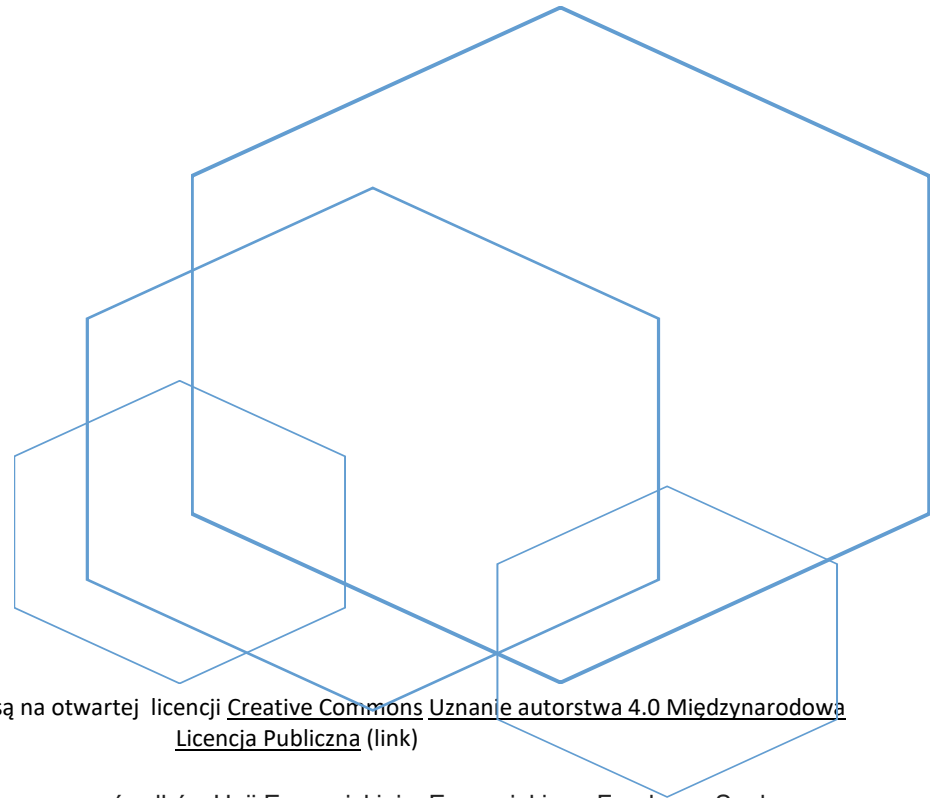
Materiały szkoleniowo-dydaktyczne do szkolenia dla nauczycieli z zakresu kompetencji cyfrowych w kontekście prowadzenia zajęć z zakresu robotyki i programowania w ramach projektu „Podniesienie kompetencji cyfrowych wśród uczniów i nauczycieli województwa podlaskiego”

Autorzy:

Małgorzata Chmielewska

Jacek Chmielewski



Projekt jest współfinansowany ze środków Unii Europejskiej z Europejskiego Funduszu Społecznego w ramach Regionalnego Programu Operacyjnego Województwa Podlaskiego na lata 2014-2020

Spis treści

1. Zajęcia z wykorzystaniem robota edukacyjnego.

Robot edukacyjny to elektroniczne urządzenie, które możemy sterować oraz zaprogramować. Roboty edukacyjne odpowiedzialne są za wykonywanie wcześniej ustalonych czynności. W zależności od modeli, wyposażone są najczęściej w czujnik ruchu, dotyku, odległości i diody. Jest duże spectrum możliwości przygotowywania programów np. możemy narysować linię na podłożu po której robot się będzie poruszał. Inny pomysł to przygotowanie programu tak aby robot zbliżając się do przeszkody zatrzymał się lub ją ominął (zasada działania odkurzaczy np. iroomba). Roboty edukacyjne mogą być aktywowane również poprzez impulsy świetlne np. przygotowanie linii z wykorzystaniem odpowiedniej sekwencji kolorystycznej tj. czerwony, zielony, niebieski odpowiada za obrót w prawo – przykład ozobot. W takim przypadku do kodowania wystarczą flamastry i kartka papieru.

Roboty edukacyjne, podobnie jak nauka programowania, pobudzają kreatywność jak również umiejętność samodzielnego poszukiwania rozwiązań. Stanowią składową dla metod aktywizujących w nauczaniu. To bardzo dobry przykład połączenia ścisłego, inżynieryjnego myślenia z cyfrową twórczością. W pracy z robotem liczy się efekt – skuteczność, a nie dokładne wypełnienie poleceń nauczyciela jak również praca według klucza.

1.1 Skribot

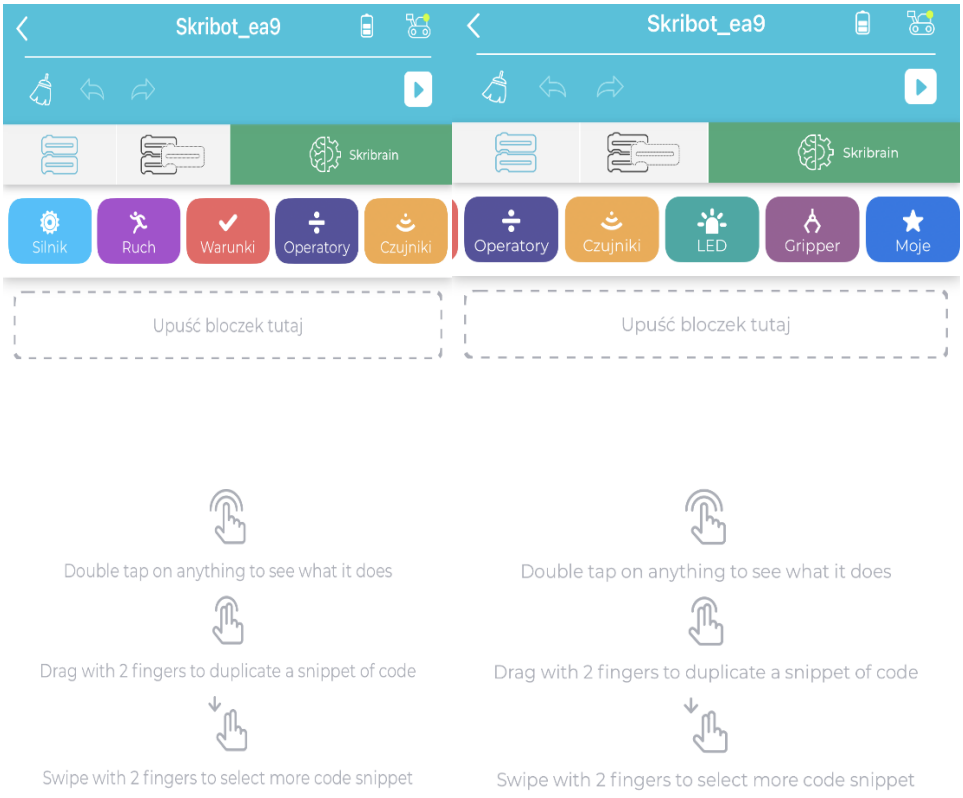
Jeden z robotów edukacyjnych, który bardzo dobrze wprowadza w świat mechaniki, elektroniki i programowania wykorzystując różne poziomy zaawansowania. Zestaw Skribot zawiera również klocki SkriKit, które świetnie pobudzają wyobraźnię przestrzenną ale również rozwijają umiejętności manualne. Klocki SkriKit, w każdym momencie możemy wydrukować na kompatybilnej drukarce Skrinter, korzystając przy tym z Creatora.

Robot posiada: czujniki odległości (ultradźwiękowe), czujniki światła/linii (możemy wykleić drogę z taśmy, po której będzie podążał), silniki, diody LED, chwytak.

Skribot działa na 3 płaszczyznach:

1. konstruowanie - uczniowie poprzez konstrukcję robota nie tylko poznają świat inżynierii ale przede wszystkim rozwijają zdolności manualne jak również uczą się organizacji pracy.
2. elektronika - samodzielne podłączanie elementów pozwala poznać podstawy elektroniki jak również zdefiniować działanie mikrokontrolerów.
3. programowanie - podniesienie wiedzy i umiejętności programistycznych, niezależnie od poziomu zaawansowania, dają możliwość odkrywać i stosować w praktyce zróżnicowane technologie.

Aby móc sprawnie skorzystać ze Skribota musimy zainstalować aplikację SkriApp. Poprawnie złożony model robota, daje możliwość zastosowania różnego rodzaju sekwencje programistyczne. Aby rozpocząć przygodę z programowaniem z SkriApp, należy wyjaśnić kilka ważnych pojęć.



Ilustracja 1.1, autor Małgorzata Chmielewska

Bloczek „**Porusz silnikiem do przodu**” oraz „**Porusz silnikiem do tyłu**”. Bloczek odpowiada za pracę silnika, w pierwszym przypadku silnik wiruje w stronę przodu robota,

w drugim zaś przypadku silnik wiruje w stronę tyłu robota. Po wybraniu bloczka, pojawi się możliwość dodania danych tj. wyboru silnika(są dwa), ustawienia czasu oraz mocy.

Bloczek „**Jedź do przodu**" oraz „**Jedź do tyłu**" odpowiada za obrót jednocześnie dwoma silnikami do przodu oraz do tyłu . Mamy możliwość dokonania wyboru czasu mierzonego w sekundach oraz prędkości mierzonej w procentach, gdzie 100% oznacza maksymalną prędkość a 1 % minimalną.

Bloczek „**Obróć się w lewo**" oraz „**Obróć się w prawo**" odpowiada za obrót jednocześnie dwoma silnikami w lewą stronę oraz w prawą stronę . Mamy możliwość dokonania wyboru czasu mierzonego w sekundach oraz prędkości mierzonej w procentach, gdzie 100% oznacza maksymalną prędkość a 1 % minimalną.

Bloczek „**Jeżeli**" – nazywany funkcją warunkową. Dzięki tej funkcji określamy co nasz robot ma zrobić tylko w określonej sytuacji. Przykład – „ Jeżeli pada deszcz, to załóż płaszcz przeciwdeszczowy oraz kalosze"

Bloczek „**Jeżeli ….w przeciwnym razie..**" **–** nazywany funkcją warunkową IF ELSE, w odróżnieniu od bloczka „Jeżeli", w tym przypadku określamy jakie zadanie ma wykonać robot jeżeli nasze warunki nie zostały spełnione.

Bloczek „**Zapętl**" - nazywany pętlą. Dzięki temu wyborowi możemy powiedzieć robotowi, aby wykonywał tę samą czynność np. wiele razy.

Bloczek „ **Jeżeli to powtarzaj**" – nazywany pętlą warunkową WHILE. Jeżeli warunek został spełniony czynność, którą powinien wykonać robot będzie powtarzana dopóki okoliczności się nie zmienią.

Bloczek „**Czekaj**" – robot odpoczywa.

Bloczek „**+, -, x, ÷** - odpowiadają za operacje na zmiennych.

Bloczek „**Odczyt czujnika linii"** – odpowiada za odróżnienie barwy białej oraz czarnej.

Bloczek „ **Odczyt czujnika odległości**" – podpowiada robotowi czy w jego okolicy nie znajdują się żadne przeszkody.

Bloczek „**Włącz światło**” oraz „**Wyłącz światło**” – odpowiadają włączenie oraz wyłączenie świateł Led w robocie.

Bloczek „**Gripper**” – „**Złap**”, „**Puść**”, „**Podnieś**”, „**Opuść**” – odpowiada za pracę chwytaka.

Aplikacja pozwala na sterowanie mechaniczne robotem.

Ilustracja 1.2, autor Małgorzata Chmielewska

Przykład 1.

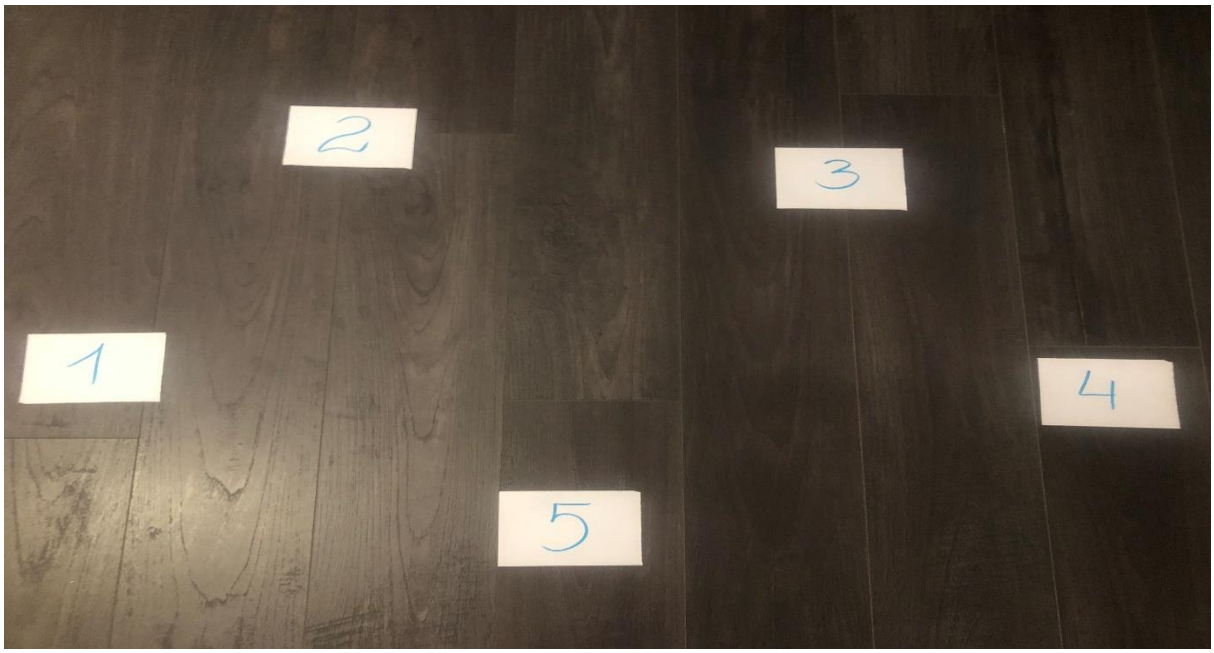
Przed rozpoczęciem zadania powinniśmy dokonać analizy zaspokajania indywidualnych potrzeb rozwojowych i edukacyjnych ucznia. Musimy mieć pewność, że treści zadań będą dopasowane pod odbiorców. W wyborze ćwiczeń musimy zwrócić uwagę na możliwości psychofizyczne ucznia jak również czynniki środowiskowe wpływające na jego funkcjonowanie.

Misja: „Labirynt z działaniami matematycznymi”

Do przygotowania zadania potrzebne będzie:

- fiszki z zadaniami (ponumerowane, numer zadania powinien znajdować się na górze) – 5 sztuk.

Na podłodze rozkładamy fiszki z zadaniami matematycznymi. Za pomocą sterowania mechanicznego, poruszamy się robotem na zadanie 1. Jeżeli zadanie 1 zostanie rozwiązane, podąża robot do zadania drugiego. I tak aż do momentu udzielenia przez ucznia odpowiedzi na wszystkie zadania. Wyniki poszczególnych zadań niech będą kodem/rozwiązaniem misji.



Ilustracja 1.3, autor Małgorzata Chmielewska

To samo zadanie wykonajmy tworząc już program z bloczków!

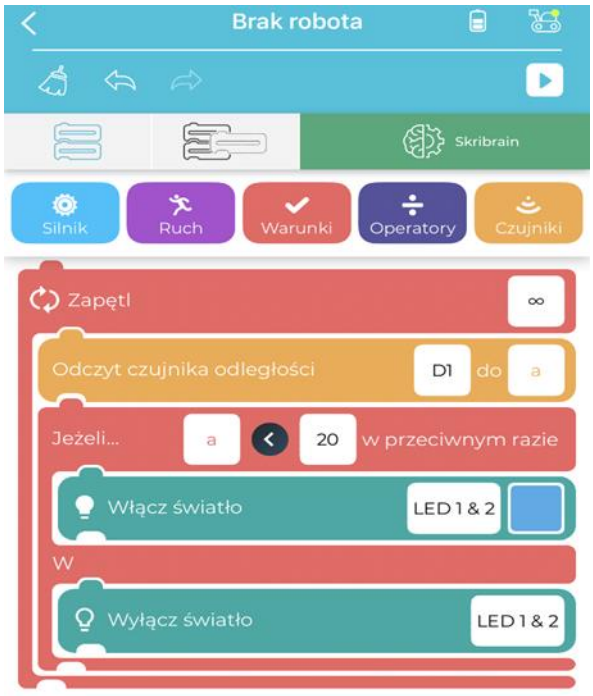
Mając na uwadze pracę z dziećmi jak również młodzieżą ze specjalnymi potrzebami rozwojowymi i edukacyjnymi, możemy na potrzeby uczniów modyfikować ww. przykład 1. Zamiast zadań, rekomendujemy przygotowanie fiszek z kolorami/ wzorami/ figurami/ strzałkami. Zadaniem ucznia będzie za pomocą sterowania mechanicznego robota, dotrzeć w wyznaczone miejsce przez nauczyciela. Nauczyciel wydając polecenie, weryfikuje na bieżąco postęp ucznia oraz odpowiednio motywuje i zachęca do kolejnego kroku.

Przykład 2.

„Przywitaj się z robotem”

Zaprogramujmy robota z wykorzystaniem bloczków. Przygotuj program w taki sposób, aby robot widząc twarze uczniów w odległości mniejszej 20 cm, zaświecił Ledami na kolor niebieski.

Rozpoczynamy pracę od umieszczenia czujnika („Odczyt czujnika odległości") na panelu roboczym aplikacji. Następnie określamy nazwę czujnika (D1), który wysyła sygnał o odległości do zmiennej a. Przeciągamy bloczek „Jeżeli w przeciwnym razie", określamy warunek pierwszy, wybierając zmienną a, wstawiamy znak < oraz wartość 20. Kolejny krok to wstawienie w pierwszy wiersz, bloczek „Włącz światło" (wybierając kolor niebieski obydwu diod), w drugi wiersz powinniśmy wstawić bloczek „Wyłącz światło". Ostatni krok to wykorzystanie bloczka „Zapętl", czyli powtarzanie w nieskończoność. Podsumowując jeżeli nasz robot dostrzeże twarze w odległości mniejszej 20 cm to zaświecą się Ledy, jeżeli zaś będziemy za daleko – Ledy w robocie będą wyłączone. Będzie wykonywał taki proces nieskończenie wiele razy. Przygotowaliśmy dwa warunki. Poniżej program z przykładowym rozwiązaniem.



Ilustracja 1.4, autor Małgorzata Chmielewska

1.2 Lego Mindstorms EV3

LEGO Mindstorms jest jednym z najpopularniejszych narzędzi do nauki robotyki dostępnych na rynku. Każdy zna LEGO i prawdopodobnie zdecydowana większość lubi tworzyć konstrukcje z klocków. W przypadku starszych dzieci popularnością cieszy się seria TECHNIC umożliwiająca tworzenie konstrukcji mechanicznych.

LEGO Mindstorms to zestawy łączące LEGO TECHNIC z elementami elektronicznymi (mikrokontroler, silniki, sensory). Użytkownik ma możliwość programowania tekstowego lub blokowego w przyjaznym środowisku programistycznym o przyjemnej szacie graficznej (szczególnie przydatna opcja w pracy z młodszymi dziećmi).

Na rynku dostępne są dwie wersje zestawów – domowa i edukacyjna. Pomimo pewnych podobieństw różnice między zestawami są znaczące, szczególnie w zakresie możliwości dydaktycznych.

