fischertechnik

Tutorial ROBO Pro Coding

STEM Coding RoboMission

2024.12

Índice

ROBO Pro Coding y controlador TXT 4.0 2
Tutoriales STEM Coding RoboMission 3
★ **Motor codificador y servo** 4
★ **Sensores ópticos de color** 7
★ **Prueba de interfaz** 8
★★ **Módulos, funciones e importaciones de Python** 9
★★ **Control de motores como biblioteca [RoboMission_motor_servo]** 10
★★ **Botón START y GUI TXT 4.0 [RoboMission_display_start]** 12
★★ **Seguidor de línea con sensores ópticos de color [RoboMission_line]** 15
★★ **Código Python con Blockly** 19
★★ **Sensor de color RGB como biblioteca [RoboMission_rgbw_hsv]** 20
★★ **Medición de la tensión de la batería [test_battery]** 22
★★ **Plantilla para proyectos propios [RoboMission_template]** 23
★★★ **Seguidor de línea con controlador PID [RoboMission_linepid]** 24
Outlook 26

ROBO Pro Coding y controlador TXT 4.0

En su entorno de programación multilingüe, el software ROBO Pro Coding de fischertechnik ofrece la opción de programación gráfica con **Blockly** así como programación basada en texto mediante **Python3** para el controlador TXT 4.0.

ROBO Pro Coding está disponible para los sistemas operativos **Windows**, **Mac**, **Linux, Android** e **iOS**:

https://www.fischertechnik.de/es-es/aplicaciones-y-software#aplicaciones

La documentación para **Blockly** en ROBO Pro Coding se puede ver en el siguiente enlace:

https://doc.fischertechnik-cloud.com/de/blockly

La documentación del fischertechnik TXT 4.0 se encuentra aquí:

https://www.fischertechnik.de/es-es/juguete/e-learning/txt-4-0-controller

Si tiene alguna pregunta sobre el contenido de este documento o tutorial, póngase en contacto con Esta dirección de correo electrónico está siendo protegida contra los robots de spam:

fischertechnik-technik@fischer.de

Tutoriales STEM Coding RoboMission

Este documento describe ejemplos de cómo se pueden utilizar los sensores y actuadores del set de robótica **STEM Coding RoboMission**. Todos los ejemplos se basan en la propuesta de modelo de robot, pero también se pueden utilizar para sus propios modelos.

Todos los ejemplos relevantes de este documento pueden ser importados directamente a ROBO Pro Coding desde **fischertechnik GitLab**. Para ello puede introducirse el nombre del programa en el campo de búsqueda en *Menú -> Importar -> fischertechnik GitLab* y cargarse directamente el ejemplo de programa correspondiente

GitLab

fischertechnik GitLab

Por favor, ten en cuenta que este tutorial asume que ya estás familiarizado con ROBO Pro Coding y conoces los fundamentos de ROBO Pro Coding. Los siguientes temas relevantes para RoboMission se cubren en los siguientes capítulos, por lo que el nivel de dificultad se puede clasificar de la siguiente manera:

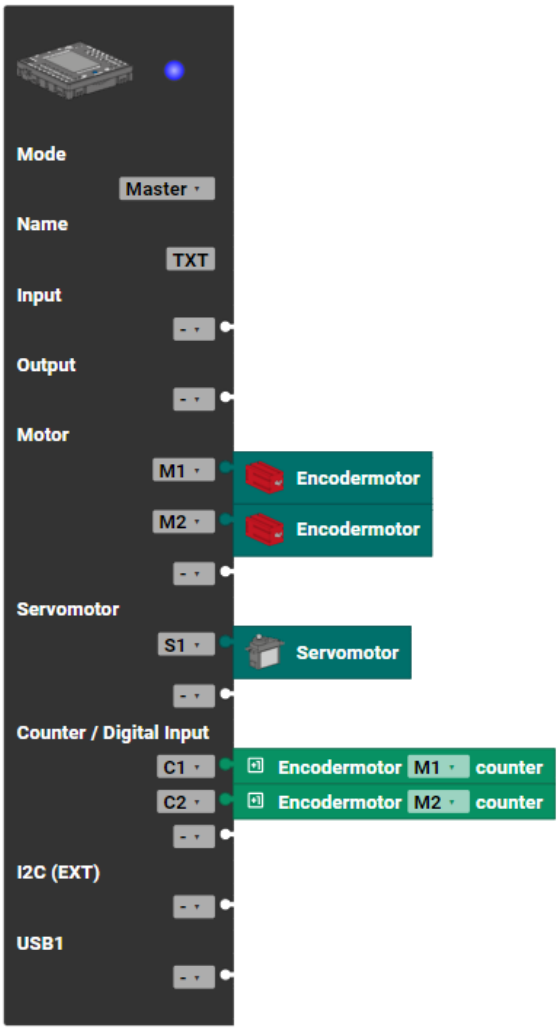
★	simple
★★	avanzado
★★★	complejo

Para familiarizarte con los componentes incluidos y dar tus primeros pasos con ROBO Pro Coding, te recomendamos que veas los vídeos de e-learning:

- STEM Coding RoboMission - fischertechnik

★ Motor codificador y servo

A un controlador TXT 4.0 se pueden conectar simultáneamente un máximo de 4 motores codificadores (M1-M4, C1-C4) y un máximo de 3 servomotores (S1-S3

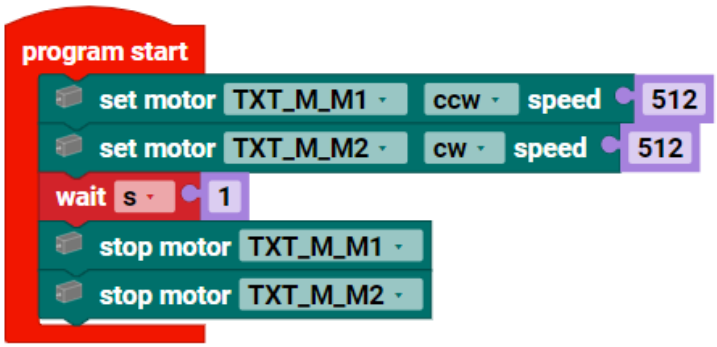


Para ello, primero se asignan los motores correspondientes a las salidas en la configuración del controlador en ROBO Pro Coding y luego se pueden seleccionar en el programa principal.

Control directo del motor

En el caso más sencillo, los dos motores codificados pueden controlarse sin utilizar los codificadores configurando directamente el sentido de giro -horario (CW) o antihorario (CCW)- y el valor de velocidad (0 ... 512) para la modulación por ancho de pulsos (PWM).

El siguiente programa enciende los dos motores durante 1 segundo y los detiene una vez transcurrido el tiempo de espera.

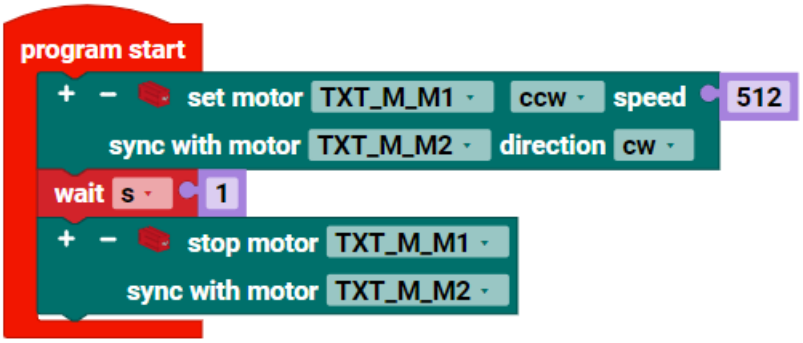


La desventaja del control directo es que los dos motores no están sincronizados entre sí, es decir, siempre se enciende y apaga primero un motor y poco después el otro. Además, las influencias externas sobre los motores y la diferente fricción entre las ruedas influyen negativamente en el comportamiento del robot. Por tanto, no es posible conducir el robot con precisión en línea recta.

Control avanzado del motor con sincronización

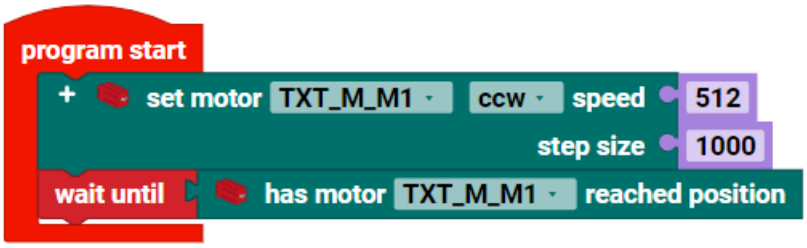
El firmware del controlador TXT 4.0 incluye un sistema ampliado de control del motor que ya realiza un control sincronizado basado en las señales del codificador.

Con los siguientes bloqueos, los dos motores M1 y M2 se sincronizan entre sí y el funcionamiento síncrono se detiene de nuevo al cabo de 1 segundo. Si un motor se bloquea durante este tiempo, el otro motor sigue al primero. La sincronización es relevante, por ejemplo, para que el robot se mueva en línea recta.

