Solution sheet

# Task 5: Communication – Morse-Code – Encryption

While solving the tasks, students learn how a transmitter and receiver are synchronised during data transmission, and how data is encoded and encrypted for transmission on a communication channel. At the same time, they learn how to handle strings and indexed lists.

Students receive the program fragment with the alphabets (“Morse\_Code\_Template.ft”) in addition to the building instructions.

## Construction task

See building instructions.

The transmitter LED is connected to the 9V output and a GND connection on the TXT via the Morse button; the Morse button only needs the TXT as a power source. The receiver LED is connected to O2, and the phototransistor to I1.

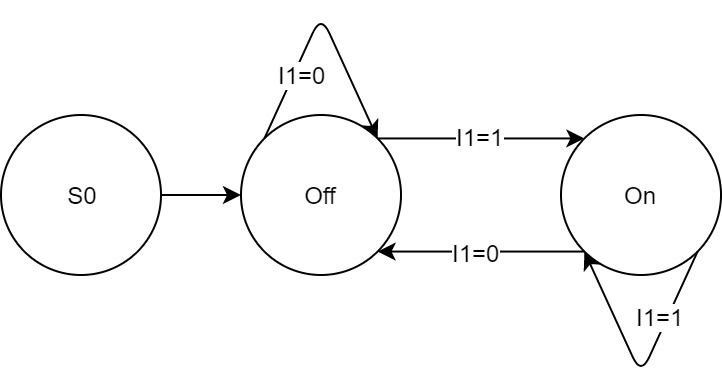
## Programming tasks

**1. Light signals**

Connection configuration:

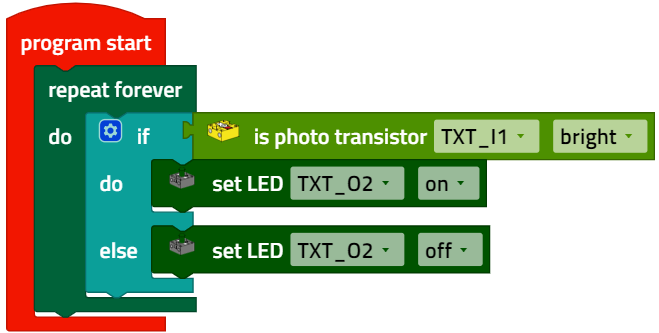


1a. State diagram for the receiver:



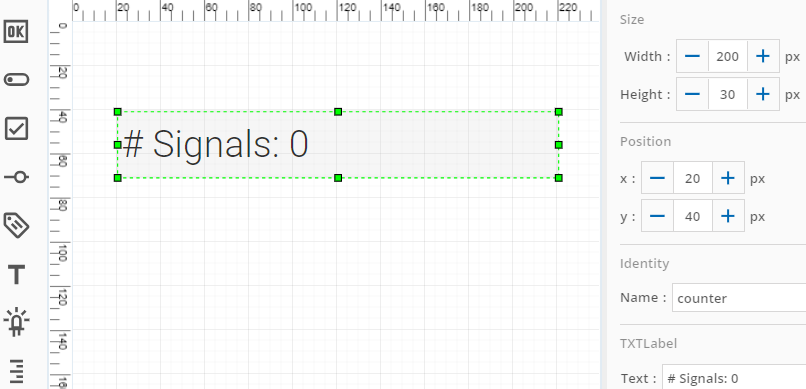
*State-Transition\_Diagram\_Light\_Signal.drawio*

1b. Program (example) receiver:

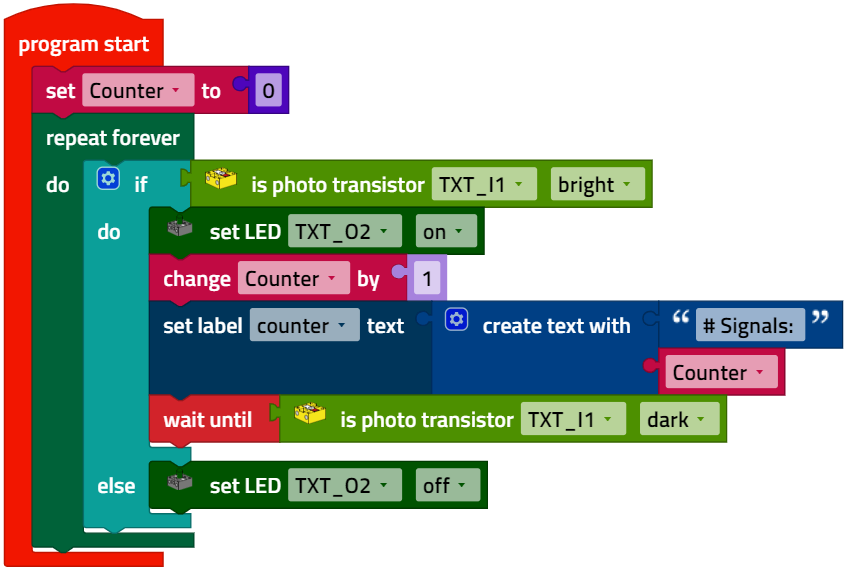


*Light\_Signal.ft*

1c. Configuring the display:



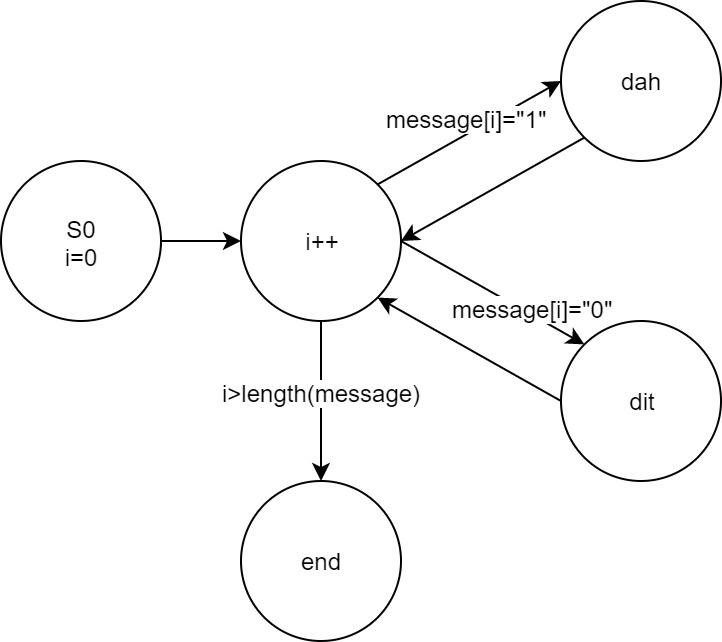
Program (example) receiver with counter:



*Light\_Signal\_Counter.ft*

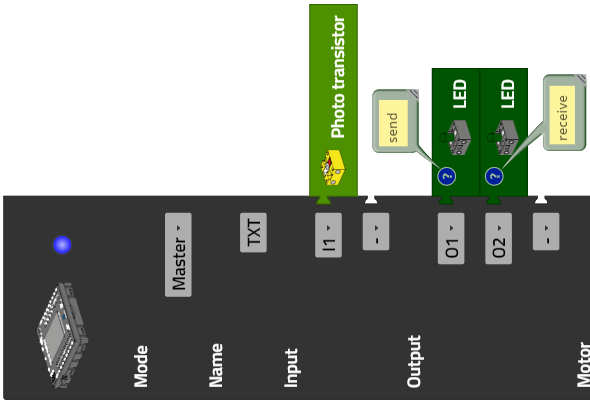
**2. Binary code transmitter**

2a. State diagram for the bit sequence transmitter:

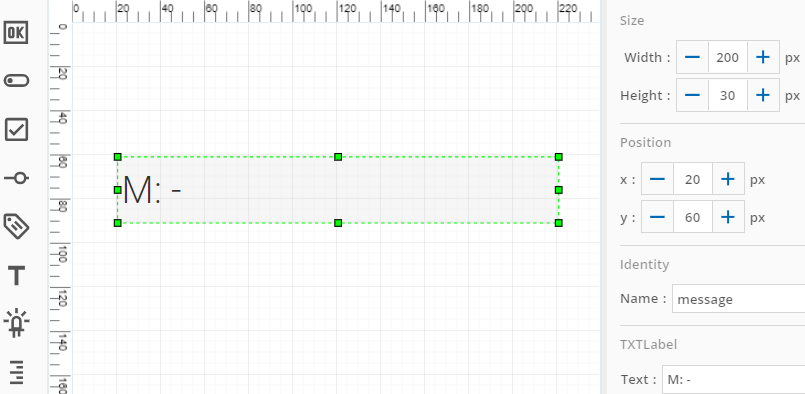


*State-Transition\_Diagram\_Bit\_Sequence\_Sender.drawio*

2b. Connection configuration for the transmitter LED:



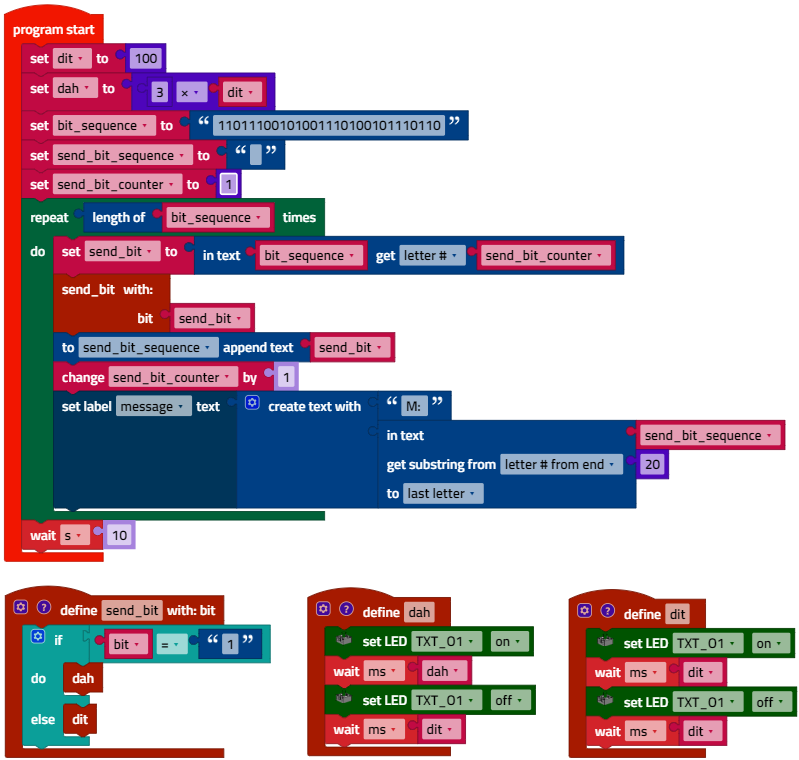
Configuring the display:



The last 20 characters of the transmitted bit sequence are output on the display.