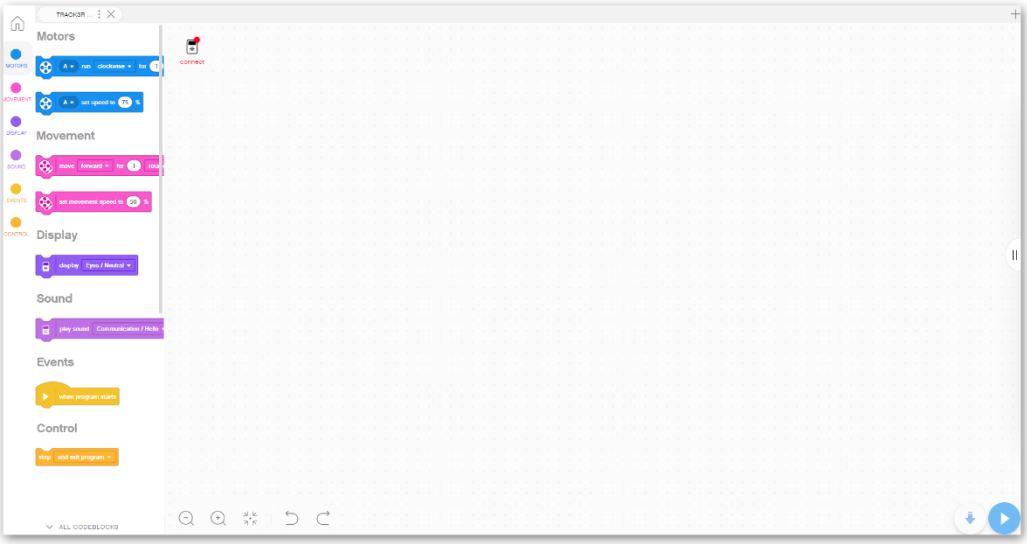
Wykorzystując wersję domową w szkole warto dodatkowo wyposażyć pracownię w plastikowe pojemniki na poszczególne elementy zestawu lub organizery. Wersja domowa w przeciwieństwie do edukacyjnej nie posiada dedykowanego pojemnika z przegrodami, zapakowane jest jedynie w standardowe dla LEGO kartonowe opakowanie.

W kwestii elementów konstrukcyjnych LEGO przyzwyczaiło już do wysokiej jakości pasujących do siebie elementów, a tych mamy w zestawie dość dużo, np.: koła zębate i belki konstrukcyjne różnych wielkości, koła podporowe, oraz mnóstwo innych elementów. W odróżnieniu od wersji edukacyjnej w wersji domowej jest więcej klocków natomiast nie przekłada się to na możliwości konstrukcyjne. Duża część klocków stanowi wyłącznie elementy dekoracyjne o niskiej wartości konstrukcyjnej.

Wersja Home Edition podobnie jak wersja edukacyjna zawiera mikrokontroler, oraz trzy silniki wyposażone w czujnik obrotu (dwa duże i jeden mały). Mikrokontroler wyposażony jest w sześć przycisków, 4 porty wyjścia, port USB, port mini USB, wyświetlacz oraz gniazdo na kartę micro SD. Do zasilenia mikrokontrolera warto wyposażyć się w akumulatory AA lub akumulator dedykowany do zestawu. W wersji domowej dostępne są czujniki dotyku,

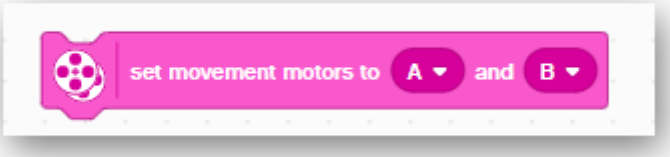
koloru oraz czujnik odległości. Czujnik odległości wykorzystuje podczerwień do wykrycia przeszkód.

Stworzone roboty można programować zarówno prze aplikację dedykowaną dla komputerów jak i na tabletach. Aplikacje do programowania na tablety są uboższe w funkcje w porównaniu do aplikacji komputerowej. W zakresie programowania blokowego Mindstorms EV3 został oparty na języku Scratch, co można zauważyć w aplikacji (ilustracja 1.5).

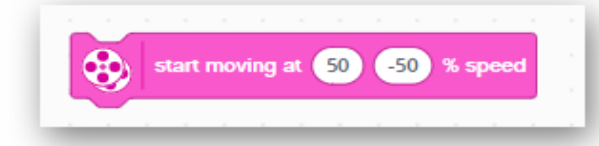


Ilustracja 1.5, autor Jacek Chmielewski

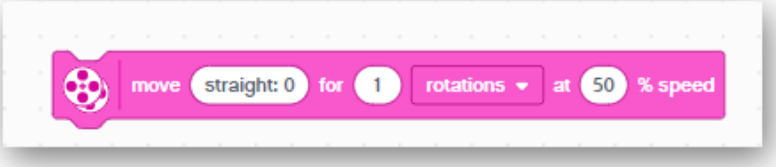
W nowej wersji aplikacji został zmieniony sposób programowania robota. Różnice pojawiają się już na etapie wprawiania w ruch obu silników wykorzystanych w konstrukcji. Aby silniki pracowały w sposób zsynchronizowany wystarczy użyć tego bloku)Ilustracja 1.6). Polecenie poruszania się w danym kierunku w określonej prędkości wydajemy poprzez przeciągnięcie bloków odpowiadających za ruch (ilustracja 1.7) oraz kierunek (ilustracja 1.8).



Ilustracja 1.6, autor Jacek Chmielewski

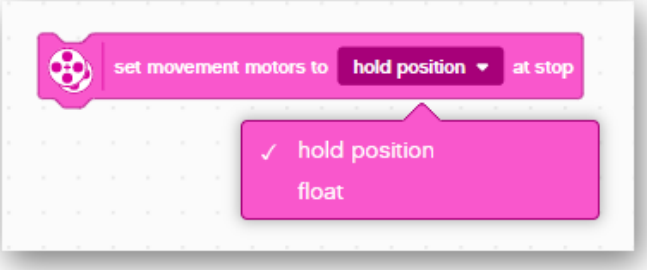


Ilustracja 1.7, autor Jacek Chmielewski



Ilustracja 1.8, autor Jacek Chmielewski

Aby zatrzymać silniki należy przeciągnąć polecenie jak na ilustracji 1.9.



Ilustracja 1.9, autor Jacek Chmielewski

Po lewej stronie w aplikacji jest do wyboru kilka zakładek poleceń i wyrażeń programistycznych, które są do dyspozycji:

- **Motors** – polecenia związane z serwomotorami podłączonymi do mikrokontrolera, znajdują się tam polecenia, które jednorazowo wprawią poszczególny silnik w ruch lub zatrzymają;
- **Movements** – polecenia dedykowane dla całej konstrukcji związane z jej poruszaniem się;

- **Display** – polecenia związane z operacjami wykonywanymi na wyświetlaczu;
- **Sound** – odpowiadają za programowanie wszelkich efektów dźwiękowych;
- **Events** – bloki warunkujące wykonanie określonej czynności od spełnienia danego warunku;
- **Control** – warunki, pętle oraz pętle warunkowe;
- **Sensors** – wszelkie polecenia związane z działaniem czujników;
- **Operations** - polecenia związane z logiką i arytmetyką;
- **Variables** – zmienne, które można wykorzystać w tworzeniu kodu;
- My blocks – stworzone przez użytkowników bloki poleceń.

Wykorzystanie LEGO Mindstorms EV# Home Education do stworzenia czujnika antywłamaniowego.

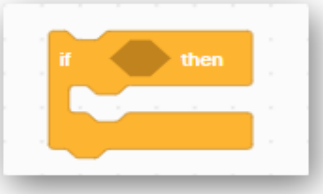
Aby stworzyć należy stworzyć dowolną konstrukcję, w skład której będzie wchodził mikrokontroler, a także czujnik odległości. Czujnik odległości w wersji domowej działa na podczerwień przez co pomiary nie są dokładne, ale do stworzenia czujnika antywłamaniowego są wystarczające.

Po stworzeniu stabilnej konstrukcji niewielkich rozmiarów, w której czujnik odległości będzie skierowany przed siebie (nie do góry ani w dół), należy przejść do stworzenia kodu i wgrania go na kostkę EV3.

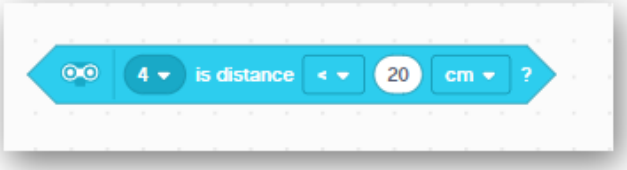
Programowanie można ograniczyć do wykorzystania bloków z zakładek sensors, control, sounds oraz display.

Tworząc algorytm programu należy uwzględnić, że czujnik powinien włączyć alarm w momencie otwarcia drzwi. W związku z tym funkcja warunkowa będzie optymalna (ilustracja 1.10). Warunkiem uruchomienia alarmu będzie otwarcie drzwi, aby czujnik zareagował nasza konstrukcja musi zostać ustawiona przy samej framudze drzwi po stronie klamki, żeby otwierane drzwi przecięły wiązkę podczerwieni emitowaną przez czujnik wzdłuż drzwi (drzwi muszą się otwierać w stronę pomieszczenia z czujnikiem. W celu stworzenia prawidłowego programu należy do wybrano bloku (ilustracja 1.10), dodać blok określający pracę czujnika (ilustracja 1.11), oraz polecenie włączeniu alarmu (ilustracja

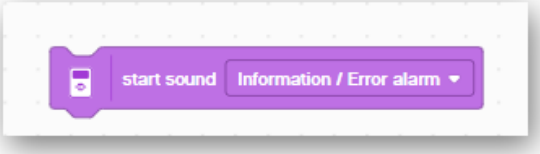
1.12). Wszystkie polecenia powinny stworzyć następującą konstrukcję (ilustracja 1.13). Cały program można umieścić w pętli **REPEAT** lub **FOREVER** odpowiadające kolejno za określoną ilość powtórzeń wykonania przygotowanego programu (ilustracja 1.14) lub nieskończoną ilość powtórzeń (ilustracja 1.15).



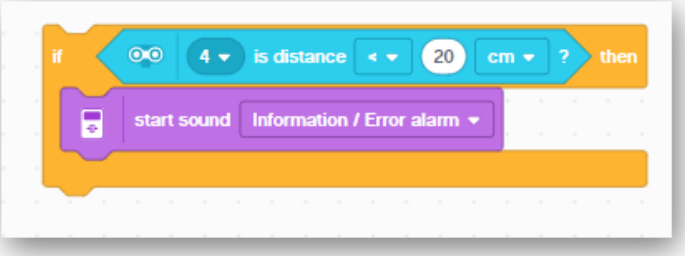
Ilustracja 1.10, autor Jacek Chmielewski



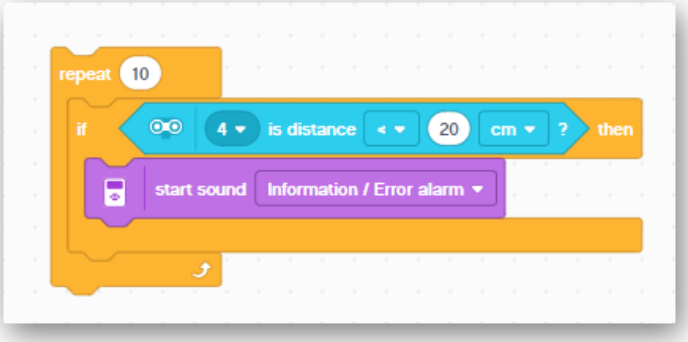
Ilustracja 1.11, autor Jacek Chmielewski



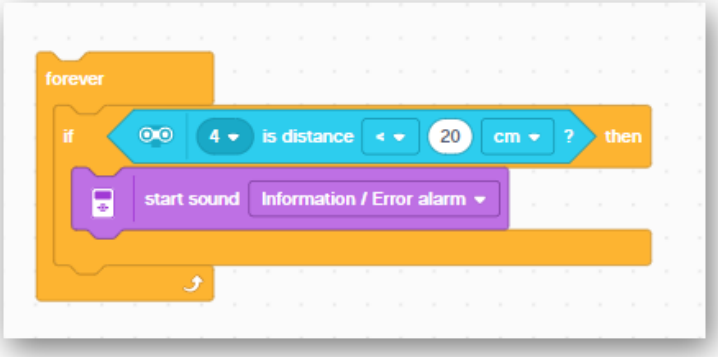
Ilustracja 1.12, autor Jacek Chmielewski



Ilustracja 1.13, autor Jacek Chmielewski



Ilustracja 1.14, autor Jacek Chmielewski



Ilustracja 1.15, autor Jacek Chmielewski

Po uruchomieniu programu każde otwarcie drzwi przerwie wiązkę podczerwieni co czujnik odczyta jako zmniejszenie odległości od przeszkody i uruchomi alarm.

1.3 Kształtowania systemu wartości i postaw zawodowych, przygotowujących do pracy z dziećmi i młodzieżą ze specjalnymi potrzebami rozwojowymi i edukacyjnymi

Wykorzystując roboty w pracy z dziećmi oraz młodzieżą ze specjalnymi potrzebami rozwojowymi i edukacyjnymi, warto pamiętać o odpowiednim podejściu do tematu:

- Warto mieć na uwadze ilość dostarczanych bodźców związanych z przekazywanym materiałem. Należy pamiętać, aby nie było ich za dużo np.

konstruowanie robota jak również programowanie podczas jednego spotkania, okazać się może zbyt obciążające;

- Poznawanie kilku funkcji w programowaniu jednocześnie, również może zniechęcić ucznia do dalszej aktywnej pracy. Z dziećmi mającymi specjalne potrzeby rozwojowe ze względu na pewne deficyty warto stosować metodę małych kroków;
- Powinno się również pamiętać o zastosowaniu dodatkowych środków dydaktycznych i środków technicznych np. prezentacji z dokładniejszymi instrukcjami / poleceniami;
- Programowanie robotów opiera się głównie na systemie blokowym/graficznym oraz w mniejszym stopniu wykorzystaniem bardziej zaawansowanego języka np. Phyton, który jest w języku angielskim. Mając to na uwadze, należy zapewnić uczniowi możliwość korzystania z nauki języka angielskiego na miarę jego możliwości i potrzeb;
- Materiały dydaktyczne do samodzielnego rozwiązania powinny być zróżnicowane np. różniły się poziomem trudności. Materiały dydaktyczne przekazane uczniom do samodzielnego rozwiązania, nauczyciel powinien na bieżąco monitorować, np. uczeń ma za zadanie napisać prosty program w aplikacji SkriApp odnoszący się do oczekiwań nauczyciela;
- Dziecko na każdym poziomie pracy powinno odczuwać obecność nauczyciela - osoby, która rozwieje nieoczekiwane wątpliwości;
- Dzieci pochodzące ze środowisk wykluczonych mają dużo mniejszy dostęp do narzędzi informatycznych jak roboty, komputery czy urządzenia mobilne, w związku z czym należy takim osobom w miarę możliwości umożliwić swobodne zapoznanie się ze sprzętem, żeby nie tworzyć potencjalnej sytuacji stresującej dla dziecka;
- Dzieci szczególnie zdolne powinny mieć możliwość szybszego realizowania zadania i otrzymania zadań dodatkowych dostosowanych do jego możliwości;

- Dzieci nadprzeciętnie uzdolnione powinny otrzymywać materiał wykraczający poza zakres materiału przygotowanego dla reszty dzieci i dostać wsparcie nauczyciela przy jego rozwiązywaniu.

2. Wstęp do programowania

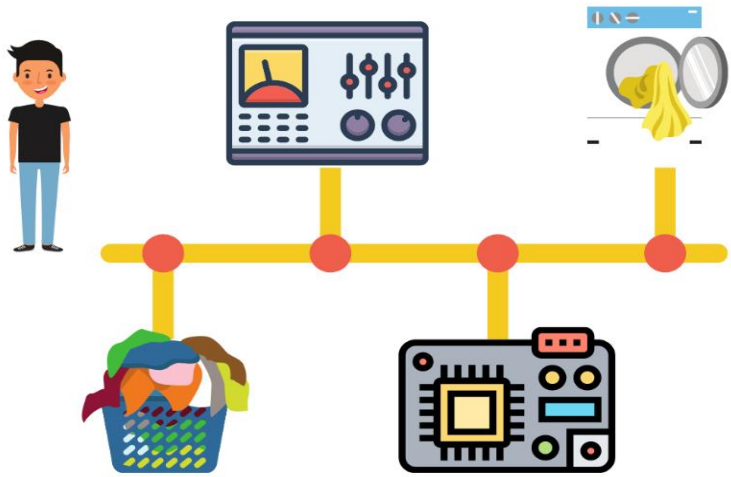
2.1 Istota programowania

Programowanie jest elementem niezbędnym do funkcjonowania dzisiejszego świata. Jesteśmy otoczeni przez komputery wykonujące bez przerwy napisane przez kogoś programy. Stały rozwój nowych technologii wpływa na zwiększanie możliwości komputerów. Dla zobrazowania zjawiska warto zaznaczyć, że procesory dzisiejszych standardowych smartfonów są dużo wydajniejsze od procesora w komputerze wykorzystanym w misji kosmicznej Apollo 11, której celem było lądowanie człowieka na księżycu.

Komputery to nie tylko laptop, PC, smartfon czy tablet. Komputery wykorzystane w określonym środowisku wykonują konkretne zadania, m. in. komputery wykorzystane w samochodach odpowiadają m. in. za automatyczną regulację temperatury, właściwą pracę układu zasilania silnika czy działanie układów wpływających na bezpieczeństwo kierowcy i pasażerów. Zdarza się również, że na co dzień programujemy w domu. Każde urządzenie wyposażone w programator wymaga zaprogramowania do właściwej pracy. Tym sposobem włączając pralkę, zmywarkę czy piekarnik stajemy się domowym programistą.

Obsługa urządzenia domowego jakim jest pralka przypomina proces tworzenia programu komputerowego. Na ilustracji 1.1 są wyszczególnione poszczególne etapy składające się na czynność prania ubrań:

a. Jest problem do rozwiązania – wypranie ubrań;
b. Zaplanowanie zadania, które wykona urządzenie/pralka – wybór właściwego trybu prania, ilości obrotów i temperatury;
c. Wprowadzenie odpowiednich danych do urządzenia/pralki - zaprogramowanie odpowiedniego trybu przez programator;
d. Wprowadzony przez nas program jest kompilowany na kod maszynowy zrozumiały przez pralkę – komputer odczytuje instrukcję i uruchamia właściwy tryb pracy;
e. Pralka wykonała zadanie zgodnie z otrzymanymi instrukcjami – ubrania zostały prawidłowo wyprane – problem rozwiązany



ilustracja 1.1, autor: Jacek Chmielewski

Przykład programowania pralki można traktować jako sposób zobrazowania uproszczonego procesu tworzenia **kodu źródłowego**. Kod źródłowy jest instrukcją lub instrukcjami napisanymi w określonym języku programowania. Te instrukcje w czytelny sposób dla człowieka opisują operacje na danych, które komputer powinien wykonać. Należy zaznaczyć, że komputer nie potrafi odczytać kodu źródłowego. Aby komputer był w stanie wykonać dane operacje musi otrzymać instrukcje w postaci **kodu maszynowego**, który jest tak zwanym **kodem wynikowym**. Za proces tłumaczenia kodu źródłowego na kod maszynowy odpowiada **kompilator.**

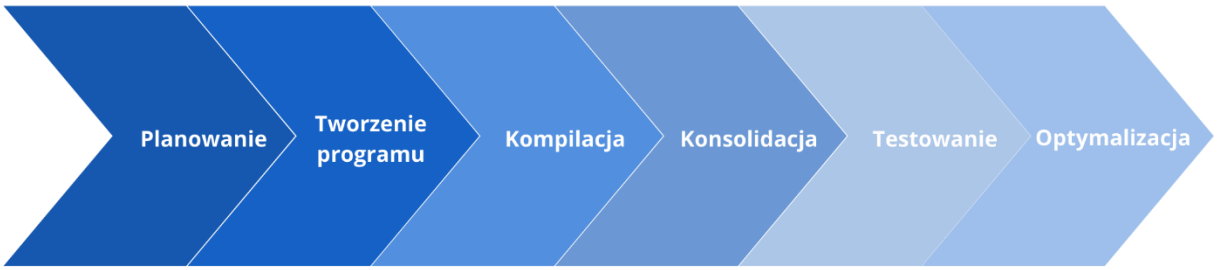
Żaden komputer ani urządzenie posiadające wbudowany komputer nie jest w stanie działać bez programu. Komputery pracują według szczegółowych instrukcji zapisanych w określonym języku programowania. Programista jest odpowiedzialny za napisanie programu, poprzez który komputer będzie pobierał określone dane i przetwarzał je w informacje istotne dla użytkownika komputera. W skrajnym uogólnieniu można przyjąć, że praca komputerów polega na przetwarzaniu wprowadzonych danych w informacje.

