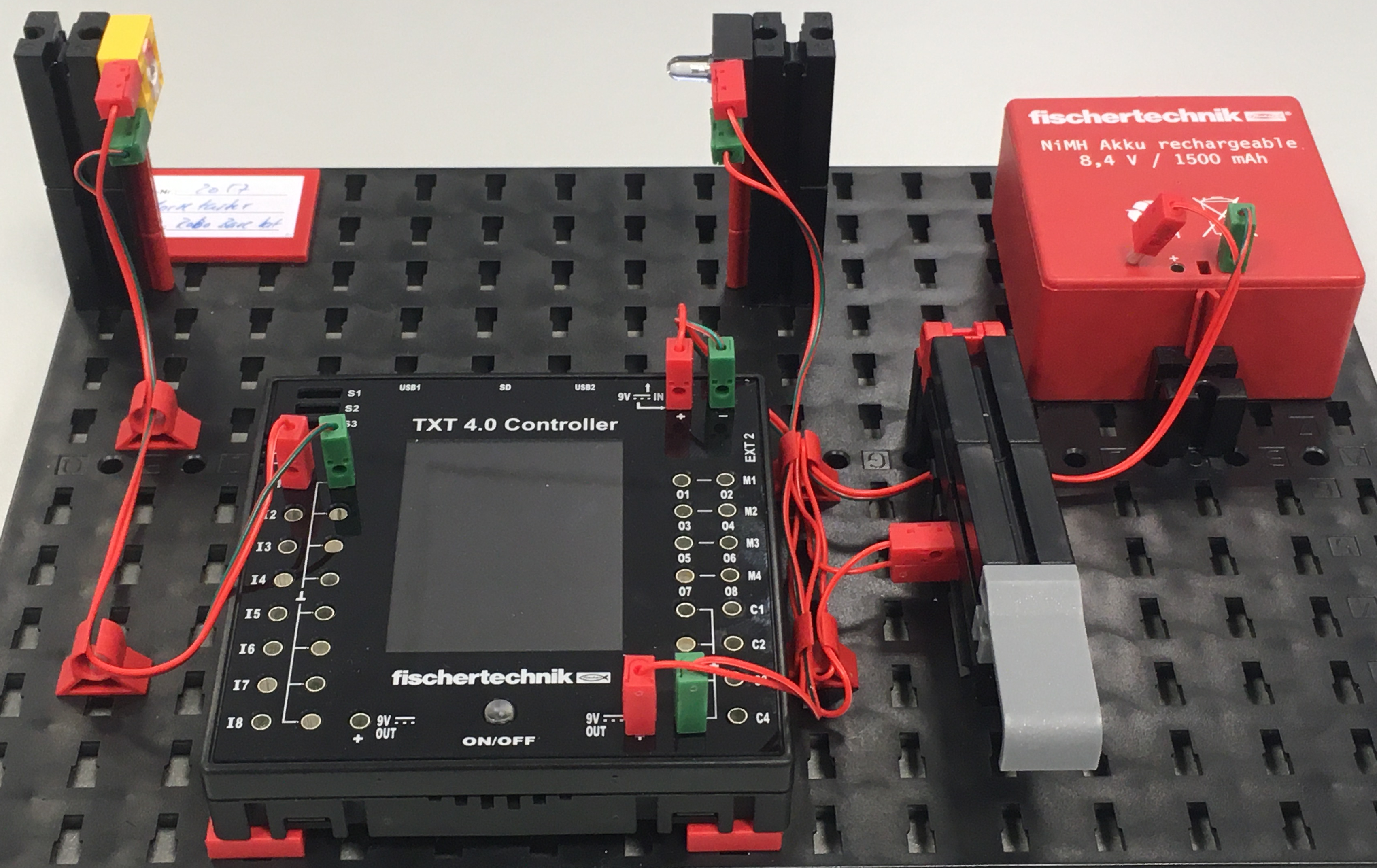
Aufgabe 5:

# Kommunikation – Morse-Code – Verschlüsselung

## Konstruktionsaufgabe

Baue den Morse-Taster. Wird er betätigt, soll die Sende-LED aufleuchten.

Installiere auf der Grundplatte eine Lichtschranke aus der Sende-LED und einem Foto­transistor (siehe Bauanleitung). Schließe den Fototran­sistor an I1 und die LED für die Empfangs­anzeige an O2 an (siehe Schaltplan).