To solve the task, it is a good idea to use the text function to evaluate the message texts, and to outsource parts of the program to functions so that the overall program remains easy to understand.

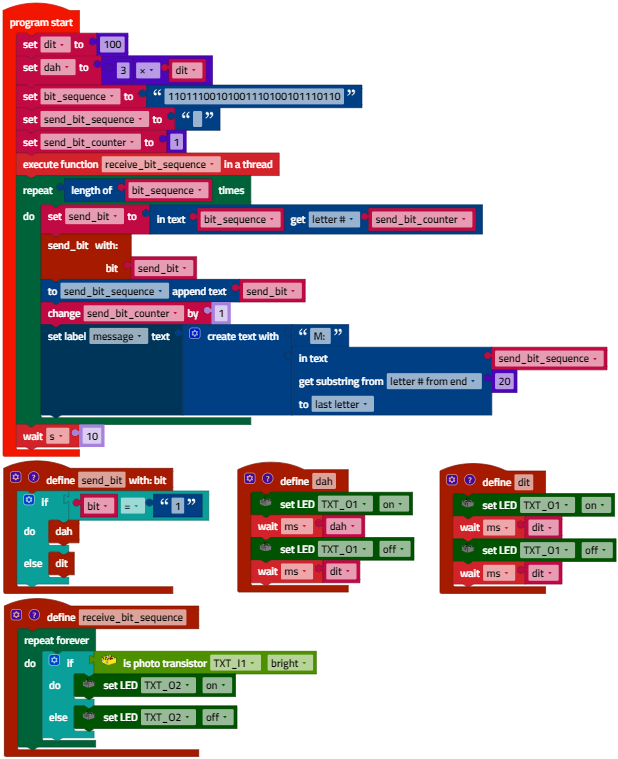
Program (example) bit sequence transmitter:



*Bit\_Sequence\_Sender.ft*

2c. The receiver routine (“receive\_bit\_sequence”) is started as a concurrent process (thread). It can be expanded with a decoder in the experimental tasks.

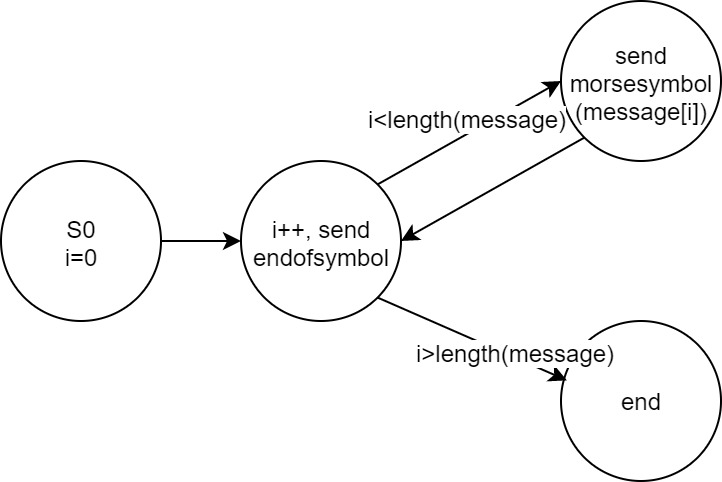
Program (example) bit sequence transmitter with receipt display (LED).



*Bit\_Sequence\_Sender\_Receive\_LED.ft*

**3. Morse Code transmitter**

3a. State diagram for the Morse Code transmitter:



*State-Transition\_Diagram\_Morse\_Code\_Sender.drawio*

Transmitting a character in the message corresponds to transmitting a Morse symbol - a sequence of “0”s and “1”s as in the bit sequence transmitter from sub-task 2a.