Ilustracja 1.2, autor: Jacek Chmielewski

Przed rozpoczęciem pisania programów należy zapoznać się z narzędziami wspierającymi proces tworzenia kodu. Najważniejsze z nich to **edytor**, **debuger** oraz **IDE –**

- **Edytor** – edytor tekstu jest najprostszym narzędziem, które można wykorzystać do pisania programu. Przy wyborze prostego edytora tekstu, np. Notatnika w systemie Windows należy dużo uwagi poświęcić na pilnowanie prawidłowej składni oraz właściwych konstrukcji językowych dla danego języka programowania. Istnieją edytory wyposażone w narzędzia przydatne przy pisaniu kodu, np.: autouzupełnianie, podswietlanie składni, pomoc kontekstową, podpowiedzi.
- **Debugger** – narzędzie służące do lokalizowania i usuwania błędów w analizowanym programie. Debuger jest niezwykle przydatny w sytuacji, w której napisany kod po skompilowaniu i uruchomieniu nie działa prawidłowo i trudno jest zlokalizować wadliwy fragment kodu. W takim przypadku debuger umożliwia uruchomienie programu i analizowanie każdej kolejnej linijki kodu. Po znalezieniu błędu debuger wskazuje jego dokładną lokalizację co pozwala przyspieszyć i ułatwić pracę.
- **IDE** - zintegrowane środowisko programistyczne (ang. Integrated Development Environment - IDE). Jest to zbiór programów umożliwiających tworzenie, modyfikowanie i testowanie oprogramowania. IDE pozwala przyspieszyć pracę nad programem. Poszczególne IDE służą do tworzenia programów w określonych językach programowania. Najczęściej zawiera narzędzia do edycji kodu, kompilacji oraz tworzenia interfejsu graficznego.

Po wybraniu środowiska, w którym powstanie program można przejść do procesu tworzenia. Proces ten można podzielić na sześć etapów wskazanych w ilustracji 1.3.



Ilustracja 1.3, autor: Jacek Chmielewski

- Projektowanie – Na tym etapie należy określić problem, który trzeba rozwiązać, a także wskazać cele i zadania do wykonania przez program, innymi słowy programista musi wiedzieć jakie dane wyjściowe program powinien generować. Programista musi zdecydować na jakiej platformie będzie pracował tworzony program. Jeśli nad programem pracuje zespół to należy przydzielić zadania. Po przyjęciu odpowiedniej koncepcji i zdefiniowaniu problemu warto nadać mu postać algorytmu, oraz opracować logikę programu. Brak opracowanego projektu może wydłużyć pracę nad kodem.
- Pisanie programu – Opracowanie algorytmu ułatwi proces pisania kodu. Każdy język programowania ma swoje reguły gramatyczne i składnię. Niezbędne jest przestrzeganie tych reguł. Przy pisaniu programów należy pilnować odpowiedniej organizacji i struktury kodu poprzez zastosowanie wcięć, oraz umieszczanie kolejnych instrukcji w nowych wierszach. Taka metoda ułatwia odszukanie konkretnego elementu kodu i ewentualne naniesienie modyfikacji.
- Kompilacja – Podczas pisania programu należy testować jego fragmenty, w tym celu kod się kompiluje i uruchamia. **Kompilator** analizuje poprawność kodu pod kątem gramatycznym i składniowym (poprawność syntaktyczna), a także poprawności na poziomie znaczenia poszczególnych instrukcji i całego programu (poprawność

semantyczna). Kompilator zidentyfikuje, zlokalizuje i wskaże błędy w kodzie. Po wykonaniu kompilacji zostanie wygenerowany kod wynikowy. Program jest gotowy do uruchomienia.

- Konsolidacja – W wyniku konsolidacji poszczególne moduły programu, które powstały w trakcie kompilacji łączone są w jeden program. Na etapie konsolidacji można dołączyć do programu dodatkowe zasoby.
- Testowanie – Etap niezbędny do stworzenia prawidłowo działającego programu. Tworzony program warto regularnie testować w trakcie pisania, umożliwia to naprawianie błędów na bieżąco. Ponadto, program testuje się w celu weryfikacji pod kątem zgodności ze specyfikacją oraz oczekiwaniami użytkownika.
- Optymalizacja – Ostatnim etapem jest poprawienie wydajności kodu.

2.2 Programowanie wizualne czy tekstowe

Od kilkunastu lat trwa trend przesuwania granic edukacji w stronę kompetencji cyfrowych i koncepcji „myślenia komputacyjnego" opierającego się na umiejętnościach rozwiązywania problemów przy wykorzystaniu metod informatycznych. Obecnie idee myślenia komputacyjnego są wdrażane od najmłodszych klas szkoły podstawowej. Duży nacisk kładzie się na edukacyjne podejście STEAM, wykorzystujące naukę, technologię, inżynierię, sztukę i matematykę do rozwijania w dzieciach m. in.: kreatywności, innowacyjności, umiejętności myślenia krytycznego i analitycznego.

Jedną z najskuteczniejszych metod rozwijania wymienionych wyżej cech jest nauka programowania. Aby nauka programowania dla dzieci była możliwa niezbędne było dostosowanie języków programowania do ich wieku i możliwości. W tym celu zostały stworzone wizualne języki programowania, które charakteryzują się przyjemnym środowiskiem programistycznym oraz łatwym i intuicyjnym sposobem tworzenia kodu. Wizualne języki programowania umożliwiają programowanie poprzez odpowiednie ustawianie bloków z poleceniami (np. poprzez przeciąganie i upuszczanie). Do najpopularniejszych języków wizualnych zaliczamy LUA, Scratch i Blockly.

3. Przygotowanie do programowania tekstowego

3.1 Podstawowe pojęcia związane z programowaniem

Tekstowe języki programowania zmuszają programistę do ręcznego pisania kodu źródłowego. Wiąże się to z koniecznością znajomości reguł syntaktycznych oraz semantyki charakterystycznych dla danego języka.

Języki programowania można sklasyfikować ze względu na:

- Przeznaczenie;
- Poziom ;
- Sposób wykonywania;
- Sposób kontroli typów:
- Generację;
- Paradygmat

Przed rozpoczęciem pisania kodu warto zapoznać się z pojęciami i terminami związanymi z programowaniem:

- **Algorytm** – zestaw zdefiniowanych i logicznych instrukcji, które umożliwiają wykonanie określonego zadania – rozwiązanie problemu;
- **Back-end** – elementy aplikacji lub strony internetowej znajdującej się na serwerze, do której nie ma dostępu użytkownik. Odpowiada za zarządzanie i działanie całej aplikacji lub strony internetowej;
- **Biblioteka** – plik zawierający zasoby, które mogą zostać wykorzystane z poziomu kodu źródłowego programu;
- **Błąd logiczny** – błąd w logice programu, często niepowodujący przerwania pracy programu;
- **Błąd składniowy** – błąd gramatyczny w kodzie lub literówka;
- **Debuger** -program służący do lokalizowania i usuwania błędów w programie;
- **Debugowanie** – usuwanie błędów z programu;
- **Edytor** - edytor tekstu wykorzystywany przez programistów do pisania kodu;
- **Funkcja** – procedura przetwarzająca dane w programie;

- **Framework** – platforma programistyczna stanowiąca szkielet tworzonej aplikacji. Określa strukturę oraz mechanizm działania aplikacji;
- **Front-end –** widoczna część aplikacji lub strony internetowej, którą może zarządzać użytkownik;
- **Instrukcja warunkowa** – funkcja, która umożliwia dalsze działanie programu JEŻELI jest spełniony określony warunek;
- **Interpreter** – program wykonujący inne programy. Zmienia tekst źródłowy programu wiersz po wierszu w kod maszynowy – ułatwia kontrolę pisanego kodu;
- **Kod źródłowy** – napisane przez programistę instrukcje programu;
- **Kod wynikowy** – kod utworzony w wyniku tłumaczenia kodu źródłowego;
- **Kompilator** – program przekształcający cały kod źródłowy na język maszynowy;
- **Kompilacja** – przetłumaczenie kodu źródłowego na kod maszynowy;
- **Oprogramowanie Open Source** – Otwarte zasoby, oprogramowanie z otwartym kodem źródłowym, którego licencja pozwala na legalne i darmowe kopiowanie, a także samodzielne modyfikowanie;
- **Pętla** – zestaw określonych instrukcji, wielokrotnie powtarzanych w trakcie jego wykonywania;
- **Pętla „for”** – zestaw instrukcji, które będą wykonane określoną ilość razy;
- **Pętla „while”** – zestaw instrukcji, które będą wykonywane dopóki będzie spełniony określony warunek;
- **Składnia** – zbiór reguł syntaktycznych dla określonego języka;
- **Skrypt** – pliki tekstowe zawierające instrukcje, które można uruchomić jak zwykły program;
- **Zintegrowane środowisko programistyczne** – Zestaw programów umożliwiających tworzenie, edytowanie, testowanie i modyfikację oprogramowania;
- **Zmienne** – nazwana lokalizacja miejsca przechowywania danych w programie;

3.2 Przykłady tekstowych języków programowania i praktyczne informacje dotyczące ich wykorzystania, w tym do tworzenia gier i programów edukacyjnych

Przed wyborem języka, który posłuży do napisania kodu programista musi wiedzieć jakie ma być jego przeznaczenie. Back-end developer do pracy wykorzystuje język **Python** lub **Java,** natomiast front-end developer z tych języków być może nawet nie skorzysta, ale niezbędny do pracy będzie język J**avaScript.**

Obok gradacji ze względu na przeznaczenie jednym z głównych sposobów podziału języków programowania jest podział na języki skryptowe (często są językami interpretowanymi) i kompilowane.

Programy napisane w językach skryptowych mają postać plików tekstowych, które można otwierać, edytować i modyfikować w edytorach tekstu. Do uruchomienia potrzebują dodatkowego programu, który interpretuje kod źródłowy wiersz po wierszu. W momencie napotkania błędu program zatrzyma wykonywanie Charakterystyczną cechą dla języków skryptowych jest natychmiastowe wykonanie zmian wprowadzanych w kodzie. Języki skryptowe są prostsze od kompilowanych, ale mniej wydajne. Zaletą jest łatwość pisania kodu ze względu na mniej rygorystyczną składnię. Są przeznaczone do tworzenia mniejszych projektów, przy większych pojawiają się problemy ze sprawnym działaniem. Brak ścisłych reguł syntaktycznych może powodować nieczytelność kodu.

Programy napisane w językach kompilowanych nie są interpretowane i wykonywane na bieżąco. Przed wykonaniem muszą zostać skompilowane. W odróżnieniu od języków skryptowych kod źródłowy musi zostać skompilowany przed uruchomieniem, w związku z tym każda modyfikacja i naprawianie błędów wymaga ponownej kompilacji. Programy pisane w językach kompilowanych są szybsze i wydajniejsze.

Warto zaznaczyć, że zarówno języki skryptowe i kompilowane służą do innych celów przez co nie można przyjąć, że jedne są lepsze od drugich. Obie grupy języków mają swoje zalety i wady.

Poniżej zostały przedstawione popularne języki programowania z informacjami dotyczącymi praktycznego ich wykorzystania. Należy pamiętać, że większość języków

została stworzona do konkretnych zadań (JavaScript do Front-endu, Java do tworzenia aplikacji na urządzenia mobilne). Umieszczone poniżej praktyczne informacje ułatwią dobór języka do realizacji konkretnego celu.

Ilustracja 2.1, autor: Jacek Chmielewski

Java

OPIS

Język wysokiego poziomu o ogólnym przeznaczeniu. java jest językiem używanym do tworzenia aplikacji na komputerze. Wtyczka internetowa Java umożliwia uruchamianie aplikacji w internecie.

DO CZEGO MOŻNA WYKORZYSTAĆ?

Aplikacje na urządzenia mobilne

Programowanie gier wideo

Graficzne interfejsy użytkownika

Oprogramowanie

- Ciągle rozwijany język programowania
- Jeden z najlepszych języków do tworzenia aplikacji mobilnych

- Wykorzystuje bardzo dużo pamięci
- Trudny do nauki
- potrzebuje dużo czasu na uruchomienie

Język Java został wykorzystany przy tworzeniu:

Usługi Gmail

Minecraft

Ilustracja 2.2, autor: Jacek Chmielewski

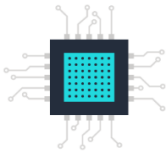
C

OPIS

Język wysokiego poziomu o ogólnym przeznaczeniu. Pierwotnie przeznaczony do pisania oprogramowania systemowego. Posiada prostą składnię.

DO CZEGO MOŻNA WYKORZYSTAĆ?

Systemy operacyjne



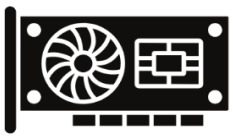
Sprzęt komputerowy

Oprogramowanie

- Niezwykle uniwersalny, można w nim pisać programy na wiele platform
- Programy pisane w C wykorzystują bardzo mało miejsca i pamięci

- Nie obsługuje programowania obiektowego
- Trudny do nauki

Język C został wykorzystany przy tworzeniu:



OpenGL

Linux

Minecraft

Ilustracja 2.3, autor: Jacek Chmielewski

C++

OPIS

Obiektowy język programowania ogólnego przeznaczenia. Jeden z najpopularniejszych i najlepiej rozwiniętych języków. Znajduje zastosowanie w wielu dziedzinach związanych z programowaniem.

DO CZEGO MOŻNA WYKORZYSTAĆ?

Wyszukiwarki

Programowanie gier wideo

Systemy operacyjne

Oprogramowanie

- Łatwo uzyskać wsparcie w przypadku napotkania problemu
- Jest przenośny
- Podobny do Javy i Pythona, które się opierają na C

- Niezwykle rozbudowany i skomplikowany język

Język C++ został wykorzystany przy tworzeniu:

Google

Outlook

Ilustracja 2.4, autor: Jacek Chmielewski

C# (SHARP)

OPIS

Obiektowy język programowania ogólnego przeznaczenia. Służy głównie do pisania aplikacji na system Windows. Ponadto, umożliwia tworzenie aplikacji webowych po stronie serwera, a także tworzenie gier poprzez wykorzystanie go w silniku UNITY.

DO CZEGO MOŻNA WYKORZYSTAĆ?

Aplikacja na Windows

Programowanie gier wideo

Aplikacje biznesowe

Oprogramowanie

- Opiera się na języku C,
- Znając język c# łatwo zacząć programowanie w Javie, PHP i C++
- Zapewnia dostęp do zasobów zawierających funkcjonalności i wsparcie

- Bardzo trudny do nauki dla początkujących programistów
- Jest ściśle zintegrowany ze środowiskiem .net. przez co trudno go wykorzystać na innych platformach niż Windows

Język Java został wykorzystany przy tworzeniu:

Stackoverflow

Evernote

Ilustracja 2.5, autor: Jacek Chmielewski

OPIS

Skryptowy język programowania po stronie serwera. Jest wykorzystywany do tworzenia stron internetowych, traktowany również jako język ogólnego przeznaczenia. Powszechnie używany przy tworzeniu for internetowych i blogów.

DO CZEGO MOŻNA WYKORZYSTAĆ?

Wtyczki WordPress

Strony internetowe

Strony z obsługą baz danych

- Łatwy i wszechstronny
- Większość stron internetowych oparta jest na PHP
- Nie wymaga kompilacji przy poprawianiu kodu

- Słaba obsługa błędów.
- Zmiany w języku są na bieżąco wprowadzane przez co wsparcie czasami nie nadąża z wprowadzaniem rozwiązań
- Łatwo pomylić się w stylu kodu

Język php został wykorzystany przy tworzeniu:

facebook

WordPress

Ilustracja 2.6, autor: Jacek Chmielewski

JavaScript

OPIS

Skryptowy język programowania wysokiego poziomu. Jest obok HTML i CSS podstawowym narzędziem do tworzenia stron internetowych. Wykorzystywany głównie do zwiększania interaktywności stron. Ponadto, umożliwia tworzenie aplikacji desktopowych.

DO CZEGO MOŻNA WYKORZYSTAĆ?

Front-end

Widżety

Elementy interaktywne w sieci

Analityka

- Programy pisane w JS są bardzo szybkie w działaniu
- Skrypty można tworzyć na każdej stronie internetowej, niezależnie od rozszerzenia
- Posiada prostą składnię

- Istnieje ryzyko ataku przez złośliwe oprogramowanie
- Ten sam program może być inaczej interpretowany przez różne przeglądarki

Język JavaScript został wykorzystany przy tworzeniu:

PayPal

YouTube

Ilustracja 2.7, autor: Jacek Chmielewski

4. Programowanie w różnych programach i językach

4.1 Python

Python jest jednym z łatwiejszych tekstowych języków programistycznych, z tego względu jest często polecany jako język do rozpoczęcia nauki programowania, ponadto korzystanie z niego jest zupełnie darmowe. Python umożliwia tworzenie skryptów, czyli prostych programów mogących zautomatyzować i ułatwić pracę na komputerze przez co korzystają z niego również m. in. programiści w innych językach i administratorzy sieci.

Dobrym środowiskiem do rozpoczęcia nauki programowania w Pythonie jest „Świat Reeborga" – narzędzie udostępnione przez firmę ABIX. Jest to środowisko do pracy w przeglądarce w związku z czym nie jest konieczne instalowanie żadnego dodatkowego programu. Świat Reeborga jest platformą programistyczną, na której możemy programować zarówno blokowo i tekstowo wirtualnego robota.

Należy pamiętać, że tego typu platformy są świetne do rozpoczęcia nauki programowania tekstowego, ale w pewnym momencie należy przejść na naukę programowania w „czystym" kodzie Pythona. Świat Reeborga jest stworzony na odmianie Pythona o nazwie Brython, która umożliwia programowanie w przeglądarce. Zastosowanie odmiany Brython sprawia, że kod w niej stworzony może nie działać w czystym Pythonie.

