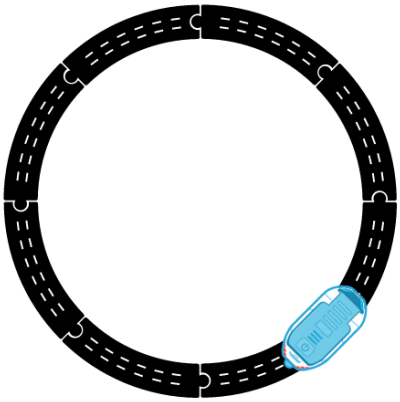
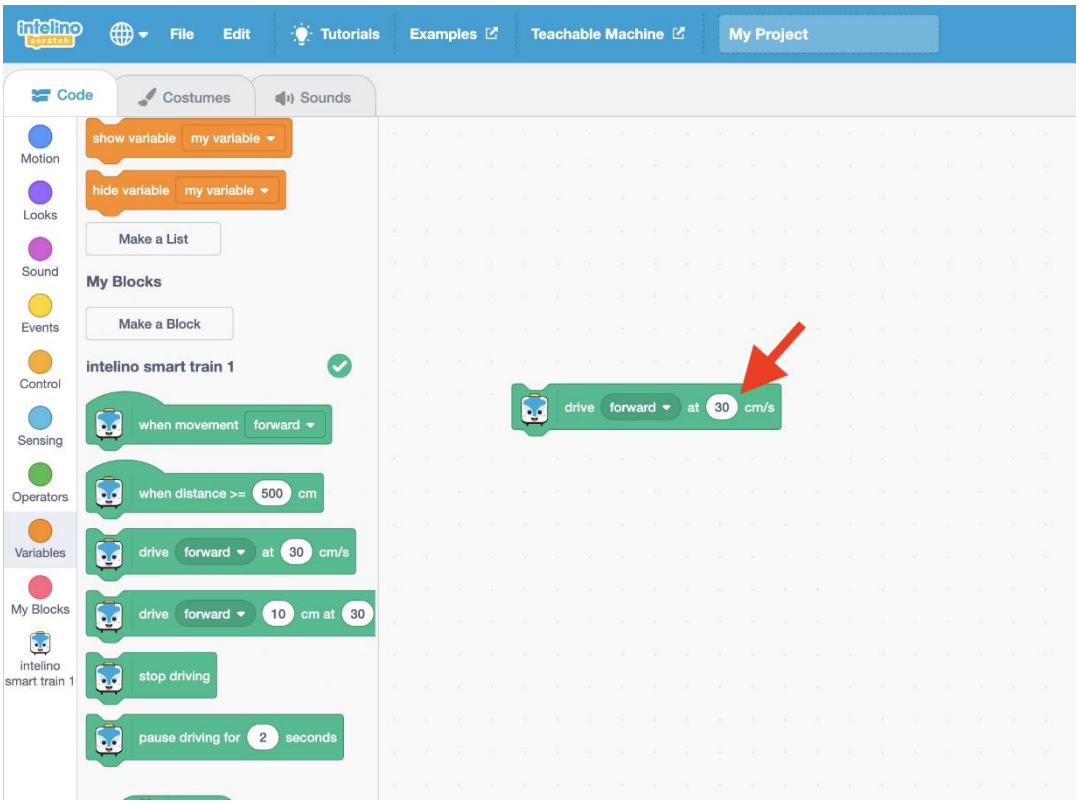

intelino
scratch

Primeiros Programas: 1. Condución

Imos dar unha volta co tren do intelino! Prende o tren e conéctao a intelino Scratch. Agora, imos configurar unha pista circular sinxela e colocar o tren nela:



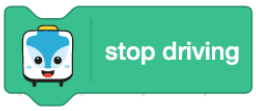
Busca o bloque de **condución**, arrástrao ao espazo de traballo do programa e fai clic nel.



Deberías ver o tren comezar a avanzar! Simple, non? O teu programa ten só un comando que lle indica ao tren que comece a conducir cara adiante a 30 cm/s.