Kommunikation mit Licht: Lichtschranke als Kommunikationskanal

## Programmieraufgaben

**1. Lichtsignale**

In Aufgabe 2 hast du eine Lichtschranke kennengelernt, bestehend aus einer LED und einem Fototransistor. Eine solche „Licht-Strecke“ werden wir nun zur Übertragung von Signalen verwenden: Wenn der Sender den Morse-Taster drückt, soll ein Lichtsignal ausgesendet werden. Empfängt der Fototransistor das Lichtsignal, soll er eine Signal-LED beim Empfänger einschalten. Eine solche Verbindung zur Übertragung von Informationen nennen wir Kommunika­tionskanal.

1a. Zeichne ein Zustandsübergangsdiagramm für den Empfänger.

1b. Schreibe ein Blockly-Programm für den Empfänger des Lichtsignals.

1c. Erweitere das Programm um einen Zähler, der die Anzahl der empfangenen Signale auf dem Display anzeigt.

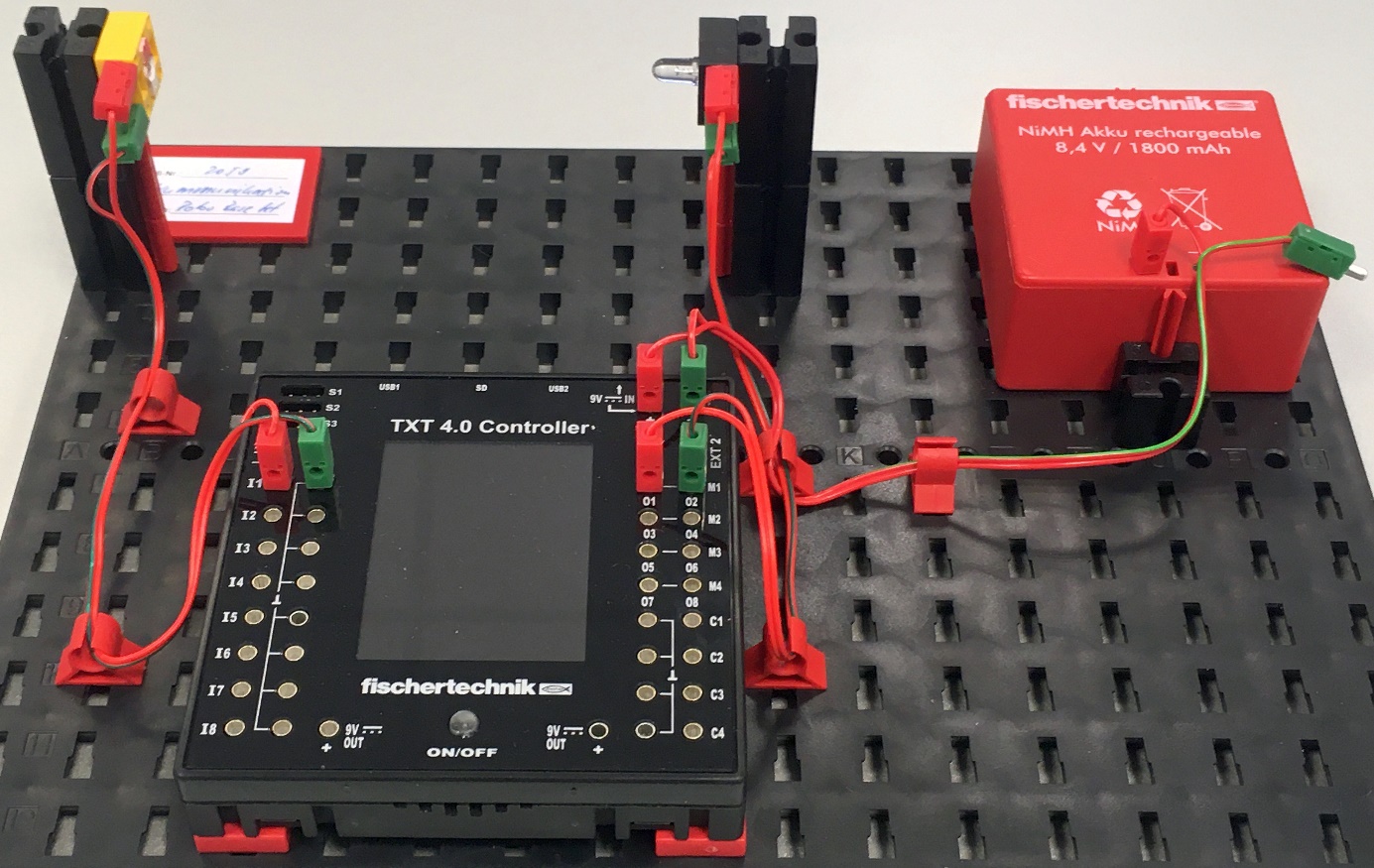
Dieser einfache Licht-Kommunikationskanal ermöglicht zunächst nur, ein einfaches Signal, nämlich eine Ein-Bit-Status-Information (an/aus) zu übertragen.

