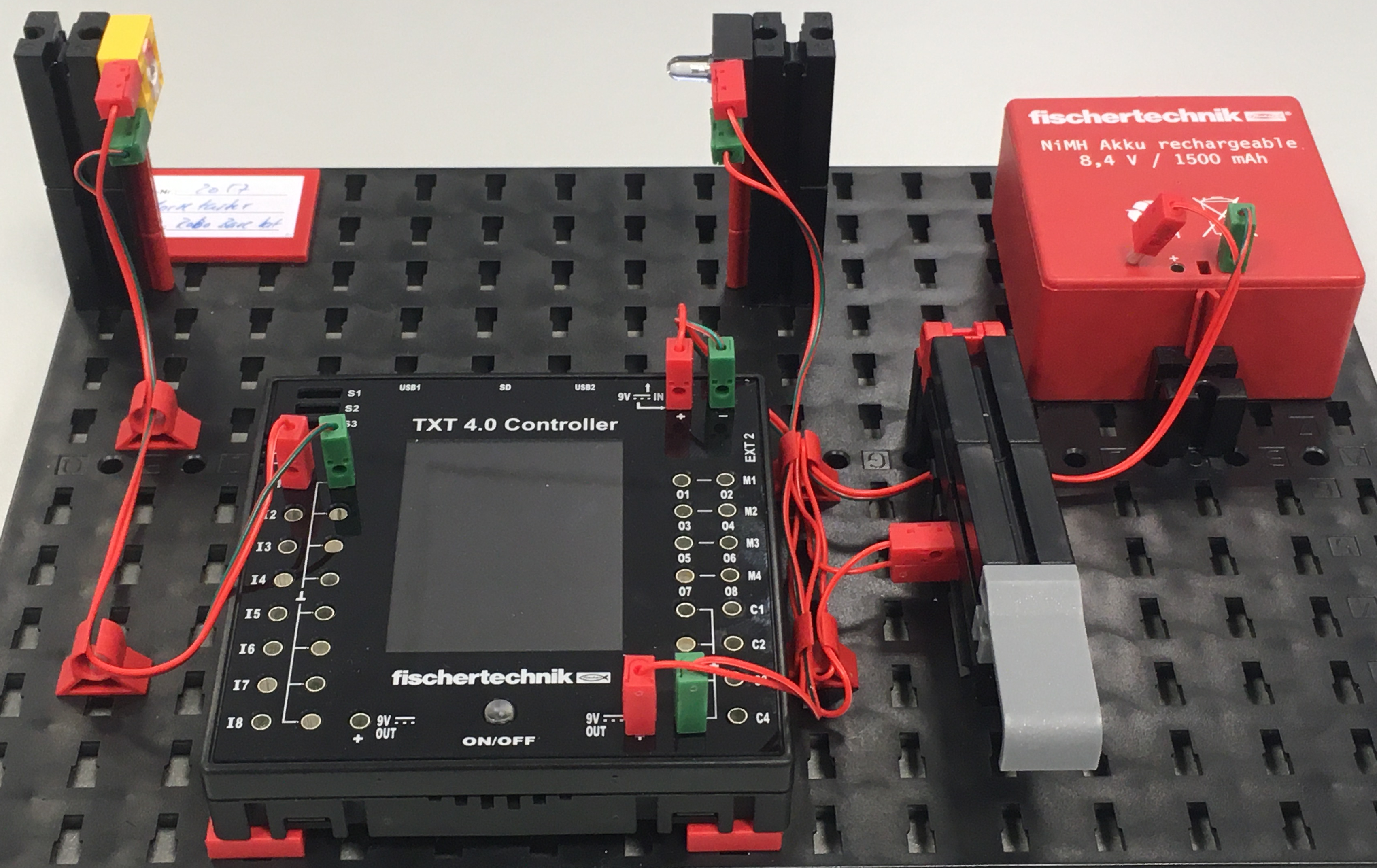
Task 5:

# Communication – Morse-Code – Encryption

## Construction task

Build the Morse button. When it is activated, the transmission LED should light up.

Install a light barrier on the base plate consisting of the transmission LED and a phototransistor (see building instructions). Connect the phototransistor to I1 and the receiver indication LED to O2 (see wiring diagram).



Communication through light: The light barrier as a communication channel

## Programming tasks

**1. Light signals**

In task 2, you worked with a light barrier consisting of an LED and a phototransistor. We will now use this kind of light path to transmit signals. When the sender presses the Morse button, a light signal will be transmitted. When the phototransistor receives the light signal, it should switch on a signal LED at the recipient. We call this kind of connection used to transmit information a communication channel.

