Nombre: _____ Clase: ___ Fecha: _____

Tarea n.º 1

Vehículo Mecanum con detección de obstáculos

Tarea de construcción

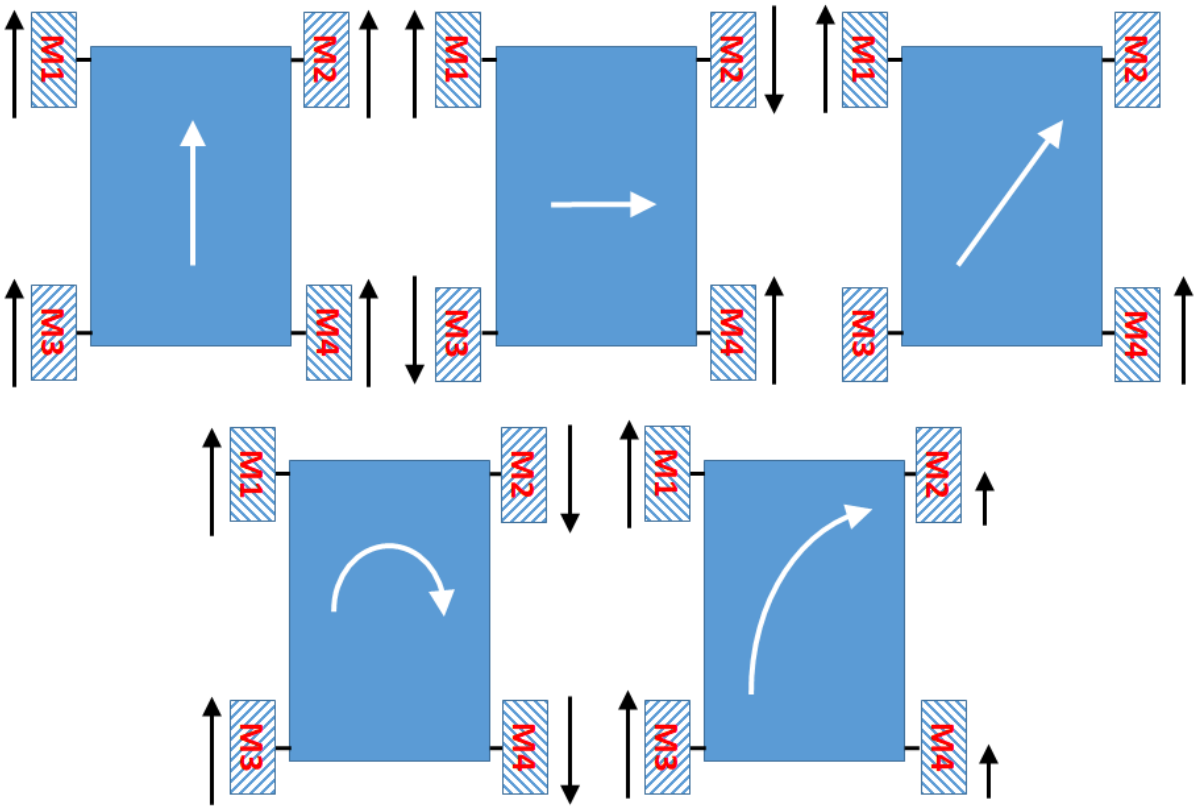
Monta el modelo básico de vehículo Mecanum con sensores como indica el manual de instrucciones. Los cubos de las ruedas dentadas Z20 y de las ruedas Mecanum deben estar apretados para que no se produzca un «deslizamiento» al conducir.

En esta tarea, el sensor de pistas (frontal) y el sensor de distancia por ultrasonido se utilizan para detectar obstáculos.

El sensor de distancia por ultrasonido se conecta a I6 (cable negro), los dos sensores infrarrojos se conectan a I7 (sensor derecho, cable amarillo/azul) e I8 (sensor izquierdo, cable azul). Los sensores infrarrojos y el sensor ultrasónico, además, deben alimentarse a través de la salida de tensión de 9 V del TXT.

Comprueba a través de la prueba de interfaz si los cuatro motores están conectados correctamente (hacia adelante: rotación izquierda, es decir, en sentido contrario a las agujas del reloj), los cuatro codificadores están conectados a las entradas correctas del contador (M1: C1, M2: C2, M3: C3 y M4: C4) y los dos sensores infrarrojos emiten los valores correctos (superficie blanca: 1, superficie negra: 0).

Tareas de programación



Tipos de movimiento más relevantes con las ruedas Mecanum

1. Accionamiento sincrónico en todas las direcciones

La figura ilustra el control del vehículo Mecanum. Los tres tipos de movimiento superiores muestran la tracción de las ruedas necesaria para el desplazamiento en línea recta, en diagonal y lateral. Cada uno de los tres tipos de movimiento incluye el movimiento hacia adelante y hacia atrás o el movimiento hacia la izquierda y hacia la derecha. Si la flecha negra junto a la rueda apunta hacia arriba, la rueda respectiva debe girar hacia adelante.

Así se pueden distinguir los siguientes ocho movimientos del vehículo Mecanum (véase también la animación en [1]):

- Hacia adelante
- Hacia atrás
- Lateral hacia la izquierda
- Lateral hacia la derecha
- Diagonal hacia la izquierda, adelante

- Diagonal hacia la derecha, adelante
- Diagonal hacia la izquierda, atrás
- Diagonal hacia la derecha, atrás

Desarrolla un subprograma Blockly (una función) para cada uno de estos movimientos al que puedas remitir la velocidad de los motores como parámetros.

Las funciones siempre serán necesarias en las siguientes tareas. Los programas de control serán así más claros y fáciles de entender. Prueba tus funciones con programas de ejemplo sencillos.

Observa: Para las ruedas Mecanum, la sincronización de los motores es especialmente importante (véase también la tarea n.º 6 del Robotics TXT 4.0 Base Set).

2. Giro sincrónico

Los dos tipos de movimiento inferiores de la figura muestran los dos movimientos de giro más importantes que pueden realizarse con las ruedas Mecanum: el giro sobre el propio eje y la conducción en una curva. También se incluyen varias direcciones de movimiento en cada uno, como girar hacia la izquierda y hacia la derecha o doblar a la izquierda o a la derecha, así como conducir en curvas hacia adelante y hacia atrás.

En total, se producen otros seis movimientos:

- Girar sobre el propio eje hacia la derecha (en el sentido de las agujas del reloj)
- Girar sobre el propio eje hacia la izquierda (en el sentido contrario a las agujas del reloj)
- Doblar a la derecha
- Doblar a la izquierda
- Doblar hacia atrás a la derecha
- Doblar hacia atrás a la izquierda

Desarrolla un subprograma Blockly (una función) para cada uno de estos movimientos al que remitas la velocidad de los motores como parámetros. Prueba tus funciones con programas de ejemplo sencillos.

3. Accionamiento sincrónico con valor predeterminado de velocidad

Para navegar el vehículo Mecanum con precisión ahora necesitas una parte del conjunto de comandos de las tareas de programación n.º 1 y 2, añadiendo respectivamente un valor predeterminado de distancia – el número de impulsos con los que deben girar los motores.

3a. Añade a cada una de las siguientes seis funciones de movimiento de las tareas de programación n.º 1 y 2 un valor predeterminado de distancia: un número fijo de impulsos

- para el desplazamiento en línea recta (hacia adelante/hacia atrás),
- para el desplazamiento lateral (hacia la izquierda/hacia la derecha) y
- para girar sobre el propio eje (hacia la izquierda: en el sentido de las agujas del reloj, hacia la derecha: en el sentido contrario a las agujas del reloj).

3b. Prueba las seis funciones en un trayecto de prueba previamente medido y, a través de la experimentación, determina los factores (impulsos por cm o impulsos por °) para calcular los impulsos necesarios de

- una distancia en cm para el desplazamiento en línea recta
- una distancia en cm para el desplazamiento lateral y
- un ángulo de giro determinado en °.

Para el factor de conversión del desplazamiento en línea recta puedes (de forma análoga a la tarea n.º 6 del Robotics TXT 4.0 Base Set) medir la circunferencia de una rueda Mecanum como una primera aproximación y derivar el factor a partir de esta medición.

4. Detección de líneas

Con el sensor de pista, el vehículo Mecanum ahora también debe identificar las líneas límite y los bordes y esquivarlos: Si se detecta una línea límite negra o un abismo, el vehículo debe retroceder 10 cm, apartarse del borde un octavo de giro (45°) y continuar el recorrido.

4a. Dibuja el correspondiente diagrama de transición de estados.

4b. Ahora diseña un programa Blockly adecuado utilizando las funciones de navegación de las tareas de programación n.º 1 y 2 (véase también la tarea n.º 6 del Robotics TXT 4.0 Base Set).

Tareas experimentales

1. Detección de obstáculos por ultrasonidos

El vehículo Mecanum está equipado con un sensor ultrasónico que proporciona la distancia a un objeto en cm (véase también la tarea n.º 1 del Robotics TXT 4.0 Base Set).

Diseña un programa Blockly que impida que el vehículo se acerque a menos de 15 cm de un obstáculo. Si detecta un obstáculo, debe esquivarlo desplazándose lateralmente hasta que el sensor ultrasónico no detecte ningún obstáculo a menos de 25 cm de distancia (véase también la tarea n.º 6 del Robotics TXT 4.0 Base Set), y después continuar con su recorrido.

2. Navegación por codificador

También puede determinar la distancia recorrida por el vehículo Mecanum a partir de los valores del codificador. Así, debes utilizarlo para navegar hacia un destino determinado para el que se ha establecido la distancia en línea recta en cm. Para las pruebas, marca un punto de destino a 3 m (en línea recta) en el suelo.

Ahora bloquea el camino directo al destino primero con uno y después con varios obstáculos. Piensa en una estrategia para que el vehículo esquive los obstáculos y continúe el recorrido hasta el punto de destino predeterminado por el trayecto más corto.

2a. Ilustra tu estrategia en un esquema.

2b. Amplía el programa Blockly de la tarea experimental n.º 1 como corresponde.

Consejo: El giro del vehículo Mecanum se realiza sobre el centro del vehículo. Por lo tanto, todos los cálculos del trayecto se refieren siempre al centro del vehículo. Si el recorrido debe comenzar y finalizar en una línea delante del parachoques, al final, el vehículo debe volver a girar en la dirección de desplazamiento para detenerse frente a la línea de meta.

Anexos
Vehículo Mecanum con detección de obstáculos

Material necesario

- Ordenador para el desarrollo de programas, localmente o a través de la interfaz web.
- Cable USB o conexión BLE o wifi para transferir el programa al TXT4.0.
- Hoja de ruta con línea circular negra cerrada de 2 cm de ancho (del Robotics TXT 4.0 Base Set).

Más información

[1] FRC Team 2605 (Bellingham, Washington): *How a Mecanum Drive Works*. github.io

[2] Editor de diagramas en línea para crear diagramas de transición de estados (formato drawio): https://www.diagrammeditor.de/

Tarea n.º 1: Vehículo Mecanum con detección de obstáculos

En esta tarea de iniciación, las alumnas y los alumnos serán introducidos en la navegación de un vehículo con ruedas Mecanum. En ella se desarrollarán funciones para abstraer los movimientos. El vehículo aprende a moverse en el espacio y a esquivar obstáculos y líneas límite.

Las ruedas omnidireccionales son aquellas con las que un vehículo puede moverse en cualquier momento en cualquier dirección. Con la rueda Mecanum, los rodillos forman un ángulo con respecto al eje principal y permiten así maniobras de conducción omnidireccionales.

Tema

Control de un vehículo con ruedas Mecanum y detección de obstáculos.

Objetivos de aprendizaje

- Comprender el funcionamiento de las ruedas Mecanum y su control
- Programación clara mediante variables de estado y funciones
- Detección de líneas y obstáculos a través de sensores (infrarrojo, ultrasonido)
- Navegación por impulsos del motor

Tiempo necesario

Para montar el modelo básico de vehículo Mecanum con sensores siguiendo el manual de instrucciones, las alumnas y los alumnos necesitan 45-90 minutos, dependiendo de la experiencia previa (una o dos clases).

Para el desarrollo del programa para resolver las tareas de programación, las alumnas y los alumnos, siempre y cuando tengan experiencia con el Robotics TXT 4.0 Base Set, necesitan dos o tres clases (90-135 minutos).

La realización de las tareas experimentales requiere otros 90-135 minutos.

Anexos
Tarea n.º 1: Vehículo Mecanum con detección de obstáculos

Material necesario

- Ordenador para el desarrollo de programas, localmente o a través de la interfaz web.
- Cable USB o conexión BLE o wifi para transferir el programa al TXT4.0.
- Hoja de ruta con línea circular negra cerrada de 2 cm de ancho (del Robotics TXT 4.0 Base Set).

Más información

[1] FRC Team 2605 (Bellingham, Washington): *How a Mecanum Drive Works*. github.io

[2] Editor de diagramas en línea para crear diagramas de transición de estados (formato drawio): https://www.diagrammeditor.de/

Nombre: _____ Clase: ___ Fecha: _____

Hoja de soluciones de la tarea temática n.º 1

Vehículo Mecanum con detección de obstáculos

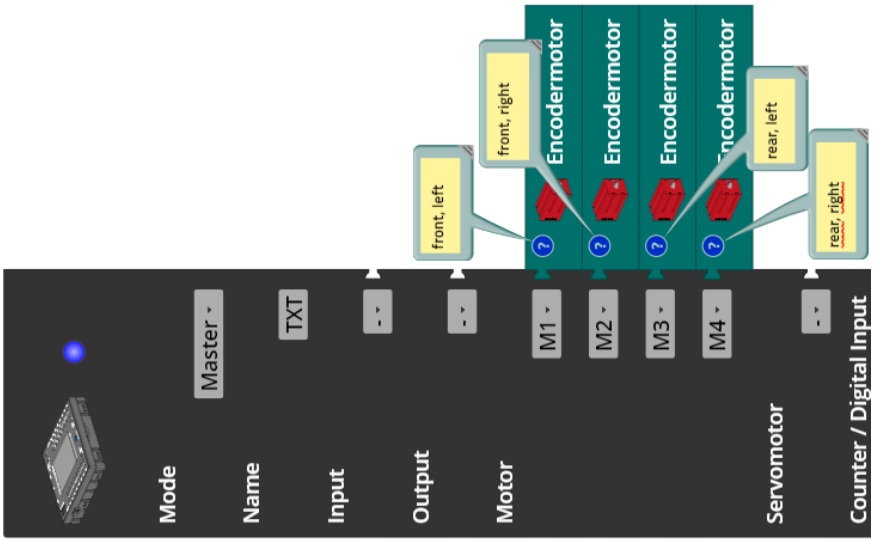
Las funciones de control desarrolladas en la primera tarea de programación constituyen la base de todas las tareas posteriores.

El uso de los sensores (sensor de pista, sensor ultrasónico) se asemeja al de la tarea n.º 6 del Robotics TXT 4.0 Base Set. Gracias a las opciones de movimiento más flexibles de las ruedas Mecanum surgen soluciones adicionales y especialmente más sencillas.

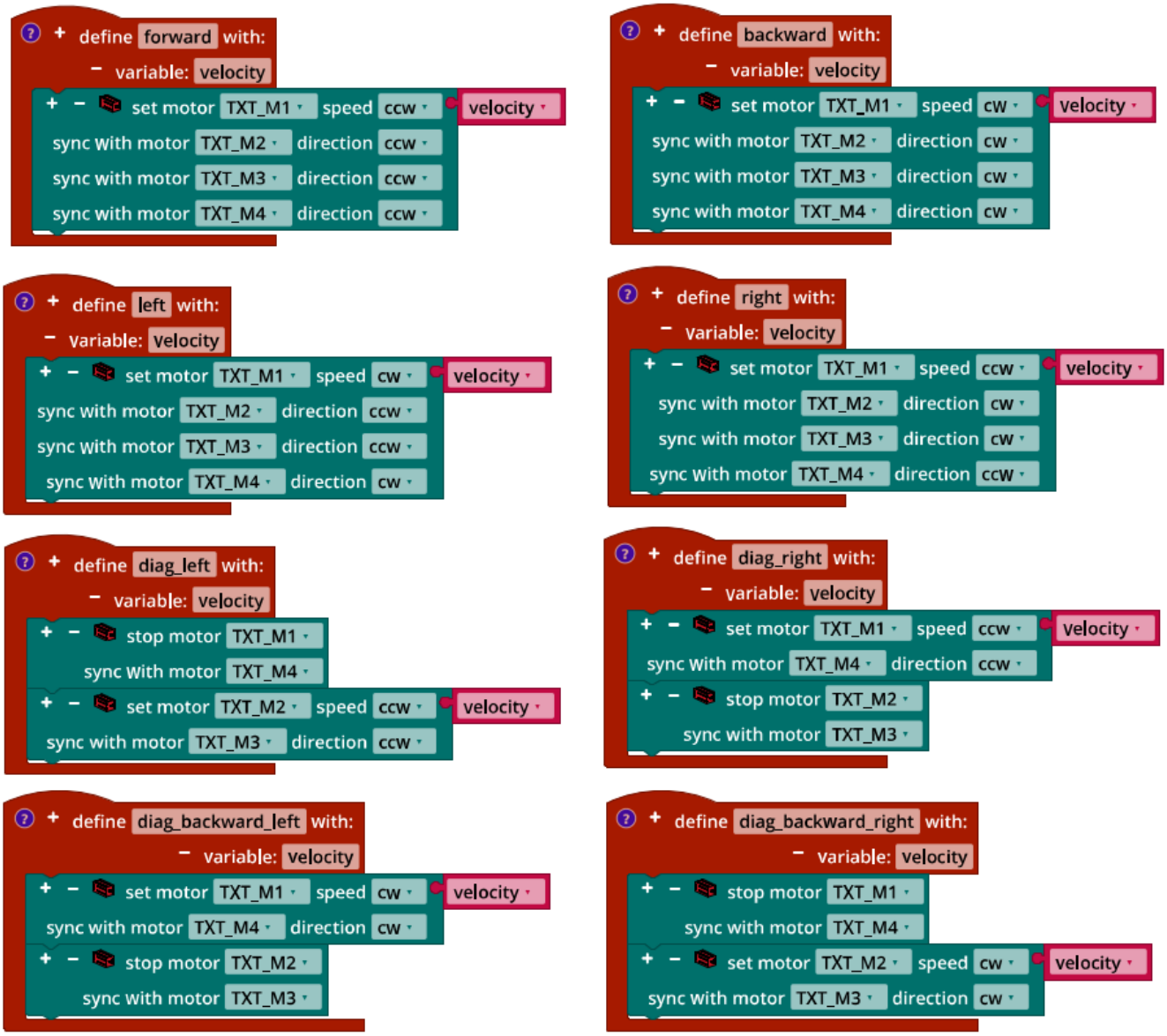
Tareas de programación

1. Accionamiento sincrónico en todas las direcciones

Configuración de los sensores:



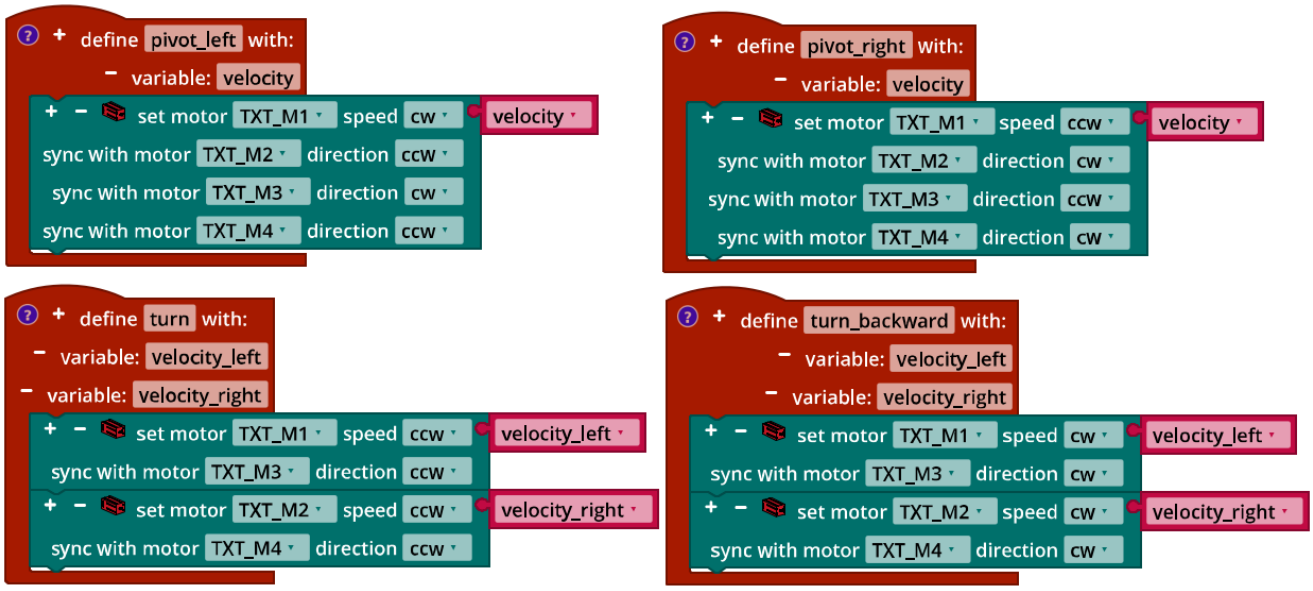
Funciones de control (ejemplo):



Mecanum_Synchronous_Driving_Funktions.ft

2. Giro sincrónico

Funciones de control (ejemplo):

