Lekcja 5:
Śledzenie linii

Tematyka: STEAM
Klasa: 5 (i wyższe)
Czas: 45 minut
Stopień trudności: początkujący

Cele lekcji

Po ukończeniu lekcji uczniowie będą wiedzieli:

- Jak działają czujniki koloru
- Jakie jest zastosowanie tych czujników w życiu codziennym i w robotyce

Informacje ogólne

Wyobraźcie sobie, że jesteście w obcym mieście i jedziecie autokarem turystycznym, który obwozi Was po najpiękniejszych miejscach. Być może odbyliście już taką podróż, a jeśli nie, to pewnie możecie sobie ją wyobrazić ze zdjęć lub z telewizji czy internetu. Podczas tej lekcji przekształcimy naszego mBota2 w taki właśnie autokar wycieczkowy.
Ponieważ jazda tą samą trasą i objaśnianie tych samych punktów turystycznych każdego dnia byłyby nudne dla kierowcy, zaprogramujemy nasz autokar, aby automatycznie wykonywał te czynności. W ten sposób turyści będą mogli cieszyć się przyjemną jazdą, a kierowca nie będzie się nudził powtarzalnymi czynnościami.

Najważniejsze zagadnienia

Po ukończeniu lekcji uczniowie będą umieli:

- Zaprogramować mBota2, aby podążał za linią
- Zaprogramować mBota2, aby wykonywał określone działania w oparciu o kolory

Lista niezbędnych narzędzi

Co należy przygotować:

- Komputer PC lub laptop (z wyjściem USB) z zainstalowanym oprogramowaniem mBlock, wersję webową (również dla ChromeBooka), lub tablet z zainstalowaną aplikacją mBlock
- mBot2 + CyberPi
- Kabel USB-C lub adapter bezprzewodowy Makeblock Bluetooth

Plan lekcji

Lekcja ta składa się z 4 kroków i trwa 45 minut.

Czas przetwarzania	Spis treści
5 minut	**1. Rozgrzewka** • Czujniki koloru w życiu codziennym • Zasada działania czujników koloru Cechy szczególne czujnika koloru w mBocie2
10 minut	**2. Teoria i praktyka** • Zapoznanie z różnymi blokami kodów wykorzystywanymi do programowania czujnika Quad RGB • Odtworzenie i przetestowanie kilku przykładów kodów do sterowania czujnikiem Quad RGB • Zaprogramowanie mBota2, aby podążał za linią
25 minut	**3. Rozwiązywanie zadania** • Pisanie własnego programu dla robota
5 minut	**4. Podsumowanie** • Czas na prezentację: pokażcie wyniki swojej pracy z mBotem2 w formie zabawnego, krótkiego filmiku, które posłuży do późniejszej dyskusji • Za zgodą nauczyciela możecie podzielić się efektem końcowym w mediach społecznościowych z hashtagiem #mBot2. • Własne przemyślenia: Z czego jesteście najbardziej dumni? Co chcielibyście poprawić w swoim robocie?

1. Rozgrzewka
(5 min.)

Krok 1: Rozgrzewka

Krok ten składa się z dwóch części:

1. Czujniki koloru w życiu codziennym
2. Zasada działania czujników koloru Jakie są cechy szczególne czujnika koloru w mBocie2?

1. Czujniki koloru w życiu codziennym

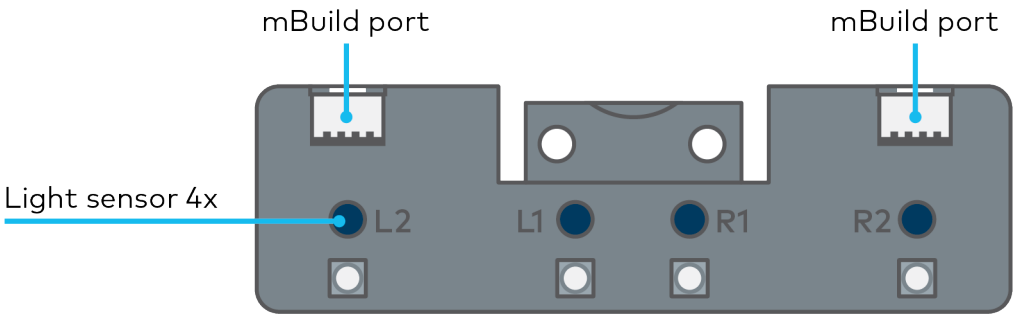
Czujniki koloru służą do wykrywania natężenia światła i barw. Znajdują one wiele różnorodnych zastosowań w życiu codziennym.

Dla przykładu, robot magazynowy może znaleźć właściwą trasę na podstawie różnych kolorów oznakowań na podłodze magazynu. Ten rodzaj czujnika jest również wykorzystywany, gdy wymagane jest precyzyjne dopasowanie kolorów, na przykład do lakierowania pojazdów. W tym przypadku urządzenie pomiarowe z czujnikiem koloru sprawdza, czy podczas malowania pojazdu sprayem użyto właściwego koloru. Czy możecie podać inne przykładów zastosowań?