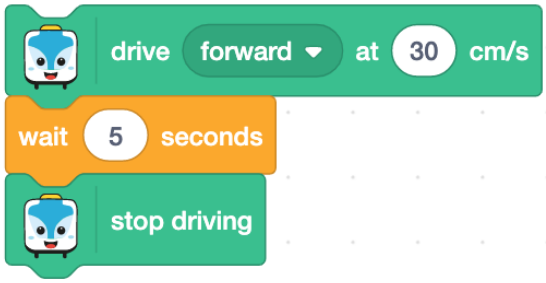
Se queres, proba a facer cambios no bloque, por exemplo. configure a dirección do movemento **cara atrás** ou cambie o valor da velocidade. A continuación, fai clic de novo no bloque para executar o teu comando modificado. Deberías ver que o tren responde co cambio desexado inmediatamente.

Xa que o tren aínda está en movemento, imos probar un comando que o **deteña.** Primeiro, elimina o bloque da unidade do espazo de traballo arrastrándoo á esquerda da pantalla. Despois atopa o bloque de parar de conducir, arrástrao ata o espazo de traballo e fai clic nel.



Como era de esperar, o tren pararase no momento en que executes o comando.

Agora que probamos o básico, imos montar unha secuencia de programa sinxela usando os dous bloques que probamos. Teña en conta que se simplemente engadimos un bloque de **parada de condución** xusto debaixo do bloque de **condución**, o tren nunca se movería. O motivo é que o comando de **parar a condución** executaríase inmediatamente despois do comando de **condución**, polo que o tren nunca tería a oportunidade de poñerse en marcha. Para que isto funcione, usemos un bloque de **espera** da categoría **Control** e insírelo xusto despois do bloque de **condución** como se mostra a continuación:

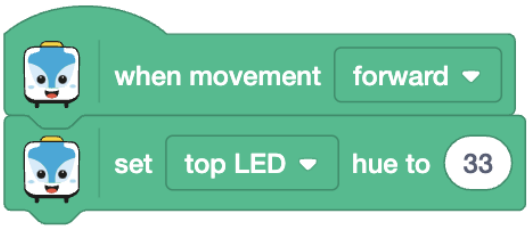


Como podes ver, establecemos a espera en 5 segundos. Agora, se fai clic no programa (o que o fai resaltado en **amarelo**), deberías ver que o tren fai exactamente o que di o programa: comezar a conducir e despois deterse exactamente despois de 5 segundos.

Primeiros Programas: 2. Eventos

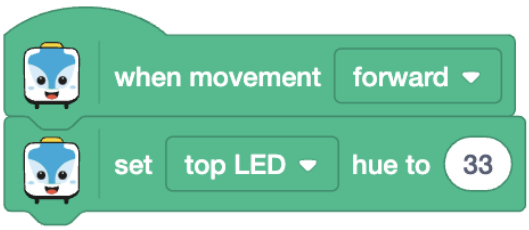
O tren Intelino usa os seus sensores para interactuar co seu entorno, incluíndo vostede, a vía, etc. Esas interaccións sensoriais chámanse **eventos** e o tren envía notificacións ao instante no momento en que ocorren tales eventos. **Scratch** pode escoitar diferentes tipos de notificacións de eventos cando chegan desde o tren. E podes usar **Scratch** para programar diferentes accións dependendo do tipo de evento recibido. Este enfoque de programación chámase **programación baseada en eventos** e é perfecto para controlar dispositivos robóticos como o tren intelixente Intelino.

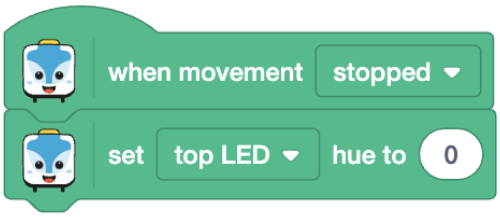
Imos probalo! Prende o tren e conéctao a **intelino Scratch**. Busca o bloque **"cando movemento"** e arrástrao ao espazo de traballo. Busca tamén o bloque **"Establecer a tonalidade [LED superior] en**" e colócao debaixo do bloque **"cando o movemento".** Establece o valor de matiz da cor en **33** que corresponde ao **verde.**



Agora prema brevemente o botón de acendido do tren para acender o motor. Deberías ver que o LED superior se ilumina en verde porque ao premer o botón cambiaches o estado de movemento a **"adiante".**

Ampliemos un pouco este programa. Fai clic co botón dereito no teu programa e fai clic en **Duplicar.** Por suposto, tamén podes crear este duplicado arrastrando novos bloques ao espazo de traballo, igual que fixeches anteriormente. No programa duplicado, imos establecer o valor do bloque **"cando movemento"** en **"detido"** facendo clic na lista despregable. E establece o valor de ton no bloque de cores en **0** que corresponde ao **vermello**. O teu programa actualizado no espazo de traballo debería verse así:

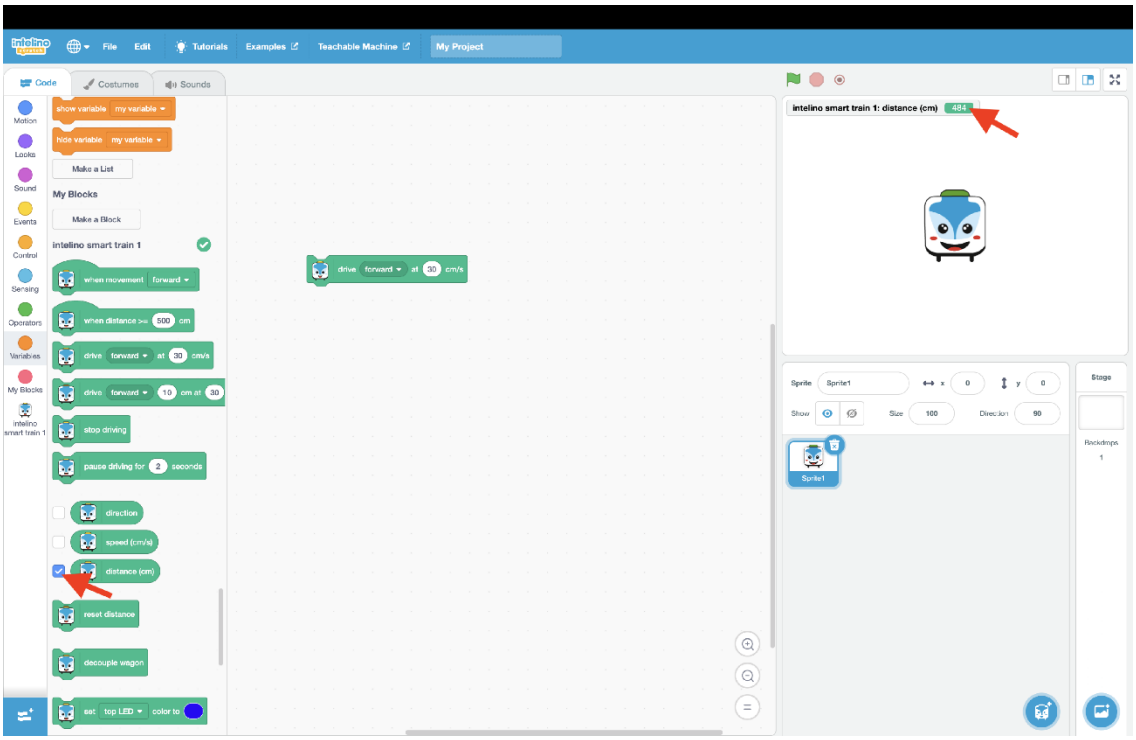




Agora, imos probalo. Do mesmo xeito que antes, premer brevemente o botón de acendido do tren debería acender o motor e iluminar o LED superior en cor verde. Pero se premes de novo o botón, o movemento deterase e o LED poñerase en vermello, tal e como programamos! Podes premer o botón tantas veces como queiras e verás que a cor do LED cambia co estado do movemento.

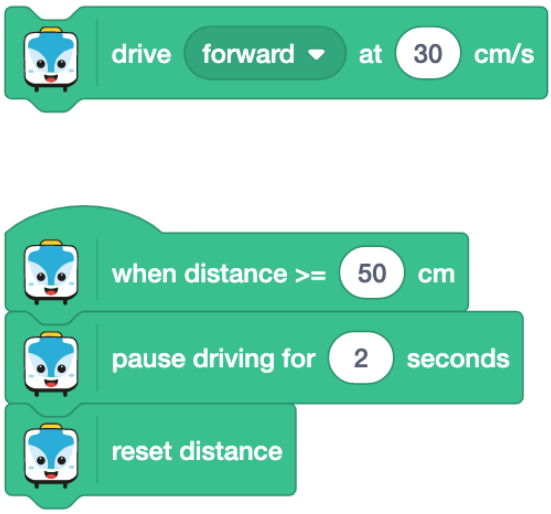
Primeiros Programas: 3. Distancia

O tren Intelino sempre segue a distancia que percorre. Cando conectas o tren a **Intelino Scratch** e creas un novo programa, establece a distancia en **0** e fai un seguimento dela mentres comezas a conducir o tren. Como se mostra na imaxe de abaixo, sempre podes ver a distancia actual na xanela superior dereita marcando a caixa situada a carón do bloque de **distancia**:



Engade un bloque de **condución** e fai clic nel para que o tren conduza ou simplemente enrólao nunha pista ou nunha mesa a man. Mentres o tren estea conectado a **Scratch**, verás que a distancia comeza a aumentar. Esta función pode ser moi útil cando queres que o tren faga algo en función da distancia percorrida.