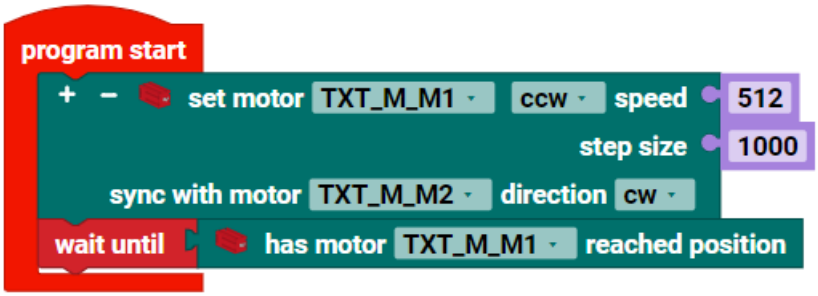


Control avanzado del motor con sincronización y especificación de la distancia

Además del funcionamiento sincronizado, se puede especificar una distancia (número de impulsos para el codificador). A continuación, puede ajustarse para un solo motor



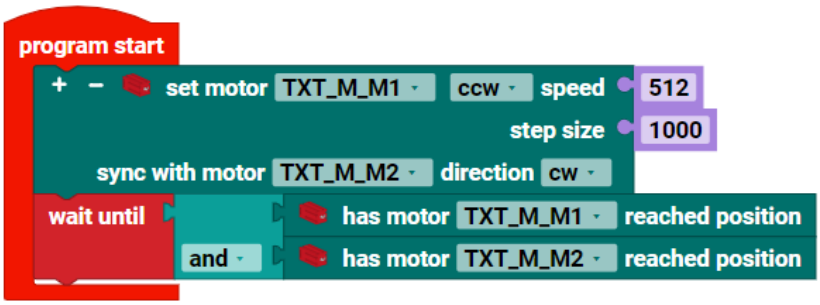
o sincronizados para varios motores.



El estado actual del control de motor ampliado puede consultarse con el bloque "**ha alcanzado el motor ... la posición**". La unidad de control del motor utiliza el estado para señalar que se ha alcanzado la posición y poder continuar con la ejecución del programa. Una vez

alcanzada la distancia, por ejemplo después de 1000 impulsos, el motor se detiene automáticamente.

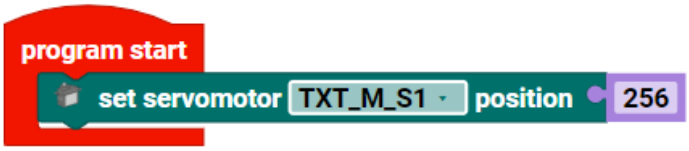
Normalmente, basta con consultar el estado de un motor. Sin embargo, también se puede consultar el estado de todos los motores para garantizar que el controlador tenga en cuenta todos los motores en caso de rotura del cable del codificador.



Control del servomotor

A diferencia del motor, que normalmente se utiliza para accionar el robot, el servomotor puede utilizarse para aproximarse directamente a ángulos absolutos de aprox. -90 ... 90 grados directamente. El control y los sensores ya están incluidos en el servomotor. El servomotor es ideal para dispositivos de agarre de robots, por ejemplo, allí donde se requieran movimientos giratorios.

servomotor se mueve a un ángulo específico especificando un valor PWM. La especificación es absoluta. La posición central del servo es aproximadamente 256 y los dos límites son 0 y 512. Los valores absolutos especificados difieren ligeramente dependiendo del servo. Por lo tanto, deben calibrarse individualmente para cada servo. Esto puede hacerse utilizando constantes o variables en el programa, por ejemplo.

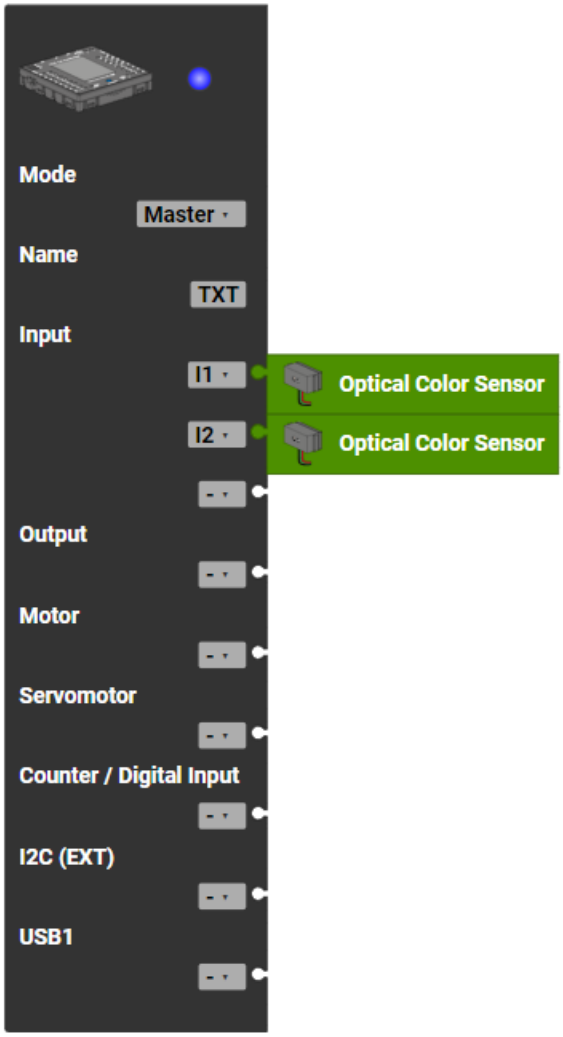


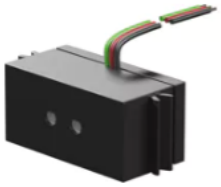
También debe tenerse en cuenta que el servo no tiene retroalimentación en relación con el controlador. El servo intenta alcanzar la posición especificada a través del valor PWM. Si la posición objetivo del servo estuviera permanentemente bloqueada mecánicamente, una corriente muy alta podría fluir a través del servo. Esto podría dañar el servo. Esto debe evitarse a toda costa cuando se utilice el servo.

★ Sensores ópticos de color

Los sensores digitales o analógicos pueden conectarse a las entradas universales (I1-I8) de un controlador TXT 4.0.

Para ello, primero se asignan los sensores correspondientes a las entradas en la configuración del controlador en ROBO Pro Coding y luego se pueden seleccionar en el programa principal.

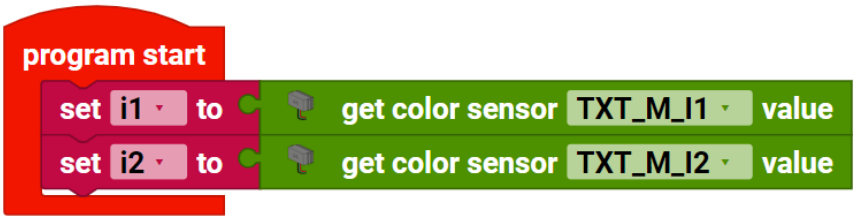




2x

El juego STEM Coding RoboMission incluye dos **sensores ópticos de color** que pueden utilizarse para detectar transiciones simples de claro/oscuro en el recorrido, por ejemplo, pista negra. El sensor óptico también puede utilizarse para detectar transiciones blanco-negro, por ejemplo, valores de gris. Esto puede utilizarse para alinear el robot con precisión a la línea o para implementar un seguidor de línea.

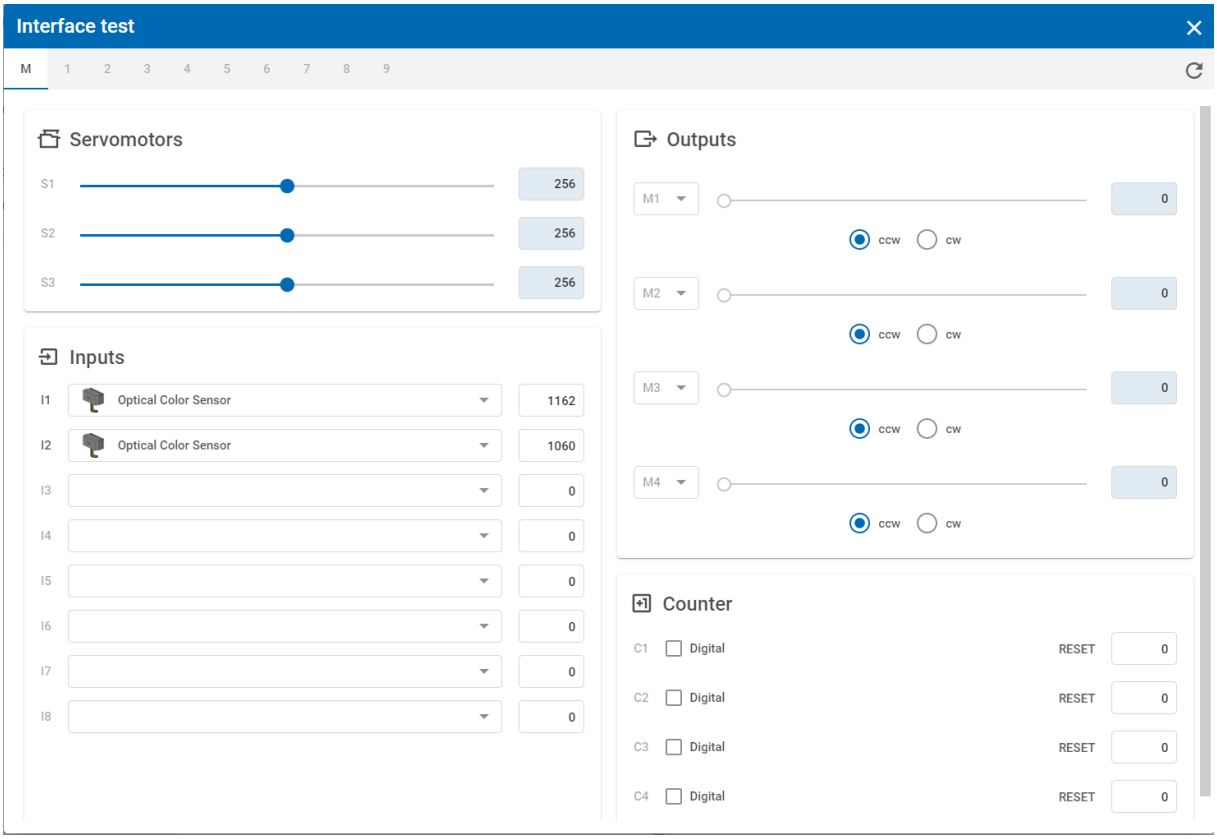
El valor analógico de los dos sensores puede leerse con los siguientes bloques: El valor analógico corresponde en mV a la tensión medida en las dos entradas I1 e I2.