Mit dem Zähler können wir einen (einzelnen) Wert übermitteln, der vom Empfänger zu interpretieren ist.

**2. Binärcode-Sender**

Wenn wir mehr Daten übermitteln möchten, benötigen wir einen Code, in dem wir die zu übermittelnden Daten darstellen (kodieren) können. Dafür verwenden wir nun einen Binärcode, der aus kurzen und langen Signalen besteht: Ein kurzes Signal („dit“) steht für eine „0“, ein langes Signal („dah“) für eine „1“. Die einzelnen Signale werden durch eine Pause voneinander getrennt.

Damit der Empfänger kurze und lange Signale unterscheiden kann, benötigen wir außerdem einen „Takt“. Den geben wir durch die Länge der Signale und Pausen vor: ein „dah“ ist dreimal so lang wie ein „dit“, die Pause zwischen zwei Signalen entspricht einem „dit“.



Nun überlassen wir das Senden der Signale dem TXT. Dafür schließen wir die Sende-LED ohne Taster direkt an den Ausgang O1 und GND an (Polarität beachten!).

2a. Zeichne ein Zustandsübergangsdiagramm für ein Programm, das eine vorgege­bene „0“/„1“-Folge in eine Folge kurzer („dit“) und langer („dah“) binärer Signale (mit einer Pause zwischen je zwei Signalen) umwandelt.

2b. Schreibe ein Blockly-Programm, das eine (als Text) vorgegebene „0“/„1“-Folge über unseren Licht-Kommunikationskanal aussendet. Verwende dabei als Basis-Zeiteinheit 100 ms (= Länge eines „dit“).

**Tipp**: Wenn du die Länge der Basis-Zeiteinheit einer Variablen zuweist, kannst du die Übertragungsgeschwindigkeit später sehr leicht durch Veränderung des Variablen­wertes anpassen.

2c. Ergänze das Programm um einen nebenläufigen Prozess, der die empfangenen Signale mit der Empfangs-LED anzeigt – die Funktion hast du schon in Teilaufgabe 1b programmiert.

**3. Morse-Zeichen-Sender**

Wir möchten jetzt eine längere Text-Nachricht in Morsezeichen umwandeln und versenden. Im Morse-Code wird jedes Zeichen des Morse-Alphabets (Buchstaben und Ziffern) als eine festgelegte „0“/„1“-Folge (Morse-Zeichen) kodiert (siehe Begleit­material). Das Ende eines Zeichens wird durch eine längere Pause (entsprechend der Länge von drei „dit“) signalisiert.

3a. Zeichne zunächst ein einfaches Zustandsübergangsdiagramm, das auf dem Zustandsübergangsdiagramm zu Teilaufgabe 2a aufbaut.

3b. Erweitere dein Programm aus Teilaufgabe 2c so, dass es einen vorgegebenen Text Buchstabe für Buchstabe erst in Morsezeichen „übersetzt“ (kodiert) und dann die entsprechenden Morsezeichen als Licht-Signale sendet.

Verwende dafür das Programm-Fragment „*Morse\_Code\_Template.ft*“, das bereits zwei vordefinierte Listen enthält: alle Buchstaben des Alphabets und das Morse-Alphabet.

Übermittle zum Testen den Text „thequickbrownfoxjumpsoverthelazydog“ – dieser verbreitete Test-Text enthält alle Buchstaben des (englischen) Alphabets.

## Experimentieraufgaben

**1. Binärcode-Empfänger**

Der Bitfolgen-Sender soll jetzt nicht nur die gesendete Bitfolge empfangen und per LED anzeigen, sondern die gesamte Binärfolge anzeigen.