2. Zasada działania czujników koloru Cechy szczególne czujnika koloru w mBocie

Czujnik koloru składa się z trzech różnych wewnętrznych czujników, które mierzą światło odbite przez obiekt w oparciu o intensywność wartości kolorów czerwonego (red), zielonego (green) i niebieskiego (blue). Dlatego tez czujniki koloru są również nazywane czujnikami RGB. Każdy kolor jest dekodowany przez czujnik na trzy wartości, a określona kombinacja tych wartości reprezentuje dany kolor.

mBot2 ma cztery takie czujniki RGB zintegrowane w jeden pojedynczy czujnik (Quad RGB). Spójrzcie na poniższą ilustrację.



Ilustracja przedstawia czujnik Quad RGB zamontowany w mBocie 2. Jego elementy są ponumerowane L1, L2, R1, i R2 (L dla lewej strony, R dla prawej). Automatycznie wykrywają one

wartości RGB odbitego koloru i porównują wewnętrznie ich kombinację z różnymi ustawieniami wstępnymi. Ułatwia to znacznie kodowanie, ponieważ nie trzeba sprawdzać kodów RGB samodzielnie. Czujnik wykrywa 6 różnych kolorów, jak również czarny i biały. Zamiast kolorów może też wykrywać odcienie szarości lub linie i skrzyżowania.

Czujnik Quad RGB pozwala mBotowi2 podążać za linią na podłodze i reagować na różnokolorowe znaczniki. Jest on zamontowany z przodu mBota2. Wszystkie cztery czujniki widoczne są patrząc od dołu. Patrząc od góry można odczytać nazwę każdego z nich, dzięki czemu można je zidentyfikować i zaprogramować w określony sposób. Spójrzcie na poniższą ilustrację.

Czujnik Quad RGB, widok z przodu

Mały przycisk na górnej stronie czujnika służy do kalibracji. Po kliknięciu go dwukrotnie, diody LED zaczną migać. Należy wtedy "przeciągnąć" mBota2 z czujnikiem po linii, którą ma śledzić. Spowoduje to kalibrację czujnika, aby odróżniał linię od tła. W przypadku czarnej linii na białej powierzchni, kalibracja nie jest zwykle wymagana.

Mapa dostarczona z mBotem2 ma zaznaczone kolory wewnątrz toru. Może się zdarzyć, że czujnik odbierze jasny kolor (np. żółty) jako tło, a nie jako tor. Dlatego też w pierwszym ćwiczeniu należy skalibrować czujnik na żółty kod koloru. Wtedy będzie on zawsze rozpoznawał ten, jak i wszystkie ciemniejsze kolory jako linie.

Podczas programowania mBota2, będziecie używać jednego lub kilku z tych czujników do sprawdzania, czy na podłodze po drodze nie ma określonych kolorów. Możecie dla przykładu kazać mBotowi2 wykonać określone polecenie po wykryciu danego koloru (np. zatrzymanie dla koloru czerwonego i skręt w lewo dla zielonego).

makeblock education

2. Teoria i praktyka (10 min.)

Krok 2: Teoria i praktyka

Krok ten składa się z trzech części:

1. Zapoznanie z różnymi blokami kodów wykorzystywanymi do programowania czujnika Quad RGB
2. Odtworzenie i przetestowanie kilku przykładów kodów do sterowania czujnikiem Quad RGB
3. Zaprogramowanie mBota2, aby podążał za linią

1. Zapoznanie z różnymi blokami kodów wykorzystywanymi do programowania czujnika Quad RGB

W mBlock dostępnych jest kilka bloków kodu, których możecie użyć do programowania czujnika Quad RGB. Bloki te można znaleźć w zakładce "Quad RGB" w mBlock. Są one zaznaczone kolorem zielonym.

W poniższej tabeli przedstawiono niektóre z nich. Pierwsze cztery bloki kodu służą do prostej identyfikacji linii. Mogą one być używane do śledzenia linii, a także do wykrywania skrzyżowań i ostrych (90°) zakrętów. Kalibracja czujnika do wykrywania różnych kolorów torów została pokrótce opisana w rozdziale powyżej. Niebieskie diody LED na górnej stronie czujnika wskazują detekcję linii/tła: jeśli są zgaszone, oznacza to, że wykryty został ciemny kolor (linia), jeśli świecą na niebiesko, wskazują jasny kolor (tło).

Bloki kodów:

Przedstawione bloki kodu wykorzystują jedynie dwa wewnętrzne czujniki L1 i R1 do wykrywania linii. Bloki te są przeznaczone do prostego śledzenia linii przez mBota2.
Pierwszy blok należy stosować w instrukcjach warunkowych. Drugi z kolei należy stosować, jeżeli wartość ma być wyświetlona na ekranie lub zapisana w postaci zmiennej.

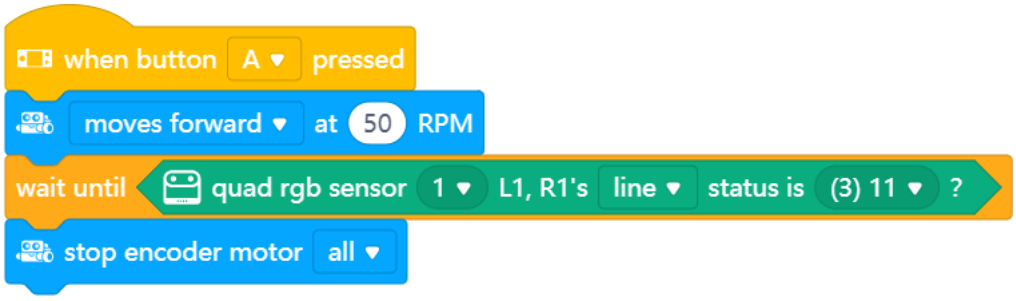
Jeśli czujnik znajduje się nad czarną linią, to obie wewnętrzne diody LED powinny być wyłączone. Ponieważ każdy czujnik (L1, R1) może wykrywać linię lub tło, w bloku tym mamy do dyspozycji 4 możliwe kombinacje: będą one reprezentowane przez liczby od 0 do 3 (blok pokazuje również wzór binarny):