Tenga en cuenta que los sensores de color RGB no se conectan a las entradas universales, sino al bus I2C (conexión EXT1 o EXT2) del controlador TXT 4.0. El sensor de color RGB también se incluye en el juego y se trata en los capítulos siguientes.

★ Prueba de interfaz

Los actuadores y sensores también pueden probarse rápidamente sin programa. Para ello se puede utilizar la prueba de interfaz en ROBO Pro Coding cuando se establece una conexión con el controlador TXT 4.0.



Para los servomotores S1-S3, el valor puede ajustarse mediante el control deslizante.

La velocidad y el sentido de giro de un motor pueden ajustarse para los motores M1-M4

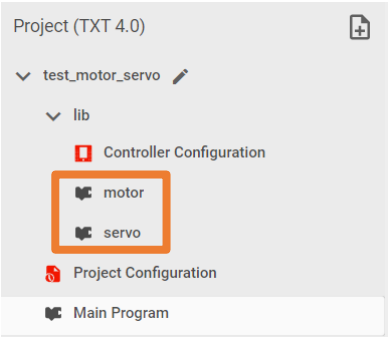
El valor se muestra para las entradas universales I1-I8, por ejemplo, para los sensores ópticos. Las entradas universales individuales deben configurarse previamente mediante el cuadro de selección.

El valor actual del contador C1-C4 para las entradas del encoder también se muestra como un valor.

Tenga en cuenta que la prueba de interfaz y un programa no pueden iniciarse al mismo tiempo.

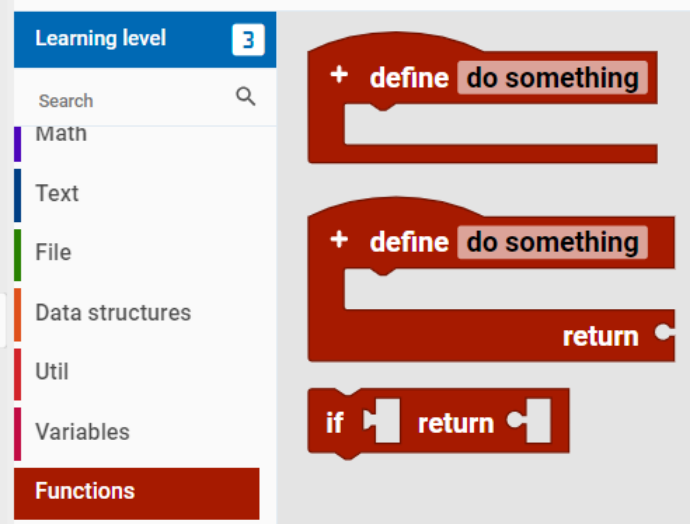
★★ Módulos, funciones e importaciones de Python

En el ejemplo de programa **RoboMission_motor_servo**, las funciones para motores y servomotores se definen en archivos adicionales que se pueden encontrar en la estructura bajo "lib". La ventaja de tener tus propios módulos de Python es que esto hace que el código sea más claro y puede ser reutilizado como una biblioteca en otros proyectos.

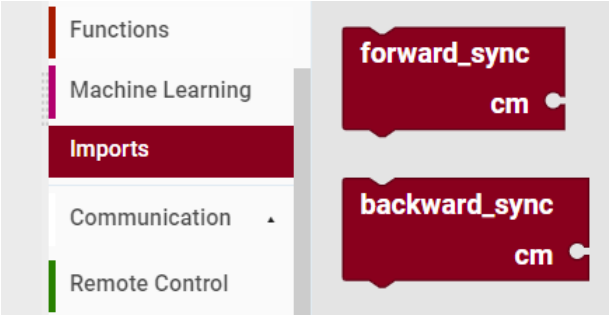


Las funciones disponibles localmente en el módulo o en el programa principal se definen con "**Funciones**". Las funciones pueden definirse tanto con parámetros como con valores de retorno.

Los parámetros se añaden con (+) y se definen en consecuencia.



Si las funciones se van a utilizar en otros ficheros, se pueden añadir a través de "**Importaciones**". Para ello, cambie al archivo en el que se va a utilizar la función y seleccione esta función de otro módulo a través de "Importaciones".

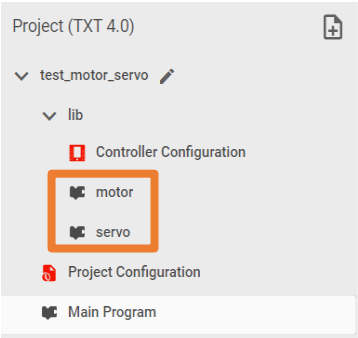


Las funciones locales "Funciones" y las funciones importadas "Importaciones" tienen colores diferentes. Esto permite diferenciarlas en la secuencia del programa.

Asegúrese de no definir referencias circulares entre los módulos. Los módulos no deben importar las funciones de los demás, ya que, de lo contrario, las importaciones no podrán resolverse correctamente

★★ Control de motores como biblioteca [RoboMission_motor_servo]

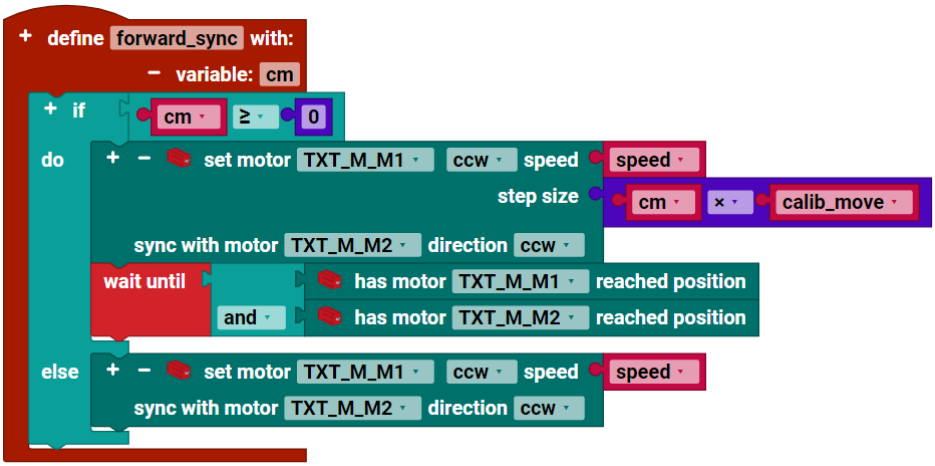
El ejemplo **RoboMission_motor_servo** muestra cómo se pueden controlar un motor (encoder) y un servomotor.

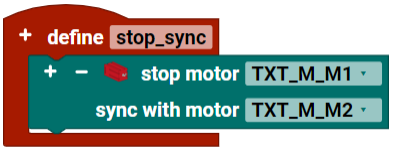


Familiarícese con el funcionamiento de los motores codificadores y los servomotores antes de consultar el programa. Encontrará los datos técnicos de los motores en las respectivas fichas técnicas.

En el módulo "**motor**" se definen varias funciones con las que se pueden controlar los motores de forma asíncrona (sin utilizar codificadores) o síncrona (sincronizados mediante codificadores).

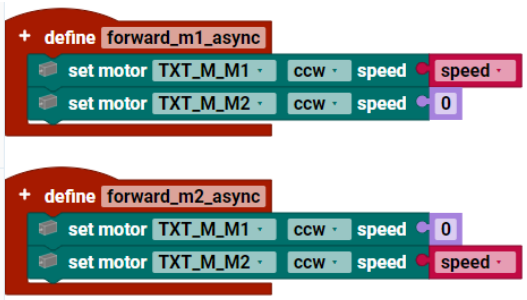
Si los dos motores se mueven **de forma sincronizada**, la unidad de control espera hasta que ambos motores hayan alcanzado un determinado número de impulsos. Si uno de los motores se bloquea mientras tanto, el segundo motor codificador permanece parado hasta que el primero también haya alcanzado la posición. Para que los movimientos del robot en el recorrido sean precisos, los dos motores deben estar siempre sincronizados. También se puede especificar una distancia. Si la distancia debe ser infinita, se puede pasar un parámetro negativo, por ejemplo -1, a la función **forward_sync**. Los dos motores se mueven entonces sincronizados sin fin hasta que ambos motores se detienen de nuevo con **stop_sync**.