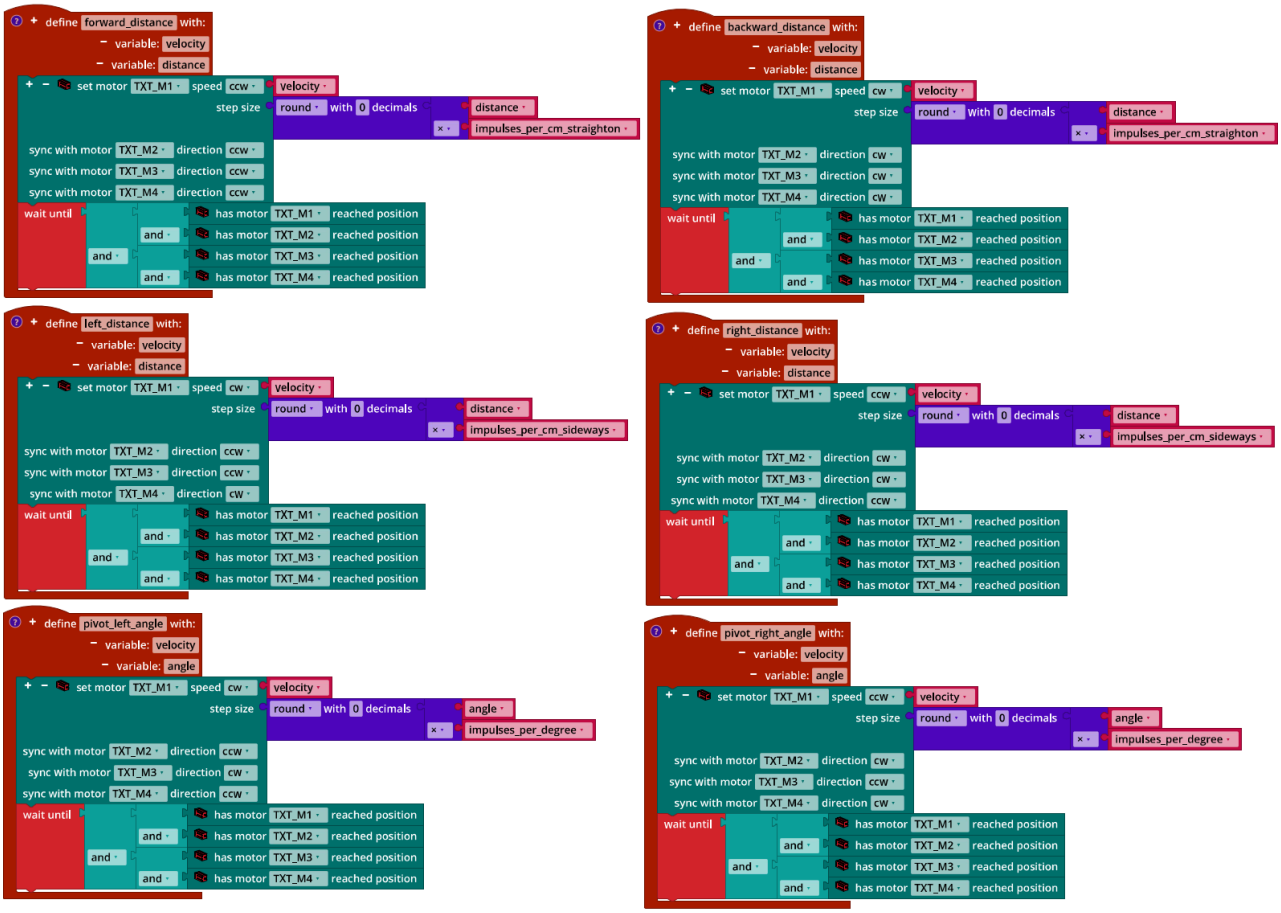


Mecanum_Synchronous_Turning_Functions.ft

3. Accionamiento sincrónico con valor predeterminado de velocidad

3a. La conversión de la distancia (en cm) en impulsos puede realizarse en la función o al solicitarla.

Funciones del programa (ejemplo):



Mecanum_Synchronous_Navigation_Distance.ft

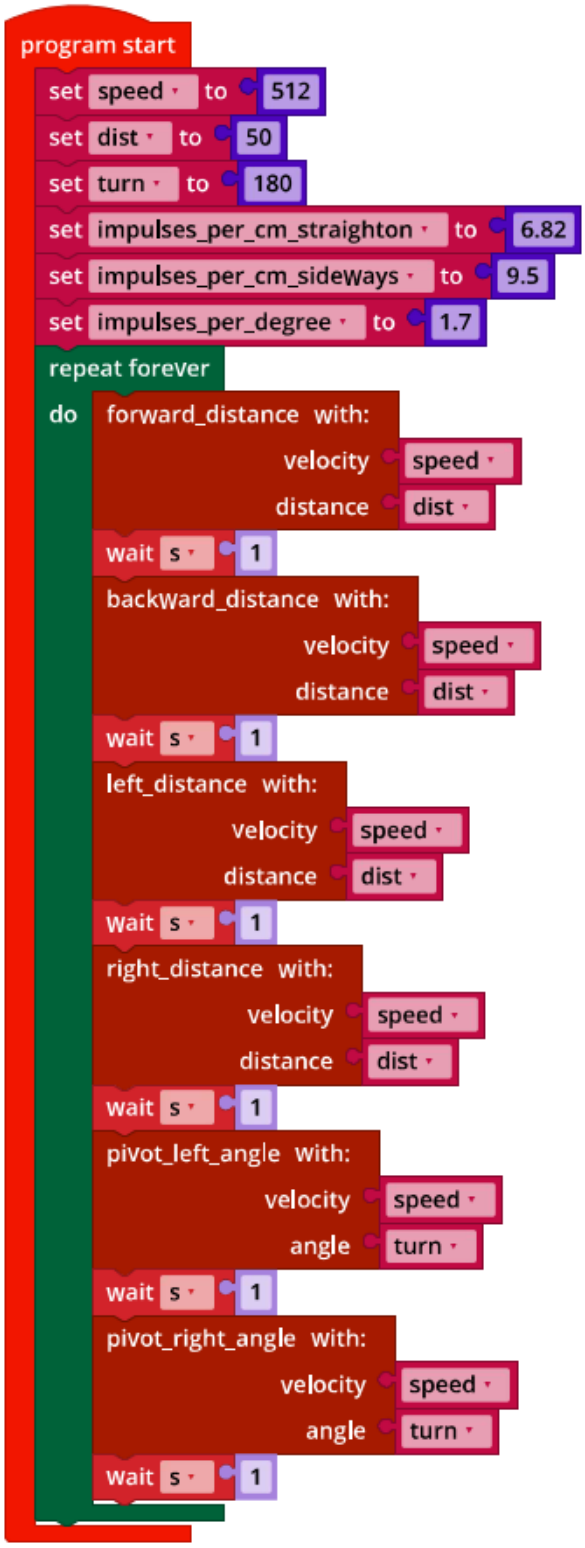
3b. La circunferencia de una rueda Mecanum es de aproximadamente 19 cm. A partir de ella se puede calcular fácilmente el número de impulsos por vuelta (al igual que en la tarea n.º 6 del Robotics TXT 4.0 Base Set): Como las ruedas son accionadas por los motores con una transmisión reductora de 1:2, son aproximadamente $\frac{63{,}9 \cdot 2}{19} \approx$ **6,726 impulsos/cm**.

Las pruebas realizadas en distancias de varios metros muestran que el valor debe corregirse a unos **6,82 impulsos/cm**.

En los casos de desplazamiento lateral y giro, solo las tareas experimentales contribuyen a determinar el factor de conversión (véase el programa siguiente). A

partir de pruebas pudo determinarse el valor para el desplazamiento lateral en aproximadamente **9,5 impulsos/cm** y el valor para el giro en **1,7 impulsos/grado**.

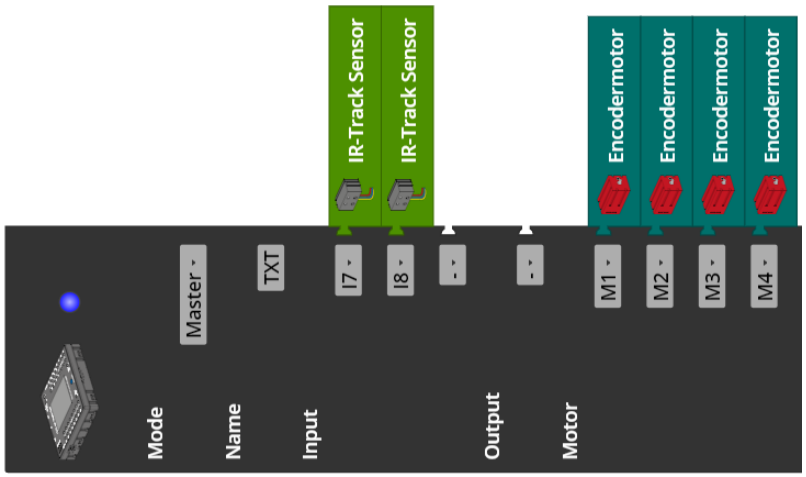
Programa (ejemplo) para probar los factores de conversión:



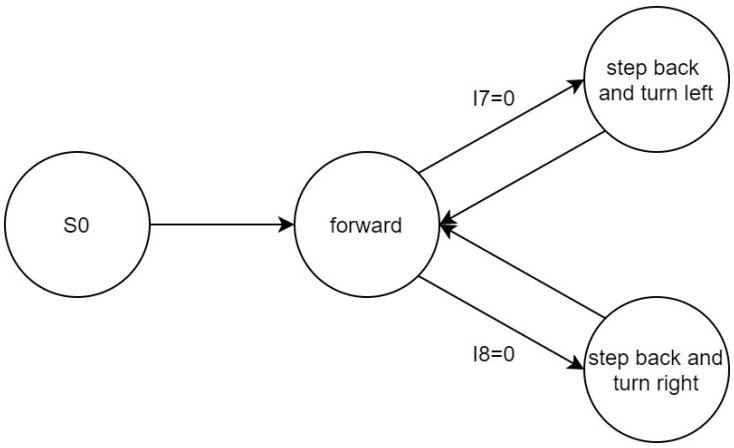
Mecanum_Synchronous_Navigation_Distance.ft

4. Detección de líneas

Configuración de sensores y actuadores:

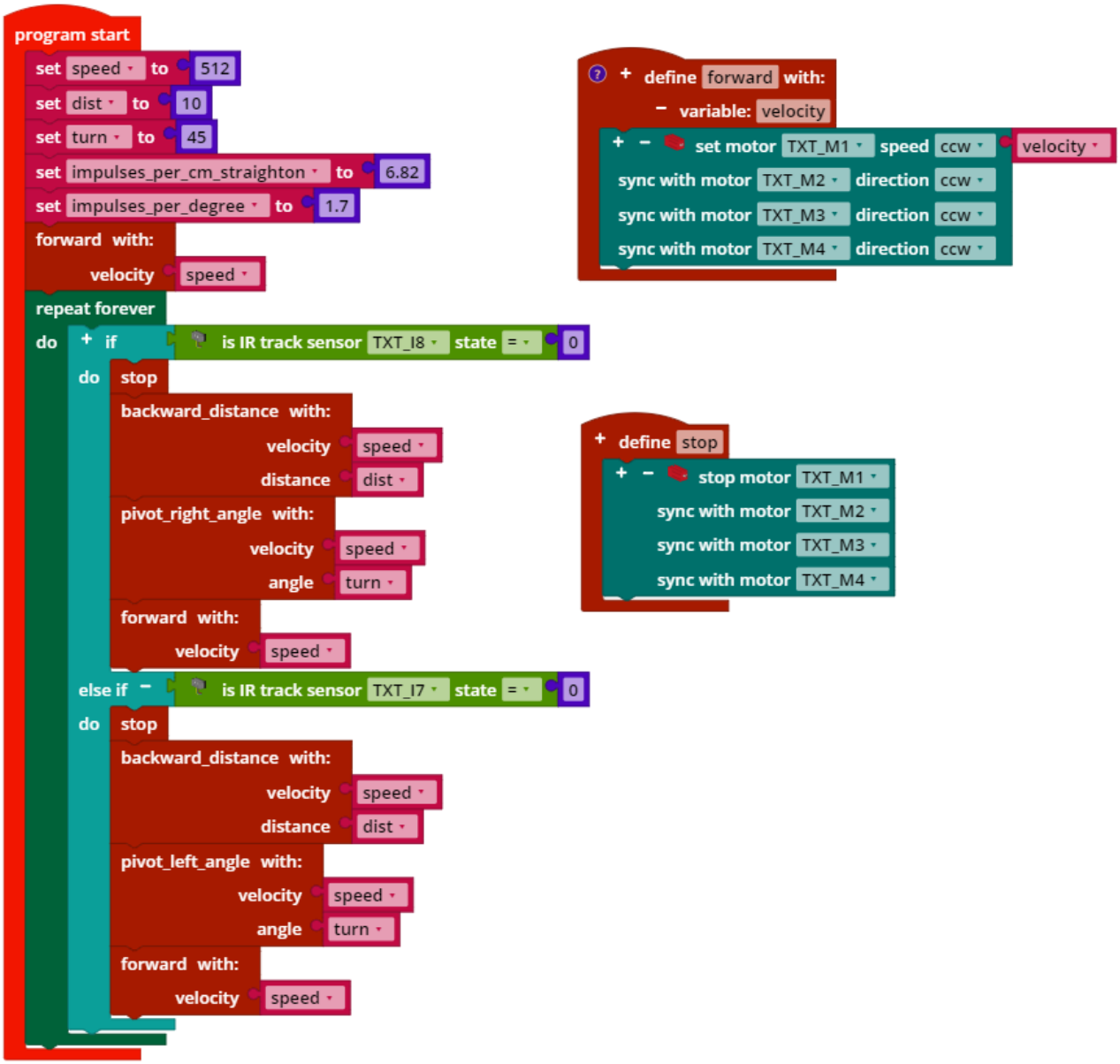


4a. Diagrama de transición de estados:



State-Transition_Diagram_Mecanum_with_IR_Sensors.drawio

4b. Extracto del programa (ejemplo):

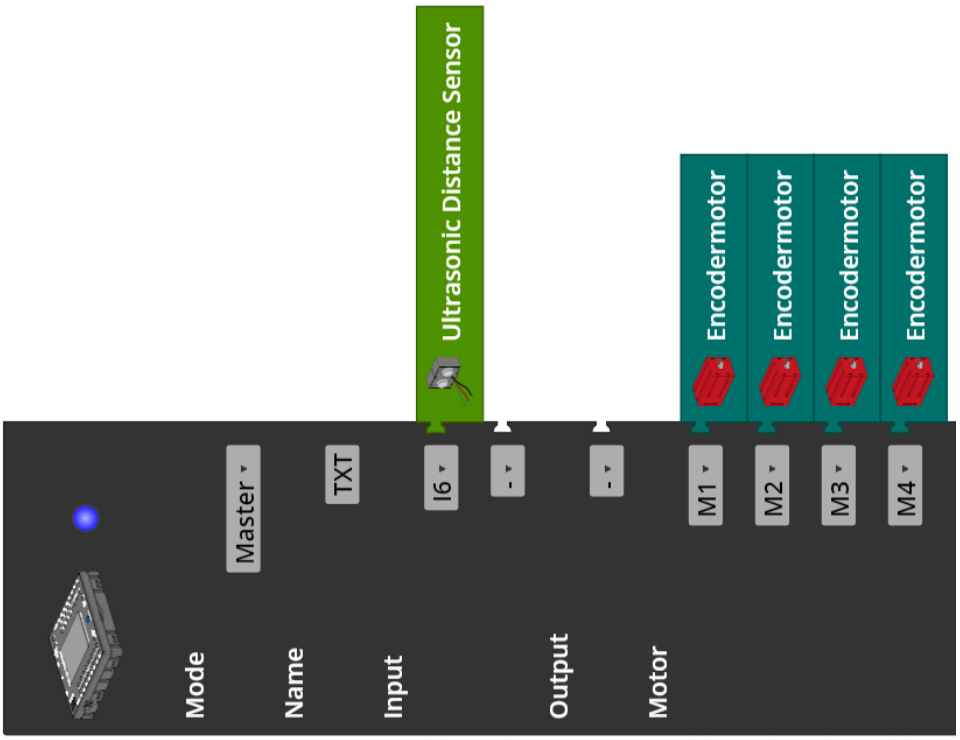


Mecanum_Boundary_Line.ft

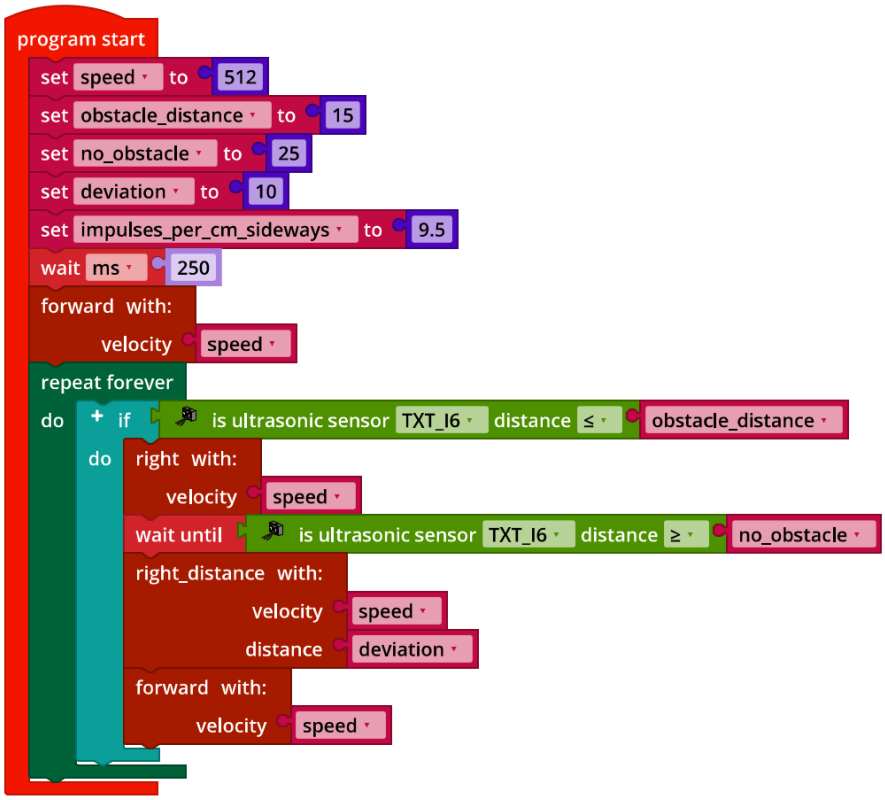
Tareas experimentales

1. Detección de obstáculos por ultrasonidos

Configuración de sensores y actuadores:



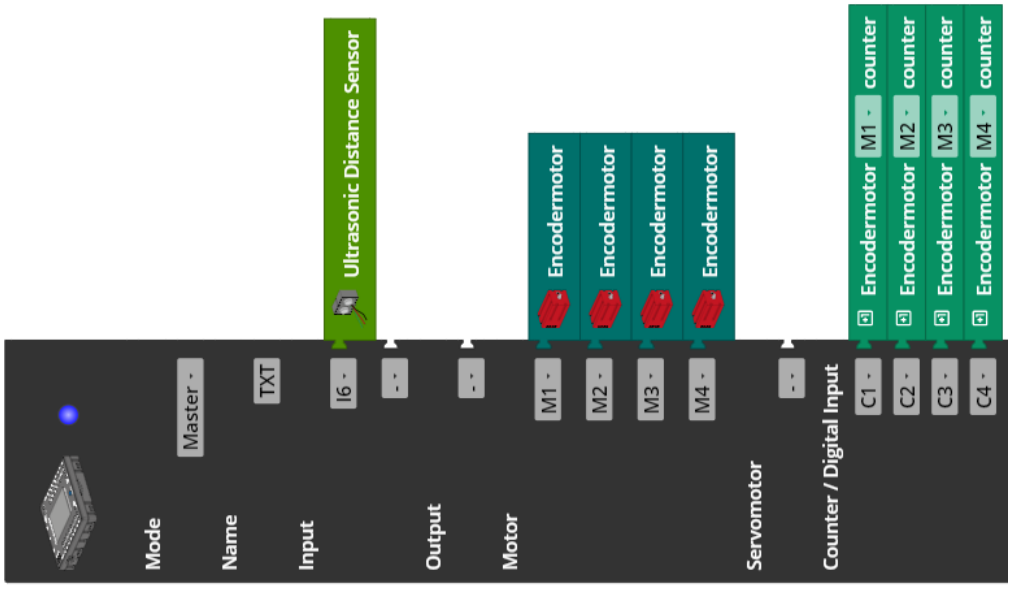
Extracto del programa (ejemplo):



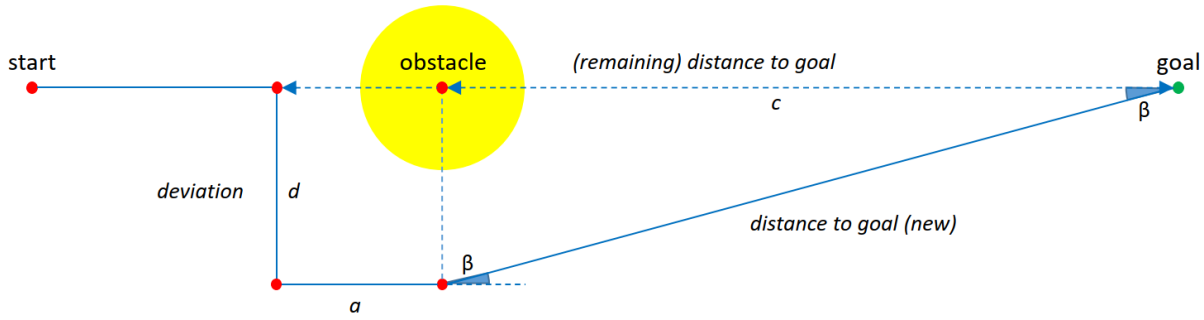
Mecanum_Collision_Prevention.ft

2. Navegación por codificador

Configuración de sensores y actuadores:



2a. Para esta tarea hay varias estrategias de solución. La que se presenta a continuación utiliza una versión simplificada por las posibilidades de las ruedas Mecanum del modelo matemático descrito como ejemplo de solución para la tarea experimental de la tarea n.º 6 del Robotics TXT 4.0 Base Set:



Mecanum_Model_Target_Reacher.jpg

Estrategia de solución:

El vehículo Mecanum se alinea en dirección al destino (*goal*) y comienza a desplazarse en línea recta. Si detecta un obstáculo en el trayecto (*obstacle*), lo esquiva desplazándose hacia la derecha hasta que el sensor ultrasónico no detecte ningún obstáculo a menos de 25 cm de distancia. Para evitar que las ruedas rocen el obstáculo al pasarlo, se desplaza otros 15 cm adicionales hacia la derecha (d).