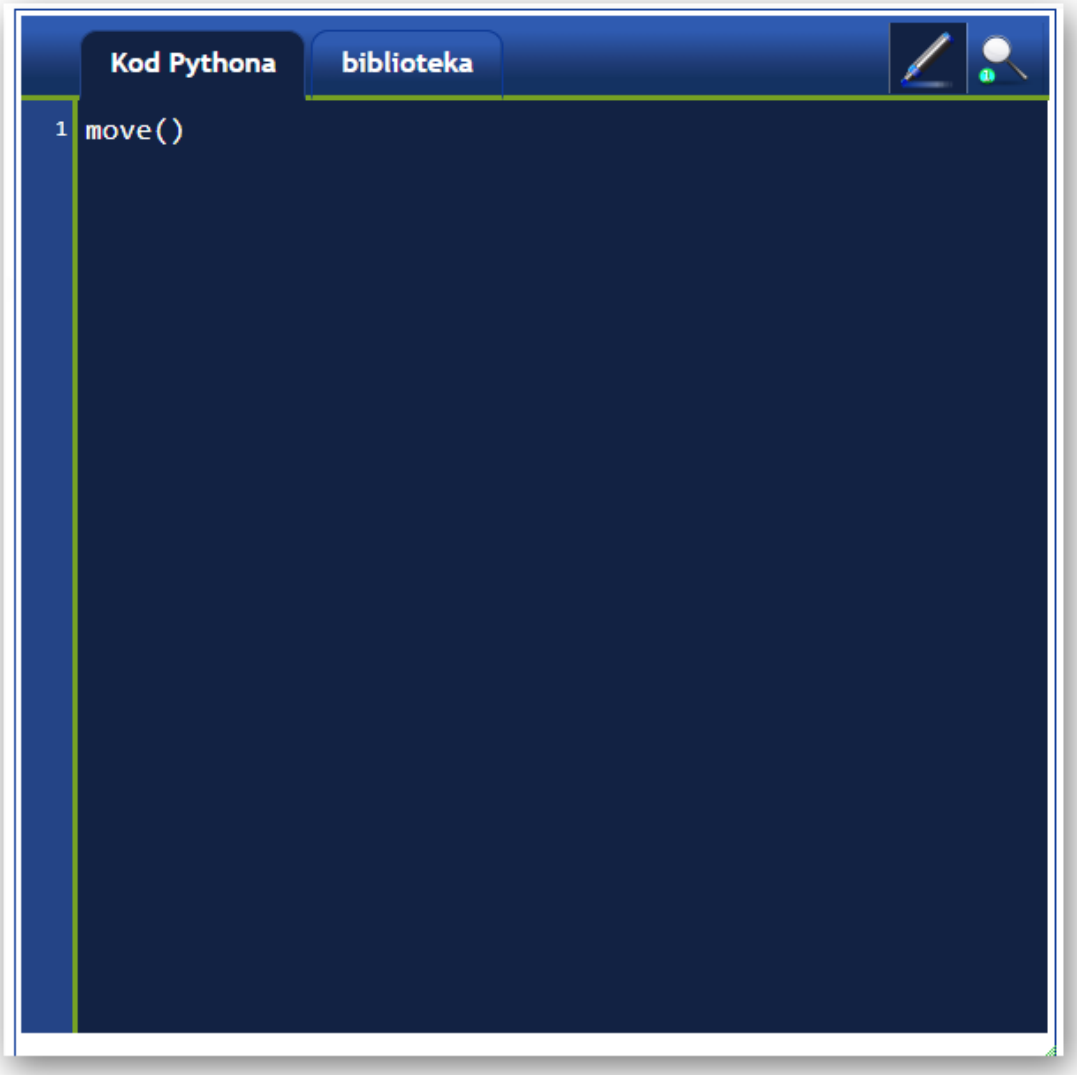
W Świecie Reeborga mamy do wyboru kilka możliwości programowania:

- **Tekstowe**
  - Python;
  - JavaScript;
- **Blokowe –** bazujące na Scratchu
  - Oparte o Pythona;
  - Oparte o JavaScript

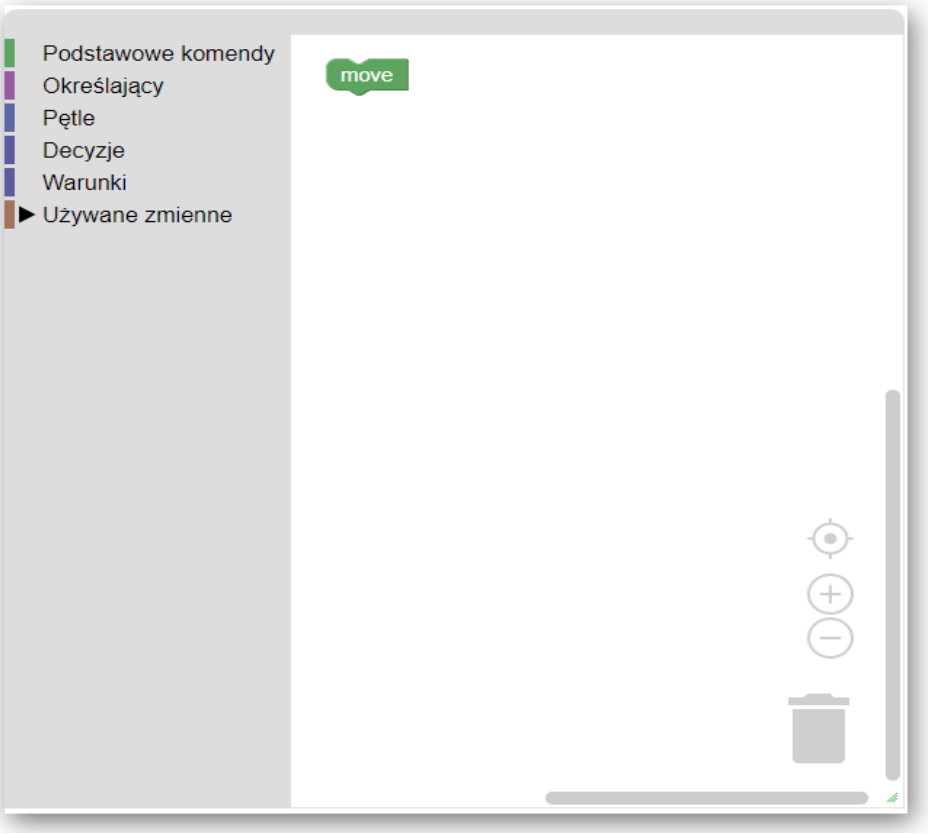
Interfejs aplikacji jest bardzo prosty i zawiera następujące elementy:

- Konsola do tworzenia kodu w opcji tekstowej (ilustracja 3.1) i opcji blokowej (ilustracja 3.2);

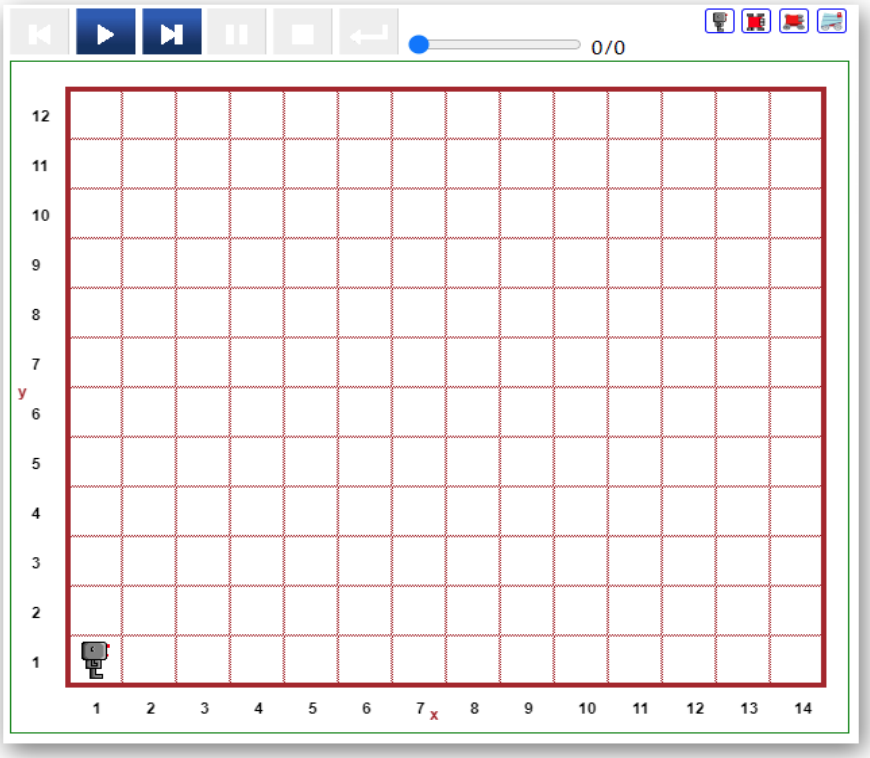
- Mapa świata w którym robot wykonuje nasze polecenia (ilustracja 3.3);
- Pasek menu u góry ekranu (ilustracja 3.4).



Ilustracja 3.1, autor: Jacek Chmielewski



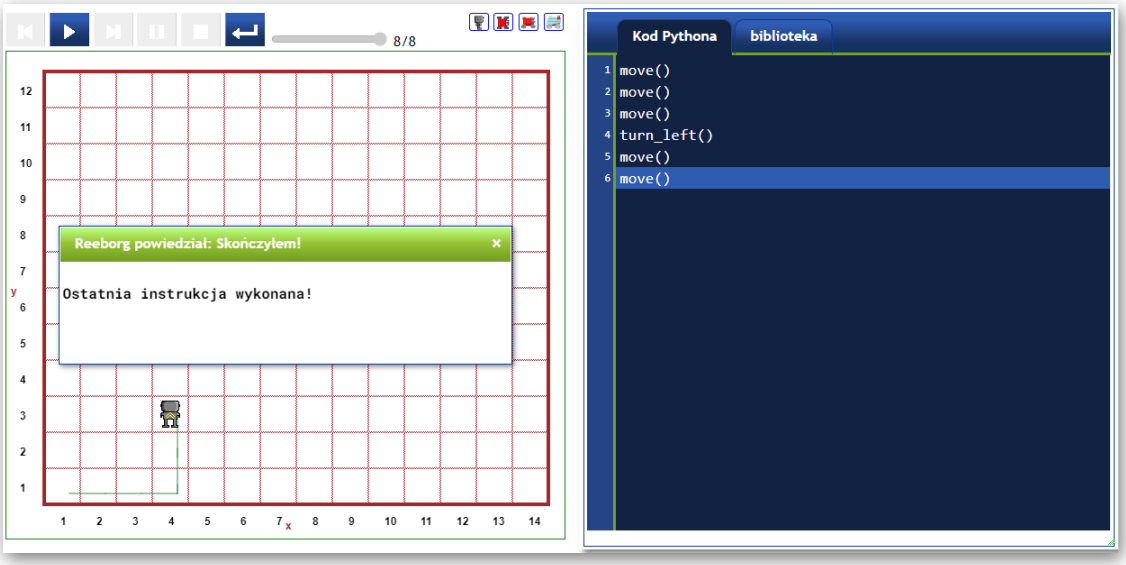
Ilustracja 3.2, autor: Jacek Chmielewski



Ilustracja 3.4, autor: Jacek Chmielewski

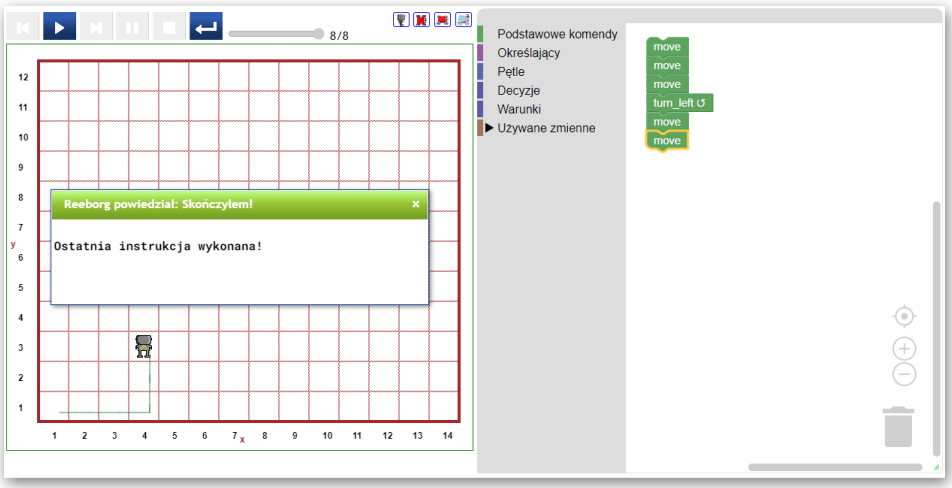
Menu górne umożliwia nam zmianę świata, w którym działa robot, informacje, które przedstawiają założenia zadania w danym świecie. W kolejnym oknie możemy wybrać język, w którym chcemy programować. Przy wyborze języka tekstowego możemy skorzystać z klawiatury Reeborga, zawiera ona gotowe komendy, warunki i polecenia, które możemy wykorzystać przy tworzeniu kodu. Dodatkowe opcje są szczególnie przydatne w tworzeniu własnego świata.

Na ilustracjach 3.5 i 3.6 przedstawiono najprostszy program w dwóch wersjach.



http://robot.abixedukacja.eu/?lang=pl-en&mode=python

Ilustracja 3.5, autor: Jacek Chmielewski

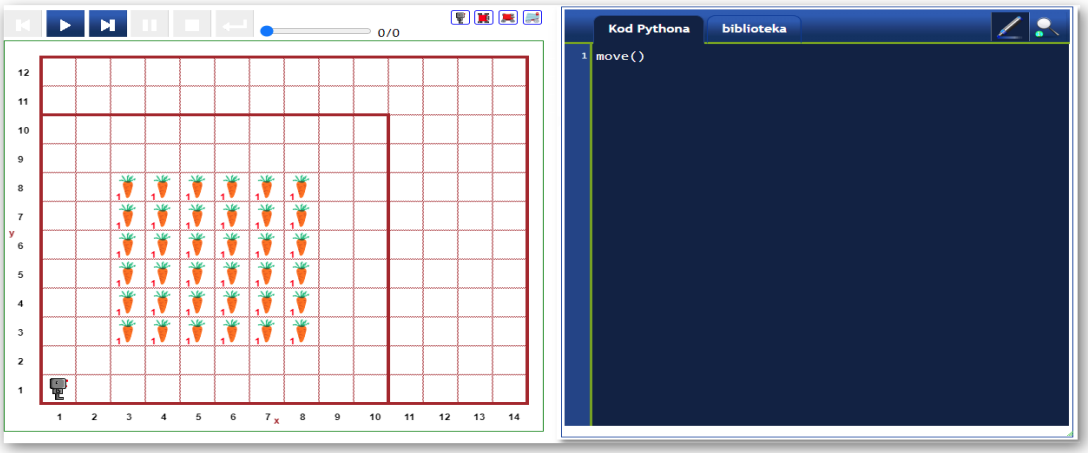


http://robot.abixedukacja.eu/?lang=pl-en&mode=blockly-py

Ilustracja 3.6, autor: Jacek Chmielewski

Programowania tekstowe i blokowe w Pythonie z wykorzystaniem platformy „Świat Reeborga”

Aby zacząć realizować zadanie należy wejść na stronę robot.abixedukacja.eu oraz wybrać świat **Harvest 1**. Po wybraniu wskazanego świata powinna pojawić się następująca mapa (ilustracja 3.7).



http://robot.abixedukacja.eu/?lang=pl-en&mode=python

Ilustracja 3.7, autor: Jacek Chmielewski

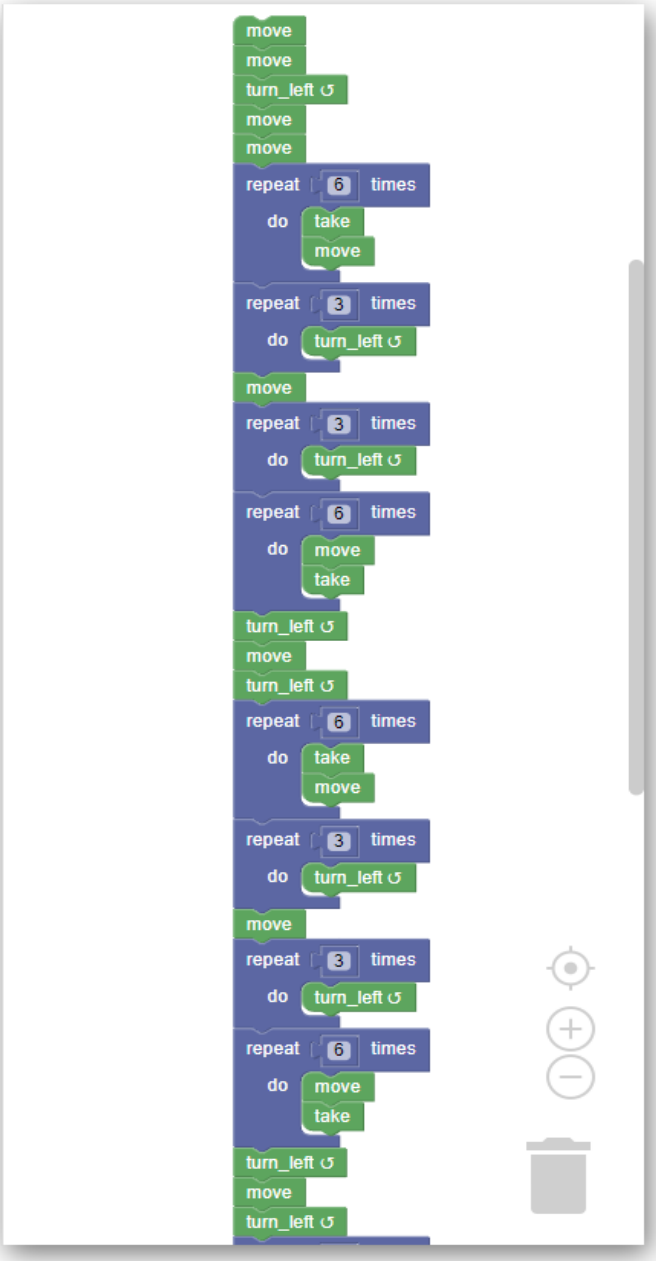
Zadaniem robota jest zebranie wszystkich marchewek za pośrednictwem jednego programu. W tym celu należy w menu wyboru języka w górnej części strony wybrać język Blockly(Py). Po przejściu na programowanie blokowe przechodzimy do tworzenia kodu. Należy pamiętać o zebraniu wszystkich marchewek. W celu ułatwienia warto wykorzystać pętle powtórzeń.