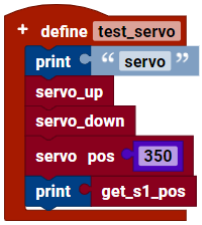


Para que el movimiento del robot pueda especificarse en las funciones del motor, por ejemplo, en **cm** o en **grados**, los contadores de impulsos de los codificadores deben convertirse con los factores correspondientes en el programa. Estos factores se incluyen en la función de calibración. Si se cambia el modelo del robot, en particular la distancia de la rueda y el diámetro de la rueda de STEM Coding RoboMission los factores **calib_move** y **calib_rotation** deben adaptarse al nuevo modelo.

Con un seguidor de línea o un sistema de control complejo, lo mejor es que los dos motores no estén sincronizados entre sí. Para estas aplicaciones, es ventajoso ajustar la velocidad directamente a través de la modulación por ancho de pulsos (PWM) para que los dos motores se puedan ajustar rápidamente y de forma independiente el uno del otro

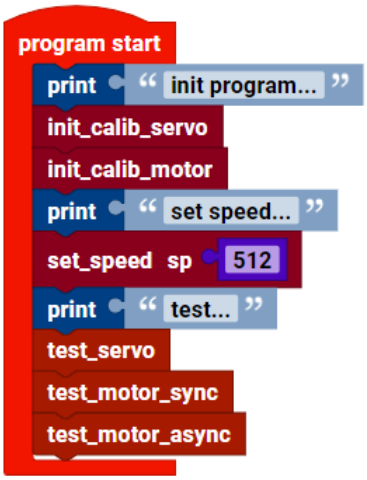


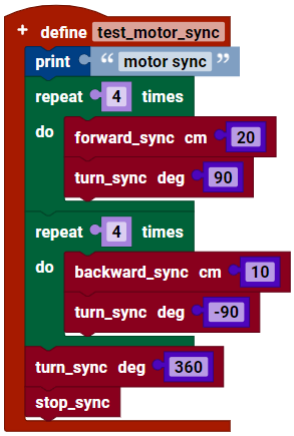
El módulo "**servo**" contiene funciones para el servomotor. El servomotor contiene su propio control y recibe una posición absoluta a través de (PWM: 0...512), que intenta controlar lo más rápidamente posible. A veces pueden fluir corrientes muy grandes durante poco tiempo y, dependiendo de la posición actual, los movimientos pueden ser muy bruscos. Para evitar movimientos bruscos en el servomotor, el programa debe especificar posiciones intermedias a determinados intervalos. De este modo, también es posible realizar movimientos lentos con el servomotor.



Los valores de las posiciones individuales también se pueden calibrar para el servomotor. Esto es necesario porque cada servo tiene posiciones absolutas ligeramente diferentes debido a las tolerancias.

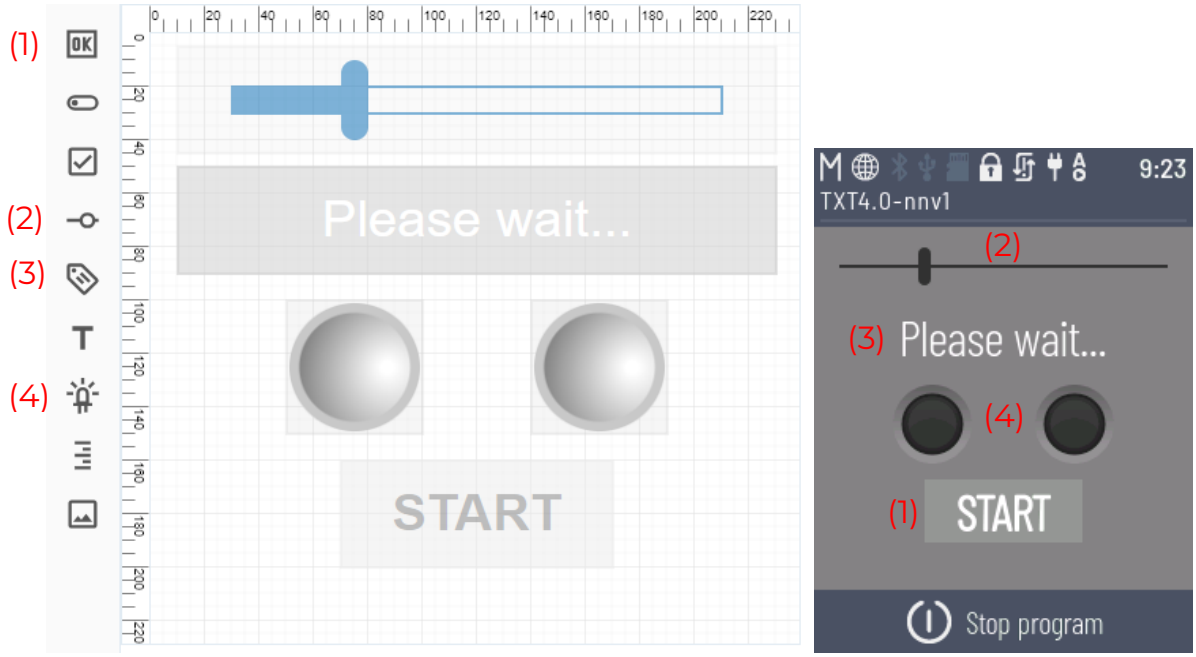
El programa principal en **RoboMission_motor_servo** entonces se parece a esto:





★★ Botón START y GUI TXT 4.0 [RoboMission_display_start]

El ejemplo **RoboMission_display_start** muestra cómo se puede crear una interfaz gráfica de usuario (GUI) en la pantalla del controlador TXT 4.0 en ROBO Pro Coding. Varios elementos están disponibles en ROBO Pro Coding en la barra lateral izquierda en el editor GUI.



En este ejemplo se utilizan los siguientes elementos:

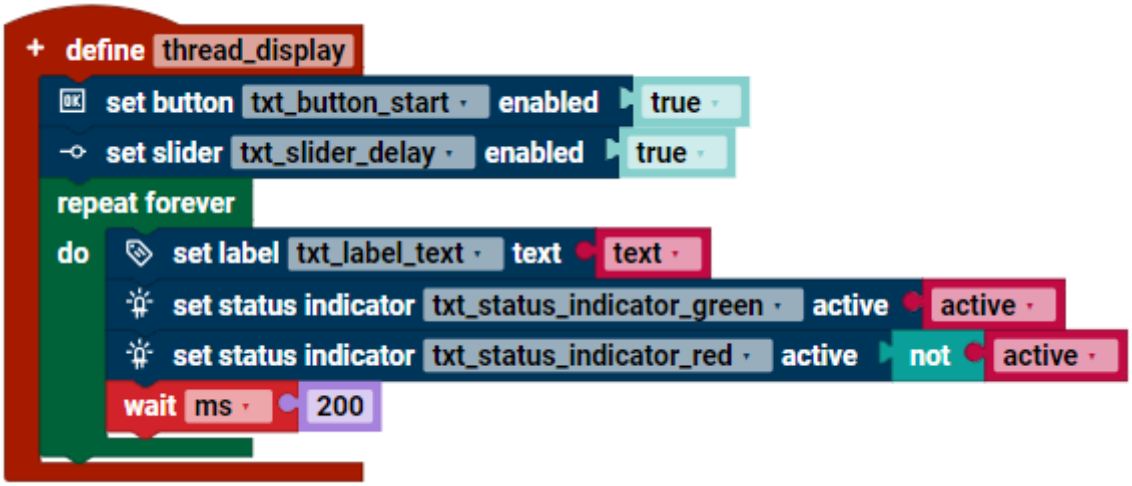
Botón (1):	Inicia la secuencia del programa
Deslizador (2):	Ajuste de valores variables
Etiqueta (3):	Salida de texto
Indicador de estado (4):	Salida de estados binarios

A la izquierda, la definición en el editor GUI; a la derecha, la representación en la pantalla TXT 4.0.

La interfaz de la pantalla puede configurarse como se desee. El tamaño y el estilo de la fuente también pueden configurarse en las propiedades de los elementos.

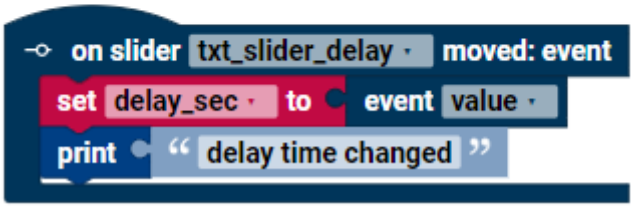
La secuencia del programa debe definirse de forma que se puedan activar acciones a través de la interfaz o se pueda mostrar el estado del programa en la pantalla. También se recomienda prever un hilo separado (un bucle sin fin separado) para actualizar la pantalla. Como el ojo es lento, la pantalla debería actualizarse como máximo cada 200ms ... 1s. Además, un hilo separado para la pantalla alivia el bucle principal, en el que a menudo se controla el robot y el tiempo de reacción debe ser muy corto.