A continuación, avanza 25 cm (a) en línea recta. Después se utilizan los impulsos restantes del camino directo (original) al destino (c) y la distancia recorrida lateralmente para esquivar el objeto (d) para calcular la nueva distancia hasta el destino (*distance to goal (new)*) y el ángulo de giro β para corregir la alineación hacia el punto de destino.

El cálculo de la nueva distancia hasta el destino es sencillo; la obtenemos a partir del teorema de Pitágoras:

$$distance\ to\ goal\ (new) = \sqrt{c^2 + d^2}$$

La distancia d se obtiene a partir de la conversión de los impulsos contados en el desplazamiento lateral más 15 cm; la longitud c se obtiene restando el desvío a (25 cm) de la distancia calculada (*remaining distance to goal*) a partir del número de impulsos que quedan por recorrer (*remaining impulses to goal*) al inicio de la maniobra evasiva.

Podemos determinar el ángulo β, en el que debemos girar el robot hacia la izquierda para que vuelva a desplazarse en dirección hacia el destino, a partir de un cálculo:

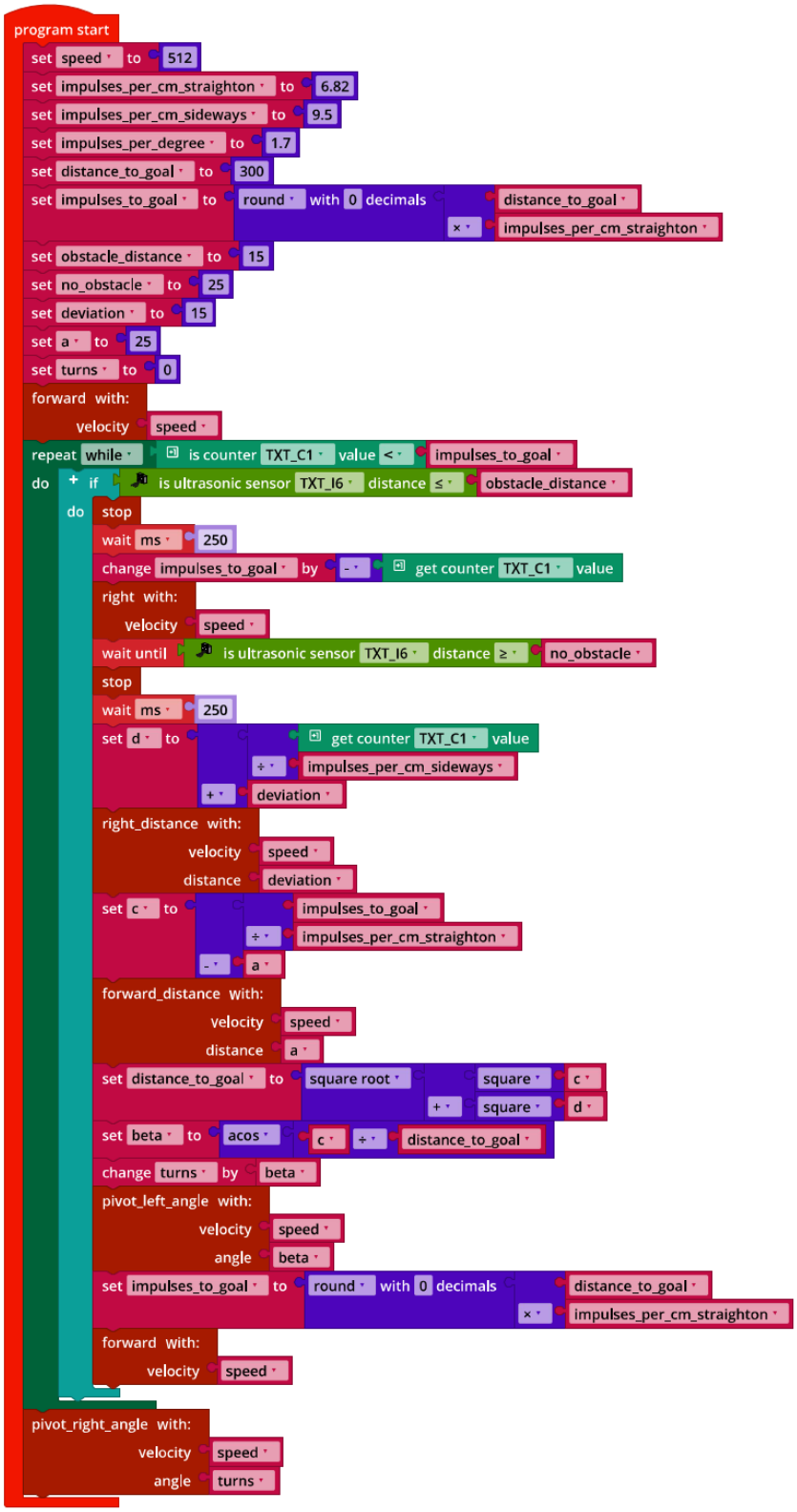
$$\beta = \arccos\left(\frac{c}{distance\ to\ goal\ (new)}\right)$$

La estrategia de solución también funciona con varios obstáculos consecutivos.

Al alcanzar la meta, el vehículo vuelve a girar a la derecha en la dirección de desplazamiento original. Para ello, a la distancia recorrida se suman todos los ángulos β, en los que el vehículo Mecanum tuvo que cambiar su dirección tras las maniobras evasivas.

Importante: Dado que los valores de conversión de los impulsos por cm difieren en el desplazamiento en línea recta y lateral, los cálculos de la nueva distancia al destino y del ángulo de giro β deben realizarse en valores expresados en cm.

2b. Extracto del programa (ejemplo):



Mecanum_Target_Reacher.ft

Anexos
Vehículo Mecanum con detección de obstáculos

Material necesario

- Ordenador para el desarrollo de programas, localmente o a través de la interfaz web.
- Cable USB o conexión BLE o wifi para transferir el programa al TXT4.0.
- Hoja de ruta con línea circular negra cerrada de 2 cm de ancho (del Robotics TXT 4.0 Base Set).

Más información

[1] FRC Team 2605 (Bellingham, Washington): *How a Mecanum Drive Works*. github.io

[2] Editor de diagramas en línea para crear diagramas de transición de estados (formato drawio): https://www.diagrammeditor.de/

Nombre: _____ Clase: ___ Fecha: _____

Tarea n.º 2

Seguidor de línea

Tarea de construcción

Para la tercera tarea de programación y las tareas experimentales necesitamos la cámara del modelo básico del vehículo Mecanum con sensores. Conéctala al puerto USB1 del TXT.

Importante: Cuando se inicia el TXT, el servomotor se coloca automáticamente en «posición línea recta». A continuación, coloca la palanca del servomotor de forma que la cámara se incline hacia delante en un ángulo de aproximadamente 45° con respecto a la vertical.

Atención: Si mueves el servomotor con la mano cuando el TXT está encendido, puedes dañarlo.

Tareas de programación

1. Seguidor de línea con sensor de pistas

Ahora el vehículo Mecanum debe seguir la pista negra de aproximadamente 2 cm de ancho del recorrido del Robotics TXT 4.0 Base Set (véase también la tarea n.º 8 del Robotics TXT 4.0 Base Set).

1a. En primer lugar, crea un diagrama de transición de estados que describa los posibles estados y el comportamiento del vehículo Mecanum.

1b. Convierte tu diagrama de transición de estados en un programa Blockly. Utiliza para ello una variable de estado cuyo valor de estado determines a partir de los valores de los sensores infrarrojos.

Prueba el programa en el recorrido circular del Robotics TXT 4.0 Base Set.

1c. ¿Cómo se puede incrementar la velocidad del seguidor de línea Mecanum? Realiza pruebas y mide el tiempo que necesita el seguidor de línea para completar el recorrido.

Solución alternativa (ajustes, parámetros)	Tiempo

2. Seguidor de línea con detección de obstáculos

Es posible encontrar obstáculos en la pista. Si el sensor ultrasónico detecta un obstáculo a una distancia máxima de 2 cm delante del vehículo, este debe desplazarse lateralmente, pasar el obstáculo y volver a encontrar la pista.

Piensa en diferentes soluciones para encontrar la línea. Amplía el programa Blockly como corresponde y prueba las variantes. ¿Qué ventajas y desventajas se presentan?

3. Seguidor de línea con control de color

Notarás cuatro superficies de color en las esquinas del recorrido. Con estas superficies de color puedes dar órdenes a tu vehículo Mecanum.

3a. Añade un reconocimiento de colores a tu programa Blockly. Para calibrar el reconocimiento, haz que se visualice en la consola el código de color RGB-HEX del color reconocido.

3b. Determina a qué colores (al menos dos) debe reaccionar el vehículo Mecanum y de qué manera y adapta el programa Blockly como corresponde.

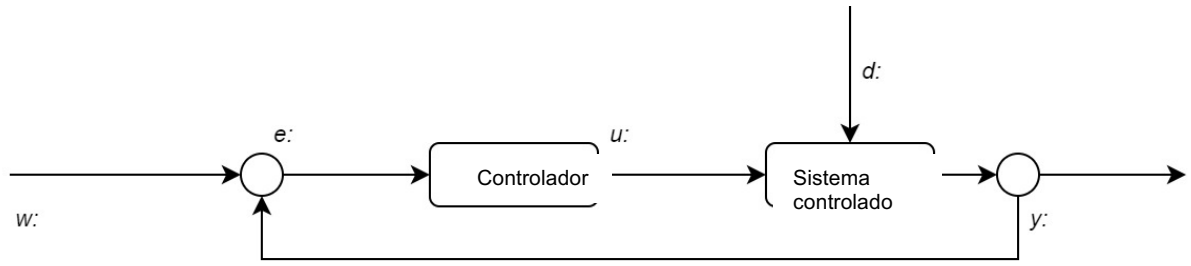
Nota: No te sorprendas si el valor HEX visualizado en la consola parece fluctuar considerablemente – el valor Hue es estable. Para el reconocimiento puedes utilizar cualquiera de los códigos de color HEX visualizados.

Tareas experimentales

1. Seguidor de línea con controlador proporcional

El seguidor de línea Mecanum ahora debe equiparse con un controlador proporcional (controlador P) al igual que el buggy seguidor de línea de la tarea n.º 8 del Robotics TXT 4.0 Base Set. La dimensión de la corrección de la dirección dependerá de la magnitud de la desviación de la línea. Aquí también se utilizará la función de detección de líneas de la cámara para determinar la magnitud de la desviación de la línea.

1a. En primer lugar, etiqueta el siguiente bucle de control con los valores correspondientes al seguidor de línea Mecanum:



1b. Diseña y programa un control proporcional (controlador P) para el seguidor de línea Mecanum. Puedes utilizar el método de detección de la línea negra de la tarea experimental n.º 1 de la tarea n.º 8 del Robotics TXT 4.0 Base Set .

Prueba tu programa en el recorrido simple de la línea recta iniciando el seguidor de línea Mecanum en posición paralela a la pista de modo que el centro de la pista aparezca lo más a la izquierda posible de la imagen.

Consejo: Comienza con el factor de proporcionalidad $k_p = 2$. Incrementa progresivamente el valor de a 1 hasta que el seguidor de línea Mecanum se «estabilice», es decir, que la desviación de la pista disminuya rápidamente sin que el controlador genere oscilaciones de amplitud creciente o se sobrepase.

1c. Añade a tu programa una salida de texto que, después de cada cambio de la desviación de la pista, proporcione en la consola

- el tiempo (en ms) que ha transcurrido desde el inicio del programa y
- el valor de la desviación actual

separados por un espacio.

Después de cada desplazamiento de prueba, copia la información de salida de la consola con un valor diferente de k_p en una hoja de cálculo y representa los valores en un gráfico (x: tiempo, y desviación de la pista). Adapta el factor de proporcionalidad k_p hasta que la curva se estabilice rápidamente.

Prueba el seguidor de línea con el recorrido circular del Robotics TXT 4.0 Base Set. Es probable que tengas que disminuir un poco la velocidad máxima.

2. Seguidor de línea con controlador PD

Al igual que con el buggy de la tarea n.º 8 del Robotics TXT 4.0 Base Set, puedes atenuar más el sobreimpulso añadiendo un elemento «D» (componente diferencial) al controlador, que al corregir la velocidad considera la magnitud del cambio en la desviación de la pista.

Amplía el controlador P de tu seguidor de línea Mecanum de la tarea experimental n.º 1 de forma correspondiente a un controlador PD. Realiza desplazamientos de prueba utilizando valores diferentes para el factor diferencial k_d y representa los datos en un programa de hojas de cálculo.

Consejo: Comienza las pruebas con el factor diferencial $k_d = 0{,}25$ y aumenta progresivamente el valor en 0,25 hasta que el sobreimpulso del controlador P se encuentre bien.

Anexos
Tarea n.º 2: Seguidor de línea

Material necesario

- Ordenador para el desarrollo de programas, localmente o a través de la interfaz web.
- Cable USB o conexión BLE o wifi para transferir el programa al TXT4.0.
- Hoja de ruta con línea recta negra de 2 cm de ancho.
- Hoja de ruta con línea circular negra cerrada de 2 cm de ancho (del Robotics TXT 4.0 Base Set)
- Obstáculo (caja de cartón, lata, ...)

Más información

[1] FRC Team 2605 (Bellingham, Washington): *How a Mecanum Drive Works*. github.io

[2] Wikipedia: *Autómata finito (máquina de estado finito)*

[3] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, Peter Wolstenholme: *Modeling Software with Finite State Machines. A Practical Approach*. Auerbach Publications, 2006.

[4] Editor de diagramas en línea para crear diagramas de transición de estados (formato drawio): https://www.diagrammeditor.de/

[5] Wikipedia: *Regelungstechnik*.

[6] Wikipedia: *Controlador*.

[7] RN-Wissen: *Regelungstechnik*.

[8] Tim Wescott: *PID without a PhD*. Embedded Systems Programming, 10/2000, Págs. 86-108.

Tarea n.º 2: Seguidor de línea

En esta tarea, el vehículo Mecanum se convierte en un «seguidor de línea» con la ayuda del sensor de pistas: El vehículo aprende a desplazarse de manera autónoma a lo largo de una línea negra.

En la tarea experimental, el vehículo recibe una cámara con detección de líneas: Como sensor analógico, permite el seguimiento de líneas con controladores P y PD.

La tarea se basa en la tarea 8 del Robotics TXT 4.0 Base Set.

Tema

Control digital del vehículo y control proporcional (y PD) del desplazamiento a lo largo de una línea negra; detección de obstáculos y reacción ante ellos.

Objetivos de aprendizaje

- Control sencillo de tres posiciones mediante sensores digitales
- Incorporación de la detección de obstáculos (medición de distancia por ultrasonido)
- Control analógico mediante la detección de líneas (cámara con análisis de imágenes)
- Calibración de controladores P y PD.

Tiempo necesario

En esta tarea se utiliza el modelo básico de vehículo Mecanum con sensores montado en la tarea 1.

Para el desarrollo del programa para resolver las tareas de programación, las alumnas y los alumnos, siempre y cuando cuenten con conocimientos previos del Robotics TXT 4.0 Base Set (en especial de la tarea 8), necesitan 45-90 minutos (una o dos clases).

En las tareas experimentales se desarrollan un controlador P y un controlador PD. Para programar y configurar los controladores, deberían emplearse varias horas de clase (cada una de 90-180 minutos). Se recomienda el trabajo en grupo.

Anexos
Tarea n.º 2: Seguidor de línea

Material necesario

- Ordenador para el desarrollo de programas, localmente o a través de la interfaz web.
- Cable USB o conexión BLE o wifi para transferir el programa al TXT4.0.
- Hoja de ruta con línea recta negra de 2 cm de ancho.
- Hoja de ruta con línea circular negra cerrada de 2 cm de ancho (del Robotics TXT 4.0 Base Set)
- Obstáculo (caja de cartón, lata, ...)

Más información

[1] FRC Team 2605 (Bellingham, Washington): *How a Mecanum Drive Works*. github.io

[2] Wikipedia: *Autómata finito (máquina de estado finito)*

[3] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, Peter Wolstenholme: *Modeling Software with Finite State Machines. A Practical Approach*. Auerbach Publications, 2006.

[4] Editor de diagramas en línea para crear diagramas de transición de estados (formato drawio): https://www.diagrammeditor.de/

[5] Wikipedia: *Ingeniería de control*.

[6] Wikipedia: *Controlador*.

[7] RN-Wissen: *Ingeniería de control*.

[8] Tim Wescott: *PID without a PhD*. Embedded Systems Programming, 10/2000, Págs. 86-108.

Nombre: _____ Clase: ___ Fecha: _____

Hoja de soluciones de la tarea temática n.º 2

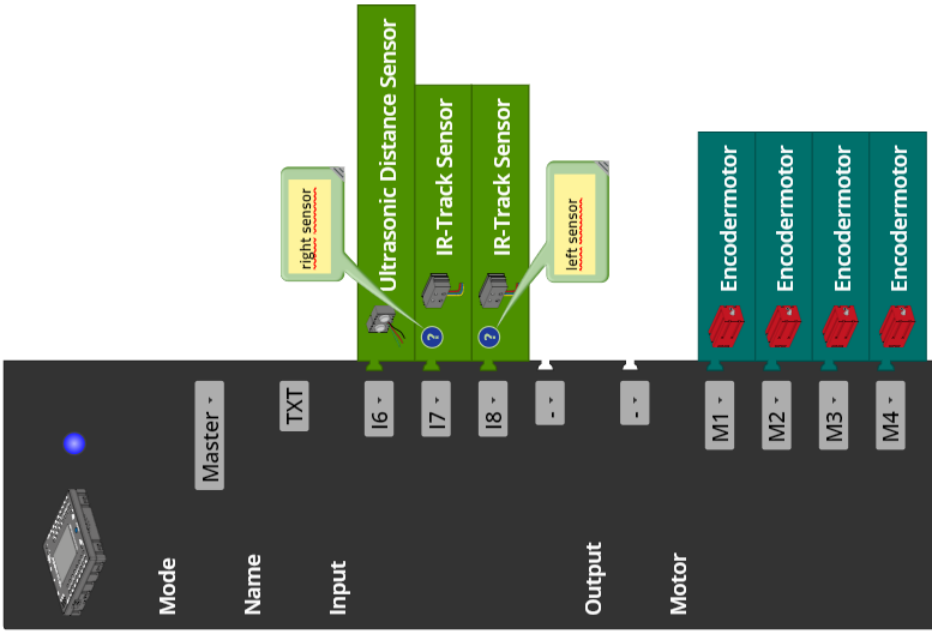
Seguidor de línea

Para controlar el vehículo, las soluciones de las alumnas y los alumnos deben utilizar las subfunciones de la tarea n.º 1. Para programar el seguidor de línea debe utilizarse – al igual que en la solución de la tarea n.º 8 del Robotics TXT 4.0 Base Set – una variable de estado («state») para diferenciar los estados. Así, los programas son claros y comprensibles.

Las dos tareas experimentales permiten comprender el desarrollo y la configuración de un controlador P y un controlador PD. La información de salida y la visualización gráfica de los valores de medición son de especial relevancia.

Tarea de construcción

Conexión de los sensores:

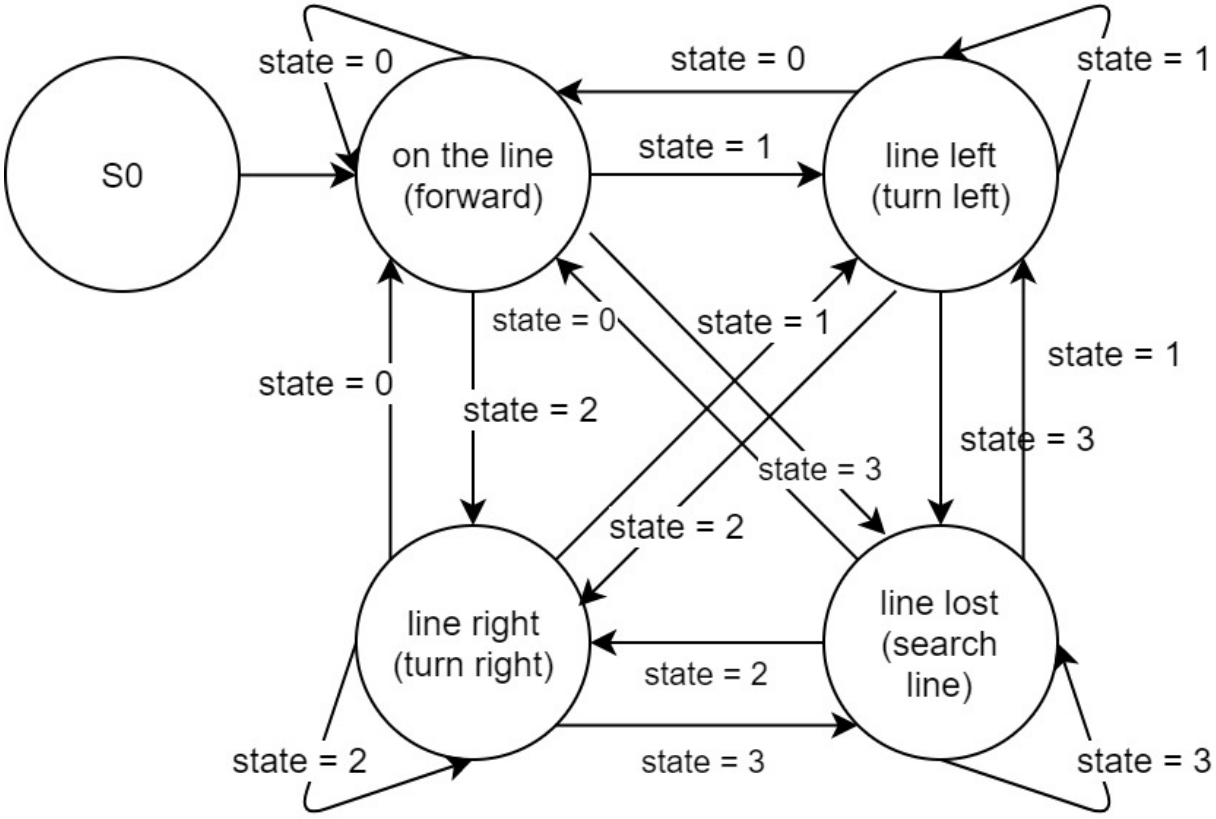


Tareas de programación

1. Seguidor de línea con sensor de pistas

Al igual que el buggy de la tarea n.º 8 del Robotics TXT 4.0 Base Set, el vehículo Mecanum debe ser controlado de modo que los dos sensores infrarrojos del sensor de pistas se encuentren centrados sobre la línea negra, es decir, que ambos arrojen el valor 0.