Aby robot wykonał zadanie zebrania wszystkich marchewek powinien wykonać następujące polecenia:

- dwa ruchy do przodu
- obrót w lewo
- dwa ruchy do przodu
- 6 powtórzeń sekwencji [podnieś - ruch do przodu]
- 3 powtórzenia obrót w lewo
- Ruch do przodu
- 3 powtórzenia obrót w lewo
- 6 powtórzeń sekwencji [ruch do przodu - podnieś]
- obrót w lewo
- Ruch do przodu
- obrót w lewo
- 6 powtórzeń sekwencji [podnieś - ruch do przodu]
- 3 powtórzenia obrót w lewo
- Ruch do przodu
- 3 powtórzenia obrót w lewo
- 6 powtórzeń sekwencji [ruch do przodu - podnieś]
- obrót w lewo
- Ruch do przodu
- obrót w lewo
- 6 powtórzeń sekwencji [podnieś - ruch do przodu]
- 3 powtórzenia obrót w lewo
- Ruch do przodu

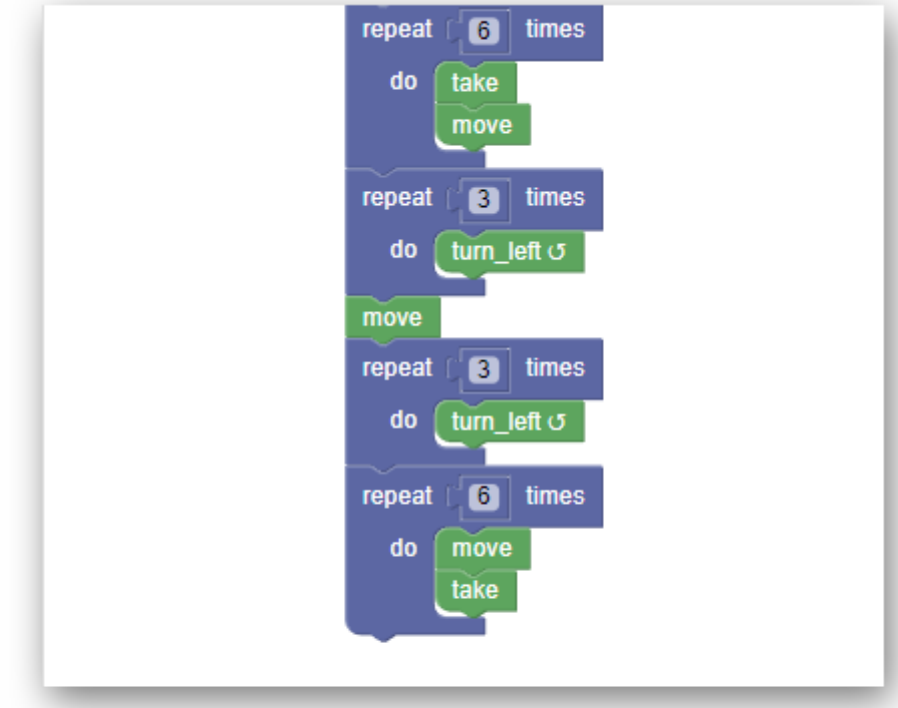
- 3 powtórzenia obrót w lewo
- 6 powtórzeń sekwencji [ruch do przodu - podnieś]

Programując w blockly(Py) kod powinien wyglądać jak na ilustracji 3.8 i 3.9.



http://robot.abixedukacja.eu/?lang=pl-en&mode=blockly-py

Ilustracja 3.8, autor: Jacek Chmielewski



http://robot.abixedukacja.eu/?lang=pl-en&mode=blockly-py

Ilustracja 3.9, autor: Jacek Chmielewski

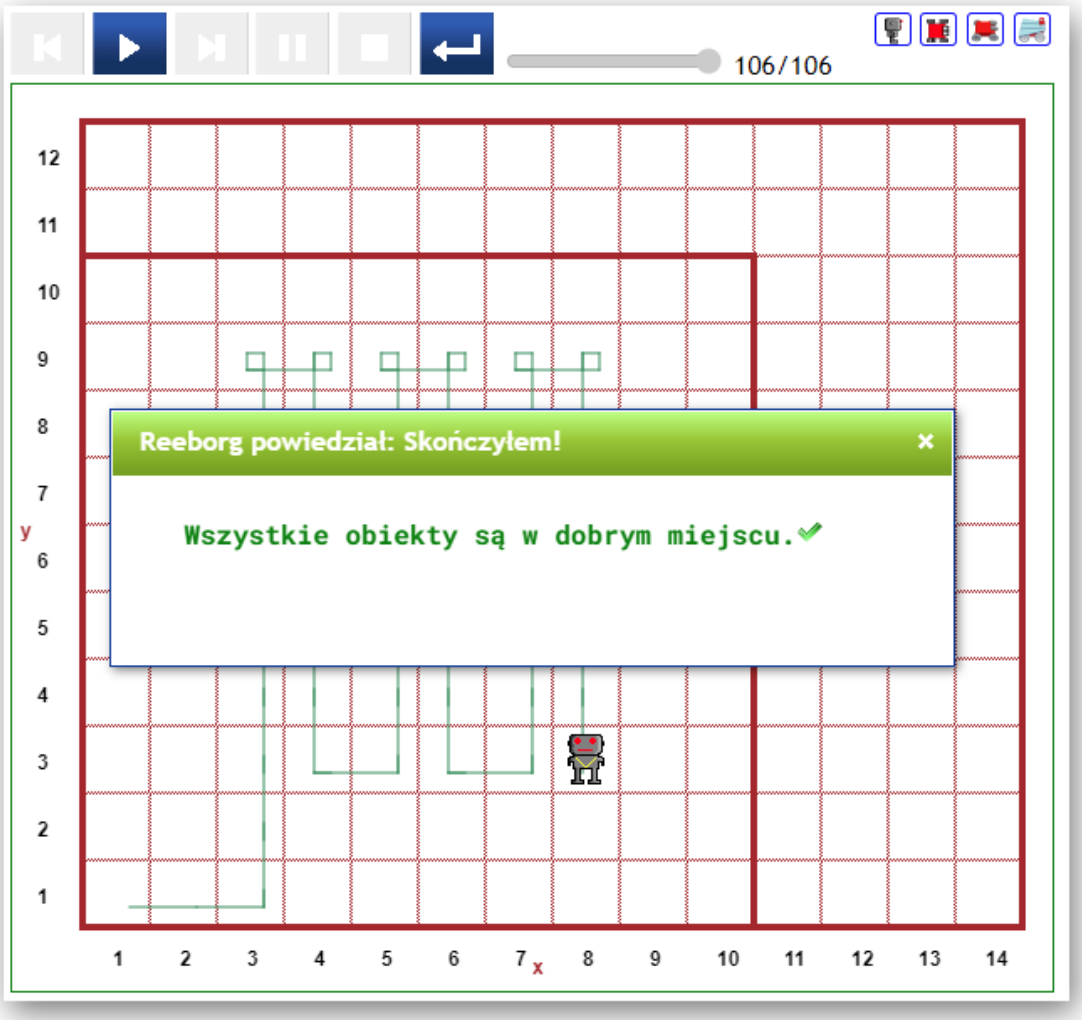
Przełączając język programowania na Pythona otrzymamy następujący kod w konsoli (ilustracja nr 3.10). Kod tekstowy, który pojawił się w konsoli po przełączeniu na język Python jest wynikiem translacji z języka Blockly(Py), w którym został stworzony. Uruchamiając zarówno jeden i drugi program otrzyma się identyczny wynik (ilustracja nr 3.11).

Kod Pythona | biblioteka

```
move()
move()
turn_left()
move()
move()
for count in range(6):
    take()
    move()
for count2 in range(3):
    turn_left()
move()
for count3 in range(3):
    turn_left()
for count4 in range(6):
    move()
    take()
turn_left()
move()
turn_left()
for count5 in range(6):
    take()
    move()
for count6 in range(3):
    turn_left()
move()
for count7 in range(3):
    turn_left()
for count8 in range(6):
    move()
    take()
turn_left()
move()
turn_left()
for count9 in range(6):
    take()
    move()
for count10 in range(3):
    turn_left()
move()
for count11 in range(3):
    turn_left()
for count12 in range(6):
    move()
    take()
```

http://robot.abixedukacja.eu/?lang=pl-en&mode=python

Ilustracja 3.10, autor: Jacek Chmielewski



http://robot.abixedukacja.eu/?lang=pl-en&mode=blockly-py

Ilustracja 3.11, autor: Jacek Chmielewski

4.2 Tworzenia gier i programów edukacyjnych w Kodu Game Lab

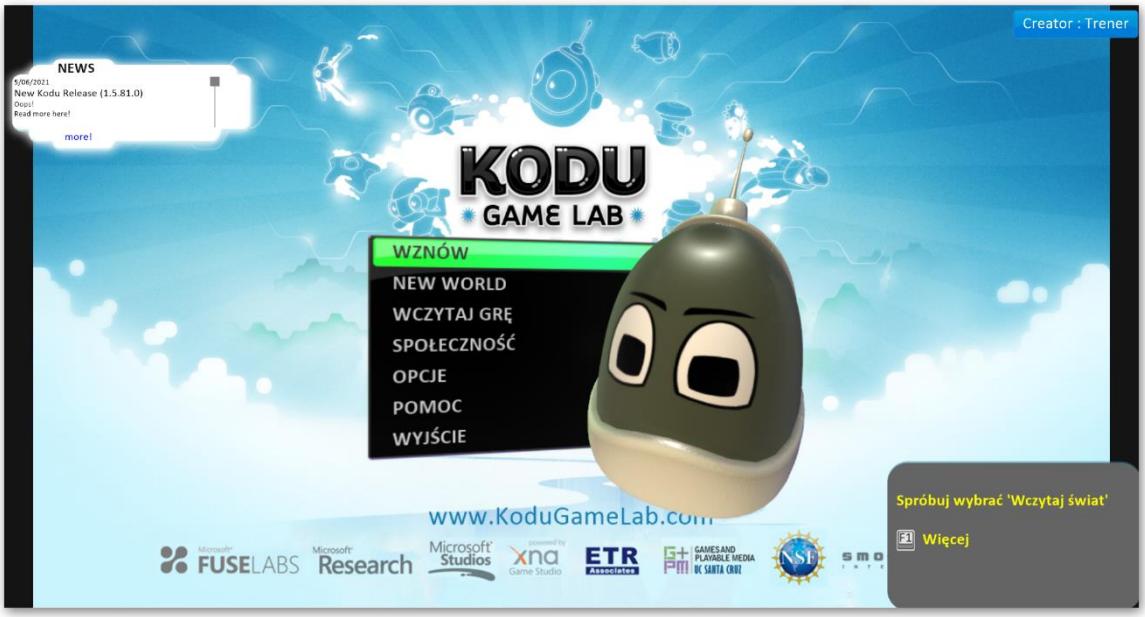
Kodu Game lab to rozbudowane narzędzie do tworzenia i programowania gier 3D, które można wykorzystać do tworzenia programów edukacyjnych. Jest prostą w obsłudze i darmową aplikacją, z której są w stanie korzystać nawet dzieci, które dopiero zaczynają czytać. Do pracy nie wymaga wiedzy programistyczne ze względu na wykorzystanie wszystkich poleceń w postaci bloków.

Programowanie w Kodu Game Lab wymaga zainstalowania programu na komputerze. Po uruchomieniu programu użytkownik musi podać swoją nazwę/login oraz PIN/hasło (ilustracja 3.12). Menu aplikacji jest proste i intuicyjne – nie powinno sprawić żadnych trudności w obsłudze. Ilustracja 3.13.

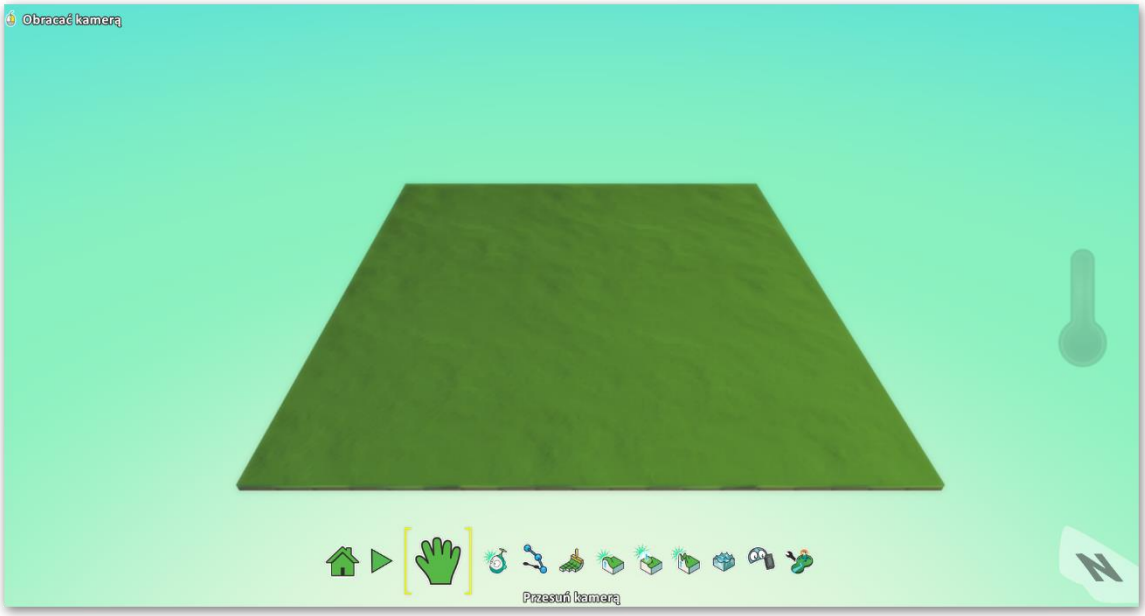
Mniej intuicyjny jest panel wyboru obiektów i funkcji. Przy pierwszym spotkaniu z programem może wprowadzić w zakłopotanie (ilustracja 3.14). W rzeczywistości program jest niezwykle prosty w obsłudze i opiera się na kilku schematach.

Programowanie gry rozpoczynamy od stworzenia świata. Po utworzeniu świata wyświetli się kwadratowa zielona plansza, którą można powiększyć, zmienić jej kształt, podłoże oraz ukształtowanie terenu. Po przygotowaniu świata gry należy wybrać postać, która będzie sterowana (ilustracja 3.15), a także obiekty ewentualnych przeciwników. Następnie postać sterowana musi zostać zaprogramowana tak, aby była w stanie wykonać polecenia programisty (ilustracje 3.16, 3.17 i 3.18). Można również zaprogramować obiekty, które stanowią przeciwników.

Ilustracja 3.12, autor: Jacek Chmielewski

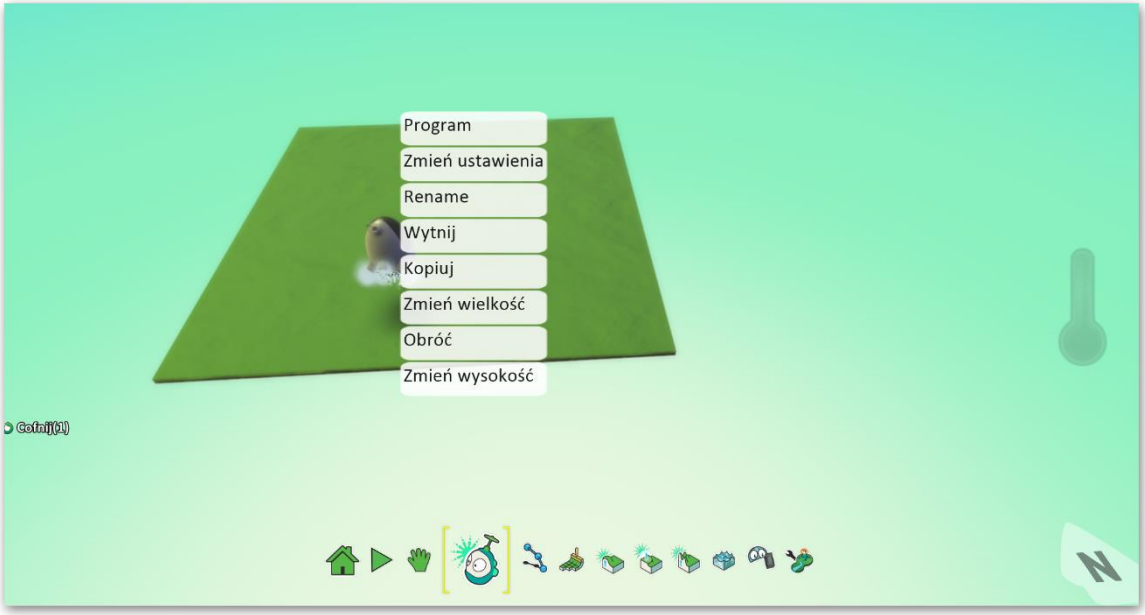


Ilustracja 3.13, autor: Jacek Chmielewski

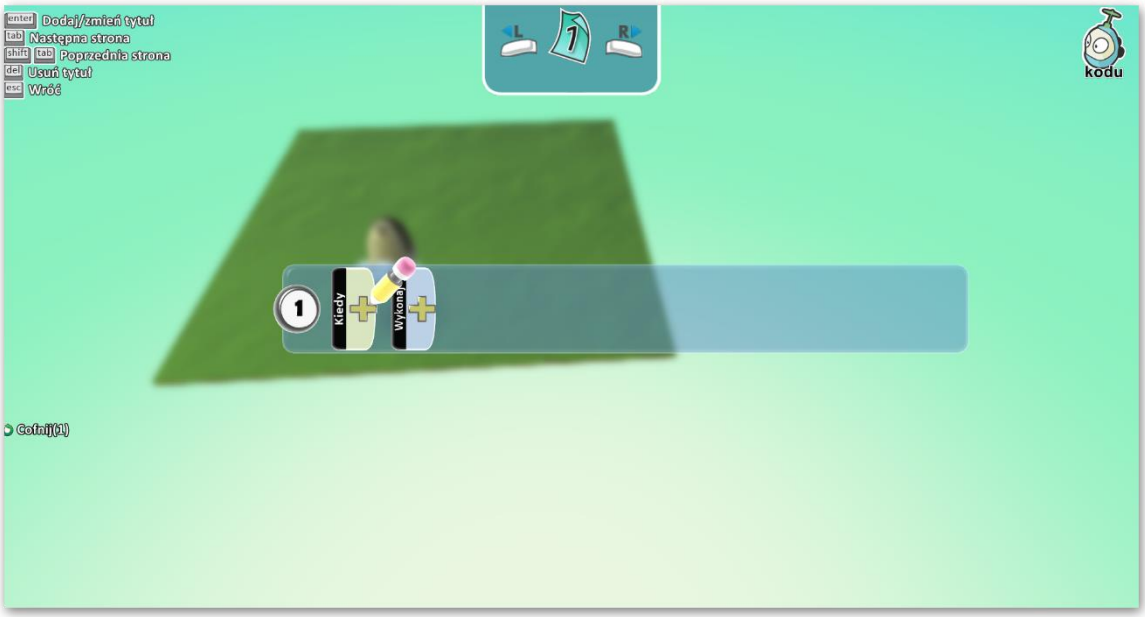


Ilustracja 3.14, autor: Jacek Chmielewski

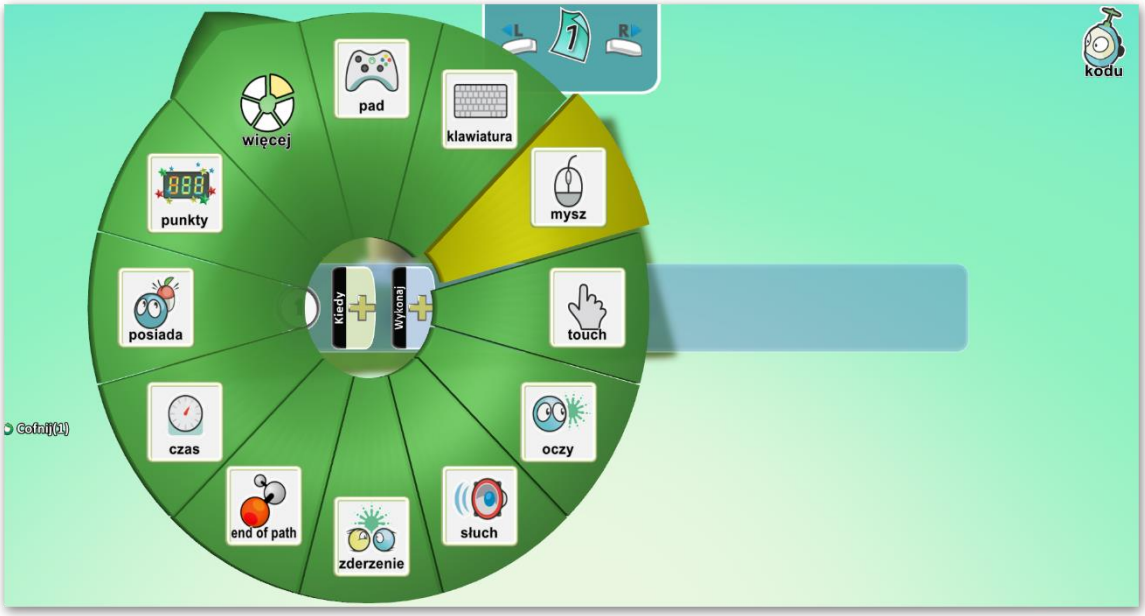
Ilustracja 3.15, autor: Jacek Chmielewski



Ilustracja 3.16, autor: Jacek Chmielewski



Ilustracja 3.17, autor: Jacek Chmielewski



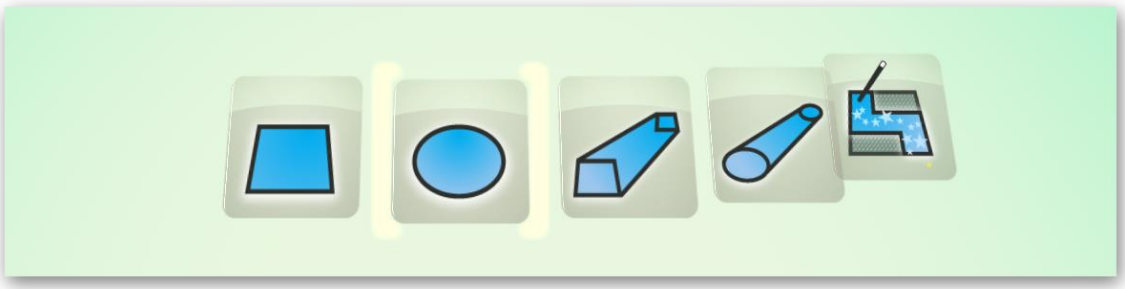
Ilustracja 3.18, autor: Jacek Chmielewski

Wykorzystanie Kodu Game Lab do stworzenia gry wyścigowej

Przygotowanie gry należy rozpocząć od stworzenia odpowiedniego świata, który będzie jednocześnie trasą wyścigu. W tym celu należy skorzystać z opcji znajdujących się u dołu

ekranu i przygotować świat o odpowiednim kształcie i wielkości dla zaplanowanej trasy (ilustracje 3.19, 3.20 i 3.21).