La carga de la interfaz suele tardar unos segundos. Para garantizar que ningún elemento pueda activarse durante la inicialización, todos los elementos que desencadenan una acción pueden desactivarse al principio en las propiedades. Cuando se inicializa el hilo de visualización, los elementos correspondientes pueden entonces activarse específicamente



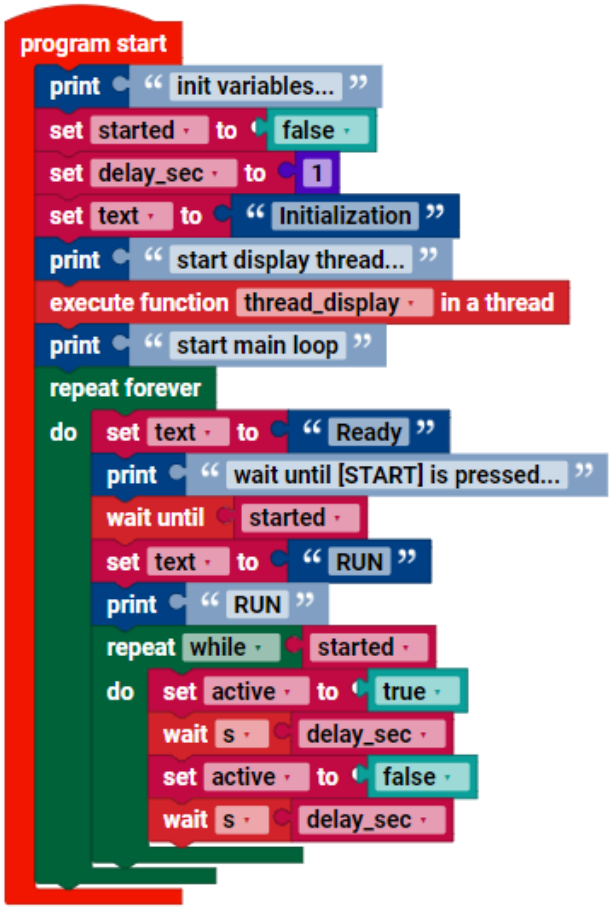
con **set ... enabled**.

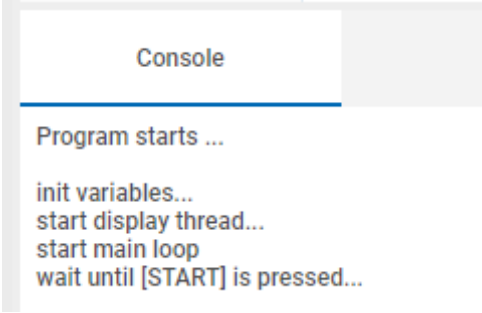
A continuación, deben definirse funciones de devolución de llamada para todas las acciones (por ejemplo, "**al pulsar el botón ...**" y "**al mover el control deslizante ...**"), que establecen variables para los eventos correspondientes. Éstas pueden evaluarse en el programa principal.



En las funciones de retrollamada, se recomienda llamar sólo a funciones cortas y no a funciones de bloqueo, ya que de lo contrario las funciones de retrollamada pueden influirse mutuamente.

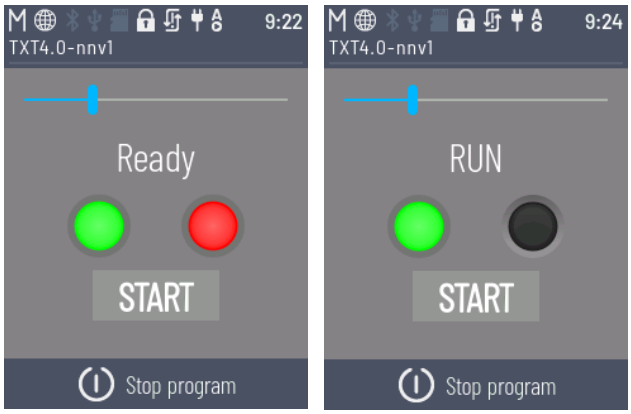
La secuencia del programa en el bucle principal puede tener el siguiente aspecto:





Para poder comprender mejor el programa, a menudo es aconsejable proporcionar salidas con print(...) en la secuencia del programa además de la salida en la pantalla. Cuando el programa se inicia entonces en ROBO Pro Coding, todas las salidas con **print()** en la consola pueden ayudar a analizar el programa para determinar cuándo se ejecutó qué instrucción del programa. Sin embargo, para bucles de control con tiempos de respuesta cortos de, por , 10ms, la salida **print()** debería evitarse.

A continuación se muestra la salida en la pantalla del controlador TXT 4.0 cuando se ejecuta el programa. El estado cambia primero de "Por favor espere..." a "Listo" y finalmente a "EJECUTAR".



Con el botón START, por , ahora es posible que el robot inicie la secuencia del programa sólo cuando se pulsa START.

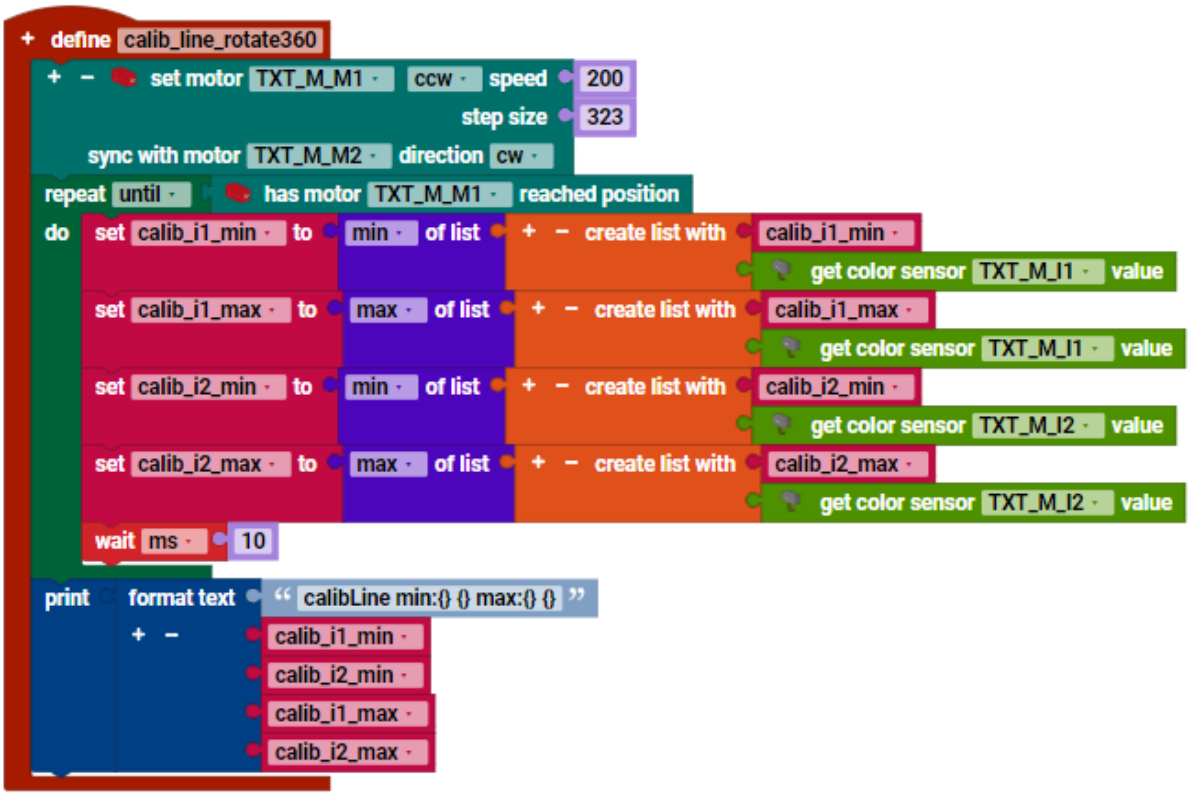
★★ Seguidor de línea con sensores ópticos de color [RoboMission_line]

Este ejemplo **RoboMission_line** muestra cómo puede implementarse un simple seguidor de línea con sensores ópticos de color en las entradas universales.

Los sensores ópticos de color proporcionan una tensión analógica como valor en mV en función de la cantidad de luz reflejada por el campo. Las zonas blancas tienen un valor bajo, las zonas negras un valor alto. Si el sensor está situado en el borde de la línea, el valor leído por el sensor se encuentra entre los dos valores mínimo y máximo.

Dado que cada sensor óptico de color tiene valores mínimos y máximos ligeramente diferentes, resulta ventajoso calibrar cada sensor individualmente. Por ejemplo, se puede realizar una rotación de 360 grados y determinar y guardar los valores mínimo y máximo de cada sensor a partir de todos los valores medidos. A continuación, los valores calibrados pueden utilizarse en el cálculo.