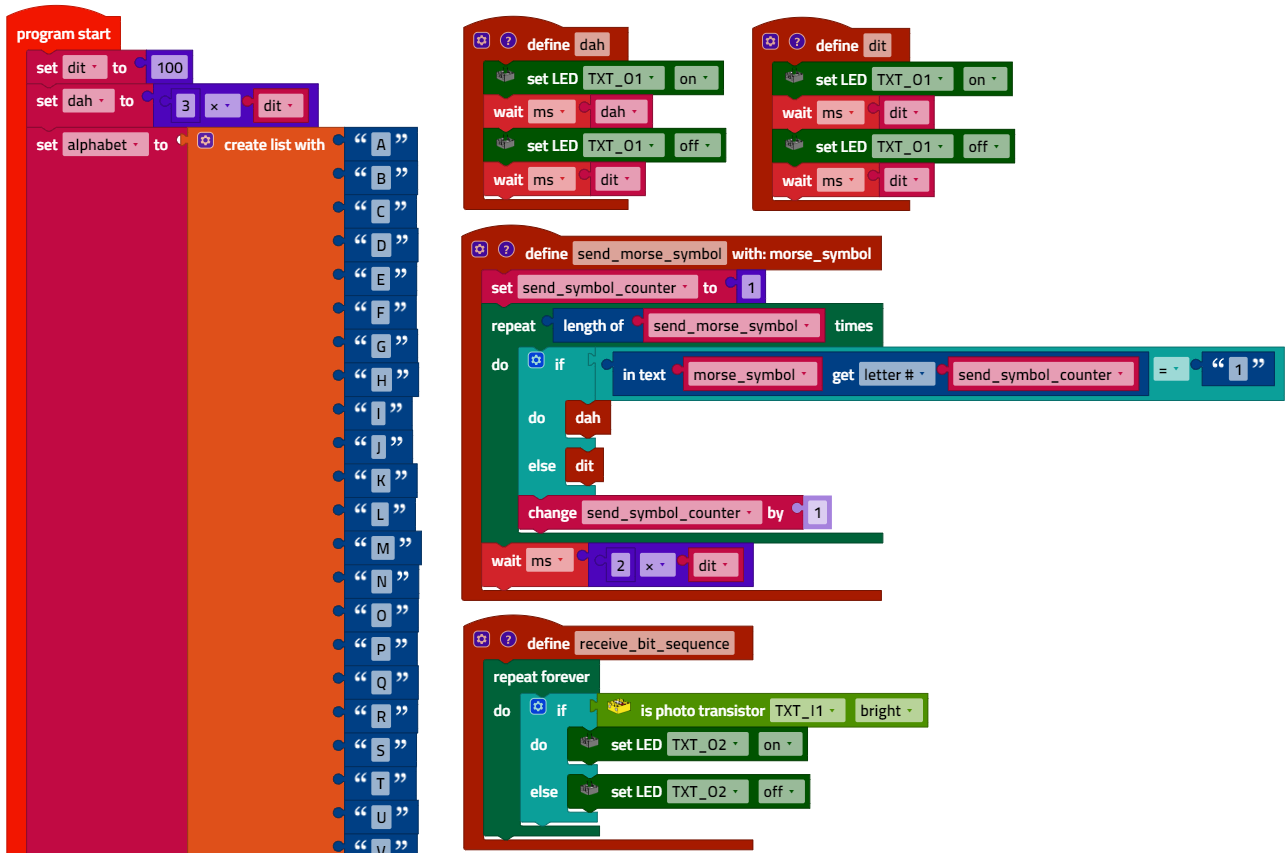
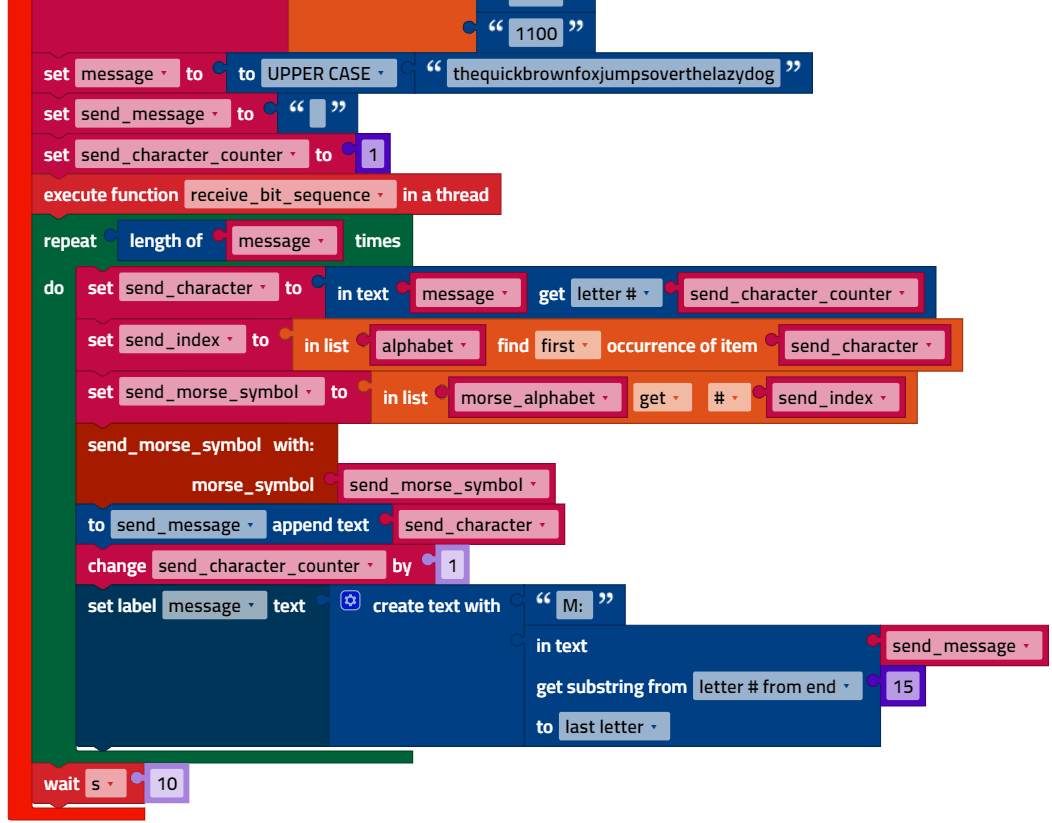
3b. The list function can be used to determine the index of a list element and read out an element with a certain index. This makes coding the text very elegant.