Ilustracja 3.19, autor: Jacek Chmielewski



Ilustracja 3.20, autor: Jacek Chmielewski

Ilustracja 3.21, autor: Jacek Chmielewski

Kolejnym krokiem jest stworzenie robota, którym będzie sterował użytkownik oraz jego przeciwnika. W tym celu należy kliknąć w ikonę robota (ilustracja 3.22) w dolnym menu i kliknąć lewym przyciskiem myszy w miejscu, w którym będzie rozpoczynał się wyścig.

Ilustracja 3.22, autor: Jacek Chmielewski

Po umieszczeniu robota na trasie należy go zaprogramować, żeby reagował na nasze polecenia. W tym celu należy kliknąć na robota prawym przyciskiem myszy i z menu wybrać **program** (ilustracja 3.23). Po pojawieniu się konsoli do programowania (ilustracja 3.24) należy kliknąć na znak plus przy polu **KIEDY** i wybrać **klawiaturę** (ilustracja 3.25) a w następnie **klawisze**, którymi użytkownik chce sterować robotem (domyślnie strzałki lub WASD). Po wybraniu sposobu sterowania robotem należy wrócić do konsoli do programowania (ilustracja 3.24) i wybrać znak plus przy polu **WYKONAJ**, po tej stronie użytkownik ustawia operacje, które robot będzie wykonywał w momencie spełnienia warunku ustawionego po stronie **KIEDY** (w tym przypadku wciśnięcie odpowiednich klawiszy na klawiaturze). Po stronie **WYKONAJ** należy wybrać opcję **ruch** oraz **szybko** (ilustracja 3.26).

Ilustracja 3.23, autor: Jacek Chmielewski

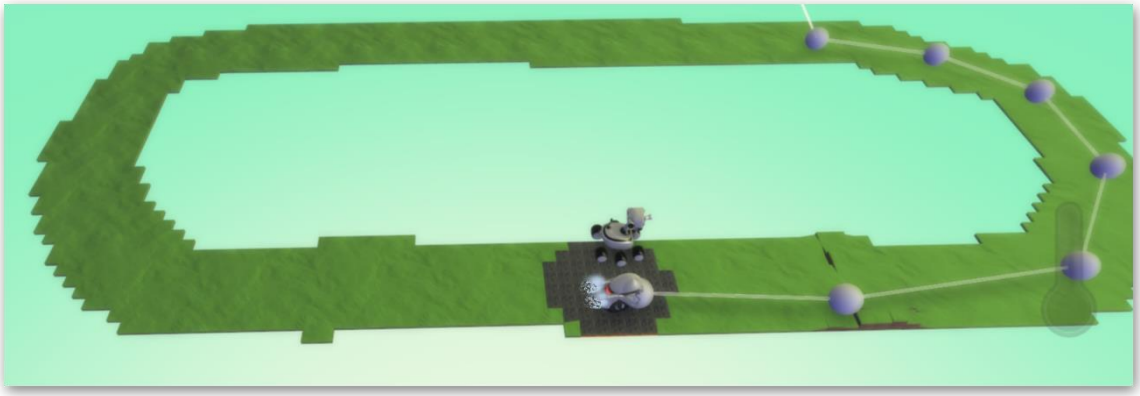
Ilustracja 3.24, autor: Jacek Chmielewski

Ilustracja 3.25, autor: Jacek Chmielewski

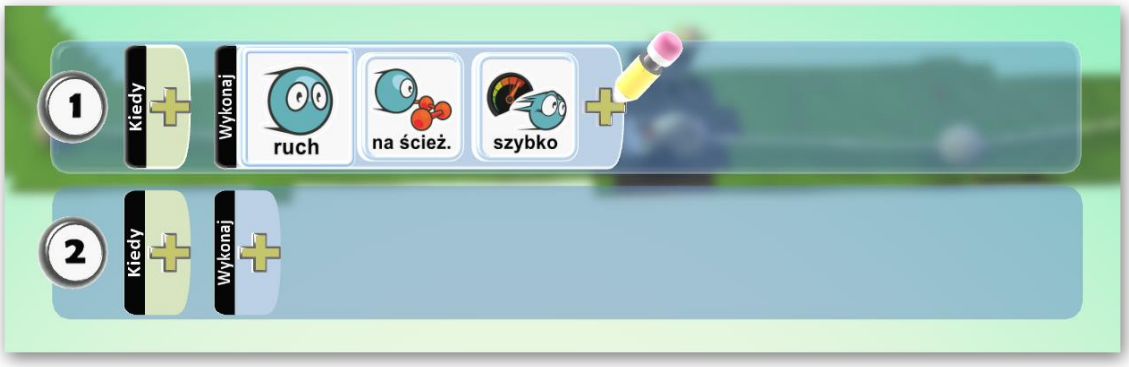
Ilustracja 3.26, autor: Jacek Chmielewski

Po zaprogramowania robota sterowalnego należy wrócić do trasy wyścigu (klawisz Esc) i stworzyć przeciwnika. Przeciwnika tworzy się w ten sam sposób jak robota sterowalnego. Po stworzeniu przeciwnika konieczne jest utworzenie ścieżki, po której będzie się poruszał, w tym celu z dolnego menu należy wybrać ikonę ścieżki (ilustracja 3.27) i poprowadzenie jej przez całą trasę (ilustracja 3.28). Po stworzeniu ścieżki niezbędne jest zaprogramowanie przeciwnika. Po kliknięciu na przeciwnika prawym przyciskiem myszy i wybraniu opcji program pojawia się konsola jak na ilustracji 3.24. Ze względu na poruszanie się przeciwnika po gotowej ścieżce nie ustawia się warunków po stronie **KIEDY**. Należy ustawić wyłącznie polecenia po stronie **WYKONAJ**. W tym przypadku robot powinien wykonać następujące polecenia: **ruch – na ścieżce – szybko** (ilustracja 3.29).

Ilustracja 3.27, autor: Jacek Chmielewski



Ilustracja 3.28, autor: Jacek Chmielewski



Ilustracja 3.29, autor: Jacek Chmielewski

Kolejnym elementem do stworzenia jest umowna meta, czyli obiekt, który będzie oznaczał koniec wyścigu. W tym celu warto stworzyć obiekt (np. kodu) na końcu ścieżki przeciwnika. Po stworzeniu mety niezbędne będzie stworzenie kolejnych programów dla uczestników wyścigu.

Po wejściu w konsolę robota sterowalnego w drugim programie po stronie **KIEDY** należy ustawić zderzenie – kodu a po stronie **WYKONAJ** gra – **wygrana** (ilustracja 3.30). Przeciwnik powinien mieć ustawiony podobny program jak robot sterowalny, jednak zamiast wygrana należy ustawić polecenie **koniec**.

Po ustawieniu powyższych programów można uruchomić grę. W tym celu należy w dolnym menu kliknąć ikonę uruchom i wziąć udział w wyścigu. Jeśli użytkownik wygra wyścig pojawi się informacja o wygranej (ilustracja 3.31), podobnie w przypadku porażki (ilustracja 3.32).

Ilustracja 3.30, autor: Jacek Chmielewski

Ilustracja 3.31, autor: Jacek Chmielewski

Ilustracja 3.32, autor: Jacek Chmielewski

5. Diagnozowanie błędów i nauka na ich podstawie

Umiejętność diagnozowania i poprawiania błędów jest jedną z najważniejszych umiejętności programisty. Wymaga to znajomości składni języka, rozumienia semantyki, testowania oraz rozumienia założeń programu. Błędy w kodzie programu często stanowią główne źródło frustracji programistów. Ma na to wpływ głównie trudność ze znalezieniem wadliwego fragmentu kodu i brak zrozumienia sytuacji, dlaczego dany kod jest wadliwy.

Ogólnym podziałem błędów programistycznych jest następujący podział:

- Błędy składniowe (Syntax errors) – skutecznie uniemożliwiają uruchomienie programu. Najpopularniejszymi błędami składniowymi są literówki, które pojawiły w trakcie pisania oraz błędy wynikające z nieznajomości składni danego języka. Narzędzia środowiska programistycznego, w którym powstaje kod wskazują lokalizację błędu z dokładnością do jednego wiersza, jednak informacja o błędzie jest bardzo uproszczona przez co często może być niezrozumiała dla niedoświadczonych programistów;
- Błędy wykonania (run-time errors) – pojawiają się podczas działania programu, powodując jego zatrzymanie. Zazwyczaj błędy wykonania są spowodowane próbą spełnienia polecenia niemożliwego do wykonania (np. dzielenia przez zero). Błędy wykonania mogą wynikać z wykorzystania konkretnych wartości danych, w takim przypadku ujawnią się przy każdym włączeniu programu. W zależności od języka komunikaty mogą mieć formę konkretnego komunikatu z dokładną przyczyną (Java, Python) lub ogólnej informacji i wystąpieniu błędu (C);
- Błędy logiczne – Skutkiem zaistnienia błędu logicznego jest sytuacja, w której program nie wykonuje zadań, których oczekuje od niego programista. W odróżnienia od poprzednich typów błędów program uruchamia się normalnie i nie informuje o zaistniałych błędach. Błędy logiczne są najtrudniejsze do zdiagnozowania.

Praktycznie w każdej sferze życia popełnianie błędów wiąże się z poniesieniem określonych konsekwencji w postaci ukarania. Forma ukarania różni się od charakteru

popełnionego błędu, błąd popełniony za kierownicą samochodu skutkuje mandatem a błąd w pracy wiąże się z ryzykiem sankcji ze strony przełożonego. Nie inaczej jest w edukacji, popełniane błędy mogą obniżyć ocenę. Bardzo ważne jest aby w nauce programowania błędy traktować jako niepożądany efekt dążenia do celu i rozwoju. Świadomość braku konsekwencji za pomyłki w kodzie sprawią, że uczniowie chętniej będą podejmować własną inicjatywę i wykorzystywać wyobraźnię.

Nauka diagnozowania błędów jest o tyle istotna, że uczy myślenia analitycznego oraz kreatywnego podejścia do rozwiązywania problemów.

Przygotowując się do nauki diagnozowania błędów warto przygotować wcześniej kilka przykładów gotowego kodu w danym programie lub aplikacji, kod powinien zawierać błędy celowo przygotowane przez nauczyciela, które powinny przynieść następujące efekty przy uruchomieniu programu:

- Brak możliwości uruchomienia programu ze względu na błąd w składni;
- Uruchomienie programu, ale zatrzymanie w trakcie pracy ze względu na błąd wykonania;
- Uruchomienie programu, który wykona zadania nie będące celem programisty.

Aby mieć możliwość przeanalizowania kodu pod kątem ewentualnych błędów uczniowie poza przygotowanym błędnym kodem, powinni otrzymać również założenia, które dany program powinien wykonać. Diagnozowanie błędów warto rozpocząć od umieszczania pojedynczych błędów w kodzie. Błędy na początku powinny być łatwe i dość oczywiste, np. nauczyciel udostępnienia uczniom kod, który sprawi, że robot zrobi dziesięć ruchów do przodu, gdzie trzy kroki przed robotem znajduje się przeszkoda.

Z czasem można zwiększać poziom trudności i dawać bardziej wymagające przykłady, oraz umieszczać po kilka błędów w jednym kodzie.

Co istotne naukę diagnozowania błędów można prowadzić zarówno z wykorzystaniem komputerów, tabletów ale również na kartkach papieru w formie drukowanej.

6. Analiza możliwości wybranych robotów w nauce programowania.

Definicji robot używamy do określenia autonomicznie działających urządzeń, które odbierają informację z otoczenia za pomocą sensorów oraz wpływające na nie przy pomocy efektorów. Budową takich robotów zajmują się badacze sztucznej inteligencji celu modelowania zdolności poznawczych, sposobu myślenia ludzi. Mimo następującego postępu technologicznego, stworzenie robota, który by dorównywał sztucznej inteligencji, wciąż wydaje się bardzo odległy. Robotyka to właśnie dziedzina sztucznej inteligencji, zajmująca się konstruowaniem i programowaniem robotów. Robot to również ogólne pojęcie, którym określamy istniejące w rzeczywistości lub wyimaginowane maszyny i automaty przypominające wyglądem człowieka.

6.1 Poznanie przykładowych robotów.

PHOTON

PHOTON robot, który posiada: czujnik światła, czujnik dotyku, głośnik, mikrofon, podświetlenie oczu i uszu.

Autorzy aplikacji, umożliwili programowanie urządzenia na cztery sposoby.

Tryb DRAW- to w nim programujemy robota rysując ścieżkę na tablecie. Mamy możliwość również nakładania na trasę akcji specjalnych, np. wydawanie dźwięku, zmiana koloru, aktywacja czujnika. Tryb ten dedykowany jest najmłodszym osobom, w początkowej fazie zaawansowania.

Tryb BADGE – to w nim znajdują się specjalne ikony z infografikami np. strzałka do przodu, światło żółte. Aby przygotować program musimy umieścić symbole w odpowiedniej kolejności. Mamy możliwość wyboru funkcji jak również powtarzać wybrane sekwencje ruchu i akcji.

Tryb BLOCKS – to w nim znajdują się kolorowe bloki komend w języku polskim. Program układa się poprzez ułożenie bloczków w odpowiedniej kolejności. Tryb ten z racji tego, że jest bardziej zaawansowany daje możliwość tworzenia skomplikowanych programów.

Tryb CODE – stworzony z myślą o zaawansowanej grupie odbiorców. Zbliżony jest do tryby BLOCKS z uwagi na komendy, jednak są w języku angielskim, nie są już w różnych kolorach. Tryb ten jest najbardziej zbliżony do powszechnych języków programowania.

mBot – stworzony do nauki programowania

Robot mBot posiada wiele czujników: odległości, linii, koloru, dotyku, wilgotności, ruchu, temperatury, gazu, płomieni, obrotu i dźwięku.

Autorzy aplikacji, umożliwili programowanie urządzenia w programie mBlock, który jest bardzo intuicyjny, opiera się na Scratch. Z jego wykorzystaniem możemy nie tylko programować roboty, daje nam o wiele więcej możliwości takich jak: tworzenie prostych gier, programów oraz aplikacji. Nauka poprzez zabawę, motywuje uczniów do logicznego myślenia.

Ozobot- zabawka roku 2016!

Ozobot zabiera dzieci (w wieku od 5 lat) w niesamowitą przygodę rysowania, rozwiązywania problemów i pracy grupowej. Za pomocą kolorowych kodów (na kartce papieru lub tablecie) dzieci programują zadania, które wykonuje robot.

Ozobot daje możliwość płynnego przejścia od kodowania offline do kodowania przy pomocy monitorów komputerów/tabletów.

Internetowy panel sterowania OzoBlockly.pl oferuje pięć poziomów programowania, od osoby początkującej (nowicjusza) do mistrza. Dlatego znajdzie śród swoich odbiorców osoby początkujące oraz bardziej doświadczonych programistów. OzoBlockly.pl bazuje na języku programowania blockly, który jest podobny do Scratch jest językiem wizualnym (blokowym).

SKRIBOT

Ilustracja 6.1, autor: Małgorzata Chmielewska

Powstał w oparciu o metodę Desingn Thinking. Design Thinking – nowa metoda projektowania, która się zaczyna od pytania dlaczego… ?

Wyróżniamy 5 etapów:

1. empatyzacja- wczuć się w osobę, której problem chcemy rozwiązać. (np. szkoła, która chciałaby uczyć robotyki a nie ma sprzętu..)
2. definiowanie problemu – potrafimy odpowiedzieć, jaki jest problem np. brak sprzętu.
3. generowanie pomysłów – burza mózgów np. jak ma wyglądać robot.
4. prototypowanie – w momencie kiedy mamy już produkt a zaczynamy go realizować, możemy zrealizować więcej prototypów.
5. testowanie – mamy przygotowany model i go testujemy.

Robot posiada: czujniki odległości (ultradźwiękowe), czujniki światła/linii (możemy wykleić drogę z taśmy, po której będzie podążał), silniki, diody LED, chwytak.

Skribot działa na 3 płaszczyznach:

4. konstruowanie - uczniowie poprzez konstrukcję robota nie tylko poznają świat inżynierii ale przede wszystkim rozwijają zdolności manualne jak również uczą się organizacji pracy.
5. elektronika - samodzielne podłączanie elementów pozwala poznać podstawy elektroniki jak również zdefiniować działanie mikrokontrolerów.
6. programowanie - podniesienie wiedzy i umiejętności programistycznych, niezależnie od poziomu zaawansowania, dają możliwość odkrywać i stosować w praktyce zróżnicowane technologie.

Dedykowane dwie aplikacje do programowania bloczkowego sprawiają, że staje się to idealne narzędzie dla osób stawiających pierwsze kroki w dziedzinie programowania. Dostępność narzędzi zarówno na komputerze, jak i na urządzeniach mobilnych sprawia, że SkriBot może być używany niezależnie od tego, w jakie urządzenia jest wyposażona jest pracownia.

7. Wykorzystanie dostępnych narzędzi do tworzenia aplikacji mobilnych.

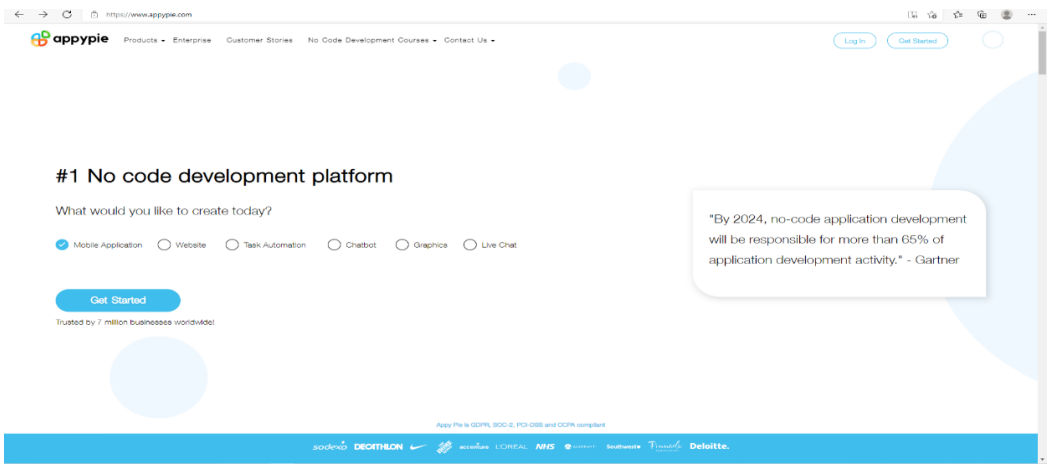
Aplikacja mobilna (ang. *mobile software / mobile application*) – ogólna nazwa dla oprogramowania działającego na urządzeniach przenośnych, takich jak telefony komórkowe, smartfony, czy tablety. Rozwój aplikacji mobilnych w ostatnim czasie odnotował znaczący wzrost liczby użytkowników. Liczy się około 5 milionów aplikacji na platformy Android, IOS, Windows na całym świecie.