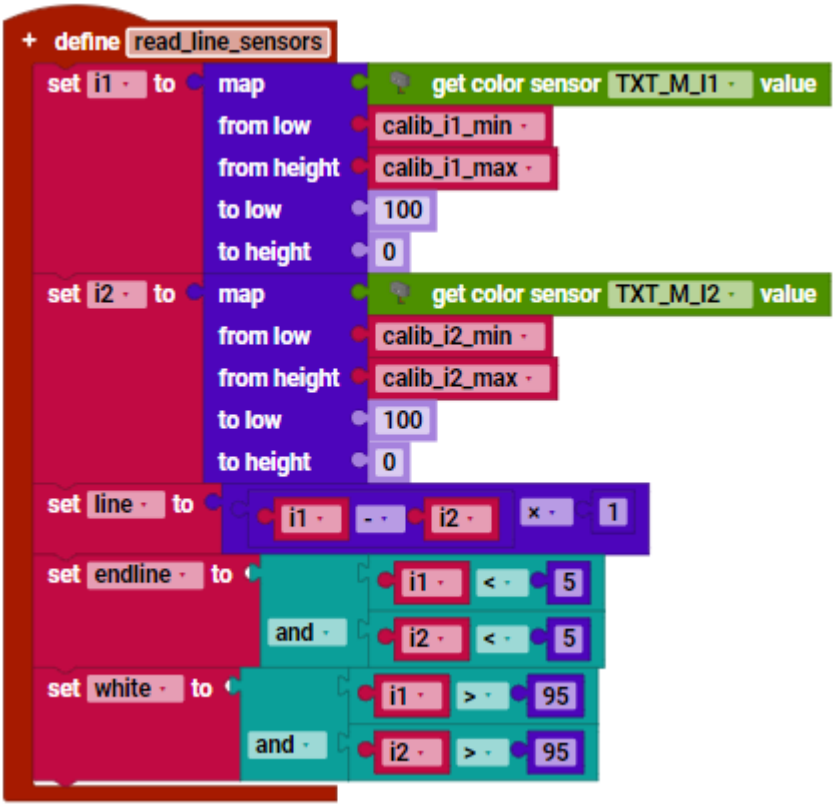
Si el robot se coloca en una línea y luego se ejecuta la función **calib_line_rotate360**, se calibran los valores mínimo y máximo por sensor y se guardan en las variables correspondientes:



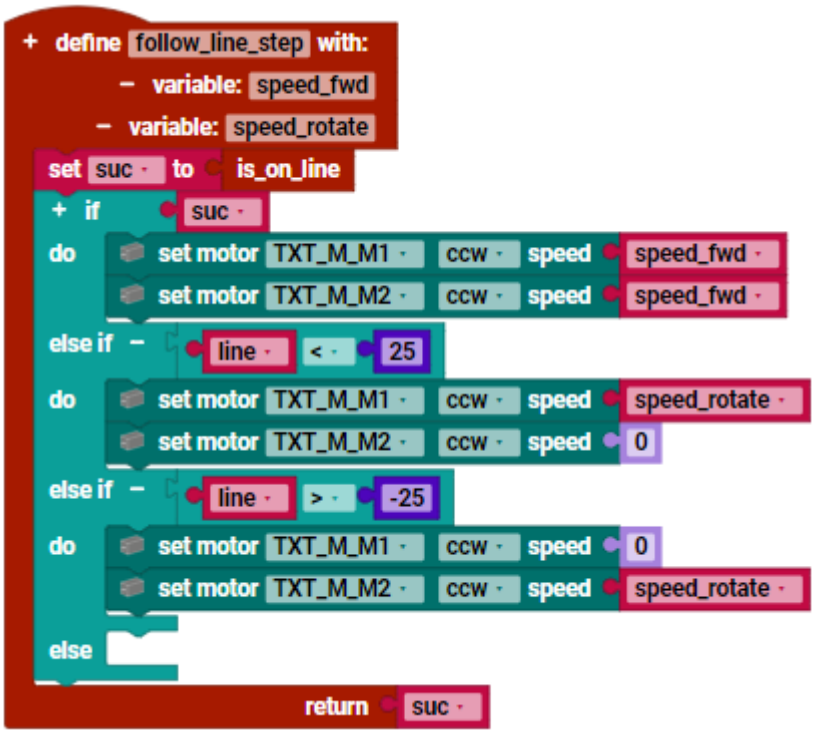
Los valores calibrados con esta función pueden tener, por ejemplo, el siguiente aspecto:

calibLine min:173 171 max:1559 1485

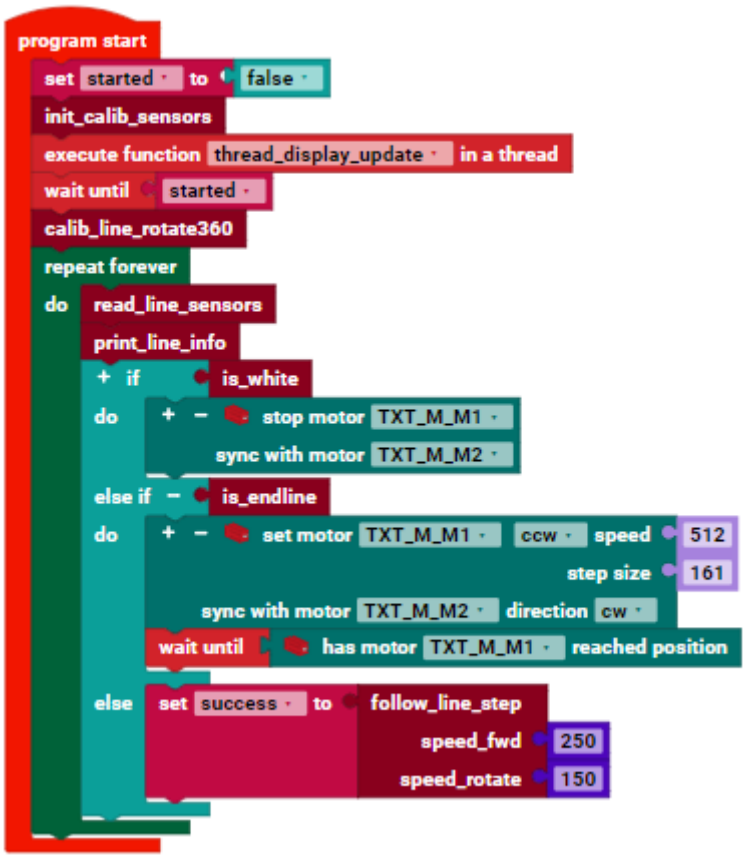
En la función **read_line_sensors** para la lectura de los sensores, los valores medidos se normalizan para que la variable de **línea** tenga valores entre -100 ... 0 ... 100. Además, se establecen varias variables booleanas para indicar si se ha reconocido un final de línea o si se ha detectado una zona blanca. Estos estados se pueden utilizar para realizar un giro de 180 grados al final de la línea o para detener el seguidor de línea cuando el robot pasa por la zona blanca.



A continuación, se llama a la función **seguir_línea_paso** en un bucle infinito, que controla los motores en función de la variable calculada línea. El robot avanza un paso en línea recta, hacia la izquierda o hacia la derecha. Esto ocurre a intervalos muy cortos.



El bucle principal del programa puede tener, por ejemplo, el siguiente aspecto:



Si la velocidad de los motores se ajusta demasiado alta, el robot perderá la línea muy rápidamente. Si la velocidad de los motores se ajusta demasiado baja, la potencia de los motores no será suficiente cuando la carga de la batería sea baja o el robot se moverá muy lentamente a lo largo de la línea.

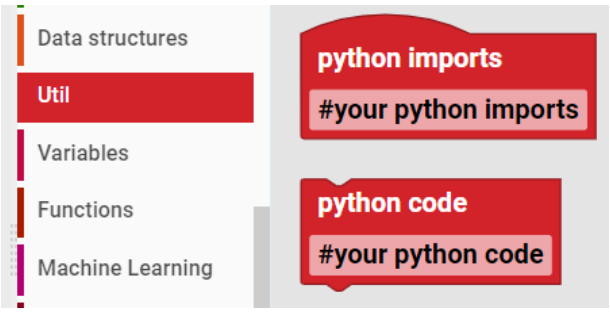
Con un seguidor de línea se pueden concebir muchos métodos diferentes. Por ejemplo, se puede seguir un borde de línea en lugar de la línea. En este caso, la función **follow_line_step** debe adaptarse en consecuencia.

El seguidor de línea puede mejorarse aún más con un controlador PID para que el robot no pierda la línea incluso a una velocidad muy alta. La implementación de dicho controlador PID requiere conocimientos de tecnología de control, que no se tratarán más en este tutorial.

★★ Código Python con Blockly

La mayoría de las funciones en ROBO Pro Coding pueden ser implementadas con Blockly. Hay muchos bloques para diferentes casos de uso. Es muy conveniente usar los bloques porque las instrucciones requeridas en Python se generan automáticamente con ROBO Pro Coding.