L1	R1	Znaczenie	Status (dziesiętny)
		Czujnik na białym tle, obie diody LED się świecą	0
		R1 wykrywa czarną linię	1
		L1 wykrywa czarną linię	2
		Oba czujniki wykrywają czarną linię	3

Pamiętajcie, że diody LED pokazują stan tła (włączone lub "prawda" logiczna = wykryte tło), podczas gdy ten blok kodu wykrywa kolor linii ("prawda" logiczna = wykryta linia).

Jeśli chcielibyście użyć więcej niż jednego czujnika Quad RGB, możecie zdecydować, który czujnik ma być użyty, podając liczbę we wszystkich kolejnych blokach kodów. Spójrzcie na poniższą ilustrację. Numer 1 oznacza pierwszy podłączony czujnik, numer 2 drugi podłączony czujnik i tak dalej.

W poniższym przykładzie wykorzystamy funkcję wykrywania linii tego bloku, aby zatrzymać robota, gdy tylko dotrze do czarnej linii. Przykład ten zostanie później zmodyfikowany na podążanie za linią.



Jako że czas ma w tym przypadku duże znaczenie, ważne jest, aby podczas programowania używać trybu Upload. Możecie sami wypróbować oba tryby, aby zobaczyć różnicę.

Bloki kodów:

Pierwszy blok należy stosować w instrukcjach warunkowych. Drugi z kolei należy stosować, jeżeli wartość ma być wyświetlona na ekranie lub zapisana w postaci zmiennej.

Bloki te działają dokładnie w ten sam sposób, co poprzednie, z tą różnicą, że mają większy obszar wykrywania i mogą identyfikować skrzyżowania, z zachowaniem śledzenia linii. Należy pamiętać, że bloki sprawdzają status linii, więc jeśli jeden z czterech czujników zidentyfikuje ciemną linię, odpowiednia cyfra binarna zostanie ustawiona na "1" (logiczna prawda), a diody LED zgasną.

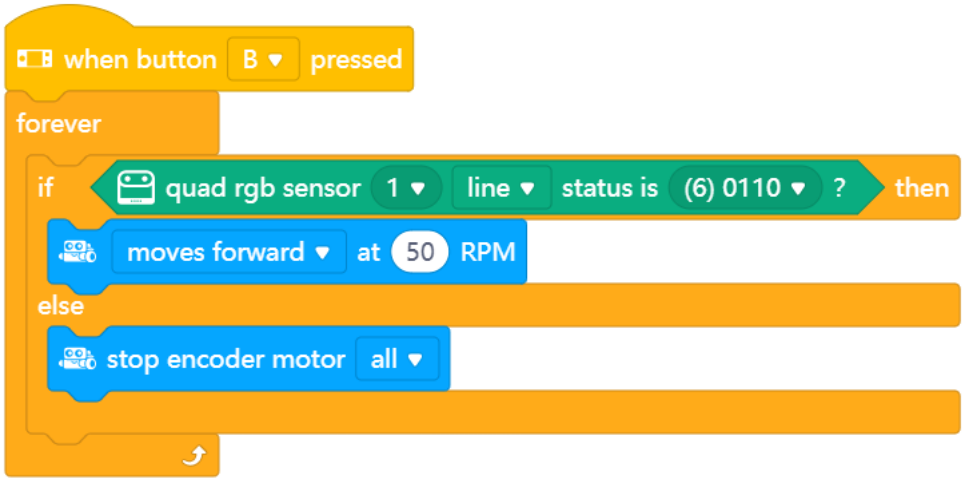
Przy 4 pojedynczych czujnikach zakres możliwości zwiększa się do 16 różnych statusów. Poniższa tabela przedstawia najważniejsze z nich:

Wzór binarny	Znaczenie	Status (dziesiętny)
0000	wszystkie 4 czujniki na białym tle (brak linii)	0
0010	tylko R1 na czarnej linii	2
0100	tylko L1 na czarnej linii	4
0110	czujniki wewnętrzne R1 i L1 na czarnej linii	6
0111	ostry zakręt w prawo (L1 i R1 na czarnej linii, skrzyżowanie w prawo; skrzyżowanie w kształcie litery L w prawo)	7
1110	ostry zakręt w lewo (L1 i R1 na czarnej linii, skrzyżowanie w lewo; skrzyżowanie w kształcie litery L w lewo)	14
1111	skrzyżowanie w kształcie litery T (L1 i R1 na czarnej linii, skrzyżowanie po prawej i lewej stronie)	15

Skrzyżowanie w kształcie litery X może być wykryte tylko poprzez przejechanie przez skrzyżowanie w kształcie litery T i sprawdzenie, czy linia środkowa biegnie dalej.