Usemos o bloque de eventos **when distance >=** para crear un programa moi sinxelo que faga que o tren se deteña sempre despois dunha certa distancia:

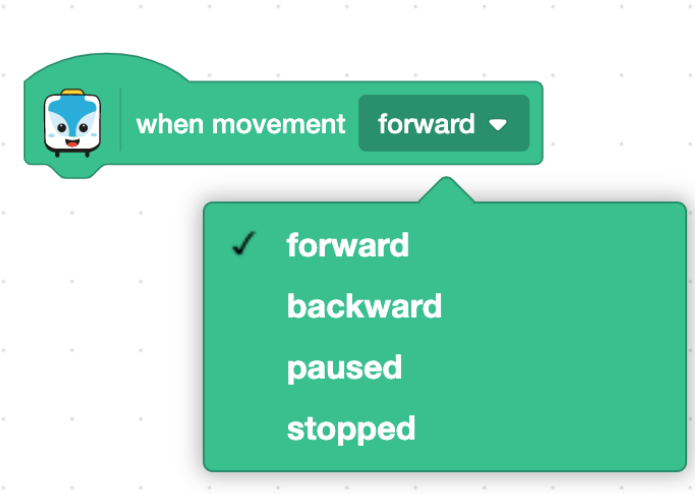


O bloque de condución inicia o movemento do tren. O bloque **when distance >= sempre** fai un seguimento da distancia percorrida e activa o código debaixo del no momento en que a distancia supera o valor escollido (**50 cm** neste exemplo). Para facer que o tren pare, engadimos o bloque de **pausa condución**. E tamén restablecemos o rastreador de distancia a **0** usando o bloque de **restablecemento de distancia** (se non, despois da primeira pausa, o tren seguiría avanzando e non volvería parar nunca máis).

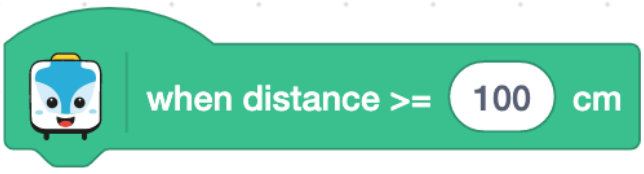
Monta este programa e probalo nunha pista que elixas. Non esquezas que tes que facer clic no bloque de **condución** para que o tren se mova. Pero tamén podes iniciar o movemento cun arranque ou premendo brevemente o botón de prendido.

Referencia de bloques de programación de Scratch

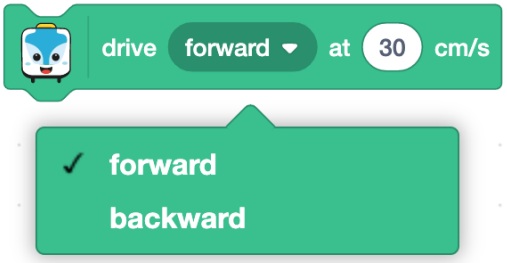
Intelino Scratch ofrece un conxunto completo de bloques de programación que controlan as moitas funcións do tren. A continuación móstrase a guía de referencia para todos os bloques de intelino soportados actualmente no noso editor:



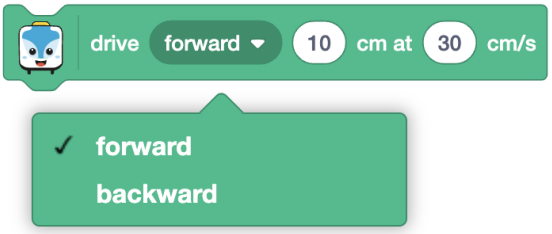
Fai algo cando o estado de movemento do tren cambie. A diferenza entre os estados en pausa e parado é que o primeiro indica un estado no que o tren está configurado para retomar a condución despois dun tempo (por exemplo, cando se le un comando de captura de acción en branco-vermello). Este último indicaba un estado no que o tren está a ralentí (por exemplo, o botón de acendido foi premido para parar o motor).