Sin embargo, a veces es necesario integrar comandos Python en programas Blockly porque el bloque correspondiente aún no existe o porque la secuencia de comandos Python debe definirse de forma diferente. Esto es muy fácil de hacer en un programa Blockly creando los comandos Python necesarios con los siguientes bloques Python:

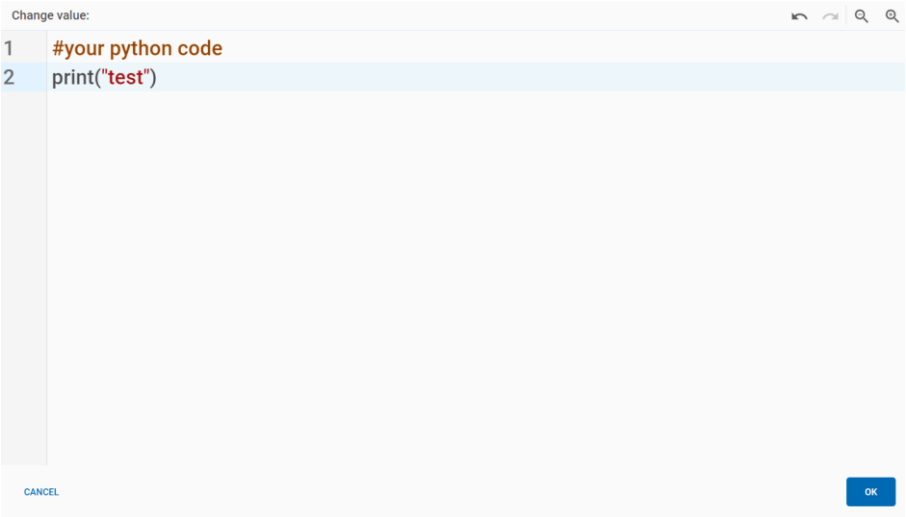


El bloque "**python imports**" puede utilizarse para importar módulos específicos de Python. El generador blockly coloca el contenido al principio del archivo.

El bloque "**código python**" puede utilizarse para definir código Python en cualquier parte del programa Blockly.

Al hacer clic en el bloque, se abre un editor de Python. El código Python también puede definirse en varias líneas.

Debe prestarse especial atención a la sangría del código fuente, ya que en Python la sangría mediante espacios o tabuladores influye notablemente en la estructura del programa

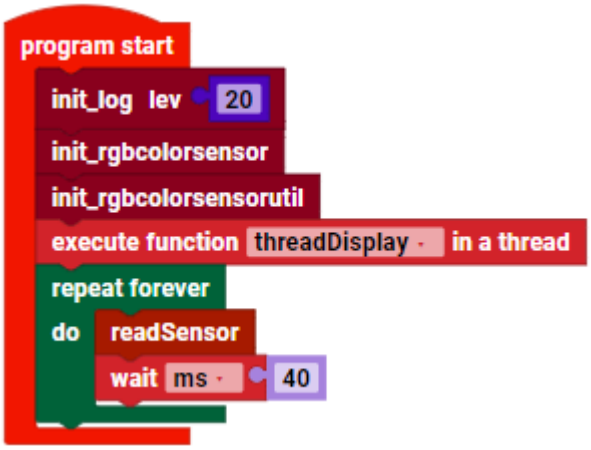


★★ Sensor de color RGB como biblioteca [RoboMission_rgbw_hsv]

El sensor de color RGB se conecta al bus I2C del controlador TXT 4.0. Para ello, debe conectarse un sensor de color RGB a la conexión EXT1 o EXT2 del controlador TXT 4.0 mediante un cable plano.

A diferencia de los sensores ópticos de color, que se conectan a las entradas universales del controlador TXT 4.0, el sensor de color RGB es muy adecuado para detectar obstáculos de distintos colores si se desea detectarlos a mayor distancia. También se pueden utilizar varios sensores con el sensor de color RGB, que se conectan en cadena al sensor principal.

El programa de ejemplo **RoboMission_rgbw_hsv** muestra cómo se pueden leer los valores de los sensores.



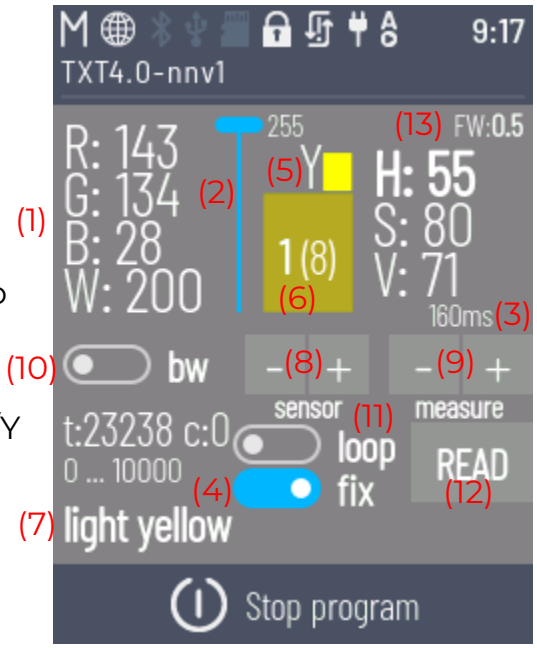
El módulo de **registro** contiene funciones para el registro, mediante las cuales se puede cambiar el nivel de registro.

El módulo **rgbcolorsensor** contiene todas las funciones básicas para la lectura de los registros del sensor de color RGB. Aquí se implementan todas las funciones necesarias para la comunicación con el sensor de color RGB.

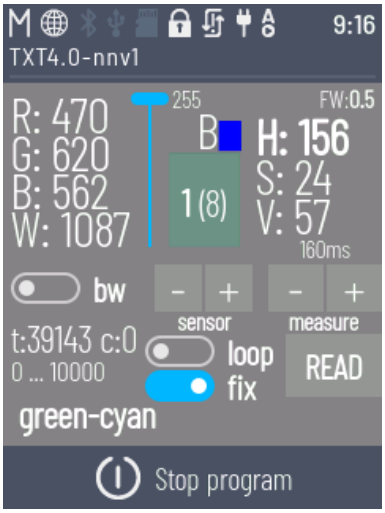
El módulo **rgbcolorsensorutil** contiene otras funciones, por ejemplo, para convertir valores de color del modelo RGBW (Rojo, Verde, Azul, Blanco) al modelo HSV (Tono, Saturación, Valor) o como valores HEX. También se han implementado ya algunas distinciones de casos para los colores estándar.

Los valores del sensor con el índice 0 (sensor principal) pueden leerse, por ejemplo, de la siguiente manera:

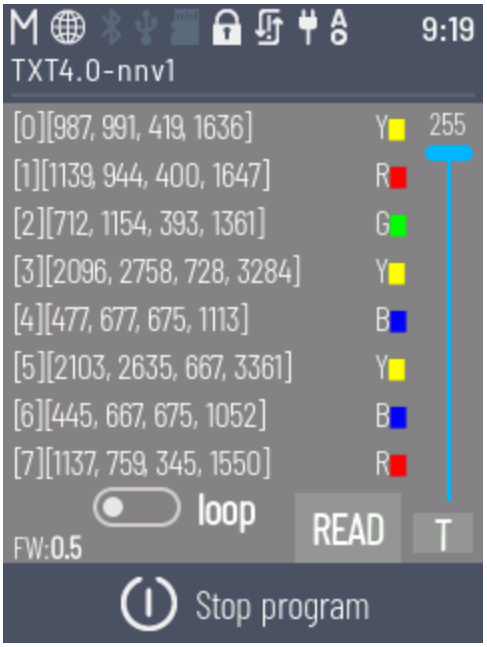
Otro programa de ejemplo **test_rgbcolorsensor_smbus** muestra cómo se puede utilizar la interfaz completa del sensor de color RGB. Varias interfaces adicionales se pueden encontrar en la captura de pantalla, ver captura de pantalla.



(1) Valores brutos del sensor RGBW: Rojo/Verde/Azul/Blanco
(2) Brillo de los LED ajustable 0...255
(3) Valores HSV calculados: Tono/Saturación/Valor
(4) fijo: Valor de referencia para el cálculo de HSV, ya sea fijo (10000) o adaptativo con valores medidos mín...máx. del historial.
(5) Color de bloque reconocido R/G/B/Y
(6) Color reconocido como valor HEX
(7) Reconocimiento general del color como texto
(8) +Sensor 1 (de 8 sensores reconocidos), se selecciona mediante los botones (-) y ( )
(9) Tiempo de medición "160 ms", seleccionado mediante los botones (-) y (+)
(10) bw: Interruptor para el modo blanco/negro
(11) bucle: medición sin fin, si está activado
(12) Iniciar medición individual con "READ" si bucle desactivado
(13) Versión de firmware FW: 0.5 del sensor conectado



El ejemplo **test_rgbcolorsensors_smbus** muestra cómo se pueden leer varios sensores de color RGB (índice 0-7) y mostrar simultáneamente los valores de todos los sensores en la pantalla.



★★ Medición de la tensión de la batería [test_battery]

El estado de carga de la batería conectada al controlador TXT 4.0 puede influir en la corriente aplicada a los motores y, por tanto, afectar al comportamiento del robot. Para poder reaccionar ante una tensión baja, se puede leer la tensión en voltios a través de una entrada de tensión interna del controlador TXT 4.0.

En el ejemplo de **test_battery**, el voltaje y, por tanto, el estado aproximado de carga puede leerse utilizando una función de Python.