1a. Diagrama de transición de estados del seguidor de línea (digital):



State-Transition_Diagram_Line_Follower.drawio

1b. En función de los valores de los sensores infrarrojos izquierdo y derecho, a la variable de estado *state* se le asignan los siguientes valores:

- *state* = 0 → sobre la pista (ambos sensores arrojan el valor 0)
- *state* = 1 → la pista se encuentra a la izquierda (solo el sensor izquierdo arroja el valor 0)
- *state* = 2 → la pista se encuentra a la derecha (solo el sensor derecho arroja el valor 0)
- *state* = 3 → desviación de la pista (ambos sensores arrojan el valor 1)

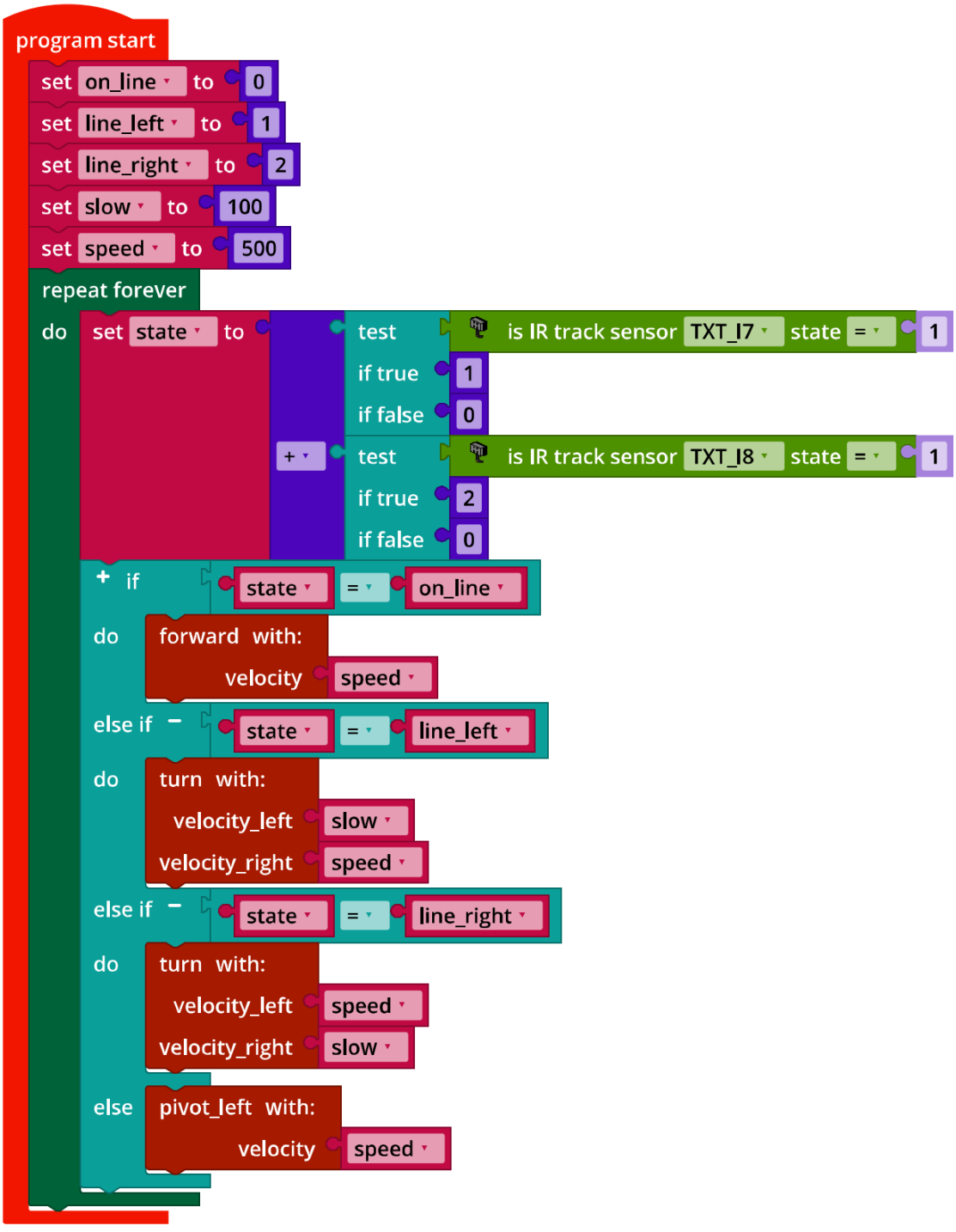
	I8 (sensor izquierdo)	I7 (sensor derecho)
Línea negra (pista)	0 → *state* += 0	0 → *state* += 0
Superficie clara	1 → *state* += 2	1 → *state* += 1

Ejemplo de lectura: Si el sensor izquierdo (I8) arroja el valor 0 y el derecho (I7) el valor 1, entonces la pista se encuentra a la izquierda del centro del vehículo. La variable de estado *state* recibe el valor 1, y en este estado el vehículo debe dirigirse hacia la izquierda para que el sensor vuelva a centrarse sobre la pista.

El siguiente ejemplo de solución utiliza las funciones de navegación del vehículo Mecanum programadas en la tarea n.º 1. Esto hace que el programa sea muy claro.

Al iniciar el bucle, la variable de estado «state» se ajusta en un valor de estado entre {0, 1, 2, 3} en función de los valores de los dos sensores infrarrojos.

Extracto del programa (ejemplo):



Mecanum_Line_Follower_digital.ft

1c. La velocidad del seguidor de línea puede incrementarse eligiendo una velocidad principal («speed») lo más alta posible y reduciendo la diferencia de velocidad de los

motores al conducir, es decir, ajustando la velocidad «slow» de modo que el seguidor de línea no pierda la pista en las curvas.

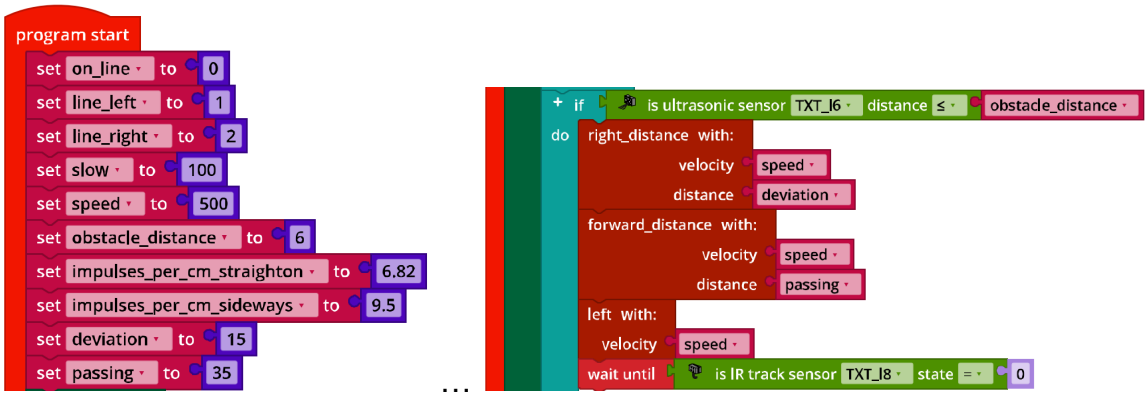
Nota: El seguidor de pista digital también puede entenderse como un controlador de tres puntos, que controla la velocidad de los motores en función de los tres estados «a la izquierda de la pista», «sobre la pista» y «a la derecha de la pista». El intervalo en el que ambos sensores infrarrojos arrojan el valor nominal «0» (ambos se encuentran directamente encima de la pista) se conoce también como *histéresis*.

2. Seguidor de línea con detección de obstáculos

Tras la desviación, es posible volver a encontrar la línea detrás del obstáculo a través de

- un nuevo desplazamiento lateral (desventaja: no se conoce la profundidad del obstáculo, por lo que es posible que se roce el obstáculo o no alcanzar la línea si hay una curva cerrada justo detrás; ventaja: se encuentra una línea recta en la posición correcta).
- un desplazamiento en diagonal (desventaja: es posible que se roce el obstáculo; hay más probabilidades de que no se alcance la línea; ventaja: se encuentra una línea recta en la orientación correcta para continuar)
- un giro de 45°-90° y un posterior desplazamiento en línea recta (desventaja: es posible que se roce el objeto, que no se alcance la línea o que se continúe en la dirección equivocada al encontrar la línea)
- una curva alrededor del obstáculo (desventaja: es posible que se continúe en la dirección equivocada al encontrar la línea)

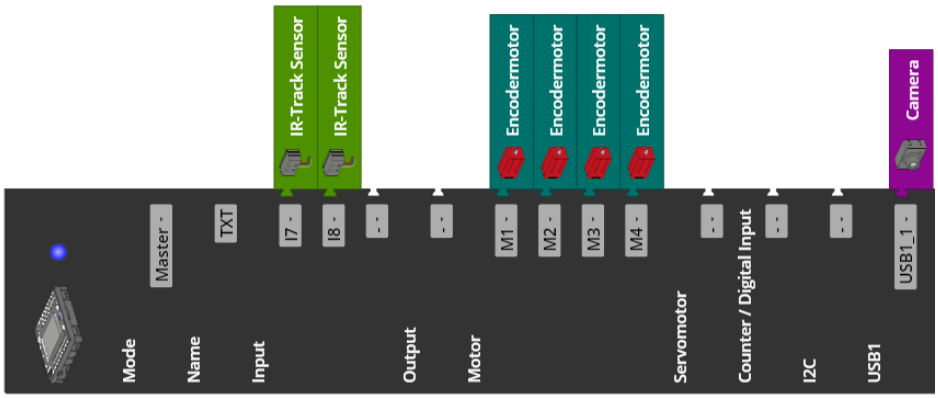
Extracto del programa (ejemplo):



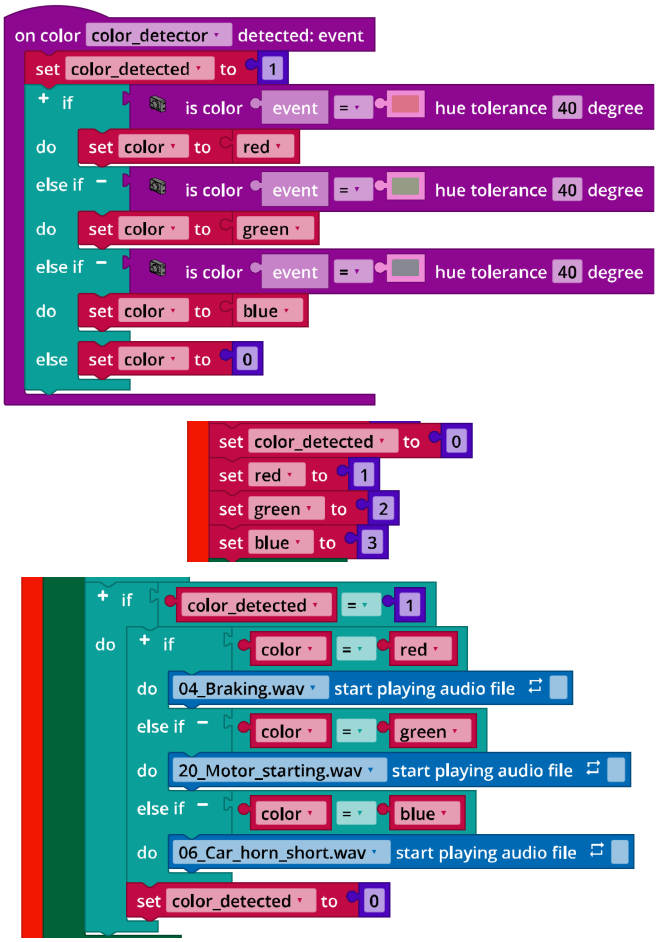
Mecanum_Line_Follower_Obstacle_digital.ft

3. Seguidor de línea con control de color

Conexión de la cámara:



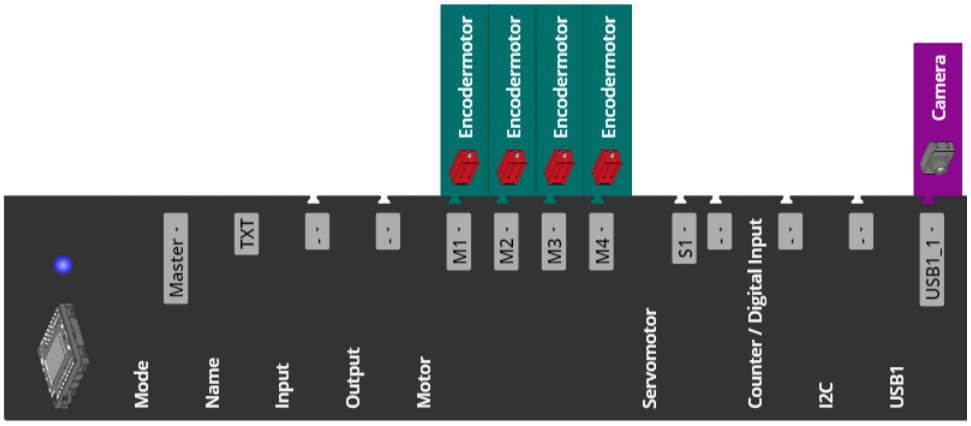
3b. Extracto del programa (ejemplo):



Mecanum_Line_Follower_with_Color_Detection_digital.ft

Tareas experimentales

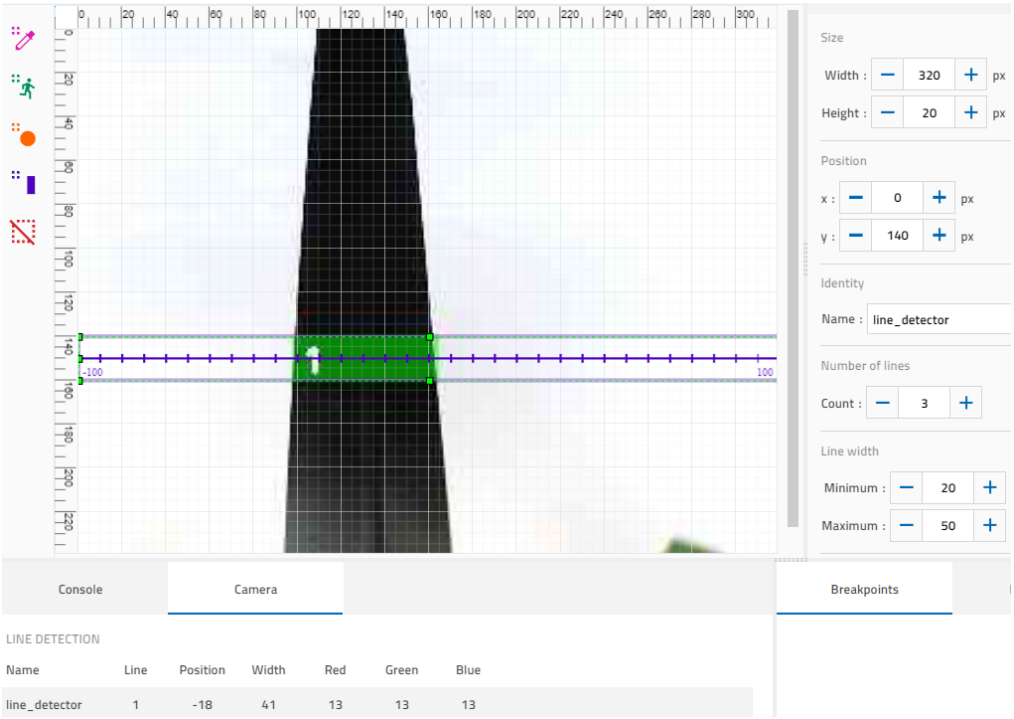
Conexión de los sensores:



1. Seguidor de línea con controlador proporcional

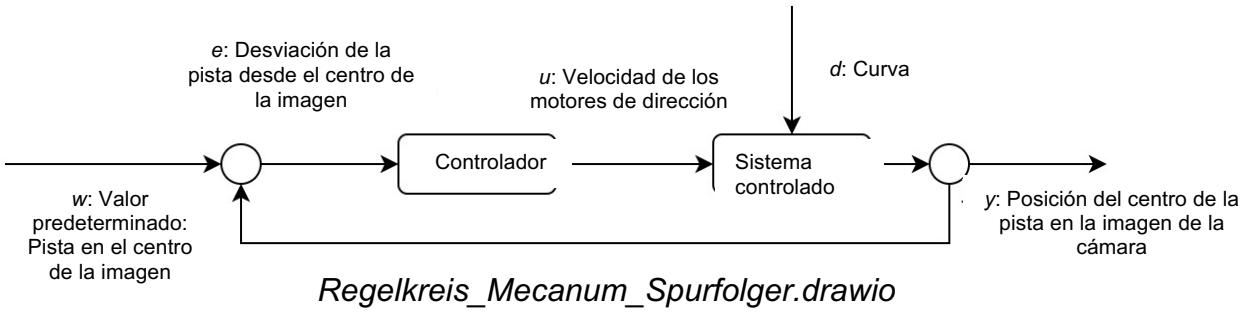
El detector de líneas de la cámara arroja la desviación desde el centro de la ventana de detección.

Configuración de la detección de líneas:



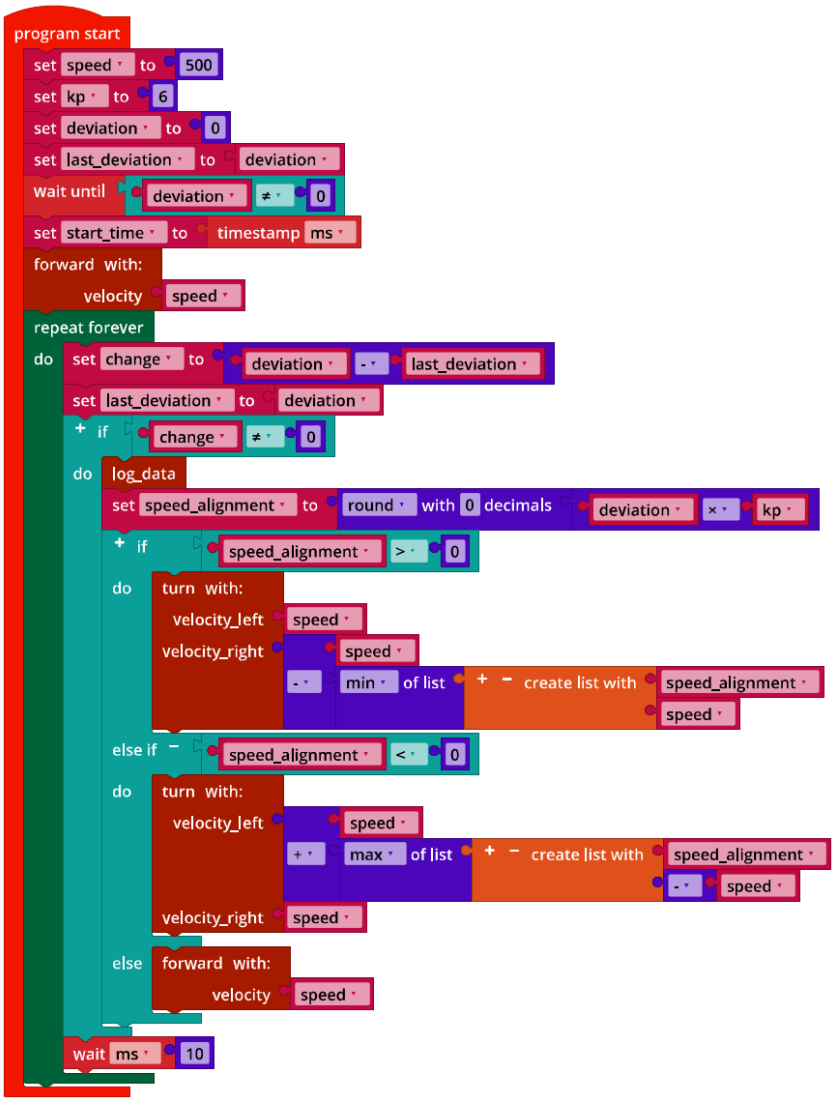
Dado que las superficies de color pueden detectarse como una línea, la detección de líneas debe configurarse como mínimo en dos líneas.