W poniższym przykładzie robot będzie jechał tak długo, jak długo czarna linia będzie

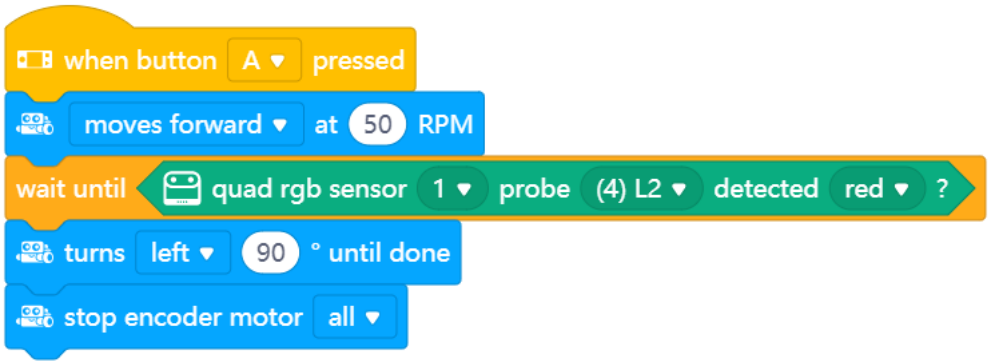
wykrywana przez dwa wewnętrzne czujniki. Zatrzyma się, jeśli zjedzie z toru, zostanie wykryte skrzyżowanie lub linia nagle się skończy:



Blok kodu:

Za pomocą tego bloku kodu można wybrać jeden z czujników Quad RGB (L2, L1, R1, R2) i określić, czy ma on rozpoznawać linię, tło, kolor biały, czarny lub dowolny inny z 5 dostępnych kolorów (czerwony, zielony, niebieski, żółty, cyjan, fioletowy). Do wykrywania i śledzenia linii zalecane są poprzednie bloki, ponieważ sprawdzają one więcej niż jeden czujnik naraz (jest to szybsza i dokładniejsza metoda, ponieważ robot nie porusza się pomiędzy odczytami). Można jednak użyć tego bloku do wykrywania na przykład czerwonego kolorowego znacznika po lewej stronie robota, gdy robot podąża za linią.

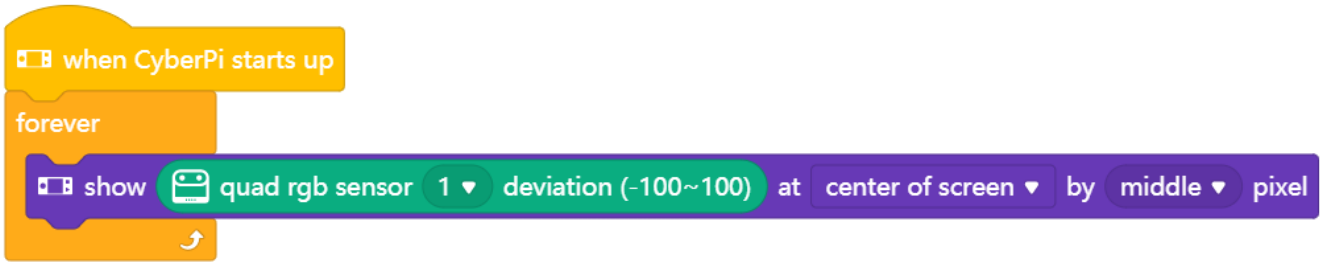
W poniższym przykładzie po naciśnięciu przycisku B, mBot2 pojedzie prosto (zakładając, że podąża za linią) i wykona obrót o 90°, gdy po lewej stronie toru zostanie wykryty czerwony znacznik.



Blok kodu:

Za pomocą tego bloku można zmierzyć, jak daleko robot jest oddalony od czarnego toru. Można go użyć do zaawansowanego i bardziej płynnego podążania za linią. Poprzedni blok kodu używany do podążania za linią wykrywa ścieżkę, odchylenie po lewej stronie lub po prawej stronie (jak również skrzyżowania w niektórych przypadkach). Jednakże, na przykład podczas jazdy na rowerze, sami potraficie z pewnością precyzyjniej dostosować skręt kierownicy do zakrętu. Ten blok pozwala na dokładniejsze i bardziej proporcjonalne podążanie robota za linią. Innymi słowy, im bardziej robot jest odsunięty od toru, tym bardziej będzie skręcał w przeciwnym kierunku. Jeśli odchylenie od toru wynosi 0 robot będzie jechał prosto.

W kroku 3 omówimy, jak można użyć tego bloku do zaprogramowania płynnego ruchu mbota2 za linią. W poniższym przykładzie mBot2 jest zaprogramowany, aby wyświetlał aktualne odchylenie na ekranie. Spróbujcie najpierw przesunąć robota po torze ręcznie:



2. Odtworzenie i przetestowanie kilku przykładów kodów do sterowania czujnikiem Quad RGB

W powyższej tabeli dla każdego bloku kodu przedstawiono przykład programowania. Waszym zadaniem będzie ich odtworzenie w mBlock i przetestowanie. Wybierzcie jeden przykład i zastanówcie się, jak można zmodyfikować kod.

3. Zaprogramowanie mBota2, aby podążał za linią

Naszym zadaniem w "Kroku 3" tej lekcji będzie zaprogramowanie mBota2, aby podążał za trasą na mapie, tak jak robi to autokar wycieczkowy. W tym celu wykorzystamy przetestowane już przez nas przykładów kodu, odpowiednio je modyfikując.