**Note**: The Morse alphabet does not differentiate between small and capital letters. Therefore, the message text must be converted to all capitals before coding.

**Solution variants**: The message can also be typed in via the input field on the TXT display. In the following example solution, it is specified in the program code as a variable.

(The initialisation of both alphabets is shown in an abbreviated form in the figure.)

Program (example):

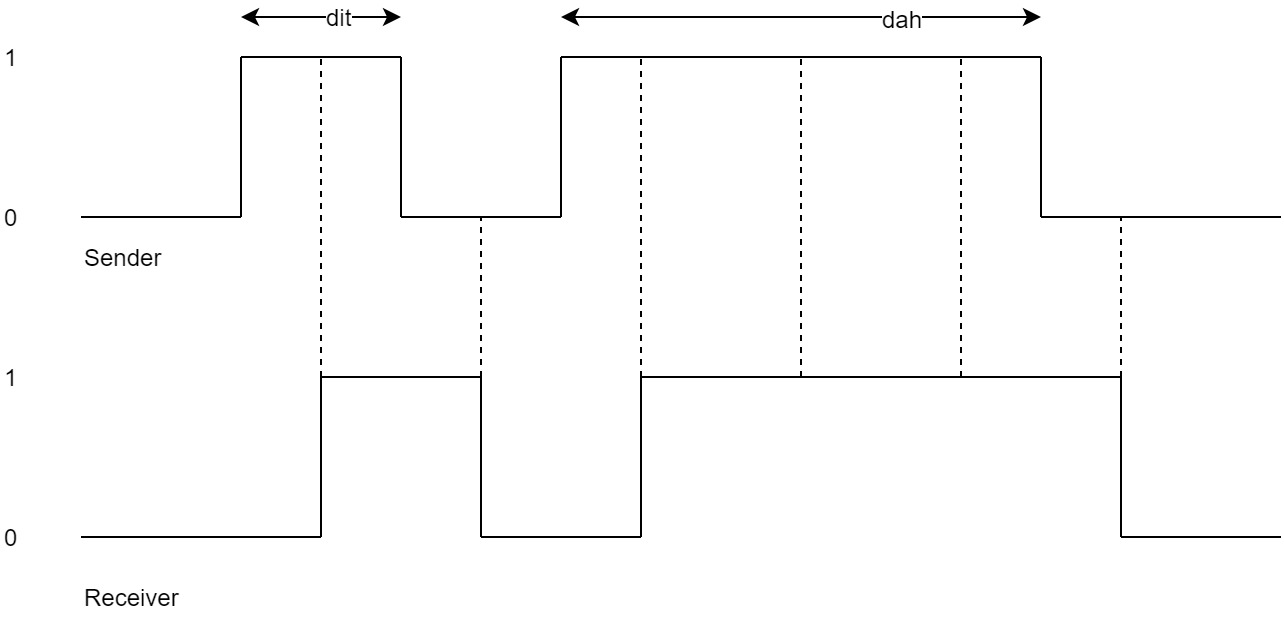
  
…  


*Morse\_Code\_Sender.ft*

## Experimental tasks

**1. Bit sequence recipient**

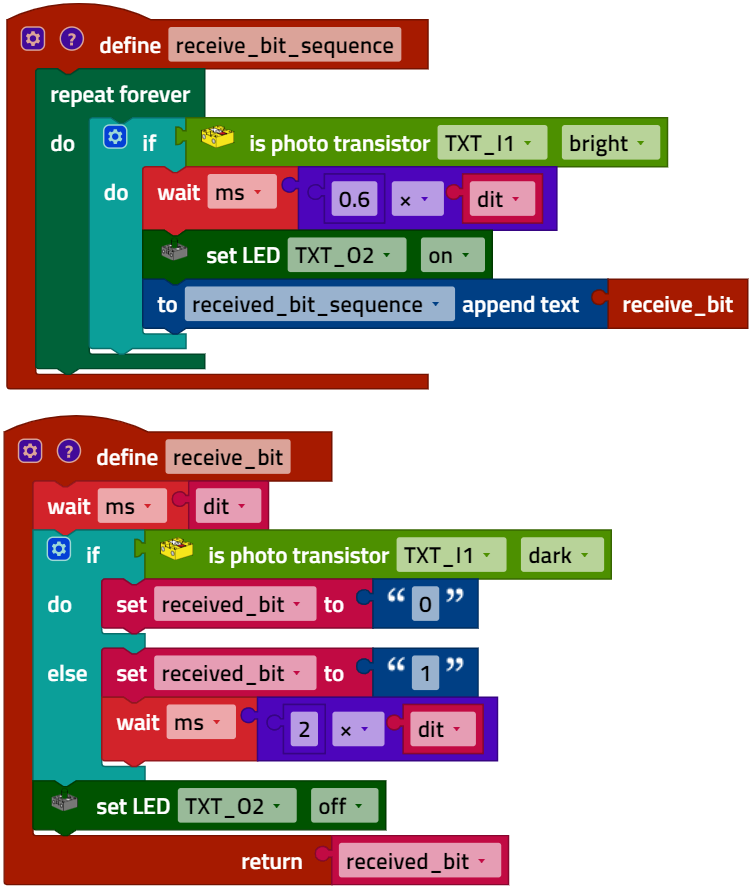
The recipient process is “triggered” by the phototransistor. Then, the process waits around half of one signal length (0.5 x “dit”) to then sample the signal again in around the middle of the signal duration (see sketch).