Fai algo cando a distancia percorrida do tren (en centímetros) sexa superior ou igual a algún valor.

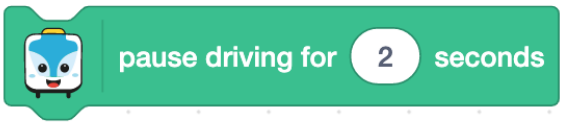


Comeza a conducir cara a adiante ou cara atrás a unha velocidade establecida (en centímetros por segundo)

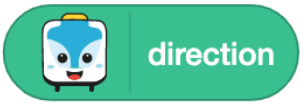


Conduce cara adiante ou cara atrás unha distancia determinada (en centímetros) a unha velocidade determinada (en centímetros por segundo).

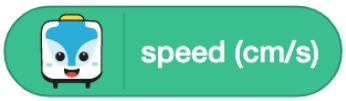
O tren deixa de moverse.



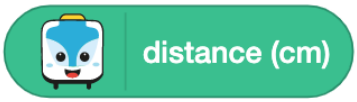
O tren deixa de moverse durante algún tempo (en segundos)



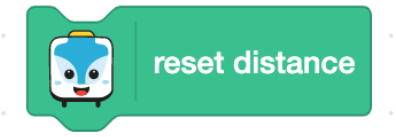
Obter o valor da dirección de xiro para a seguinte división. Este bloque pode ser útil cando necesitas comprobar se xa hai unha opción almacenada no tren.



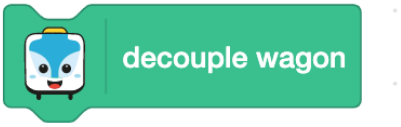
Obtén a velocidade actual do tren (en centímetros por segundo).



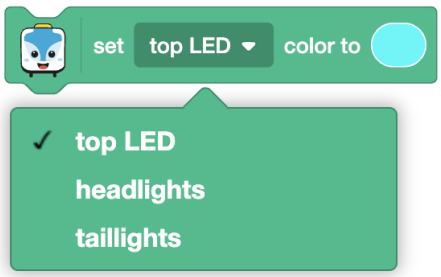
Obtén a distancia actual percorrida do tren (en centímetros).



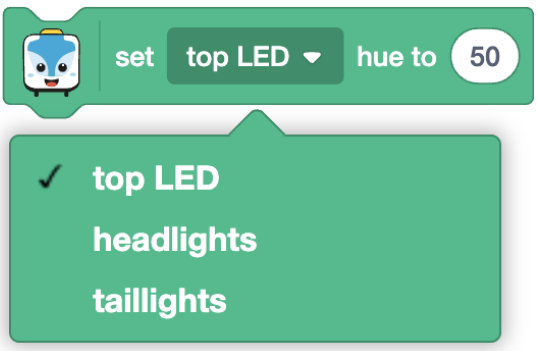
Restablece a distancia percorrida a 0.



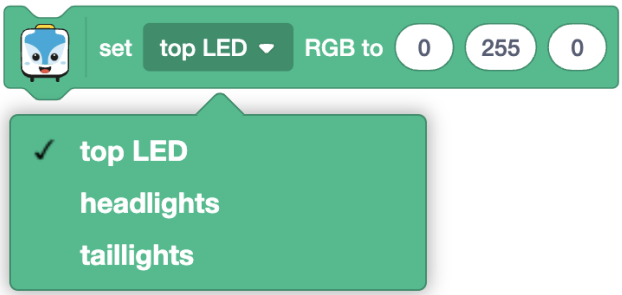
Desacoplar o vagón do tren.



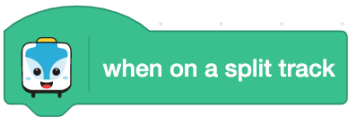
Establece a cor do LED superior, dos faros ou das luces traseiras mediante un selector de cores HSV.