Najpierw użyjemy poniższego bloku:

Jeśli robot znajduje się na torze (oba czujniki L1 i R1 wykrywają linię, wartość kodu=3), robot będzie jechać prosto. Jeśli tylko jeden z dwóch wewnętrznych czujników zidentyfikuje linię, robot odpowiednio skręci w lewo lub w prawo.

W poniższym przykładzie pokazany jest skręt w prawo:

Silniki są uruchamiane w razie potrzeby, a następny blok kodu jest natychmiast wykonany. Ponieważ nie wiemy, kiedy nastąpi kolejny zakręt, polecenia jazdy nie mogą zawierać zaprogramowanego czasu jazdy ani odległości (to samo tyczy się skręcania). Następna iteracja pętli programu sprawdzi nowe dane z czujników i odpowiednio dostosuje jazdę.

Jeśli używacie dołączonej mapy, upewnijcie się, że czujniki są skalibrowane zgodnie z opisem na najjaśniejszy kolor (żółty odcinek na torze).

3. Rozwiązywanie zadania (25 min.)

Krok 3: Rozwiązywanie zadania

Dowiedzieliście się już wiele o funkcjach i działaniu czujnika Quad RGB. W tej części napiszecie własny program komputerowy w mBlock z wykorzystaniem czujnika Quad RGB.

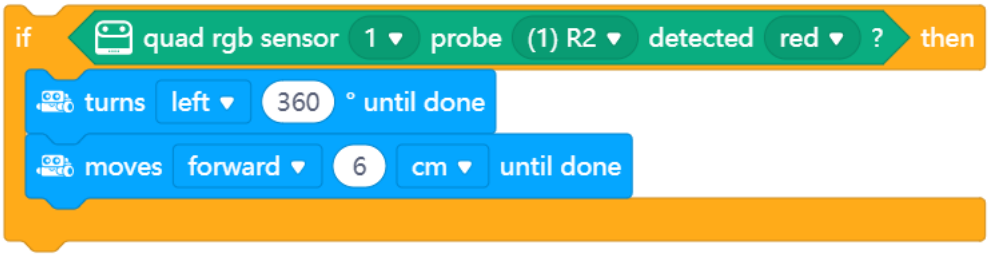
Naszym zadaniem będzie przekształcenie naszego mBota2 w autokar wycieczkowy, który obwozi grupy turystów po najważniejszych punktach turystycznych w mieście. W pudełku z mBotem2 znajdziecie mapę z zaznaczoną trasą. Autokar powinien podążać za czarną linią na mapie. W poprzedniej części programowaliśmy już naszego mBota2 tak, aby podążał za linią. W tym kroku dodamy do tego kilka ulepszeń.

Dla ułatwienia możecie również zbudować własne bloki kodów. Wszystkie elementy programowania związane z podążaniem mBota2 za linią mogą być zawarte w ramach jednego prostego polecenia (np. "simple_line"). Różowa ikonka "My Blocks" w menu kodów pozwala na zdefiniowanie i ustrukturyzowanie własnego bloku kodu według własnych preferencji. W poniższym przykładzie wykorzystane są bloki niestandardowe.

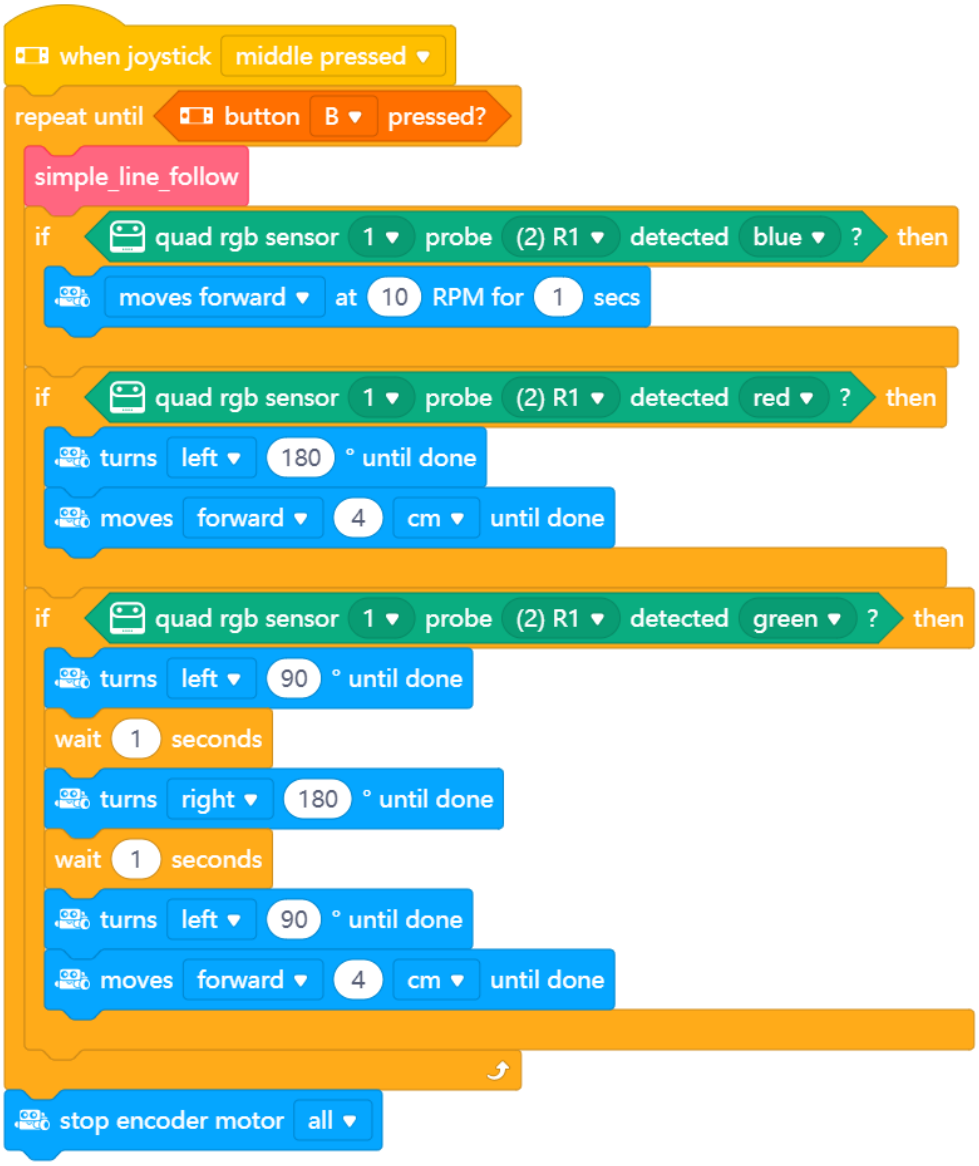
Oprócz jazdy po torze, nasz autokar musi wykonywać też inne czynności w określonych punktach widokowych. Punkty te są oznaczone kolorem czerwonym, żółtym, zielonym i niebieskim.