Na rynku jest wiele bezpłatnych narzędzi do tworzenia aplikacji mobilnych. Konieczność posługiwania się kodami od góry do dołu dla każdej aplikacji jest stresująca i wymaga szeregu umiejętności. Korzystanie z dostępnych bezpłatnych narzędzi, pozwala na prostsze i bardziej dostępne rozwiązanie

7.1 Przykładowe narzędzia do tworzenia aplikacji mobilnych.

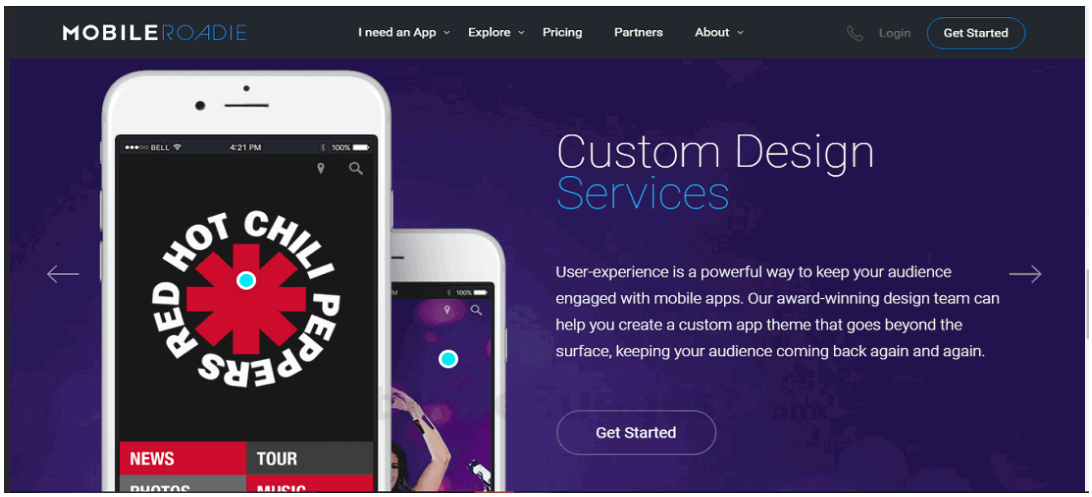
Bezpłatne narzędzia do tworzenia aplikacji mobilnych mają na celu pomoc w przygotowaniu różnych rodzajów aplikacji mobilnych.

Appy Pie – daje możliwość stworzenia aplikacji mobilną w mniej niż 30 minut. Mamy możliwość tworzyć spośród setek szablonów w wyborze branży. Możemy dodawać dowolną ilość stron lub funkcji.



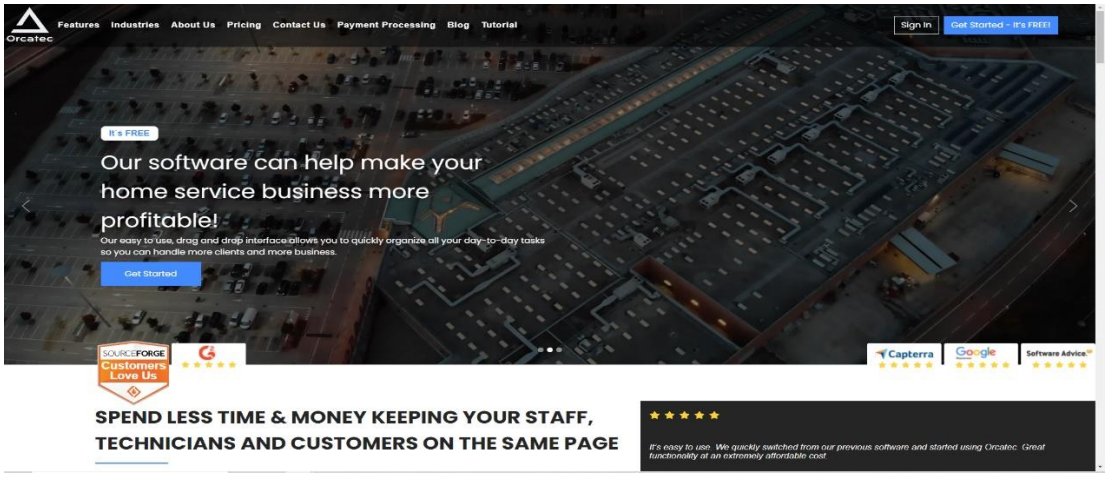
Ilustracja 7.1, autor: Małgorzata Chmielewska

Mobile Roadie – dzięki temu narzędziu powstało tysiące aplikacji, które powstały przez użytkowników na całym świecie przy użyciu jego platformy. Platforma daje możliwość pobrania aplikacji z ich strony internetowej za darmo.



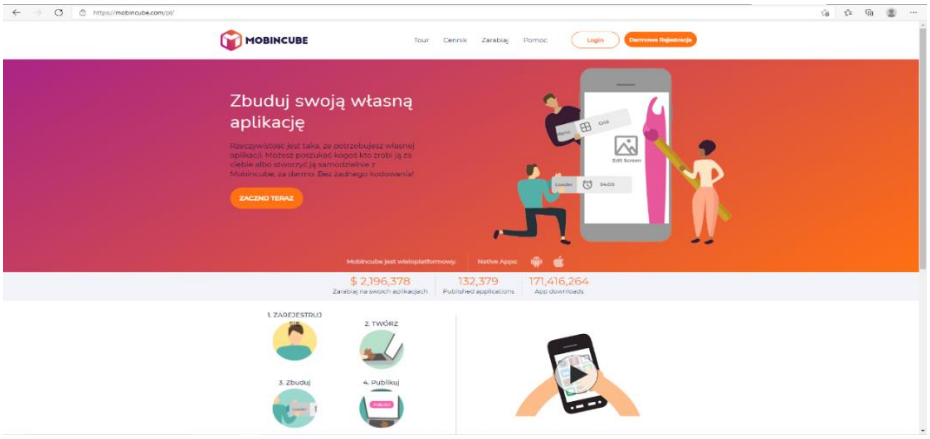
Ilustracja 7.2, autor: Małgorzata Chmielewska

OrcaTec ma również narzędzie online dostępne (bezpłatnie) o nazwie **GoMobi Create**. Jest jednak mały minus… Nie mamy możliwości projektowania wszystkich funkcji, które dają nam możliwość dwie powyższe platformy.



Ilustracja 7.3, autor: Małgorzata Chmielewska

Radminfo jest kolejną stroną internetową, która oferuje darmowe narzędzie do tworzenia aplikacji, nazywa się **Mobincube**. Jest to platforma all-in-one, na której mogą Państwo stworzyć swoją aplikację i przesłać ją do sieci lub bezpośrednio do sklepów Apple/Android!

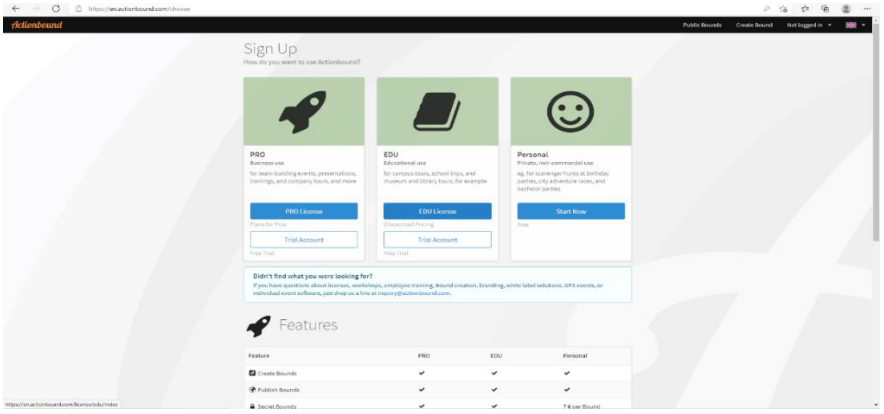


Ilustracja 7.4, autor: Małgorzata Chmielewska

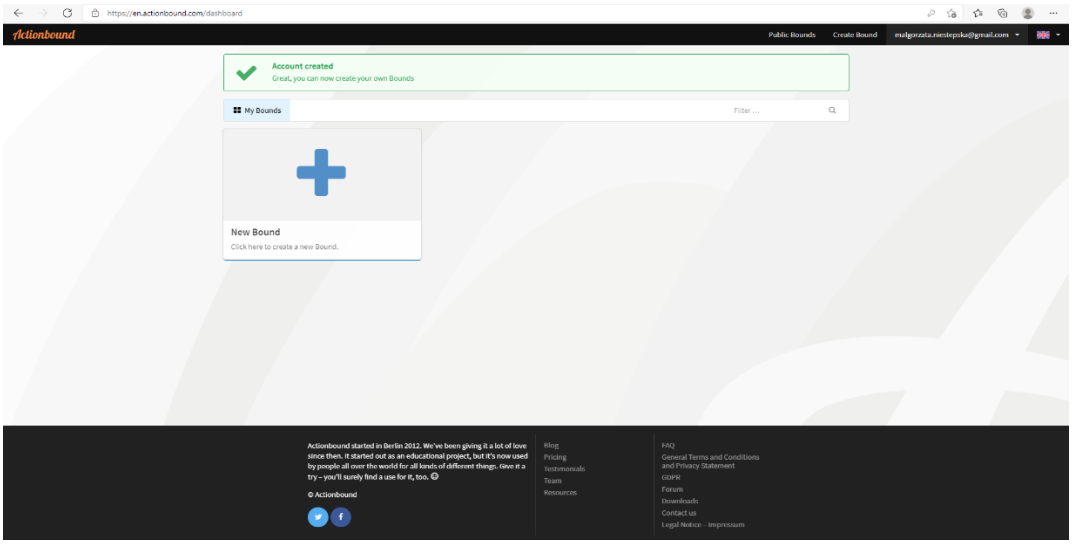
7.2 Actionbound – stwórz swoją aplikację!

Ilustracja 7.5, autor: Małgorzata Chmielewska

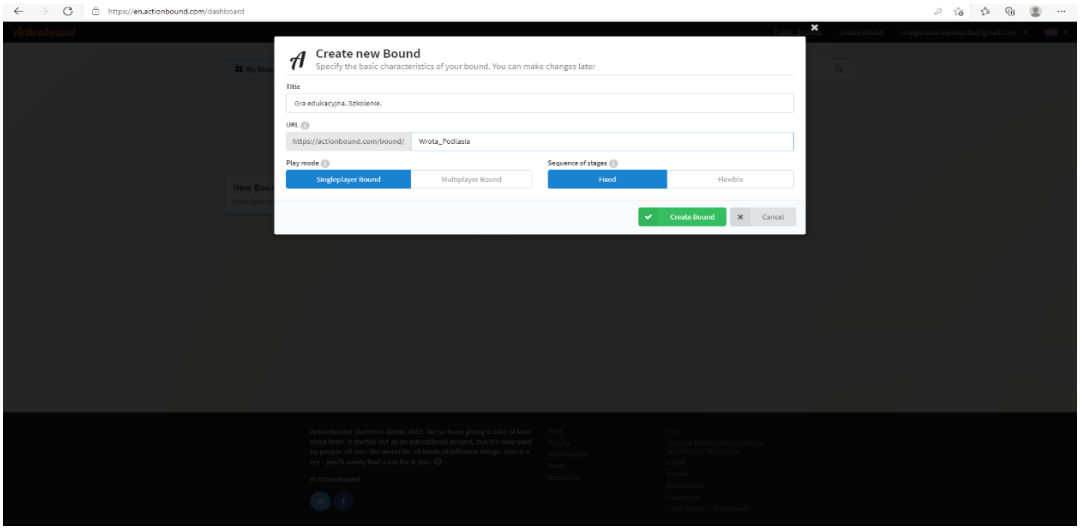
Darmowa aplikacja pod adresem: http://en.actionbound.com.



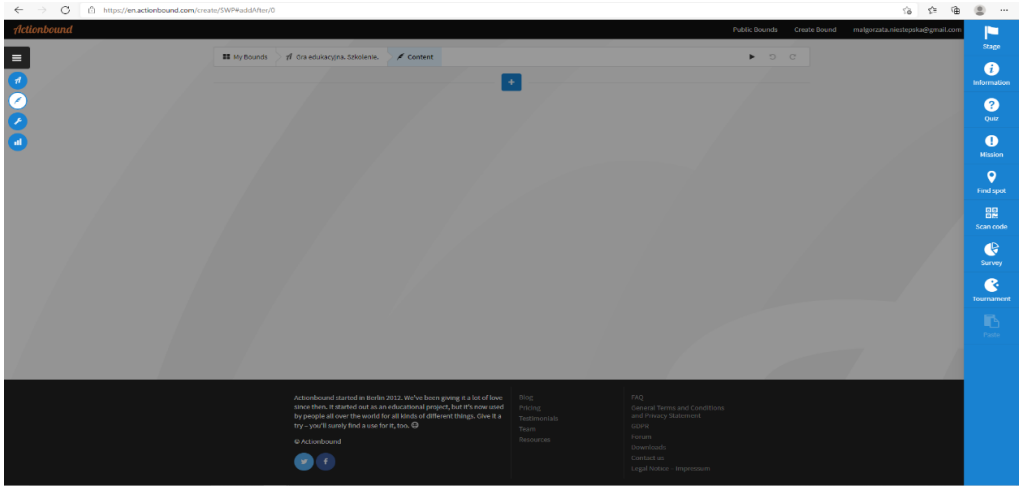
Ilustracja 7.6, autor: Małgorzata Chmielewska



Ilustracja 7.7, autor: Małgorzata Chmielewska



Ilustracja 7.8, autor: Małgorzata Chmielewska



Ilustracja 7.9, autor: Małgorzata Chmielewska

Jeżeli chcemy, aby uczestnicy gry poruszali się po terenie z wykorzystaniem mapy, klikamy w zakładkę „**STAGE”.**

Po dokonaniu wyboru, określamy nazwę miejscowości, w której rozgrywać się będzie gra. Na ekranie aplikacji zobaczymy plan terenu wraz z nazwami ulic i najważniejszych obiektów. Klikając w wybraną lokację nadajemy miejsce akcji naszej gry.

Jeżeli chcemy, aby uczniowie dotarli do konkretnej lokacji na planie, klikamy w funkcję „**FIND SPOT**”.

Wykorzystując ww. funkcję, stworzymy zadanie, które zostanie pozytywnie zaliczone wtedy, gdy GPS smartfona jednego z uczniów potwierdzi, że odnalazł się w miejscu, które przyporządkowaliśmy na planie.

Funkcja o nazwie „**QUIZ”** umożliwia nam przygotowanie grupom punktowanych pytań.

Tworząc pytania możemy dołączyć dowolne zdjęcie, obrazek, film lub plik dźwiękowy. Przydzielamy punkty możliwe do uzyskania, określamy limit czasu na udzielenie odpowiedzi oraz o ilość możliwości na udzielenie poprawnej odpowiedzi.

Funkcja „**MISSION”** pozwala nam dodanie zadania do wykonania przez uczestnika/grupę, np. podanie odpowiedzi, zrobienie i załączenie zdjęcia, filmiku, nagrania dźwiękowego.

Aplikacja nie sprawdzi wykonania poprawnie zadania – autor aplikacji Actionboud, zobaczy u siebie na profilu odpowiedzi uczestników.

Dla zwolenników kodów QR przygotowano funkcję „**SCAN CODE**”.

Jeżeli w przygotowanej przez nas grze znajdą się kody QR, uczniowie będą musieli odnaleźć przez nas ukryte kody oraz je zeskanować.

Aby móc zagrać w tak przygotowaną grę terenową, potrzebny nam będzie smartfon z zainstalowaną aplikacją Actionbound. Możemy ją pobrać z Google Play oraz w App Store.

Po otworzeniu aplikacji mobilnej należy: zeskanować kod QR, do gry. Kod ten znajduje się w profilu autora. Mamy możliwość pobrać go w formie pliku pdf oraz wydrukować. Po zeskanowaniu kodu uczestnik/grupa podaje swoją nazwę następnie imiona graczy. Autor aplikacji w czasie rzeczywistym, ma prawo obserwować postępy grup.

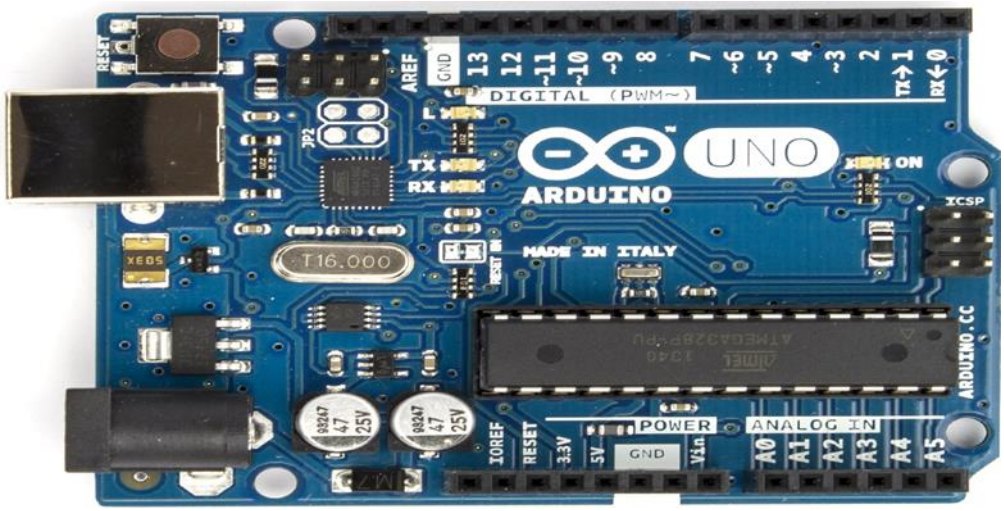
Zeskanuj kod ! I rozpocznij swoją przygodę z Actionbound !

8. Podstawy elektroniki cyfrowej, robotyki i sterowania.

8.1 Elektronika cyfrowa

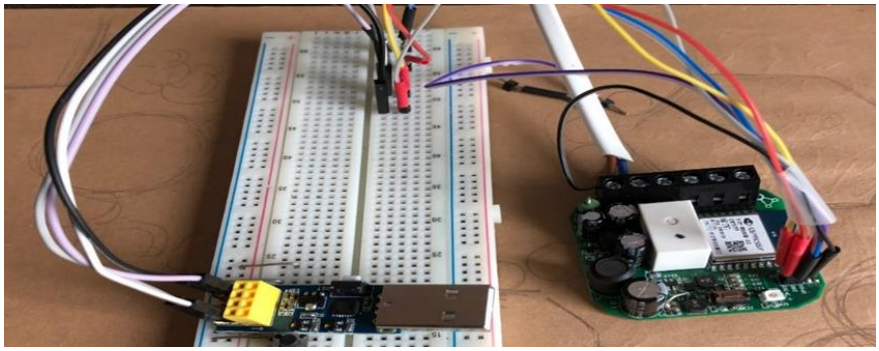
Elektronika cyfrowa – dziedzina elektroniki zajmująca się układami cyfrowymi, sygnałami cyfrowymi i cyfrowym przetwarzaniem sygnałów

Jednym z przykładów elektroniki cyfrowej jest **Arduino**: to platforma, która świetnie ułatwia tworzenie i uruchamianie konstrukcji o różnym stopniu zaawansowania.