1a. Bucle de control:

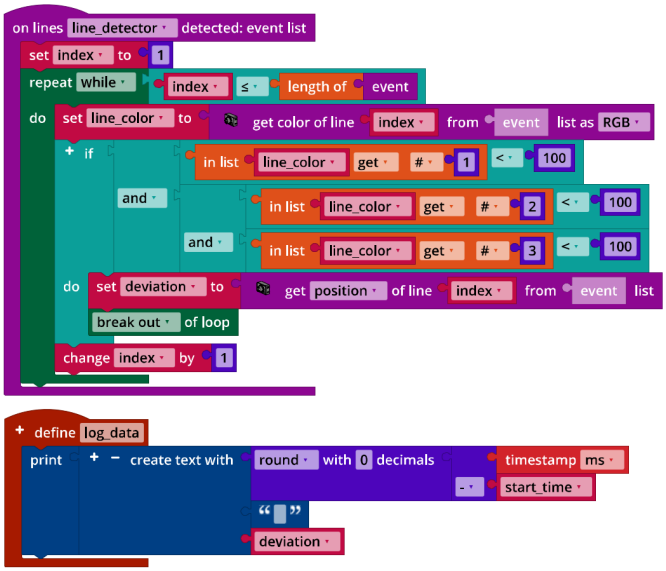


Regelkreis_Mecanum_Spurfolger.drawio

1b. Extracto del programa (ejemplo):



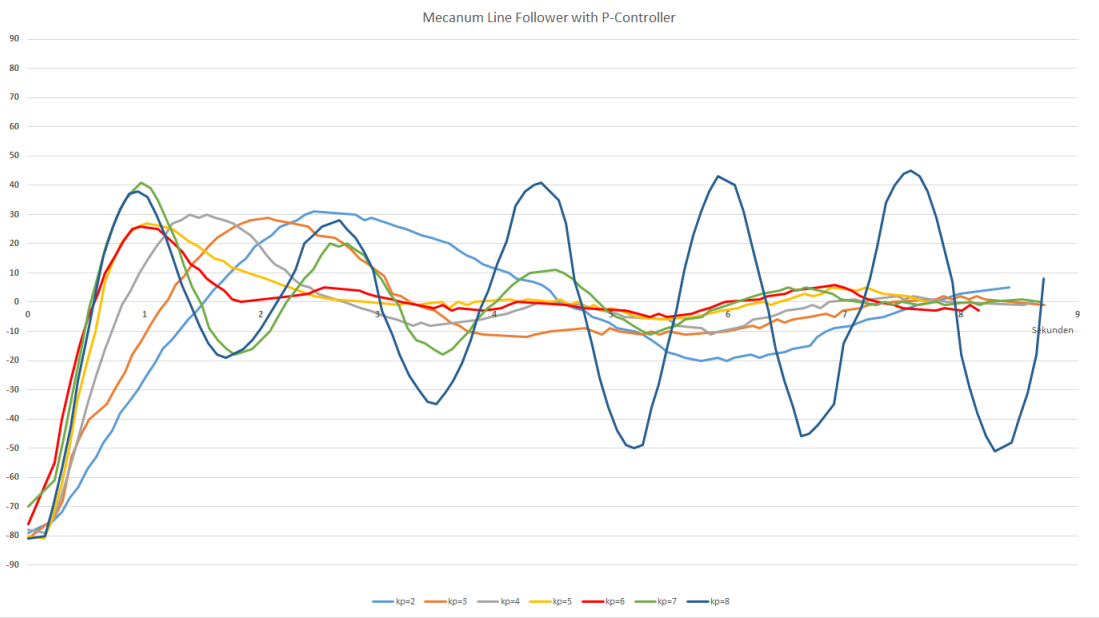
Mecanum_Line_Follower_P_Controller.ft



Mecanum_Line_Follower_P_Controller.ft

Multiplicada por el factor de proporcionalidad k_p, la posición se suma o resta a la velocidad del motor al doblar a la izquierda o a la derecha (función *turn*) de modo que la desviación de la pista disminuye con el cambio en la dirección de desplazamiento. En función de la posición vertical del detector de líneas en la configuración de la cámara, es probable que deban ajustarse la anchura mínima y máxima de las líneas en el programa.

1c. Resultados de medición del controlador P (con $k_p \in \{2, 3, 4, 5, 6, 7, 8\}$):



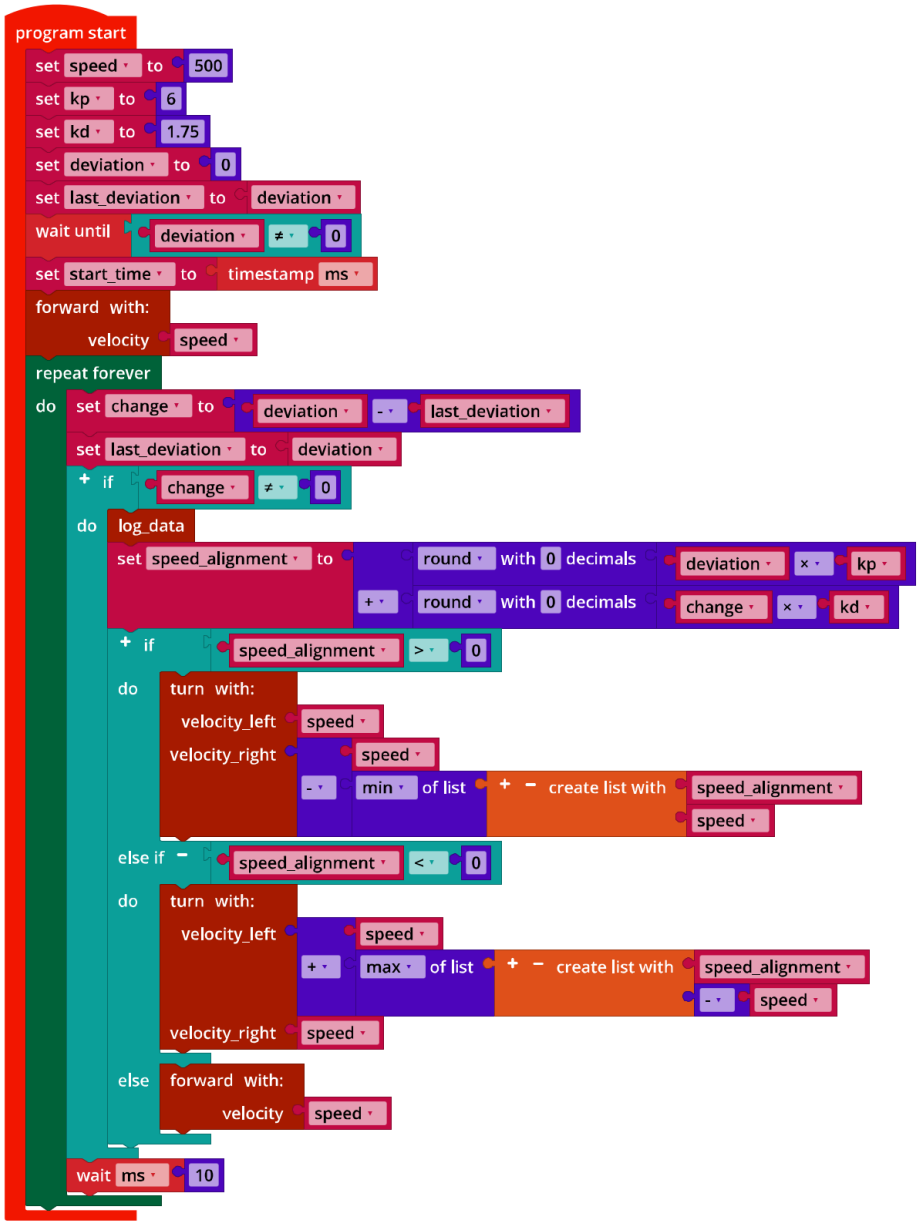
Mecanum_Line_Follower_with_P_Controller_Results.jpg

La línea roja con $k_p = 6$ se estabiliza más rápidamente tras aproximadamente 1,7 segundos, sin sobrepasarse una segunda vez. Con $k_p = 8$ (línea azul oscuro) oscila el controlador.

2. Seguidor de línea con controlador PD

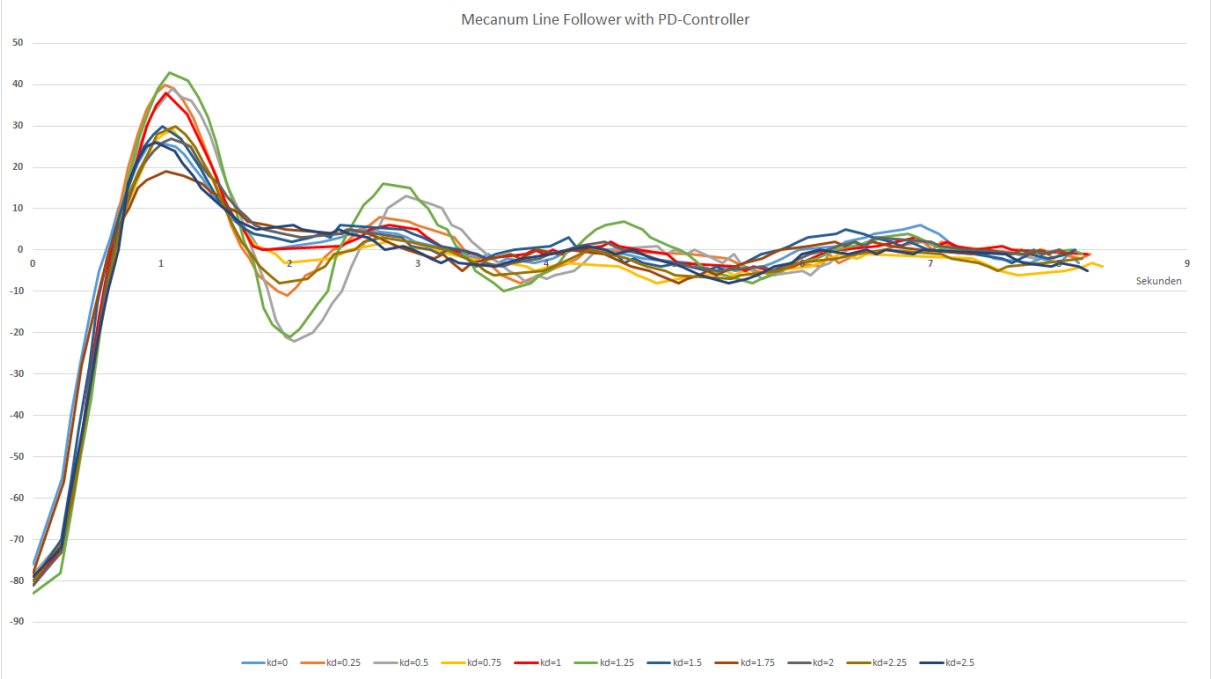
El elemento D del controlador PD es el cambio de la desviación multiplicado por el factor diferencial k_d.

Programa (ejemplo):



Mecanum_Line_Follower_with_PD_Controller.ft

Resultados de medición del controlador PD
(con $k_d \in \{0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5\}$):



Mecanum_Line_Follower_with_PD_Controller_Results.jpg

Con el factor diferencial $k_d = 1.75$ se obtuvo la máxima eficacia en la amortiguación del sobreimpulso en las pruebas realizadas. El valor puede variar de un modelo a otro en función de la posición de la detección de líneas en la imagen de la cámara y de pequeñas diferencias constructivas.

Anexos
Tarea n.º 2: Seguidor de línea

Material necesario

- Ordenador para el desarrollo de programas, localmente o a través de la interfaz web.
- Cable USB o conexión BLE o wifi para transferir el programa al TXT4.0.
- Hoja de ruta con línea recta negra de 2 cm de ancho.
- Hoja de ruta con línea circular negra cerrada de 2 cm de ancho (del Robotics TXT 4.0 Base Set)
- Obstáculo (caja de cartón, lata, ...)

Más información

[1] FRC Team 2605 (Bellingham, Washington): *How a Mecanum Drive Works*. github.io

[2] Wikipedia: *Autómata finito (máquina de estado finito)*

[3] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, Peter Wolstenholme: *Modeling Software with Finite State Machines. A Practical Approach*. Auerbach Publications, 2006.

[4] Editor de diagramas en línea para crear diagramas de transición de estados (formato drawio): https://www.diagrammeditor.de/

[5] Wikipedia: *Regelungstechnik*.

[6] Wikipedia: *Controlador*.

[7] RN-Wissen: *Regelungstechnik*.

[8] Tim Wescott: *PID without a PhD*. Embedded Systems Programming, 10/2000, Págs. 86-108.

Nombre: _____ Clase: ___ Fecha: _____

Tarea n.º 4

Robot pintor

El vehículo Mecanum Omniwheels se equipa con un rotulador y se convierte en un robot pintor.

Tarea de construcción

Construye el robot pintor siguiendo las instrucciones o convierte el vehículo Mecanum Omniwheels de la tarea 1, 2 o 3 en un robot pintor. Conecta los motores codificadores y el servomotor como se indica en el diagrama de cableado.

Utiliza la prueba de interfaz o tus programas de control de la tarea 1 para comprobar que todos los motores estén correctamente conectados.

Importante: En primer lugar, fija el rotulador (de punta fina, de fibra) en el soporte de forma que la punta se asiente sobre el papel. Después, inicia el TXT. El servo se ajusta automáticamente a la posición central. A continuación, coloca la servopalanca de forma que apunte horizontalmente hacia delante y no llegue a levantar el rotulador con el soporte.

Usa la prueba de interfaz para probar la bajada del rotulador mediante el servo.

Atención: Si mueves el servo con la mano cuando el TXT está encendido, puedes dañarlo.

Tareas de programación

1. Bajar el rotulador

Para que el robot pintor pueda moverse sin trazar una línea, el servomotor debe levantar el rotulador.

Añade una función para subir y bajar el rotulador a tu biblioteca de funciones para el vehículo Mecanum Omniwheels. Selecciona la posición del servo mediante la prueba de interfaz.

2. La casa de Papá Noel

Utilizando las funciones de navegación de la tarea 1, ahora puedes hacer que el robot pintor dibuje la «casa de Papá Noel» de un solo trazo.

Al finalizar, el robot pintor debe levantar el rotulador y moverse levemente hacia un lado.

3. Polígono

Ahora el robot pintor (como en la tarea 7 del Robotics TXT 4.0 Base Set) debe dibujar un polígono con una longitud de lado fija. Intenta que el programa sea lo más general posible, y usa bucles para que sea lo más compacto posible.

Prueba el programa haciendo que el robot pintor dibuje sucesivamente un triángulo, un cuadrilátero, un pentágono,... y finalmente un polígono de 15 lados o pentadecágono.

Consejo: No elijas una longitud demasiado grande para los lados.

Tareas experimentales

1. Desplazamiento hacia un punto de destino

Ahora el robot pintor debe desplazarse desde su posición hasta un punto (objetivo) (x, y) definido por coordenadas. Supongamos que el rotulador se encuentra en el punto cero $(0, 0)$ del sistema de referencia de coordenadas al iniciar el programa y que el robot está alineado a lo largo del eje «x» (positivo).

1a. ¿Qué movimientos debe realizar el robot pintor para desplazarse desde su posición (el punto cero) hasta un punto determinado (x, y) por el camino más corto? ¿Qué cálculos se requieren para esto?

Haz un dibujo para ilustrar tu cálculo.

1b. Utilizando las funciones de navegación de la tarea 1, escribe un programa que permita al robot pintor recorrer el camino más corto hasta el punto especificado (x, y). Prueba tu programa con puntos medidos previamente.

2. «Pintar por números»

En la plantilla del programa *«Mecanum_Drawing_Coordinates.ft»* encontrará dos listas con coordenadas «x» e «y» y una con banderas «up/down». Cada una de las coordenadas «x» e «y» designa un punto en el sistema de coordenadas con un punto en la esquina inferior izquierda del papel como punto de partida $(0, 0)$.

Coloca el robot pintor de forma que el rotulador esté directamente encima del punto $(0, 0)$ y alinea el robot de forma que quede paralelo al borde inferior del papel y

apuntando hacia la derecha. El papel de dibujo debe tener al menos 60 cm de ancho y 40 cm de alto.

2a. Amplía tu programa de control para el robot pintor de la tarea experimental 1 de forma que se desplace en orden por los puntos (coordenadas) de la lista. Si la bandera «up/down» asociada a la coordenada es igual a 0, el robot pintor debe recorrer la distancia con el rotulador levantado; si es igual a 1, debe dibujar hasta ese punto. Puedes ajustar el tamaño de la imagen multiplicando las coordenadas por un factor fijo.

2b. Toma una foto de la imagen dibujada. ¿Qué representa?

2c. Ahora puedes darle al robot pintor tus propias listas de coordenadas para una imagen.

Anexos
Tarea n.º 4: Robot pintor

Material necesario

- Ordenador para el desarrollo de programas, localmente o a través de la interfaz web.
- Cable USB o conexión BLE o wifi para transferir el programa al TXT4.0.
- Rotulador (de punta fina, de fibra), hoja de papel blanca grande
- Plantilla de programa „*Mecanum_Drawing_Coordinates.ft*“

Más información

[1] Busca «Coordinate Grid Picture» en Internet.

[2] Oliver Boorman: *Cartesian Grid Image Generator*.

Tarea n.º 4: Robot pintor

Ahora, el vehículo Mecanum Omniwheels se convierte en un robot pintor. Aprende a dibujar formas geométricas con un rotulador de punta fina o de fibra, y se le enseña a «pintar por números».

La tarea se basa en la tarea 7 del Robotics TXT 4.0 Base Set.

Tema

Manejo del vehículo Mecanum Omniwheels para dibujar formas geométricas y líneas definidas.

Objetivos de aprendizaje

- Uso de funciones para la elaboración de un programa claro
- Construcción de modelos y cálculo del control del vehículo (trigonometría)
- Elaboración de estructuras de datos

Tiempo necesario

Para convertir el vehículo Mecanum Omniwheels de las tareas 1 y 2 en un robot pintor, los alumnos necesitan aprox. 45 minutos, y para construirlo desde cero siguiendo las instrucciones (suponiendo que ya tengan experiencia con fischertechnik), hasta 75 minutos.

Para desarrollar el programa de control para resolver las tareas de programación, los alumnos necesitan aprox. 90 minutos, suponiendo que ya conocen el Robotics TXT 4.0 Base Set (en especial la tarea 7). El tiempo necesario para resolver las tareas experimentales es de 135-180 minutos, dependiendo de la edad y la experiencia.

Anexos
Tarea n.º 4: Robot pintor

Material necesario

- Ordenador para el desarrollo de programas, localmente o a través de la interfaz web.
- Cable USB o conexión BLE o wifi para transferir el programa al TXT4.0.
- Rotulador (de punta fina, de fibra), hoja de papel blanca grande
- Plantilla de programa „*Mecanum_Drawing_Coordinates.ft*“

Más información

[1] Busca «Coordinate Grid Picture» en Internet.

[2] Oliver Boorman: *Cartesian Grid Image Generator*.

Nombre: _____ Clase: ___ Fecha: _____

Hoja de soluciones de la tarea temática n.º 4

Robot pintor

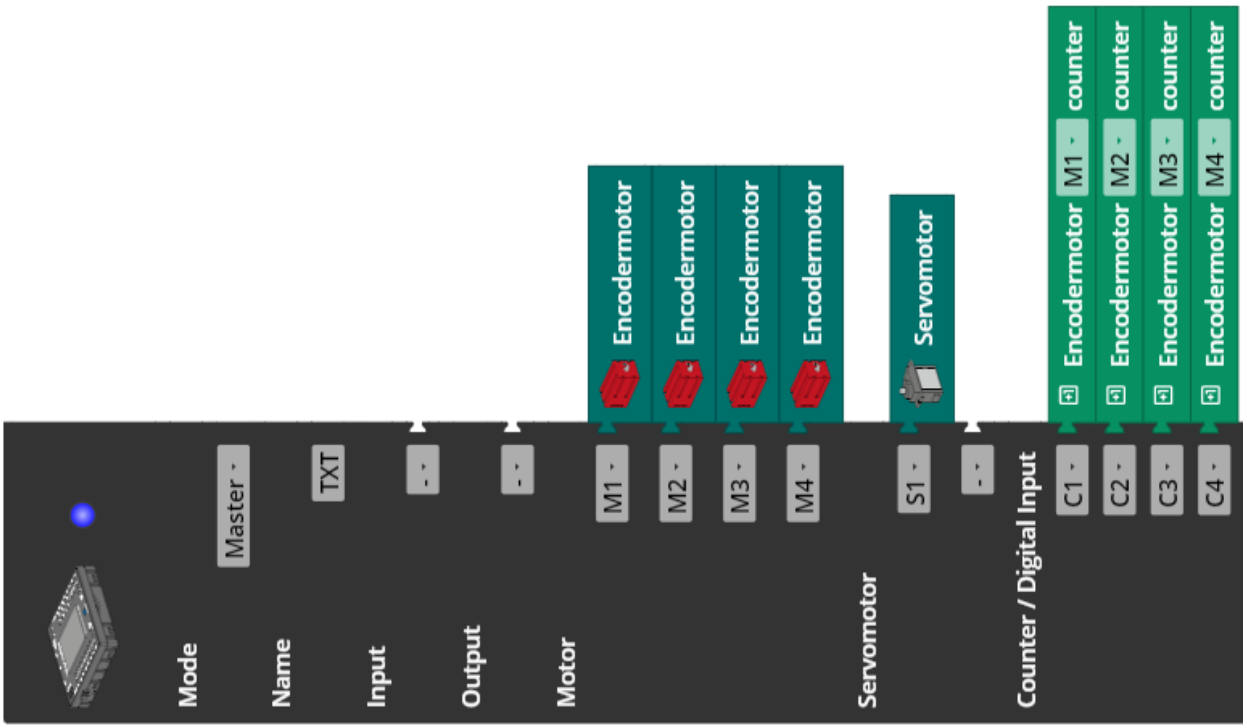
Las tareas son una práctica para el uso de bucles y funciones con el objetivo de obtener un programa claro, compacto y comprensible. Luego se compararán los resultados de los alumnos con relación a esto.

Tarea de construcción

Véase el manual de instrucciones.