Możecie również sami wymyślić inne funkcje dla autokaru. Czy chcecie, aby nasz autokar jeździł po Paryżu? W takim razie może powinien odtwarzać w określonych punktach francuski hymn narodowy? A może wolicie Londyn? W tym przypadku można odtwarzać bicie dzwonów Big Bena.

Oczywiście, można również zastosować prostsze rozwiązania. W poniższym przykładzie, mBota2 jest zaprogramowany tak, aby zawracał przy czerwonym znaczniku i wykonywał różne inne czynności przy innych kolorowych znacznikach.

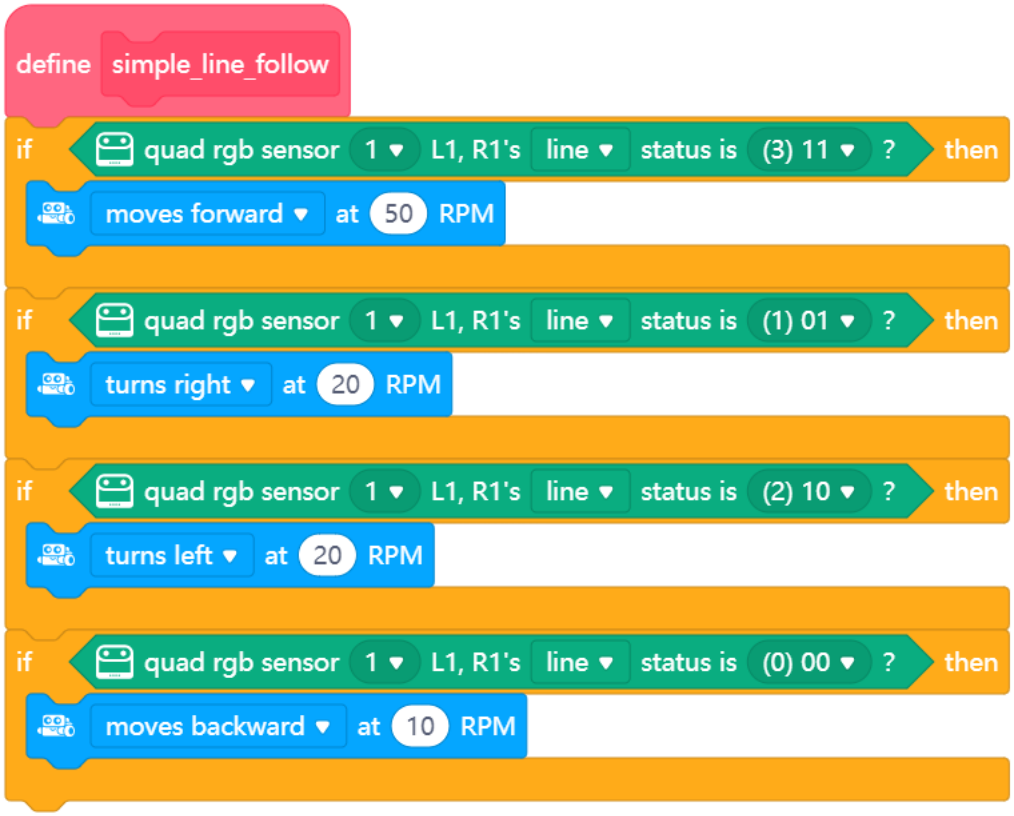


Do wykonania zadania należy wykorzystać wiedzę zdobytą w "Kroku 2" tej lekcji. Oczywiście, możecie eksperymentować z różnymi przykładami programowania w mBlock5. Jeśli macie problem z rozwiązaniem zadania, możecie użyć przykładów podanych poniżej. Możecie je skopiować i dostosować do własnych preferencji.