1a. Erweitere den Empfangs-Prozess des Binärcode-Sender-Programms um eine Dekodierung der empfangenen Signale als Bits und gib‘ statt der gesendeten die empfangene Binärfolge auf dem Display des TXT aus.

1b. Teste das Programm mit Mitschülern/innen und deren TXT, indem ihr eine Binärfolge jeweils an den Fototransistor des jeweils anderen TXT sendet.

1c. Wie kurz kannst du ein „dit“ wählen, ohne dass es zu Übertragungsfehlern kommt? Welche Übertragungsgeschwindigkeit (in bit/s) erreicht dein Kommunikationskanal?

**2. Morse-Zeichen-Empfänger**

Jetzt möchten wir den Morsecode-Sender um einen Morsecode-Empfänger erwei­tern. Der Empfangs-Prozess soll die empfangene Binärzeichenfolge zu Morse­zeichen zusammen­setzen und diese wiederum als Buchstaben des Alphabets dekodieren und ausgeben.

2a. Erweitere den Empfangs-Prozess der Binärfolge in deinem Programm um eine Erkennung der Morsezeichen. Dekodiere sie und gib‘ den empfan­genen Text auf dem Display aus.

2b. Teste das Programm mit Mitschülern/innen und deren TXT, indem ihr eure Nachricht jeweils an den Fototransistor des jeweils anderen TXT sendet.

2c. Wie kurz kannst du „dit“ für eine fehlerfreie Übertragung wählen?

2d. Die Geschwin­digkeit eines Morsesenders wird meist in „Wörtern pro Minute“ (WpM) angegeben. Dabei wird das Wort „Paris“ (inklusive Wortende-Signal) als Maß verwendet. Wie schnell ist dein Morse-Sender/-Empfänger bei einer „dit“-Länge von 100 ms, ausgedrückt in WpM?

**3. Morse-Trainer**

Mit dem Morsecode-Empfänger hast du nun einen „Morsecode-Trainer“, an dem du Morsen üben kannst. Schließe dazu die Sende-LED wieder an den Morsetaster an.

Versuche, mit dem Morse-Taster „SOS“ (oder einen anderen Text) manuell zu morsen, sodass dein Morsecode-Empfänger den Text korrekt erkennt.

**4. Verschlüsselung – Caesar-Chiffre**

Die Übertragung auf unserem Licht-Kommunikationskanal ist ungeschützt, sodass unsere übermittelten Nachrichten von Unbefugten mitgelesen werden können. Das möchten wir nun ändern, indem wir die Nachricht vor der Übertragung verschlüsseln.

Erweitere das Morse-Zeichen-Empfänger-Programm aus Experimentieraufgabe 2 um eine Caesar-Ver­schlüsselung (siehe Begleitmaterial).

**Hinweis**: Speichere den Schlüssel (die Anzahl Stellen, um die die Buchstaben des Alphabets „verschoben“ werden) in einer Variablen, damit du ihn leichter ändern kannst.

**5. Verschlüsselung – Vigenère-Chiffre**

Schutz vor einer einfachen Häufigkeitsanalyse bietet die Vigenère-Ver­schlüsselung. Ersetze die Caesar-Verschlüsselung in Experimentieraufgabe 4 durch eine Vigenère-Ver­schlüsselung (siehe Begleitmaterial).

**Hinweis**: Anstatt den Schlüssel im Programm fest vorzugeben kannst du ihn über das Display des TXT eingeben lassen.

Anlagen

# Aufgabe 5: Kommunikation – Morse-Code – Verschlüsselung

## Erforderliches Material

* PC für Programmentwicklung, lokal oder über Web-Schnittstelle.
* USB-Kabel oder BLE- bzw. WLAN-Verbindung für die Übertragung des Programms auf den TXT4.0.
* Programm-Template (für Morsecode): Morse\_Code\_Template.ft

## Weiterführende Informationen

[1] Albrecht Beutelspacher: *Kryptologie: Eine Einführung in die Wissenschaft vom Verschlüsseln, Verbergen und Verheimlichen*. 10. Auflage, Springer Verlag, 2015.

[2] Simon Singh: Codes. *Die Kunst der Verschlüsselung*. Impian, 2021.

[3] Online-Diagrammeditor zur Erstellung von Zustandsübergangsdiagrammen (Format drawio): <https://www.diagrammeditor.de/>