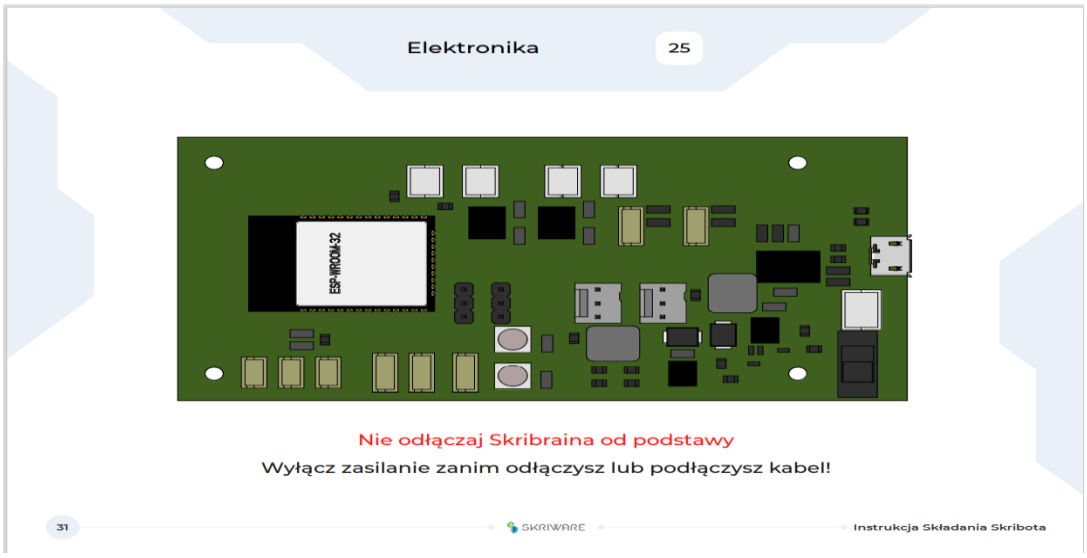


Ilustracja 8.1, autor: Nieznany autor, licencja: CC BY-SA

Dlaczego Arduino dla ucznia? Nie musimy nic lutować. Do pracy wystarczy płytka Arduino + przewody + płytka stykowa. Prototyp projektu tworzy się i uruchamia bardzo szybko: tym samym efekty widoczne są błyskawicznie. Arduino to znakomita platforma dla osób, które rozpoczynają pracę z elektroniką". Pod nazwą ARDUINO kryje się: system na który składa się płytka z mikrokontrolerem oraz ustandaryzowane środowisko do programowania Arduino IDE.



Ilustracja 8.2, Nieznany autor, licencja: CC BY-SA



Ilustracja 8.3, autor: Nieznany autor, licencja: CC BY-SA-NC

8.2 Robotyka i sterowanie.

Robotyka dla dzieci – to znakomity przykład nauki poprzez zabawę. To nauka, która bez wątpienia zaprocentuje na przyszłość.

Dziecko z wykorzystaniem różnych zestawów do robotyki (SKRIBOT, LEGO WE DO, LEGO MINSTORMS), swoją przygodę rozpoczyna od złożenia robota następnie wprowadza go w ruch. Aby wprowadzić robota w ruch, należy go zaprogramować, z wykorzystaniem dedykowanych do tego celu aplikacji oraz paneli sterujących.

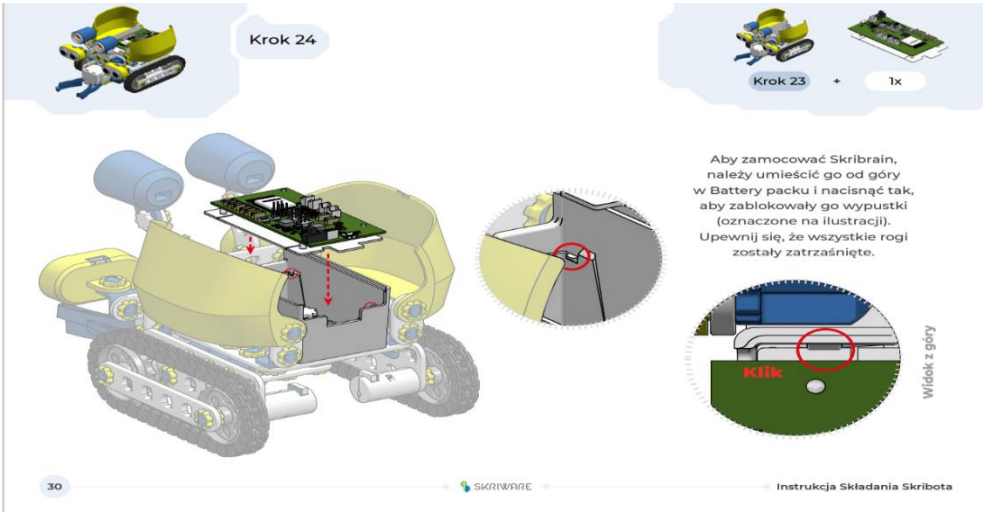
Na zajęciach z wykorzystaniem robotów, dziecko nabywa takie umiejętności jak: rozwój strategicznego oraz analitycznego myślenia. Bez wątpienia uczy się dążenia do celu i konsekwencji w działaniu.

Robotyka – dziedzina naukowa, która swój początek ma w latach 60 tych. Zaczęło się od robotów przemysłowych, które posiadały wielkie ramiona.

Wyróżniamy 4 kategorie robotów

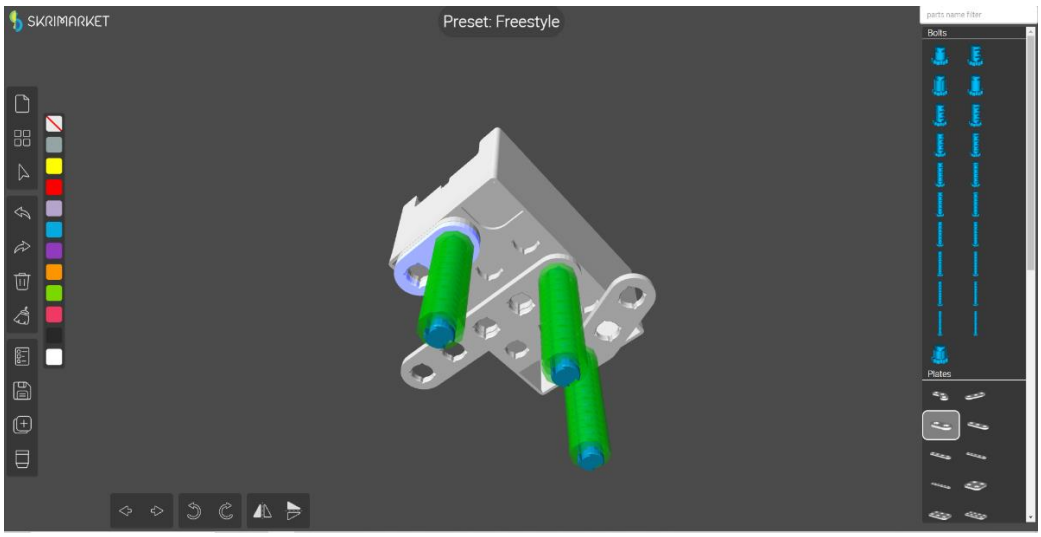
- naukowe – najbardziej szlachetne (łaziki marsjańskie).
- medyczne (cyberknife), - „cybernóż”, lekarze na końcu świata są w stanie zoperować pacjenta.

- edukacyjne do nauki programowania/kodowania dla dzieci np. Skribot, Lego Mindstorms.
- codzienne– ułatwiające prace życia codziennego np. odkurzacz iRoomba.



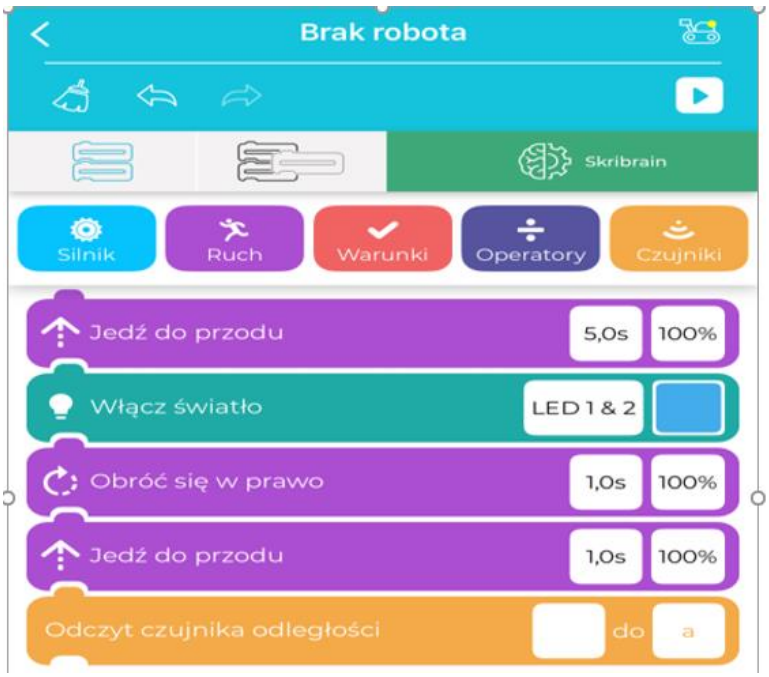
Ilustracja 8.3, autor: Małgorzata Chmielewska

Program CREATOR do tworzenia modeli 3D. Kreator wirtualny to proste w obsłudze narzędzie do przygotowania dowolnego prototypu z wykorzystaniem elementów konstrukcyjnych Skiware. Możliwość stworzenia własnej robotycznej konstrukcji. Zajęcia z Creatorem pobudzają wyobraźnię oraz rozwijają podstawy inżynierii. Elementy konstrukcyjne w Creatorze odzwierciedlją klocki w SkriBocie. Narzędzie można sparować z drukarką 3D.



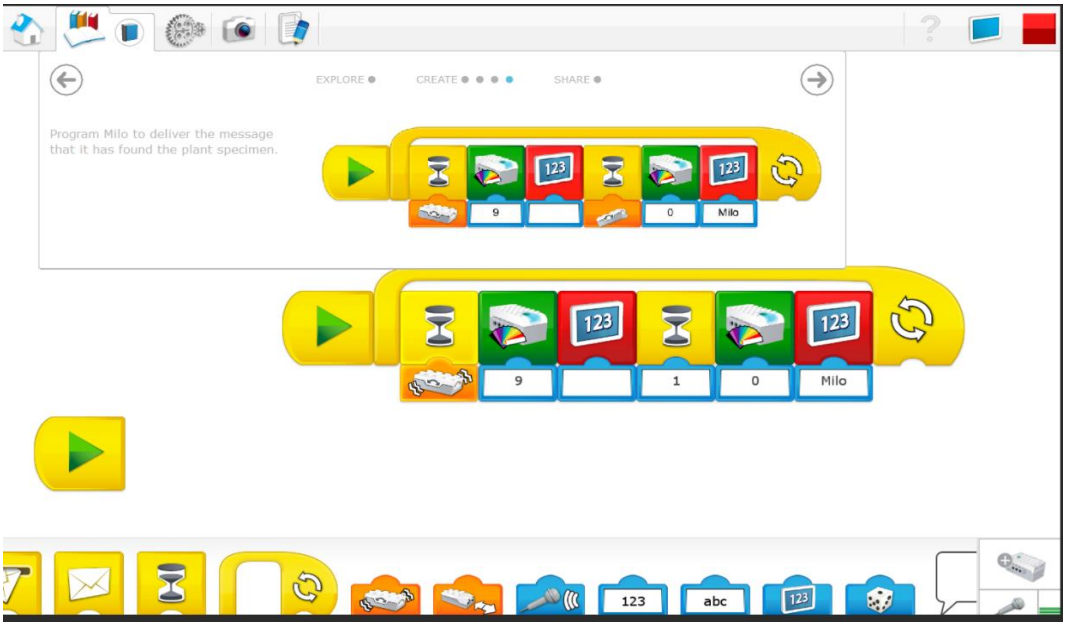
Ilustracja 8.4, autor: Małgorzata Chmielewska

Panel sterowania zestawem SKRIBOT jest: intuicyjny, z bezpłatną aplikacją, łącze Bluetooth, w polskiej wersji językowej, możliwość programowania na różnych płaszczyznach zaawansowania, dedykowany na urządzenia mobilne/komputer.



Ilustracja 8.5, autor: Małgorzata Chmielewska

Lego We Do – przykładowe konstrukcje modeli znajdują się w aplikacji oraz na stronie: Speed | WeDo 2.0 Lesson Plan | LEGO® Education.



Ilustracja 8.6, autor: Małgorzata Chmielewska

Lego Mindstorms - przykładowe konstrukcje modeli znajdują się w aplikacji oraz na stronie LEGO.

Aplikacja Robot Commander. Robot Commander to oficjalna aplikacja LEGO® MINDSTORMS® do wydawania poleceń. Łączy się z inteligentnym elementem EV3 za pomocą Bluetooth. Ta prosta aplikacja nie wymaga łączenia się robota z komputerem.

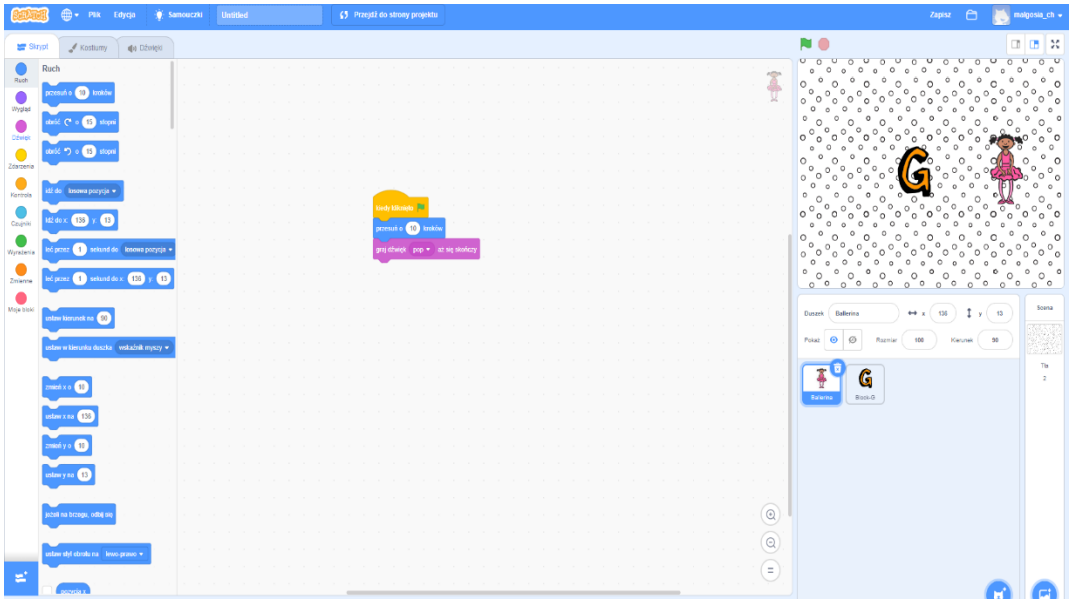
Aplikacja LEGO® MINDSTORMS® EV3 Home. Należy zainstalować oprogramowanie następnie należy nawiązać połączenie z robotem. W programie możemy wykonać misje programistyczne z robotem Hero. Aplikacja dedykowana na tablety, komputery. Pozwala rozwinąć umiejętności z robotyki oraz programowania dzięki dedykowanemu panelu sterowania.

9. Tworzenie programów sterujących, które zmieniają maszyny lub pojazdy w roboty wchodzące w interakcję z otoczeniem.

Robotyką możemy sterować na wiele sposób, w tym z pomocą sterowania bezprzewodowego, ręcznego, półautonomicznego (połączenie sterowania bezprzewodowego ze sterowaniem w pełni autonomicznym) oraz w pełni autonomicznego (wykorzystanie sztucznej inteligencji lub opcje ręcznego sterowania).

9.1 Scratch

Scratch to wizualny język programowania. Został stworzony przez pomysłodawcę serii zabawek Lego Mindstorms - Mitchela Resnicka. Edukacyjny język stworzony z myślą o nauce programowania dla dzieci i młodzieży. Służy do tworzenia programów oraz ich uruchamiania. Kod tworzy się poprzez przeciąganie puzzli, następnie zostaje przypisany obiektowi. Obiekty mogą reagować na zdarzenia zewnętrzne. Postacie możemy importować z zewnątrz lub wybierać z zasobnika.



Ilustracja 9.1, autor: Małgorzata Chmielewska

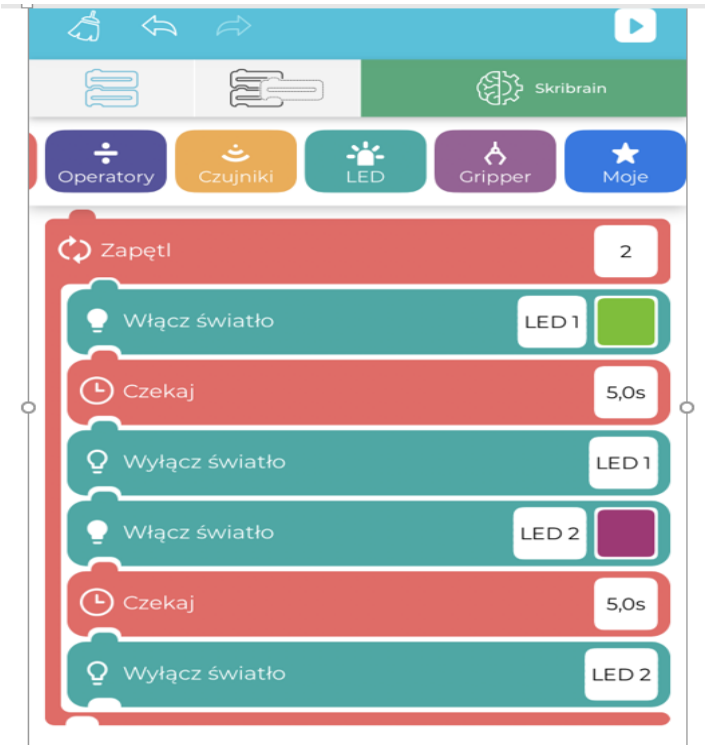
Praca indywidualna, metodą warsztową na platformie Scratch – tworzenie skryptów. https://scratch.mit.edu/

9.2 SkriApp – Skribot

Co to jest warunek? **Warunek** – wprowadzająca informacja co komputer musi wiedzieć, by podjąć decyzję. Wybór określonych decyzji nazywamy warunkiem i korzystamy z nich na co dzień np. kiedy pada deszcz zakładamy kalosze.

Co to są pętle? **Pętle** – często powtarzający się układ np. układ taneczny, urodziny, święta. Komputer można powtarzać pętle niezliczoną ilość razy i nigdy mu się nie znudzi.

Aplikacja SkriApp daje nam możliwość różnorodnie programować naszego robota, posiada wbudowany panel: silnik, ruch, warunki, operatory (typowe operacje zmiennych), czujniki, led oraz Gripper (chwytak).



Ilustracja 9.2, autor: Małgorzata Chmielewska

Programów, które umożliwiają sterowanie robotami jest bardzo wiele. Każdy wyprodukowany robot edukacyjny na wyposażeniu posiada platformę do sterowania. Każdy ma swoją oddzielną. Zasada pracy na platformach dla robotów edukacyjnych w większości jest podobna. Różnica polega na językach programowania, które zostały wykorzystane podczas ich tworzenia.