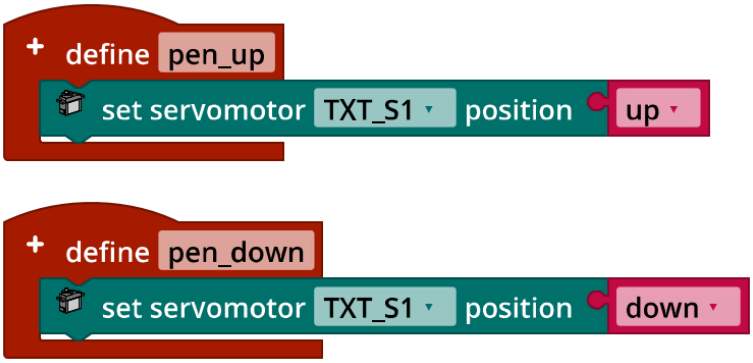
Tareas de programación

Conexión de los actuadores:



1. Bajar el rotulador

Extracto del programa (ejemplo):



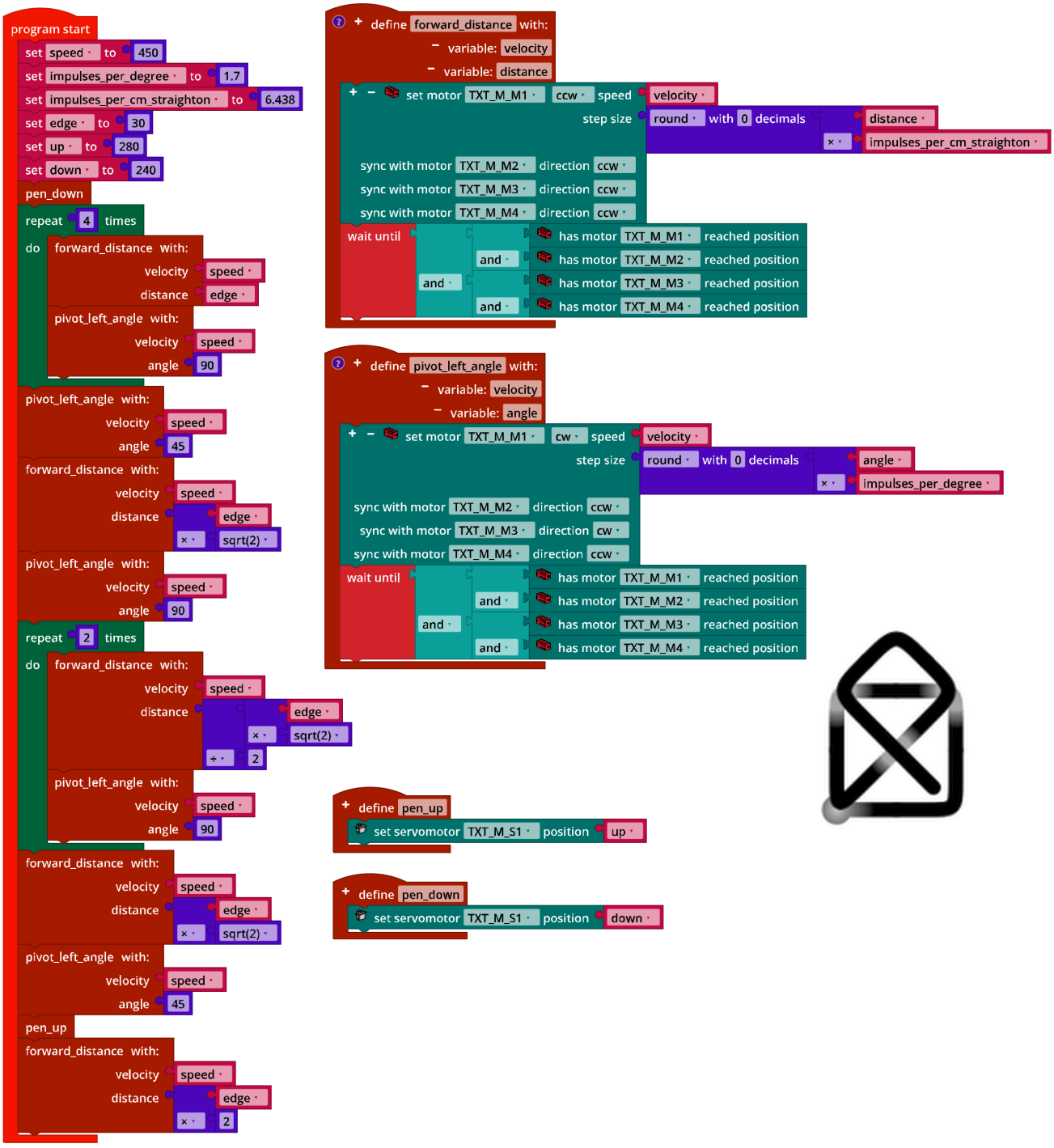
Los valores de las variables «up» y «down» se pueden determinar con la prueba de interfaz. Estas variables dependen del modelo y, en particular, de la posición del servobrazo acoplado.

2. La casa de Papá Noel

La «casa de Papá Noel» se puede dibujar con ocho líneas, sin levantar el rotulador del papel. De hecho, existen 88 distintas posibilidades correctas.

En el siguiente ejemplo de programa, el dibujo se realiza con una longitud de lado de 30 cm utilizando las funciones «forward_distance» y «pivot_left_angle» de la tarea 1.

Programa (ejemplo):

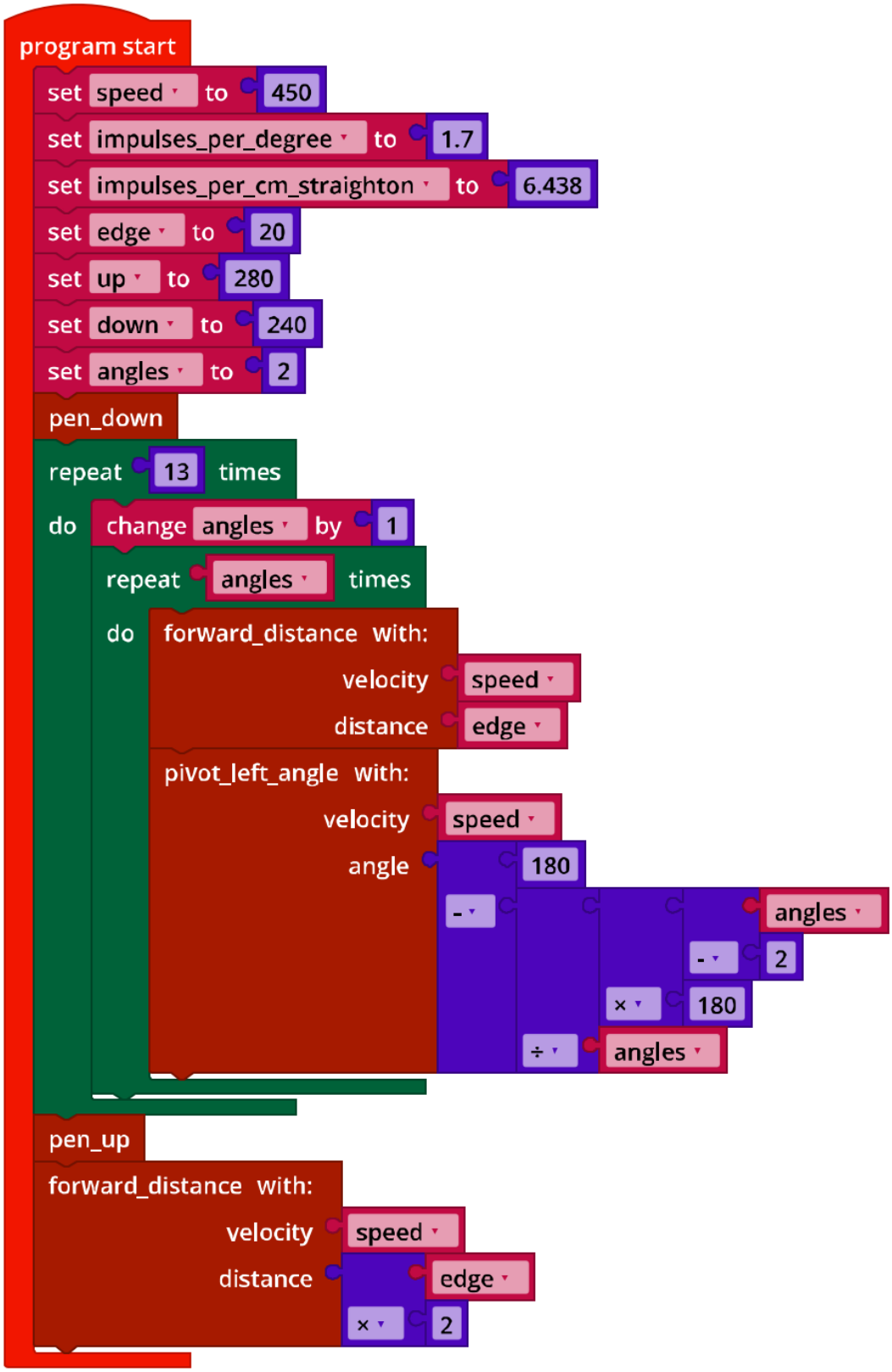


Mecanum_House_of_Santa_Claus.ft

3. Polígono

La suma de los ángulos internos de un polígono da $(n-1)\cdot 180°$. Es decir, que para dibujar un polígono, el vehículo Mecanum Omniwheels debe girar $180° - \frac{(n-1)\cdot 180°}{n}$ después de cada lado. La solución de ejemplo es una solución general para la tarea de dibujar polígonos y primero dibuja un triángulo, luego un cuadrado hasta llegar a dibujar un polígono de 15 lados o pentadecágono con lados de 20 cm cada uno. El uso de dos bucles anidados hace que el programa sea muy compacto.

Extracto del programa (ejemplo):

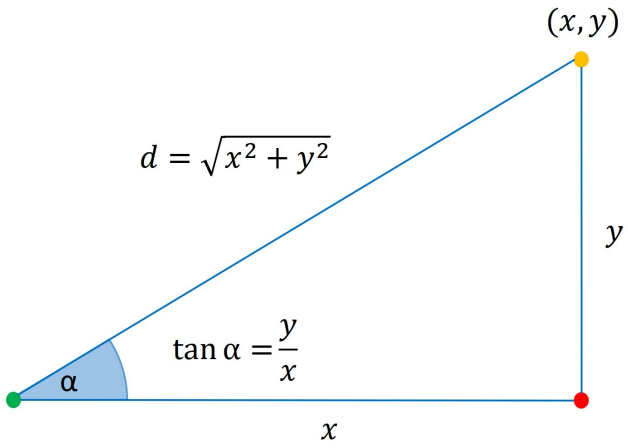


Mecanum_Drawing_Polygons.ft

Tareas experimentales

1. Desplazamiento hacia un punto de destino

1a. Para desplazarse de un punto $(0, 0)$ a otro (x, y), el robot pintor debe moverse a lo largo de la hipotenusa de un triángulo rectángulo con las longitudes de catetos x y y.



Mathematical_Model_Coordinates_Drawing.jpg

La longitud de la hipotenusa se obtiene mediante el teorema de Pitágoras:

$$d = \sqrt{x^2 + y^2}$$

El ángulo interior α del triángulo rectángulo se puede calcular fácilmente usando un poco de trigonometría:

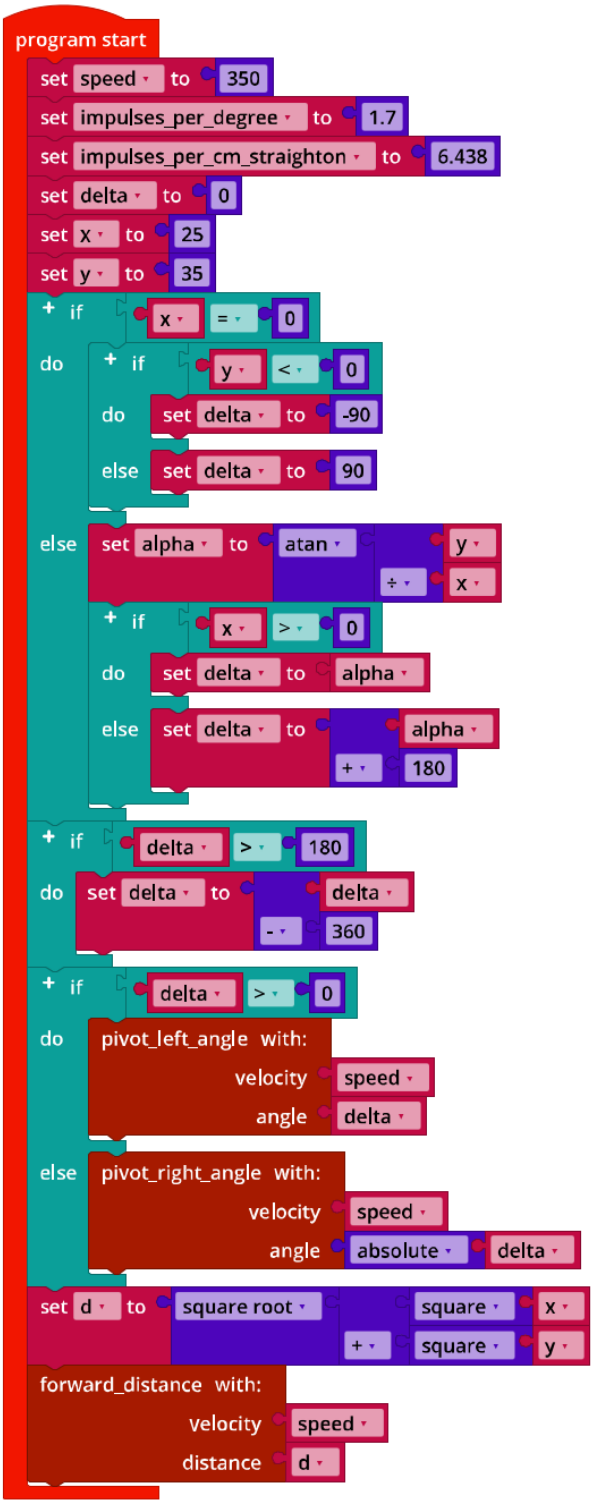
$$\tan \alpha = \frac{y}{x}$$

A partir de ahí, se debe derivar el ángulo de rotación δ en relación con el eje «x». La forma más fácil de hacerlo es con una distinción de casos:

- Siempre que $x > 0$: $\delta = \alpha$
- Si $x < 0$: $\delta = 180° + \alpha$

El caso $x = 0$ debe ser tratado por separado para que no se produzca la división por 0. Para $y > 0$, en este caso $\delta = 90°$, de lo contrario $\delta = -90°$.

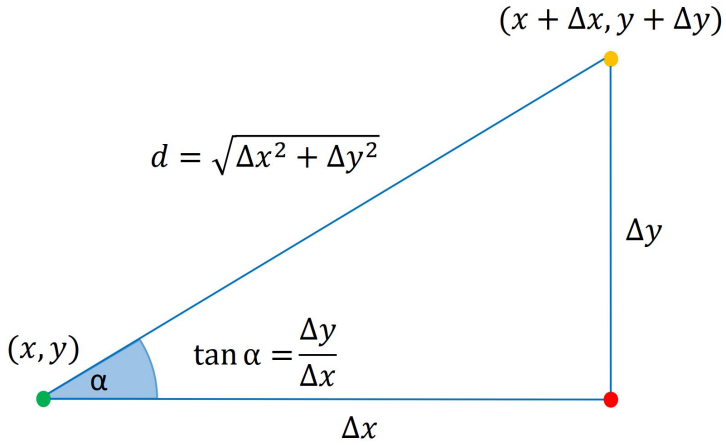
1b. Programa (ejemplo):



Mecanum_Move_2_Point.ft

2. «Pintar por números»

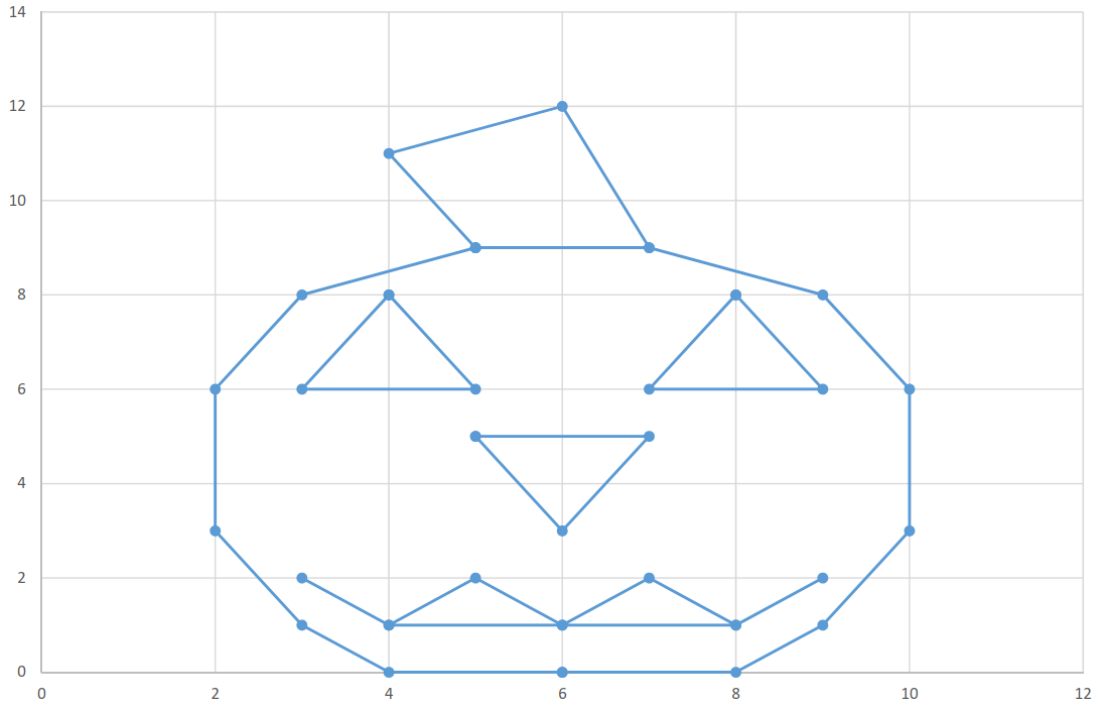
En general, el robot pintor no está alineado a lo largo del eje «x», sino que se encuentra en un ángulo γ con respecto al eje «x»; el ángulo de rotación δ debe definirse en relación con este ángulo. Además, la distancia al (siguiente) punto de destino debe calcularse en relación con la posición del robot ($(\Delta x, \Delta y)$).



Mathematical_Model_Coordinates_Drawing_generalised.jpg

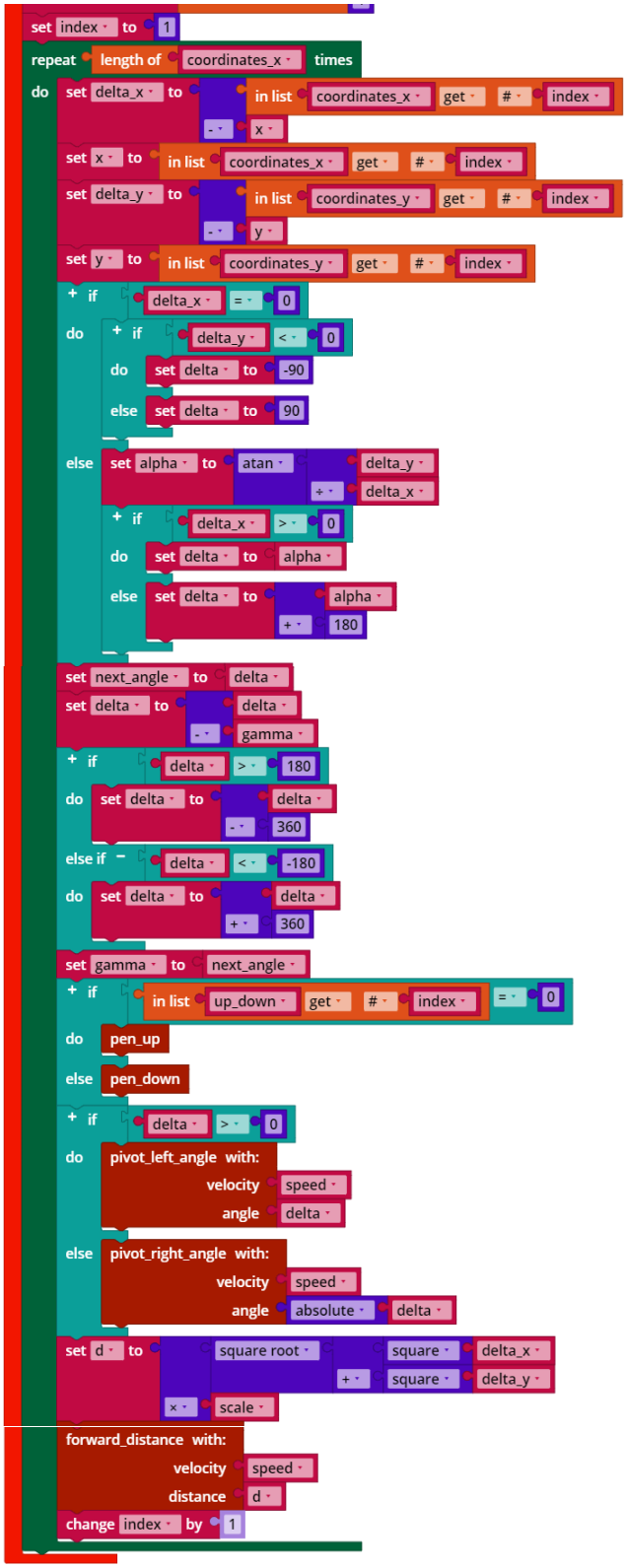
Los puntos a recorrer se almacenan en tres listas con las respectivas coordenadas «x» e «y» y una bandera «up/down» que indica si el rotulador debe bajar (1) o subir (0) en el camino hacia el punto designado por las coordenadas.

Las coordenadas dan como resultado el siguiente dibujo:



Pumpkin.jpg

Extracto del programa (ejemplo):



Mecanum_Coordinates_Drawing.ft

Anexos

Tarea n.º 4: Robot pintor

Material necesario

- Ordenador para el desarrollo de programas, localmente o a través de la interfaz web.
- Cable USB o conexión BLE o wifi para transferir el programa al TXT4.0.
- Rotulador (de punta fina, de fibra), hoja de papel blanca grande
- Plantilla de programa „*Mecanum_Drawing_Coordinates.ft*“

Más información

[1] Busca «Coordinate Grid Picture» en Internet.

[2] Oliver Boorman: *Cartesian Grid Image Generator*.

Nombre: _____ Clase: ___ Fecha: _____

Tarea n.º 5

Robot lanzabolas

En esta tarea, el vehículo Mecanum Omniwheels se equipa con un lanzador de bolas de plástico huecas, un sistema de control por voz y un sistema de puntería autónomo con la ayuda de la cámara.