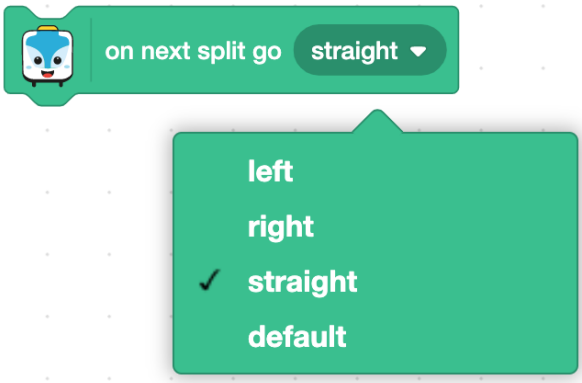
Establece a tonalidade de cor do LED superior, dos faros ou das luces traseiras. O valor da tonalidade varía entre 0 e 99. Pense na tonalidade como un espectro circular de matices de cores. Así, sempre que o seu valor desborde (por exemplo, é superior a 100), a cor volve ao principio.



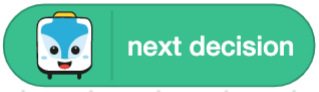
Establece a cor RGB do LED superior, dos faros ou das luces traseiras. Os valores RGB oscilan entre 0 e 255



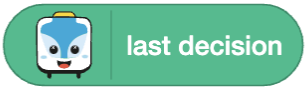
Fai algo cando o tren detecte unha vía dividida. Este bloque pódese usar para definir a decisión da dirección de dirección para a **seguinte** división.



Establece a decisión da dirección de desvío para a **seguinte** división. O valor **predeterminado** significa que a decisión se axustará á elección do tren (por exemplo, aleatoria ou definida con complementos de cores).

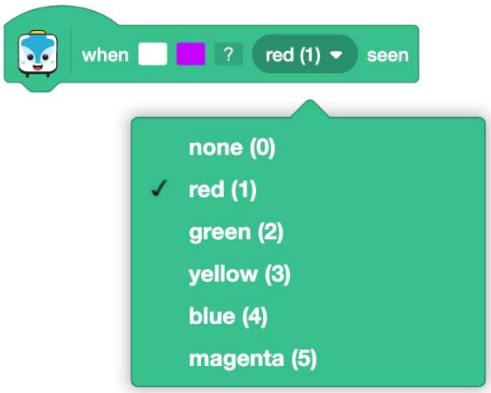


O seguinte operando de decisión de dirección usado nas instrucións condicionais.

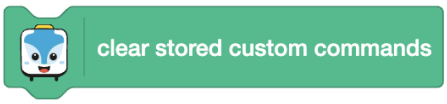


Último operando de decisión de dirección usado en declaracións condicionais.

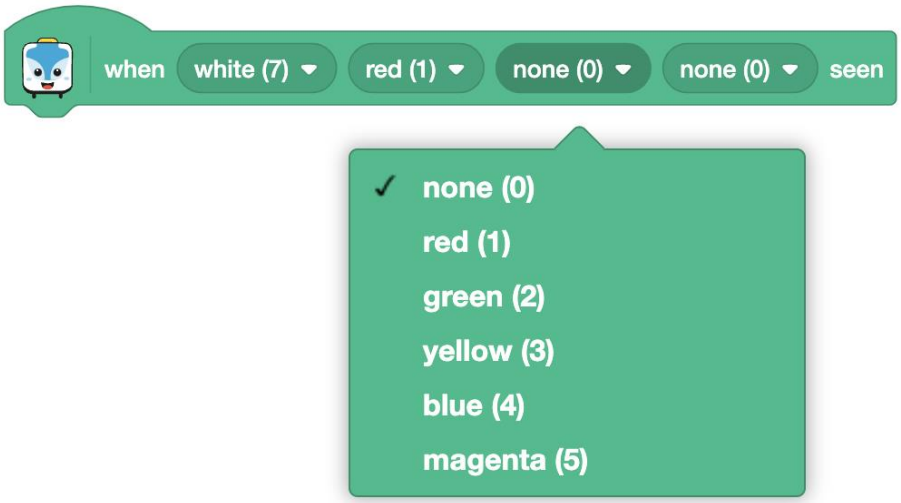
Valor do operando da decisión de dirección utilizado nas declaracións condicionais.



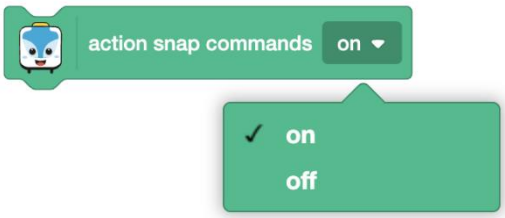
Fai algo cando o tren detecte unha secuencia de comandos de cor personalizada. O comando debe comezar con cores branco - maxenta e pode seleccionar a cor do 3º instante na lista despregable. Teña en conta que este bloque acepta variables (consulte os valores de cor numéricos entre parénteses).