1a. Draw a state diagram for the receiver.

1b. Write a Blockly program for the receiver of the light signal.

1c. Expand the program by adding a counter that shows the number of received signals on the display.

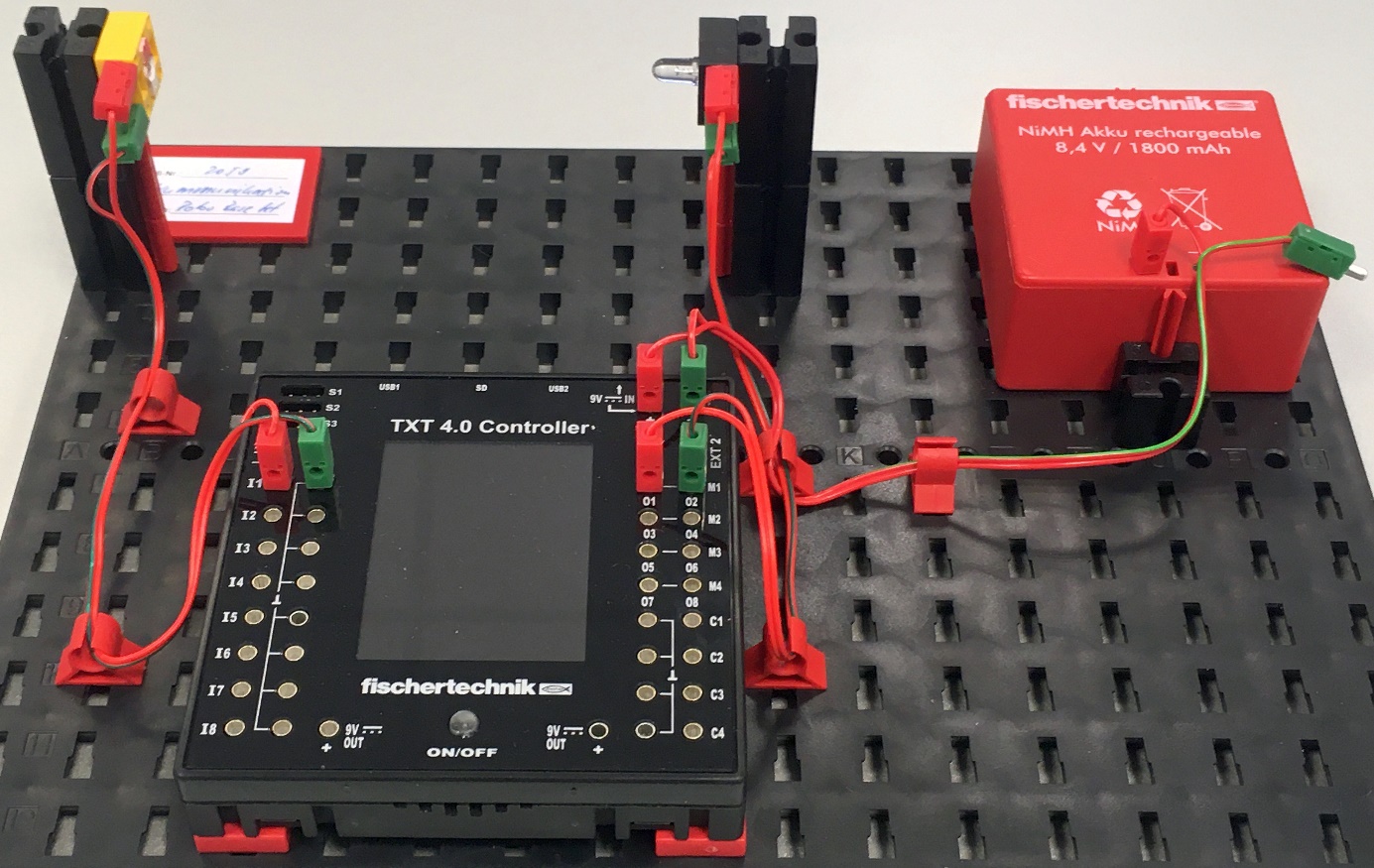
This simple light communication channel initially can only be used to transmit a simple signal, namely a single bit of status information (on/off).

We can use the counter to transmit a (single) value that is to be interpreted by the receiver.

**2. Binary code transmitter**

If we want to transmit more data, we will need a code we can use to represent (encode) the transmitted data. We will use a binary code to do so, consisting of short and long signals: A short signal (“dit”) stands for a “0”, while a long signal (“dah”) stands for a “1”. The individual signals are separated from one another by a pause.