Tarea de construcción

Construye el robot lanzabolas siguiendo las instrucciones o convierte el vehículo Mecanum Omniwheels de una de las otras tareas. Conecta los motores codificadores, la cámara y el servomotor como se indica en el diagrama de cableado.

Utiliza la prueba de interfaz o tus programas de control de la tarea 1 para comprobar que todos los motores estén correctamente conectados.

Importante: No fijes la servopalanca hasta haber iniciado el TXT. El servo se coloca automáticamente en una posición central. A continuación, coloca la servopalanca de manera que apunte «hacia adelante» (es decir, hacia la izquierda en el modelo) y se apoye detrás del soporte de lanzamiento verde.

Atención: ¡Si mueves la servopalanca con la mano cuando el TXT está encendido, el servo puede sufrir daños irreparables!

Tareas de programación

1. Mecanismo de lanzamiento

El lanzamiento de una bola de plástico hueca se activa moviendo la servopalanca hacia atrás hasta que se libera el soporte verde de disparo, que se encuentra tensado. Al mismo tiempo, una bola de plástico hueca se desliza automáticamente desde el cargador hacia la posición de lanzamiento. A continuación, la servopalanca debe volver a situarse delante del soporte de lanzamiento.

1a. Utiliza la prueba de interfaz para determinar las posiciones adecuadas de la servopalanca para disparar una bola de plástico hueca y para volver a «pretensar» el soporte de lanzamiento.

1b. Añade una función para lanzar una bola de plástico hueca a tu biblioteca de funciones del vehículo Mecanum Omniwheels.

1c. Coloca un botón en el lateral de tu vehículo y conéctalo a I1. Escribe un programa Blockly que lance una bola de plástico hueca cuando se pulse el botón.

1d. Añade un indicador de nivel de llenado del cargador a la pantalla del TXT. Añade un comando para recargar cuando todas las bolas de plástico huecas hayan sido lanzadas y haz que la recarga efectuada se confirme pulsando el botón.

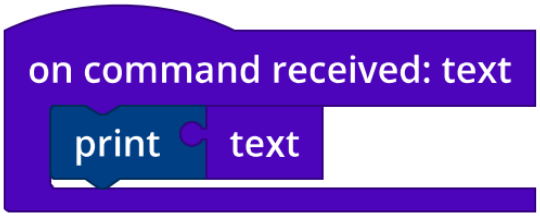
2. Control por voz

También puedes controlar tu vehículo Mecanum Omniwheel mediante comandos de voz. Para ello, descarga la aplicación «Voice Control» del App Store de Apple (para iOS) o de Google Play Store (para Android) y conéctala al TXT 4.0.

- Conexión a través de wifi: El controlador TXT 4.0 y el dispositivo (teléfono inteligente o tablet) deben estar conectados al mismo router. Además, el router debe permitir que los dispositivos se comuniquen entre sí. La dirección IP del controlador TXT 4.0 al que debe conectarse la aplicación puede consultarse a través del menú de la pantalla táctil en «Información» / «Wifi».

- Conexión a través de WAP: En lugar de «Wifi», puedes activar la opción «Access Point» en «Configuración» / «Red» en el TXT 4.0. Entonces podrás conectar el teléfono móvil directamente con el controlador. En la opción «Access Point» del menú del TXT puedes consultar (o modificar o desactivar) la clave WPA2 necesaria para la conexión wifi.

Una vez que hayas conectado la aplicación con el controlador, los comandos de voz se transmitirán en formato de texto al controlador y podrás evaluarlos con la siguiente función de eventos:



En tu programa de la tarea de programación 1, sustituye la función del botón para activar un lanzamiento por un comando de voz adecuado que tu teléfono inteligente pueda reconocer.

Tareas experimentales

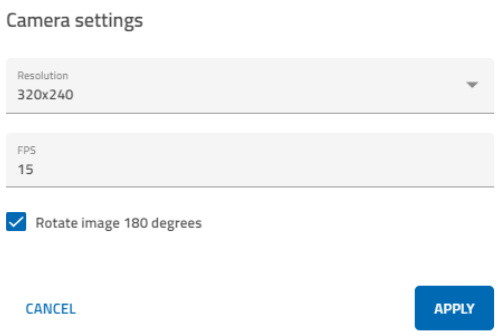
En las cuatro tareas siguientes, el vehículo Mecanum Omniwheel se equipa paso a paso con un sistema de búsqueda automática del objetivo.

Primero construye una diana de color siguiendo las instrucciones y colócala a una distancia de unos 50 cm delante de tu vehículo Mecanum Omniwheel.

1. Diana

1a. Activa la cámara en la configuración de la cámara. Configura el sistema de detección de bola de forma que el centro de la diana se encuentre exactamente en el centro de la ventana de detección cuando el dispositivo de disparo golpee el objetivo con la mayor fiabilidad posible. Prueba el ajuste con tu programa de la tarea de programación 1.

Nota: La cámara está de cabeza, por lo que se debe girar la imagen 180° en los ajustes de la cámara:



1b. Si acercas o alejas el vehículo de la diana, se modifica la coordenada «y» del sistema de detección de bola. Introduce en la siguiente tabla la coordenada «y» correspondiente a la distancia con respecto a la posición ideal.

Separación de distancia ideal	Coordenada y
15 cm	
10 cm	
5 cm	
0 cm	
-5 cm	
-10 cm	
-15 cm	

1c. A partir de esta medición, deduce una fórmula de aproximación sencilla con la que puedas calcular cuántos cm debe avanzar o retroceder el vehículo Mecanum Omniwheel para llegar a la distancia ideal con respecto a la diana.

2. Corrección de la distancia al objetivo

Escribe un programa Blockly que haga que el vehículo Mecanum Omniwheel se mueva la distancia correcta con respecto a la diana. Para esto, utiliza tus resultados de la tarea experimental 1.

Primero dibuja un diagrama de transición de estados.

3. Sistema de puntería

Escribe un programa Blockly que gire el vehículo Mecanum Omniwheel para que la diana quede exactamente en el centro de la ventana de detección.

Primero dibuja un diagrama de transición de estados para tu solución.

Consejo: La cámara tiene un ángulo de apertura de 60°.

4. Búsqueda de objetivo

Ahora debes combinar las soluciones desarrolladas en las subtareas anteriores. Escribe un programa Blockly que haga que el vehículo Mecanum Omniwheel primero busque una diana. Una vez que la haya encontrado, debe ubicar la diana exactamente en el centro de la mira y darle a la diana con las tres bolas de plástico huecas del cargador.

Primero ilustra tu concepto de solución con un diagrama de transición de estados.

Para finalizar puedes hacer que el TXT 4.0 reproduzca un sonido.

.

Anexos
Tarea n.º 5: Robot lanzabolas

Material necesario

- Ordenador para el desarrollo de programas, localmente o a través de la interfaz web.
- Cable USB o conexión BLE o wifi para transferir el programa al TXT4.0.
- Bolas de poliestireno

Más información

[1] FRC Team 2605 (Bellingham, WA): *How a Mecanum Drive Works*. github.io

[2] Wikipedia: *Autómata finito (máquina de estado finito)*

[3] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, Peter Wolstenholme: *Modeling Software with Finite State Machines. A Practical Approach*. Auerbach Publications, 2006.

[4] Editor de diagramas en línea para crear diagramas de transición de estados (formato drawio): https://www.diagrammeditor.de/

Tarea n.º 5: Robot lanzabolas

El vehículo Mecanum Omniwheels se equipa con un mecanismo de lanzamiento de bolas de poliestireno, un sistema de control por voz y una cámara que le permite apuntar de forma autónoma.

Tema

Evaluación de imágenes para determinar la distancia y apuntar al objetivo.

Objetivos de aprendizaje

- Análisis experimental de una tarea
- «Computational Thinking» (pensamiento computacional): descomposición de una tarea compleja en subtareas manejables y solucionables, que luego se combinan (principio de «divide y reinarás»).

Tiempo necesario

Para convertir el vehículo Mecanum Omniwheels de las tareas anteriores en un robot lanzabolas, los alumnos necesitan aprox. 90 minutos, y para construirlo desde cero siguiendo las instrucciones (suponiendo que ya tengan experiencia con fischertechnik), hasta 120 minutos.

Para desarrollar el programa de control que permita resolver las tareas de programación, los alumnos necesitan unos 60-90 minutos. El tiempo necesario para resolver las tareas experimentales es de 135-240 minutos, dependiendo de la edad y la experiencia.

Anexos
Tarea n.º 5: Robot lanzabolas

Material necesario

- Ordenador para el desarrollo de programas, localmente o a través de la interfaz web.
- Cable USB o conexión BLE o wifi para transferir el programa al TXT4.0.
- Bolas de poliestireno

Más información

[1] FRC Team 2605 (Bellingham, WA): *How a Mecanum Drive Works*. github.io

[2] Wikipedia: *Autómata finito (máquina de estado finito)*

[3] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, Peter Wolstenholme: *Modeling Software with Finite State Machines. A Practical Approach*. Auerbach Publications, 2006.

[4] Editor de diagramas en línea para crear diagramas de transición de estados (formato drawio): https://www.diagrammeditor.de/

Nombre: _____ Clase: ___ Fecha: _____

Hoja de soluciones de la tarea temática n.º 5

Robot lanzabolas

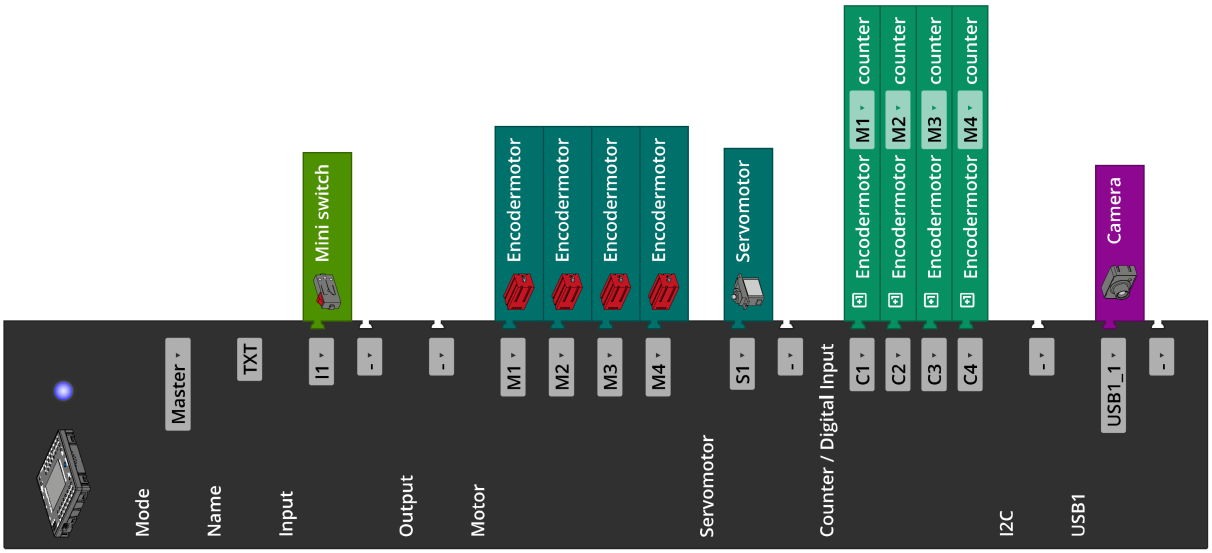
Las tareas son una práctica para el uso de bucles y funciones con el objetivo de obtener un programa claro, compacto y comprensible. Luego se compararán los resultados de los alumnos con relación a esto.

Tarea de construcción

Véase el manual de instrucciones.

Tareas de programación

Configuración de sensores y actuadores:

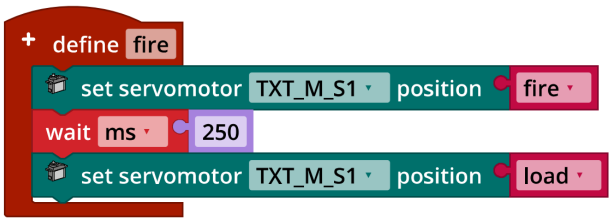


Para resolver la tarea de programación n.º 2 se necesita la aplicación «Voice Control» (para iOS o Android). Para el reconocimiento de voz, la aplicación debe estar conectada a Internet (a través de Bluetooth o wifi) y conectarse al controlador.

1. Mecanismo de expulsión

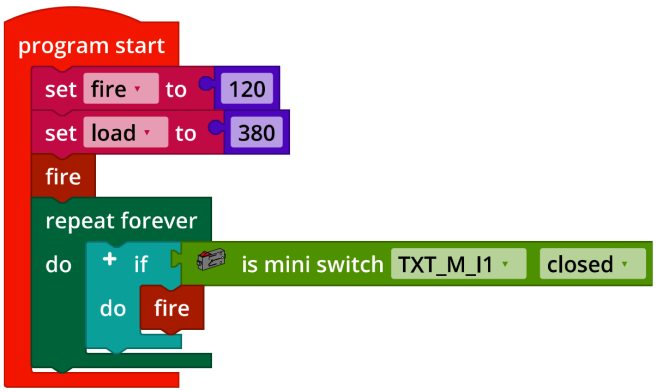
1a. Los valores adecuados para las variables «fire» y «load» (las posiciones buscadas de la servopalanca) pueden determinarse fácilmente con la prueba de interfaz. Estas variables dependen del modelo y, en particular, de la posición de la servopalanca acoplada. En el modelo utilizado aquí son 120 (fire) y 380 (load).

1b. Función «fire» (ejemplo):



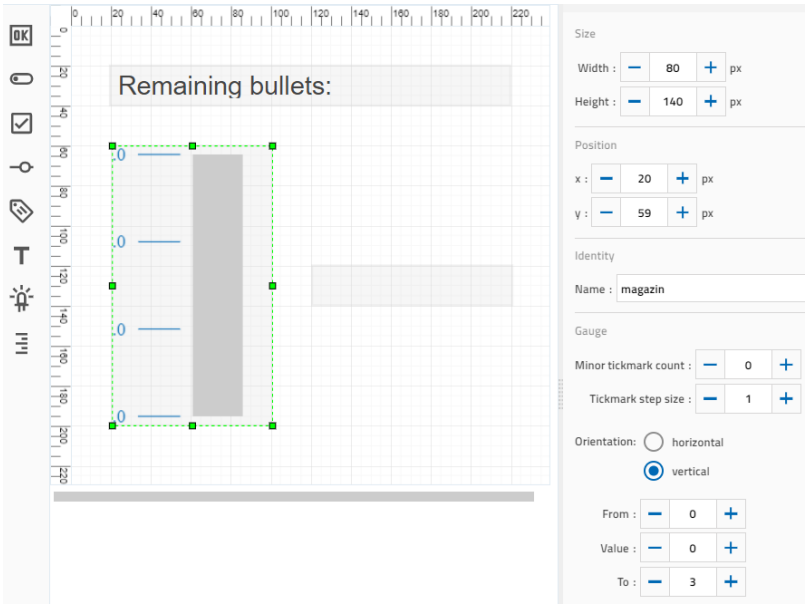
Mecanum_Fire_and_Load.ft

1c. Extracto del programa (ejemplo):

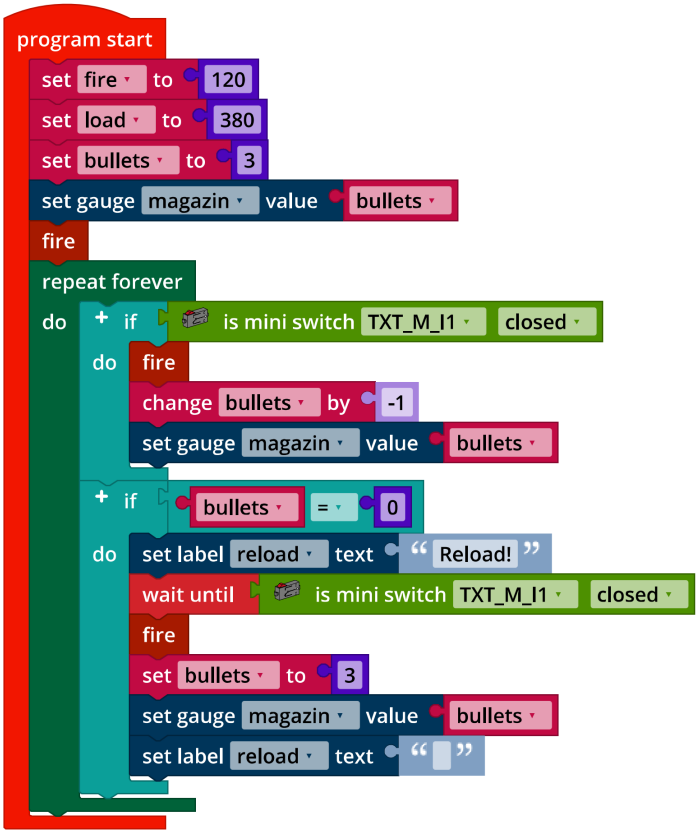


Mecanum_Fire_and_Load.ft

1d. Configuración de la pantalla TXT (ejemplo):



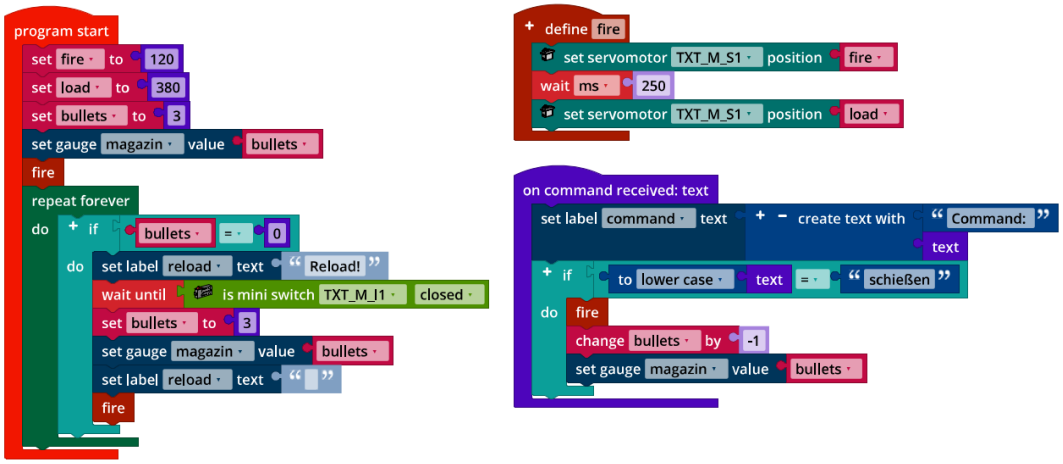
Programa (ejemplo):



Mecanum_Fire_and_Load_extended.ft

2. Control por voz

Programa (ejemplo):



Mecanum_Fire_and_Load_Voice_Command.ft

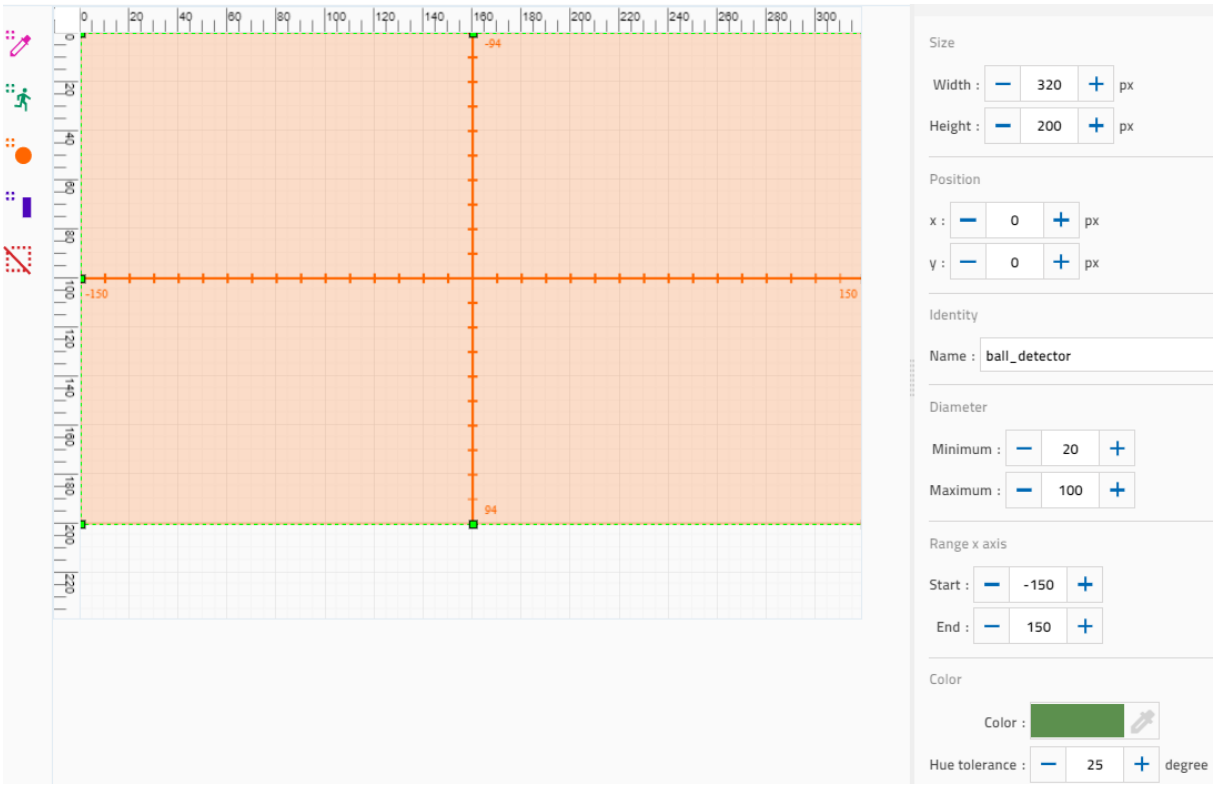
Tareas experimentales

1. Diana

1a. Lo ideal es colocar la diana lo suficientemente lejos como para que la bola de plástico hueca golpee la diana ligeramente por encima del centro.

En el modelo de ejemplo utilizado, esta distancia ideal es de unos 35 cm. Este valor puede variar ligeramente de un modelo a otro.