*Morse\_Signal\_Sampling.drawio*

The received bit sequence (“received\_bit\_sequence”) is output on the display of the TXT instead of the transmitted sequence in the main program.

1a. Program excerpt (example):



*Bit\_Sequence\_Sender\_Receiver.ft*

1b. The transmission can still be completed without errors with a “dit” length = 90 ms. This corresponds to a speed of around 11 bit/s.

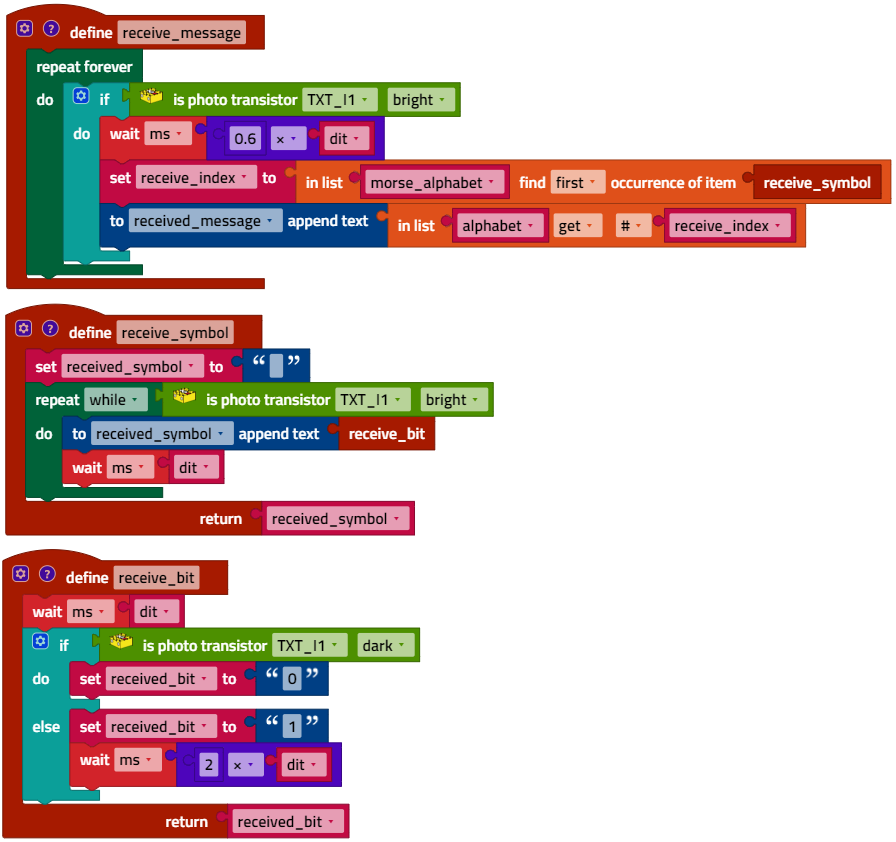
For comparison: A DSL connection today can reach 100 Mbit/s, which is around 9,000,000 times faster than this.

**2. Morse Code receiver**

The receiver function in the example program is divided into three parts:

* Receiving a single bit (as in experimental task 1)
* Receiving a Morse code
* Receiving a message (started as a thread)

2a. Program excerpt (example):



*Morse\_Code\_Sender\_Receiver.ft*

2c. Receipt is still reliable in the example program with a “dit” length = 70 ms. Therefore, the maximum transmission speed is around 14 “dit” per second.

2d. The reference word “Paris” consists of 50 “dit”. Therefore, our Morse transmitter can achieve a “dit” = 100 ms 12 WpM. This corresponds to the Morse speed that amateur radio operators have to achieve in their test. The record is 88 WpM.