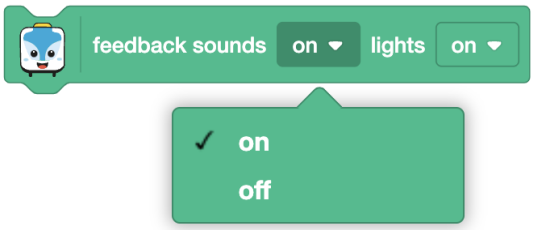
Borra as ordes personalizadas almacenadas no tren mediante o **Editor de Axustes** na aplicación móbil **intelino play**. Por exemplo, digamos que creou e almaceu un comando personalizado usando o **Editor de axustes** usando a secuencia **branco-maxenta-amarelo**. Se usas a mesma secuencia de cores en Scratch para facer algo, o tren tamén executará o que está almacenado no tren. Nalgúns casos, este pode ser o teu comportamento previsto. Pero se o comando almacenado no tren entra en conflito co que estás facendo en Scratch, podes usar este bloque de **"borrar comandos personalizados almacenados".**



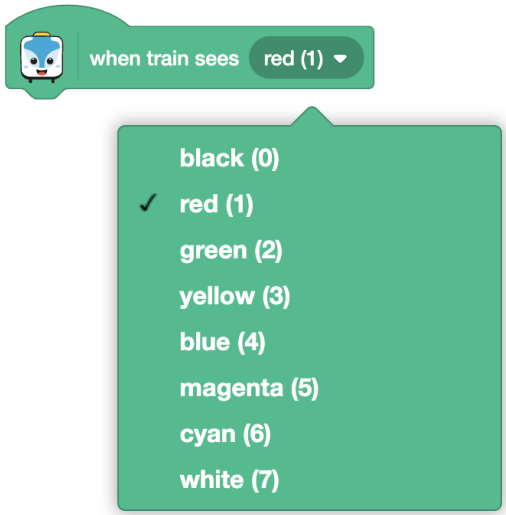
Fai algo cando o tren detecte unha secuencia de comandos personalizada de ata 4 cores. O comando debe comezar por **branco ou cian** e pode seleccionar a cor do **2º, 3º e 4º pestanas** nas listas despregábeis. Teña en conta que este bloque acepta variables (consulte os valores de cor numéricos entre parénteses).



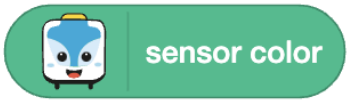
Activa ou desactiva os comandos predeterminados de captura de accións almacenados no tren. Se usas a mesma secuencia de cores en Scratch para facer algo, o tren tamén executará o que está almacenado no tren. Nalgúns casos, este pode ser o teu comportamento previsto. Por exemplo, o comando BRANCO-VERMELLO fai que o tren se pare durante 2 segundos. E se usas o bloque anterior para, por exemplo, facer que o comando BRANCO-VERMELLO cambie a cor do LED superior, ambos 2 seg. parando e executaríase a cor do LED. No caso de que o comando almacenado no tren entre en conflito co que estás facendo en Scratch, podes **desactivar** este bloque de **"comandos de captura de accións".**



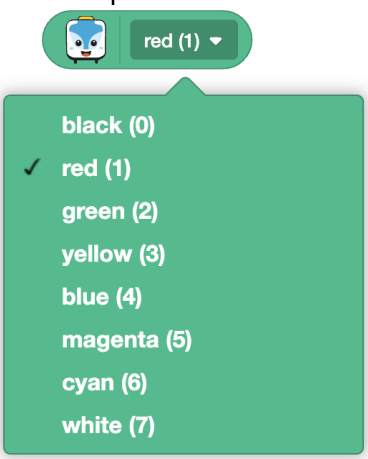
Activa ou desactiva os efectos de son e luz asociados aos comandos predeterminados de captura de accións. Pode querer configurar estes efectos no estado **"desactivado"** se interfiren co que está a programar en Scratch.



Fai algo cando os sensores de cor do tren detecten unha cor. Teña en conta que este bloque acepta variables (consulte os valores de cor numéricos entre parénteses).



Obtén o valor da cor detectado polo sensor de cor.



Operador de cor usado en instrucións condicionais.