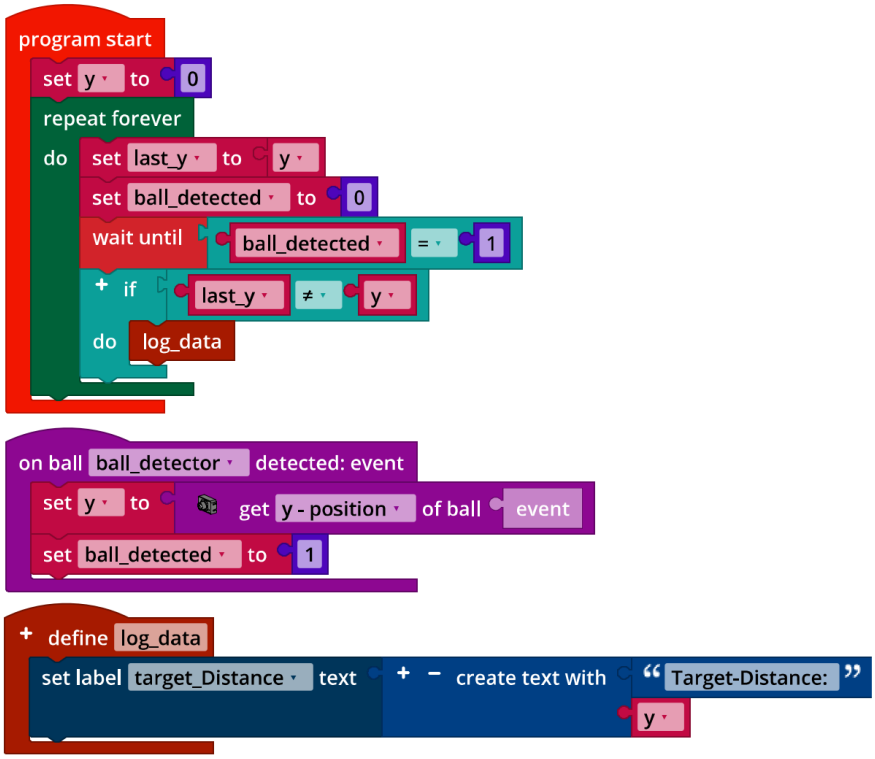
Configuración de la ventana de detección (detección de bolas):



Cuanto mayor sea la ventana de detección y el rango del diámetro de la bola a detectar, mayor será la desviación máxima de la distancia ideal que se puede detectar. Como rango para el eje X, se recomienda el intervalo [-150, 150]: Utiliza casi toda la resolución (320 píxeles) y puede convertirse fácilmente en un ángulo, ya que el ángulo de apertura de la cámara es de 60°.

El color seleccionado y la tolerancia del tono deben adaptarse a las condiciones de iluminación.

1b. Programa de medición de distancia (ejemplo):

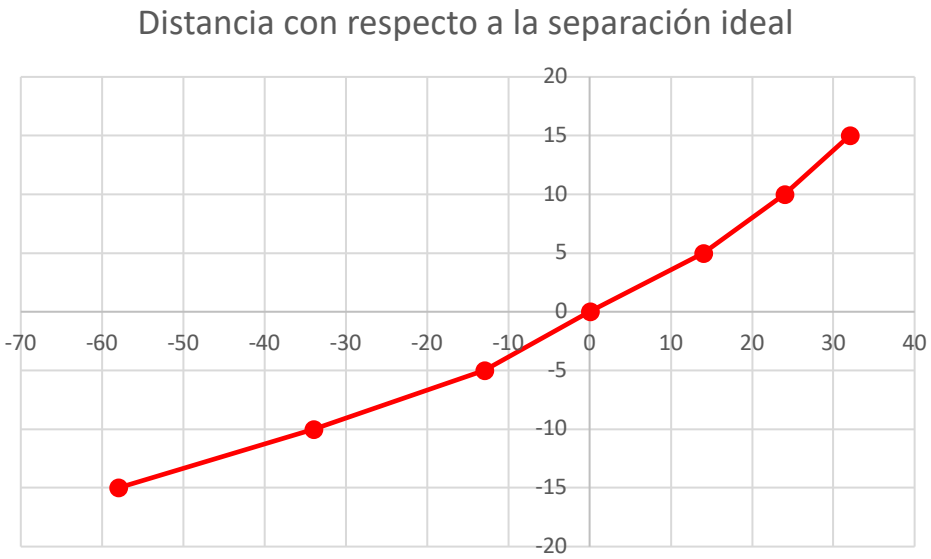


Test_Target_Distance.ft

Ejemplo de medición para la relación entre la distancia de la diana y el valor «y» del centro de la bola que proporciona el sistema de detección de bola:

Separación de distancia ideal	Coordenada y
+15 cm	32
+10 cm	24
+5 cm	14
0 cm	0
-5 cm	-13
-10 cm	-34
-15 cm	-58

Los valores de la coordenada «y» del centro de la bola devueltos por el sistema de detección de bola dependen de la escala seleccionada y de la altura de la ventana de detección. Sin embargo, los valores deben coincidir con los indicados en la tabla anterior con excepción de un factor de escala.



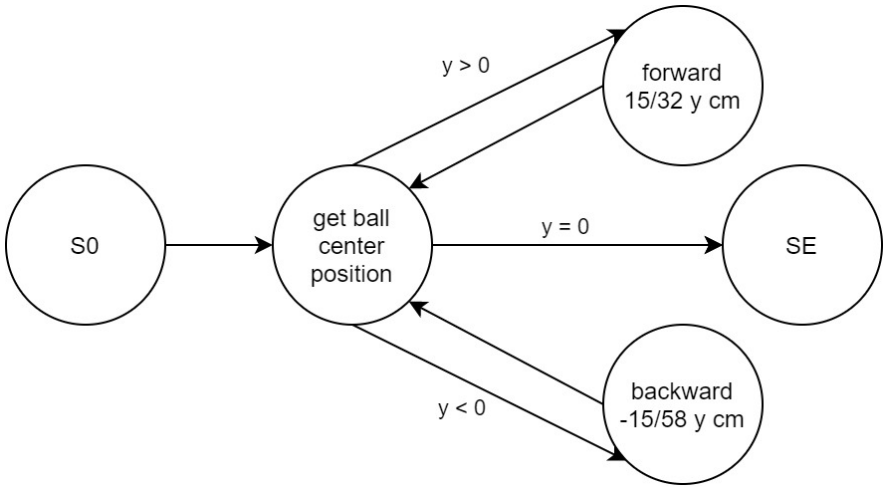
1c. La distancia d en la que el vehículo se aleja de la separación ideal con respecto al objetivo puede estimarse fácilmente usando dos funciones lineales:

$$d = -\begin{cases} y > 0: \frac{15}{32}y \\ y = 0: 0 \\ y < 0: \frac{15}{58}y \end{cases} \text{ cm}$$

Aquí también, el factor de incremento de la línea recta depende de la altura de la ventana de detección y de la escala establecida en el sistema de detección de bola y puede diferir en un factor.

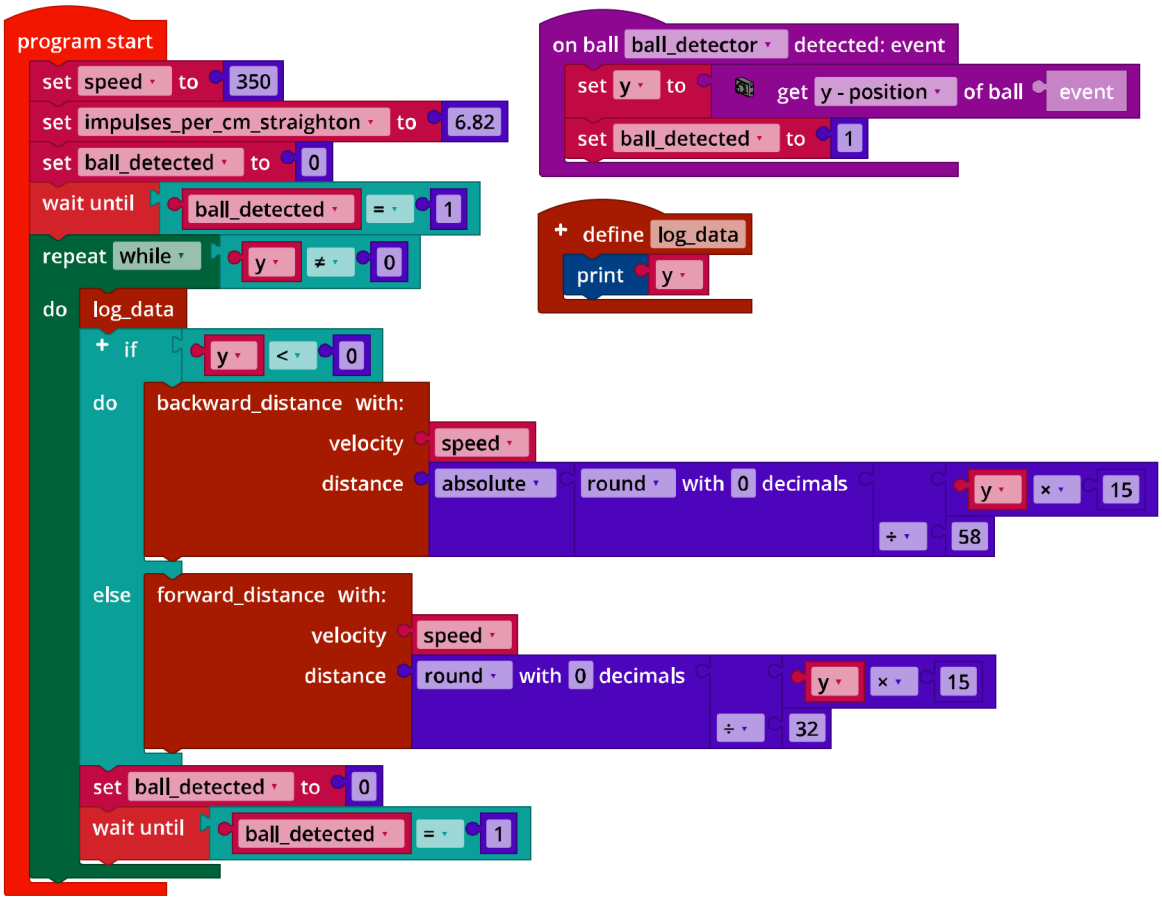
2. Corrección de la distancia al objetivo

2a. Diagrama de transición de estado:



State_Transition_Diagram_Correct_Position.drawio

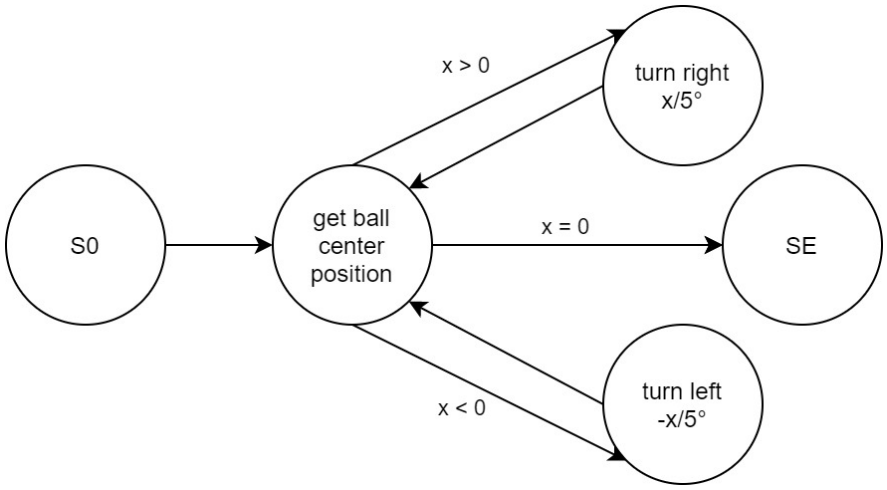
2b. Extracto del programa (ejemplo):



Mecanum_Correct_Distance.ft

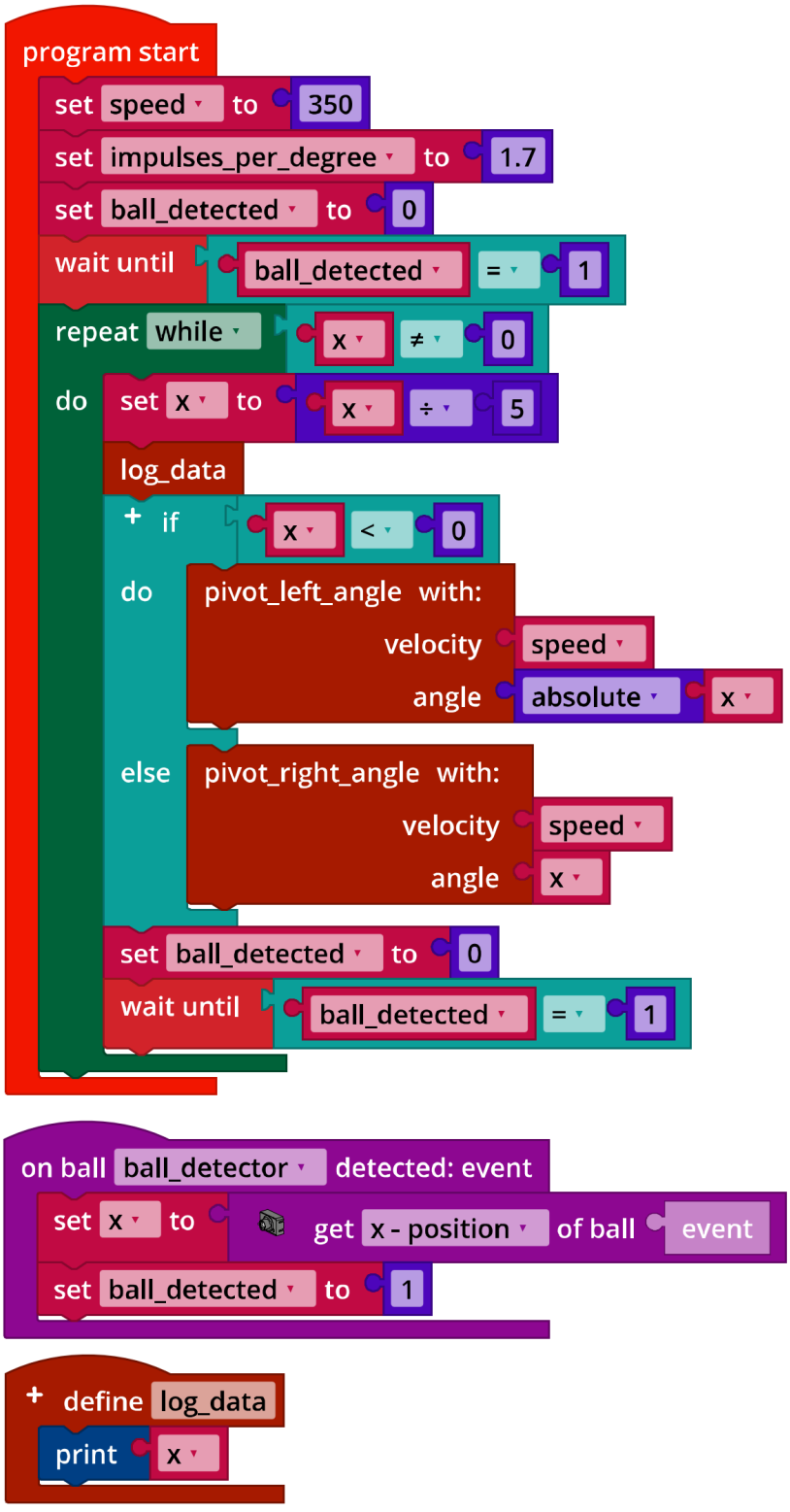
3. Sistema de puntería

3a. Diagrama de transición de estado:



State_Transition_Diagram_Correct_Position.drawio

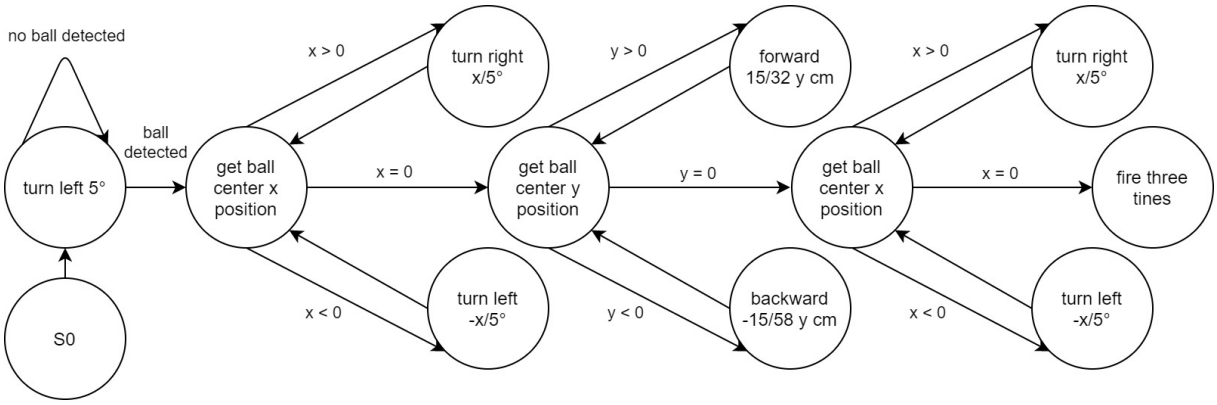
3b. Extracto del programa (ejemplo):



Mecanum_Turn2Target.ft

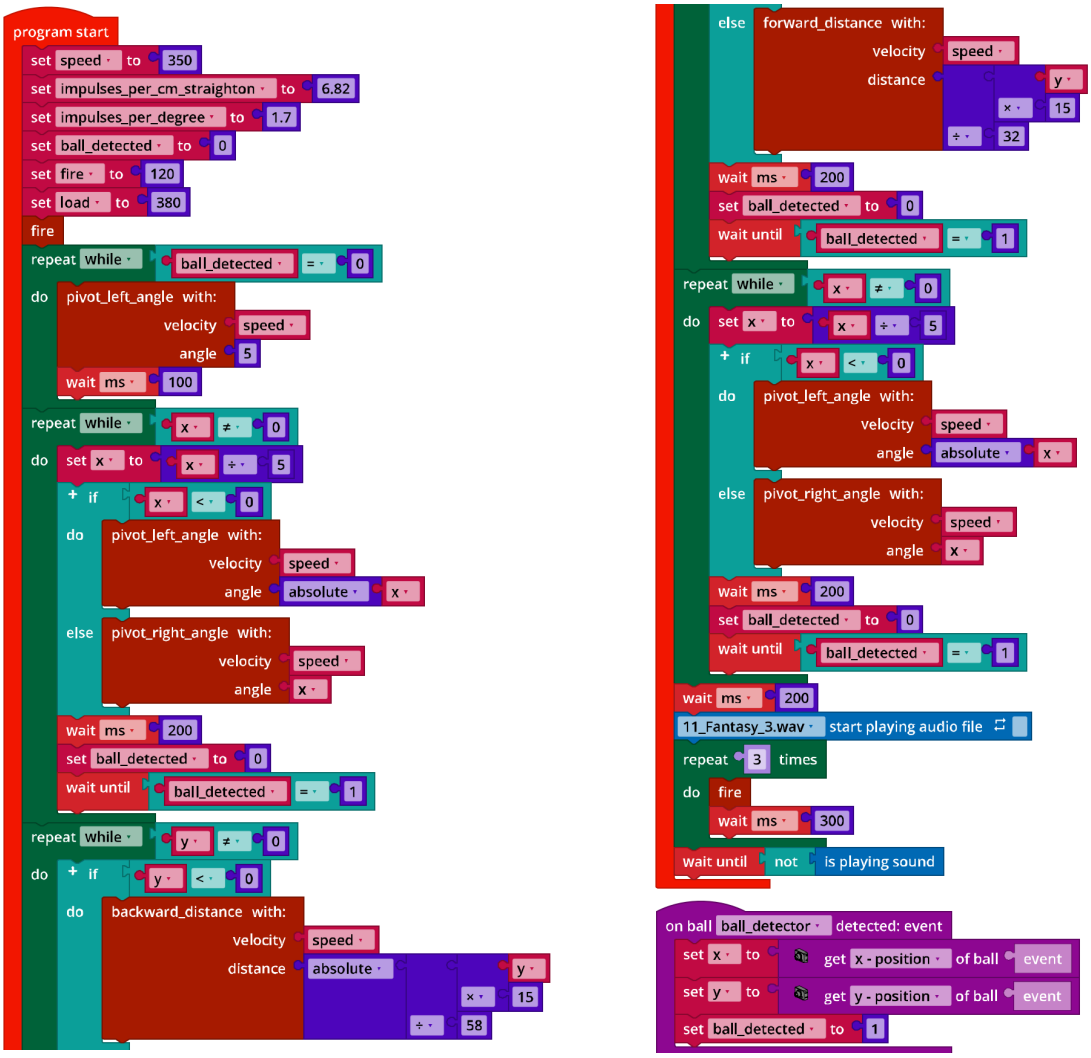
4. Búsqueda de objetivo

4a. Diagrama de transición de estado:



State-Transition_Diagram_Find_Target.drawio

4b. Extracto del programa (ejemplo):



Mecanum_Find_Target.ft

Anexos

Tarea n.º 5: Robot lanzabolas

Material necesario

- Ordenador para el desarrollo de programas, localmente o a través de la interfaz web.
- Cable USB o conexión BLE o wifi para transferir el programa al TXT4.0.
- Bolas de poliestireno

Más información

[1] FRC Team 2605 (Bellingham, WA): *How a Mecanum Drive Works*. github.io

[2] Wikipedia: *Autómata finito (máquina de estado finito)*

[3] Ferdinand Wagner, Ruedi Schmuki, Thomas Wagner, Peter Wolstenholme: *Modeling Software with Finite State Machines. A Practical Approach*. Auerbach Publications, 2006.

[4] Editor de diagramas en línea para crear diagramas de transición de estados (formato drawio): https://www.diagrammeditor.de/