Program uruchamia się po naciśnięciu przycisku B i zatrzymuje się po naciśnięciu przycisku A. Wykonuje prostą procedurę śledzenia linii przy użyciu spersonalizowanego kodu stworzonego w "My Blocks", a następnie sprawdza, czy czujnik R1 wykrywa kolor niebieski (powolna jazda), czerwony (zawracanie) lub zielony (obrót w prawą i lewą stronę). Aby upewnić się, że kolor nie zostanie ponownie rozpoznany po wykonaniu każdej z tych czynności, robot jest zaprogramowany tak, aby po każdej z nich przesunął się o 4 cm do przodu.

Blok "simple_line" jest zdefiniowany w następujący sposób:



Wersja zaawansowana:

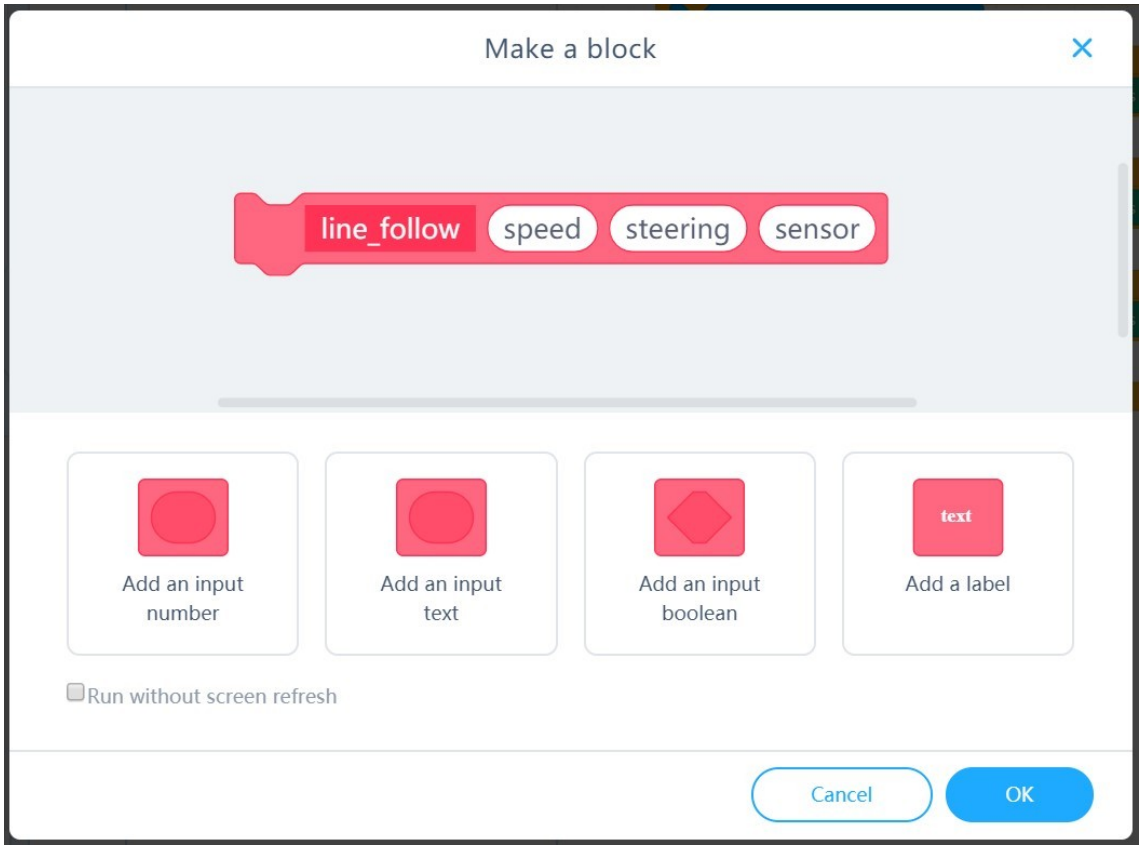
W poprzednim kroku omówiliśmy sobie odchylenie od linii. Ten blok informuje, jak bardzo robot jest odchylony od toru (od -100 do +100; 0 dla idealnego położenia w środku). Pozwala na dokładniejsze i bardziej proporcjonalne podążanie robota za linią. Innymi słowy, im bardziej robot jest odsunięty od toru, tym bardziej musi skręcić w przeciwnym kierunku. Przy odchyleniu na prawo od toru, wartość odchylenia będzie dodatnia (robot musi jechać w lewo); wartości ujemne oznaczają odchylenie na lewo od toru (robot musi jechać w prawo).

Będziemy musieli ustalić podstawową prędkość naszego mBota2 podczas jazdy do przodu po prostym torze. Przy skręcaniu w lewo lub w prawo, jedno z kół musi obracać się szybciej niż drugie - im większa różnica, tym większy skręt.

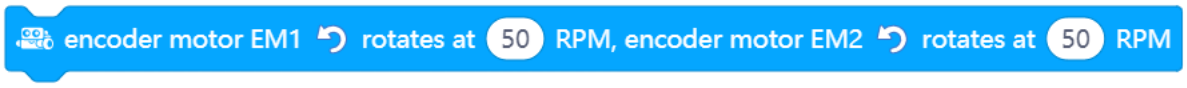
W tym celu musimy dodać odchylenie do prędkości podstawowej jednego silnika i odjąć je od prędkości drugiego. Aby umożliwić dokładniejszą regulację, użyty zostanie współczynnik wpływu odchylenia na sterowność, ponieważ zakres dla samego odchylenia jest zbyt duży, aby mógł być użyty bezpośrednio do sterowania silnikami.

