal wird mit Fehlern generiert); 2,0 (+ Korrekte Generierung des Signals); 1,7 (+ Ampel läuft richtig); 1,3 (+ Code sauber geschrieben und dokumentiert); 1,0 (+Verwendung von *bit field alignment*, möglichst speichersparende Implementierung)

### Wichtige Funktionen

- `analogWrite(...)`
- `delayMicroseconds(...)`

### Sie brauchen

- Arduino Board, USB Kabel, IR Sender, Steckbrücken, für Test und Abgabe ein „Ampel“-System

<sup>1</sup> Der Frequenzteiler für Timer 1 (verantwortlich für das PWM an Pin9) wird durch die unteren 3 Bit im TCCR1B Register des Atmel ATmega328p eingestellt – solche Details können Sie im Datenblatt des Prozessors nachschauen.