**4. Encryption – Caesar cipher**

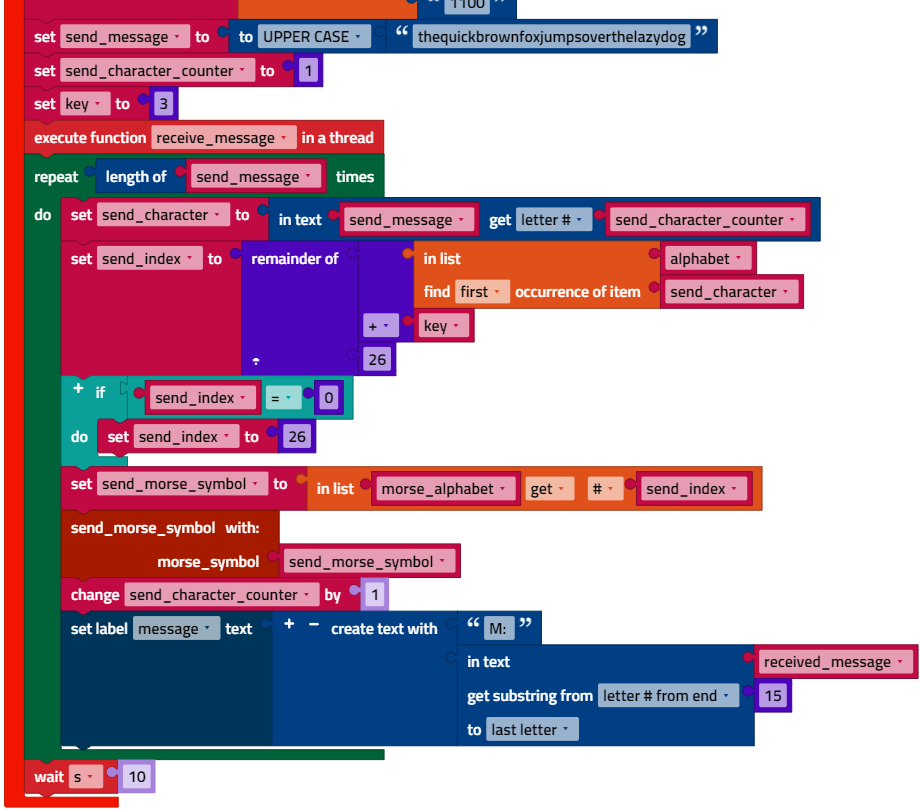
Encryption using the Caesar cipher is very easy to program by “shifting” the characters in the list element “Morse alphabet” by the appropriate number of characters: by +3 characters during encryption, then -3 characters during decryption.

Please note: The index for the list elements of the alphabet and Morse alphabet runs from 1 to the number of elements (here, 26). After adding or subtracting the key, the rest of the division must be set by 26. If the result (of the index) = 0, then the index must be set to the maximum value (26).

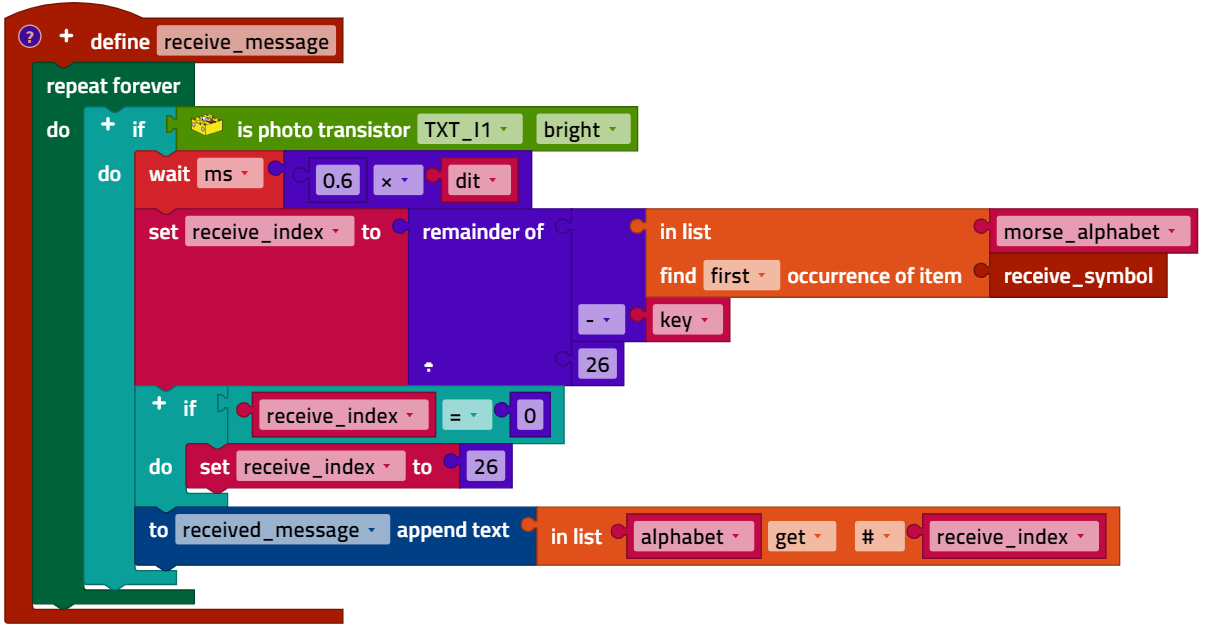
Solution variants: The key (the number of letters by which the alphabet is “shifted”) can also be selected variably via input on the display.