Ale do którego silnika należy dodać lub odjąć obliczoną wartość? Dodatnia wartość odchylenia oznacza, że mBot2 jest odchylony na prawo od toru i musi skręcić w lewo. Prawe koło musi zatem obracać się szybciej. W tym celu należy dodać wyskalowaną wartość odchylenia do prawego koła i odjąć ją od lewego.

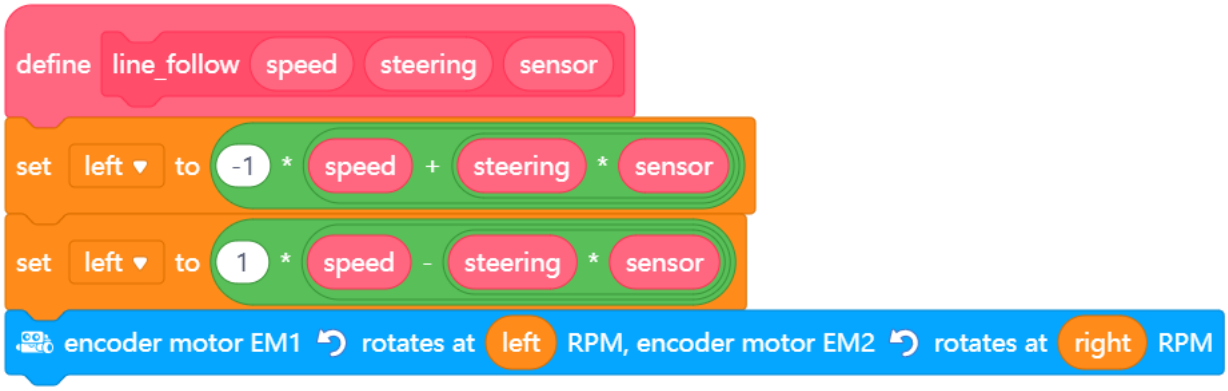
Podczas konfiguracji bloków w "My Blocks" można zmieniać ich nazwy:



Na koniec należy upewnić się, że blok (pokazany poniżej) zawiera ujemne wartości dla prawego silnika i dodatnie dla lewego podczas jazdy do przodu.



Wartości dla prawego silnika należy w tym celu pomnożyć przez -1. Poniżej przedstawiono blok służący do zaprogramowania mBota2 do płynnej jazdy po linii:



W głównej pętli programu można użyć tego bloku jednym prostym poleceniem:

Polecenie to wywołuje podprogram z prędkością podstawową 50 i współczynnikiem skrętu 0,6, które przesyła również wartość aktualnego odchylenia z odczytu czujnika. Możecie trochę poeksperymentować z tymi wartościami - jeśli robot zbyt gwałtownie skręca, oznacza to, że współczynnik sterowności jest za wysoki; jeśli zbacza z toru nawet na słabych zakrętach, współczynnik ten jest za niski.

Przy wykonywaniu tego zadania warto skorzystać z poniższego szczegółowego planu dla ułatwienia. Czy macie już pomysł na to, jak wykonacie zadanie? Jeśli tak, przedyskutujcie swój pomysł z nauczycielem.

	Wyjaśnienie
Krok 1: Co chcemy zrobić?	• Jak powinien się poruszać nasz mBot2?
Krok 2: Co należy przygotować:	• Czego będziemy potrzebować oprócz naszego mBota2?
Krok 3: Jakich bloków kodów będziemy potrzebować, aby wprawić mBota2 w ruch?	• Jak możemy sprawić, by mBot2 zmieniał kierunek na każdym rogu? • Jakich bloków kodów będziemy do tego potrzebować? • Jak możemy opisać działanie naszego programu (używając pseudokodu//języka naturalnego, schematu blokowego lub UML)? • Gdy potrzebujemy dodatkowych wyjaśnień, warto skonsultować się z kolegami, nauczycielem lub poszukać informacji na ten temat. Dla każdego

	bloku kodu w mBlock dostępna jest pomoc
Krok 4: Testowanie i wdrażanie	• Czy pierwsza wersja jest już gotowa? Przetestujmy ją! Podczas rundy testowej należy spisać aspekty rozwiązania wymagające poprawy • Rozwiązanie należy usprawniać, aż mBot2 przejedzie przez trasę bezbłędnie. • I jak Wam poszło? Sfilmujcie efekt końcowy i za zgodą nauczyciela opublikujcie go w mediach społecznościowych z hashtagiem #mBot2

4. Podsumowanie (5 min.)

Krok 4: Podsumowanie

Czy udało Wam się zaprogramować mBota2, aby jeździł po torze jak autokar wycieczkowy? Podczas tej lekcji dowiedzieliście się, jak działają czujniki Quad RGB i gdzie znajdują zastosowanie w codziennym życiu. Wiecie już, jak zaprogramować tego typu czujnik, aby podążał za linią. Potraficie również zaprogramować robota mBot2, aby jeździł po trasie i wykonywał różne czynności w oparciu o różne wykryte kolory.

Czas na krótkie podsumowanie. Zastanówcie się indywidualnie, a następnie przedyskutujcie w grupie:

- Co Waszym zdaniem dobrze się udało?
- Co należałoby poprawić?
- Które części tej lekcji były dla Was łatwe, a które sprawiały więcej trudności?
- Które aspekty warto by omówić bardziej szczegółowo?
- Kto mógłby Wam w tym pomóc?