We will also need a “rhythm” so that the receiver can differentiate between short and long signals. This is specified by the length of the signals and pauses: a “dah” is three times as long as a “dit”; the pause between two signals is the length of a “dit”.



Now, let’s have the TXT transmit the signals. To do so, we will connect the transmission LED without a button directly to the output O1 and GND (ensure correct polarity!).

2a. Draw a state diagram for a program that converts a specified sequence of “0” / “1” into a sequence of short (“dit”) and long (“dah”) binary signals (with a pause between every two signals).

2b. Write a Blockly program that transmits a specified sequence of “0” / “1” (as text) via our light communication channel. Use the base unit of 100 ms (= length of a “dit”).

**Tip**: If you assign the length of the base unit to a variable, you can later easily change the transmission speed by changing the value of the variable.

2c. Add a concurrent process to the program that displays the received signals with the receiver LED. You have already programmed this function in sub-task 1b.

**3. Morse Code transmitter**

Now, we want to convert a longer text message into Morse code and send it. In Morse code, each character in the Morse alphabet (letters and numbers) is encoded as a set sequence of “0” / “1” (Morse characters) (see the accompanying materials). The end of a character is signalled by a longer pause (corresponding to the length of three “dit”).

3a. First, draw a simple state diagram that builds on the state diagram from sub-task 2a.

3b. Expand your program from sub-task 2c so that it “translates” (encodes) a specified text letter by letter first into Morse characters, then transmits these Morse characters as light signals.

To do so, use the program fragment “*Morse\_Code\_Template.ft*”, which already contains two pre-defined lists: all of the letters of the alphabet and the Morse alphabet.

To test it, transmit the text “thequickbrownfoxjumpsoverthelazydog”. This commonly used text test contains all of the letters in the (English) alphabet.

## Experimental tasks

**1. Binary code receiver**

The bit sequence transmitter will now not only receive the transmitted bit sequence and display it using LEDs, but rather display the entire binary sequence.

1a. Expand the receiver process of your binary code transmitter program by adding decoding of the received signals as bits; then, output the received binary sequence on the display of the TXT instead of the transmitted sequence.

1b. Test the program with fellow students and their TXT, by transmitting a binary sequence to the phototransistor of the other TXT.

1c. How short can a “dit” be without causing transmission errors? What transmission speed (in bit/s) can your communication channel deliver?

**2. Morse Code receiver**

Now, we want to add a Morse Code receiver to the Morse Code transmitter. The receiver process should combine the received binary character sequence into Morse characters, then decode and output these as letters of the alphabet.

2a. Expand the receiver process for the binary sequence in your program by adding detection of the Morse characters. Decode these and output the received text on the display.

2b. Test the program with fellow students and their TXT, by transmitting your message to the phototransistor of the other TXT.

2c. How short can the “dit” be and still ensure error-free transmission?

2d. Usually, the speed of a Morse transmitter is indicated in “words per minute” (WpM). The word “Paris” (including word end signal) is used as a standard. How fast is your Morse transmitter/receiver with a “dit” length of 100 ms, expressed in WpM?

**3. Morse trainer**

With the Morse Code receiver, you now have a “Morse Code trainer” that you can use to practice Morse Code. To do so, connect the transmission LED to the Morse button once again.

Try to manually transmit “SOS” (or another text) using the Morse button so that the Morse Code receiver correctly detects the text.

**4. Encryption – Caesar cipher**

Transmission on our light communication channel is not protected, and unauthorised persons could read the messages we are transmitting. Now, we want to prevent this by encrypting the message before transmission.

Add a Caesar encryption (see the accompanying materials) to the Morse Code receiver program from experimental task 2.

**Note**: Save your key (the number of places by which the letters of the alphabet are “shifted”) in a variable, so that you can change it easily later on.

**5. Encryption – Vigenère cipher**

The Vigenère encryption offers protection against a simple frequency analysis. Replace the Caesar encryption in experimental task 4 with Vigenère encryption (see the accompanying materials).

**Note**: Instead of specifying the keys in the program, you can enter it via the display of the TXT.

Annex

# Task 5: Communication – Morse-Code – Encryption

## Required materials

* PC for program development, local or via web interface.
* USB cable or BLE or WiFi connection to transmit the program to the TXT4.0.
* Program template (for Morse code): Morse\_Code\_Template.ft

## Further information

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

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

[3] Online diagram editor for creating state diagrams (Format drawio): <https://www.diagrammeditor.de/>