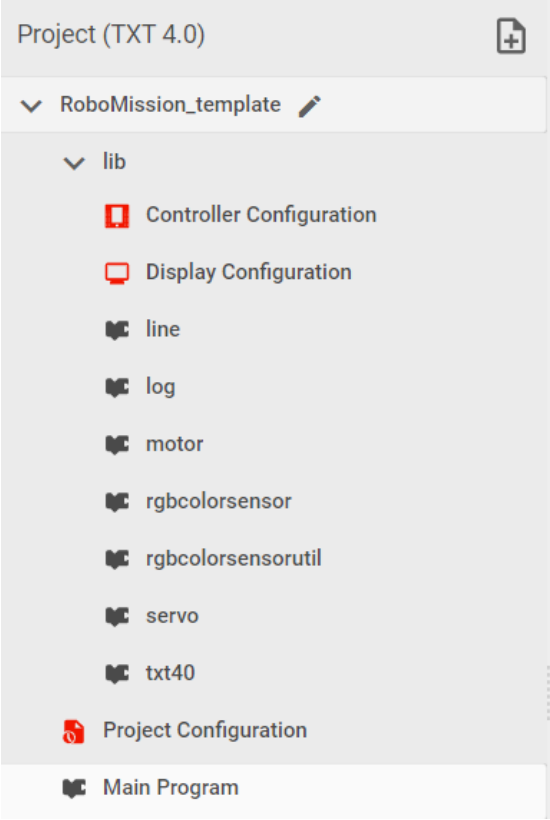
★★ Plantilla para proyectos propios [RoboMission_template]

El ejemplo **RoboMission_template** puede utilizarse para crear su propio proyecto

Importe este proyecto y modifíquelo según sus propias necesidades.

Las funciones utilizadas en este tutorial se incluyen como módulos y pueden utilizarse en el programa principal o en otros módulos de nueva creación.

Cree su propia plantilla y utilícela para sus propios proyectos.



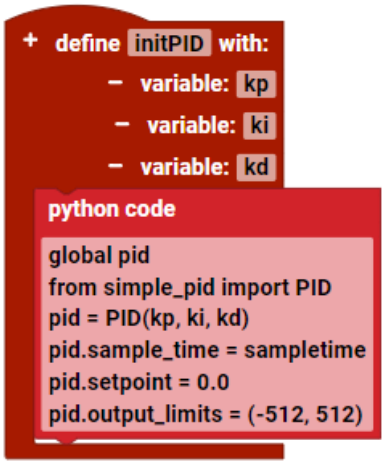
★★★ Seguidor de línea con controlador PID [RoboMission_linepid]

El ejemplo **RoboMission_linepid** muestra cómo puede realizarse un controlador PID con el seguidor de línea.

En Python, por ejemplo, existe la biblioteca **simple-pid** para este fin**.** La versión 1.0.0 de esta biblioteca externa ya está incluida en el firmware del controlador TXT 4.0 3.1.8 y puede utilizarse directamente. La documentación de simple-pid se encuentra en el siguiente enlace:

https://github.com/m-lundberg/simple-pid/tree/v1.0.0

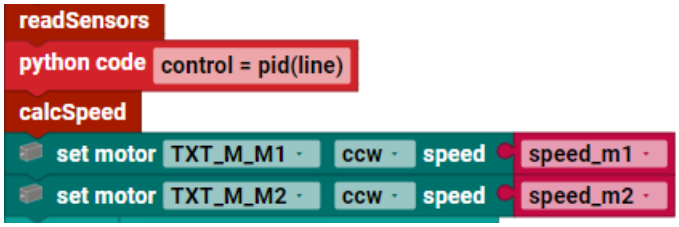
En primer lugar, se inicializa el regulador PID, para lo cual deben ajustarse los parámetros **Kp**, **Ki** y **Kd**. El ajuste de las constantes debe determinarse experimentalmente y puede llevar mucho tiempo. Para conocer los fundamentos del PID y la selección correcta de los parámetros, consulte la bibliografía externa correspondiente.



Significado de los parámetros:

- **Comportamiento proporcional** Kp
- **Comportamiento integral** Ki
- **Comportamiento diferencial** Kd

Las siguientes funciones se llaman en cada paso del bucle de control:

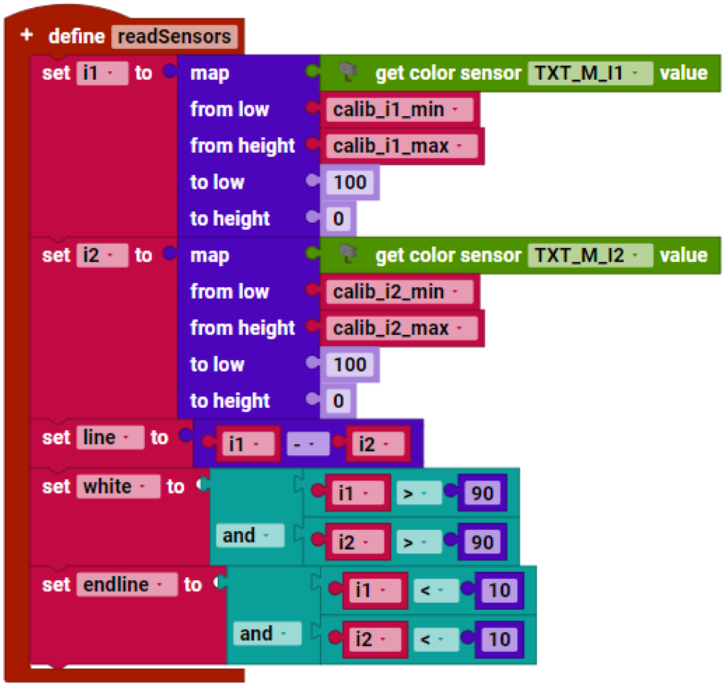


El primer paso consiste en leer los sensores.

El cálculo se realiza entonces en el regulador PID. La **línea** variable (-100 ... 0 ... 100) es la variable de control. La consigna se encuentra en el centro de la línea en el valor 0.

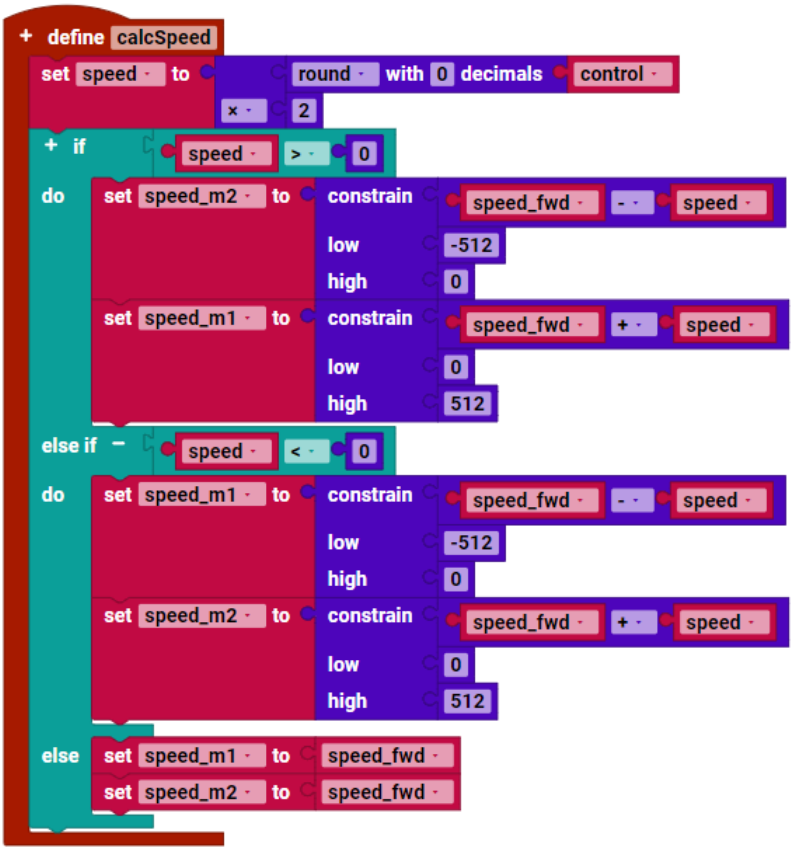
A continuación se calculan y ajustan las velocidades de los dos motores. Tenga en cuenta que la velocidad de los motores debe limitarse a valores comprendidos entre -512 y 512.

Los valores de los sensores se en la función **readSensors** y se normalizan a los valores calibrados. Para ello, al inicio del programa se llama una vez a la rutina de calibración, en la que se fijan los valores mín. y máx. Si los mín. y máx. no cambian, los valores pueden ajustarse directamente en el programa durante la inicialización. En este caso, no es necesario llamar a la función de calibrado.



La velocidad se calcula en la función **calcSpeed**.

Experimenta con diferentes parámetros Kp, Ki y Kd. Prueba también a cambiar la velocidad de los motores. Si los valores son correctos, el robot debería seguir la línea con mayor precisión.



Outlook

Además del set STEM Coding RoboMission, fischertechnik ofrece muchos otros componentes adicionales y piezas individuales que también pueden utilizarse.

Echa un vistazo a la página web de fischertechnik https://www.fischertechnik.de/es-es.