Program excerpt (example):

Encryption:

…  


Decryption:

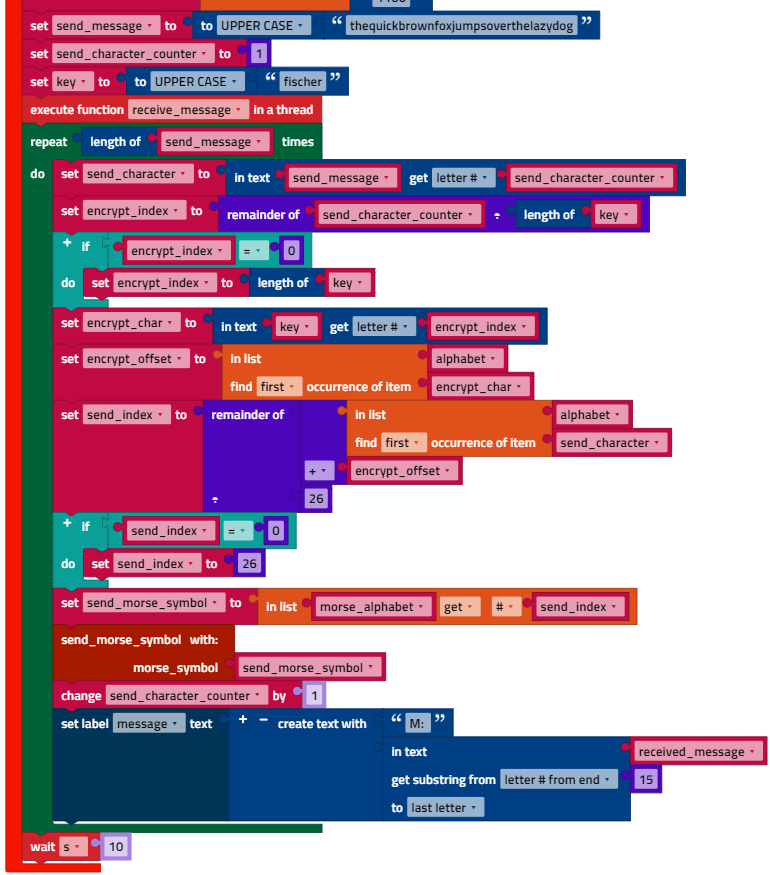


*Morse\_Code\_Sender\_Receiver\_with\_Caesar\_Encryption.ft*

**5. Encryption – Vigenère cipher**

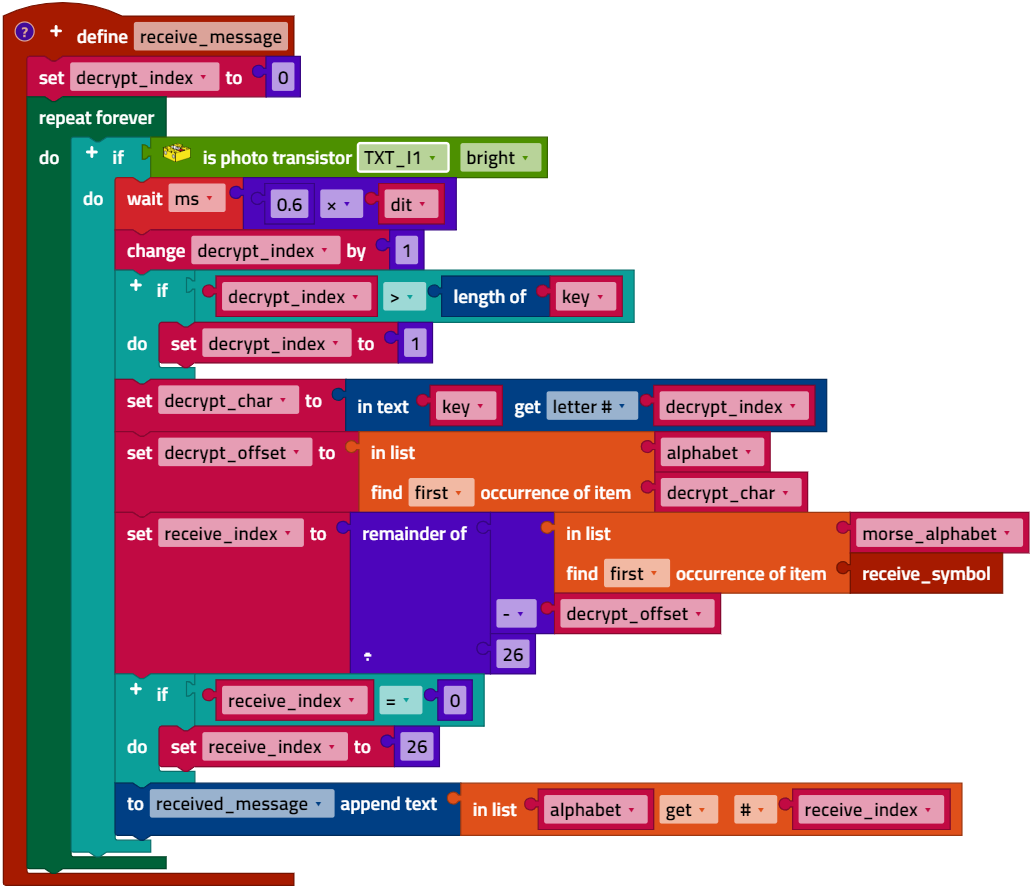
Program excerpt (example):

Encryption:

…  


Other variables must be used to determine the current key symbol for decryption, since the decryption and encryption occur concurrently.

Decryption:



*Morse\_Code\_Sender\_Receiver\_with\_Vigenere\_Encryption.ft*

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/>