MoboLab – roboty i tablety w Twojej szkole
Obszar II. „Stwórz własnego robota”
Scenariusze lekcji i zajęć pozalekcyjnych

SCENARIUSZ 25. JOYSTICK

scenariusz zajęć pozalekcyjnych

autor: Michał Kłosiński

redakcja: Agnieszka Koszowska

SŁOWA KLUCZOWE:

Arduino, programowanie, elektronika, joystick

KRÓTKI OPIS ZAJĘĆ:

Podczas zajęć uczniowie i uczennice uczą się, jak stworzyć układ elektroniczny za pomocą Arduino oraz **joysticka**. Przygotowują układ i tworzą program pozwalający na odczyt wartości X i Y z joysticka.

WIEDZA I UMIEJĘTNOŚCI ZDOBYTE PRZEZ UCZNIA / UCZENNICĘ:

- wie, czym są mikrokontrolery i do czego służą,
- zna pojęcia: mikrokontroler, skrypt, program, algorytm, sterowanie, warunek, pętla,
- zna projekt Arduino, wie, czym jest platforma Arduino, z jakich części się składa,
- potrafi w podstawowym stopniu samodzielnie obsługiwać Arduino (podłączyć płytkę do komputera, wgrać prosty program),
- wie, co to jest czujnik joystick,
- potrafi podłączyć joystick do Arduino,
- zna podstawowe elementy interfejsu środowiska programistycznego Arduino IDE i podstawowe komendy języka Arduino IDE: pinMode(), digitalWrite(), delay(),
- rozumie zasadę działania funkcji digitalWrite() i potrafi wykorzystać ją w praktyce,
- zna podstawowe elementy języka **Scratch**, potrafi stworzyć prosty skrypt w tym języku.

GRUPA DOCELOWA:

Starsze klasy szkoły podstawowej (VII-) i klasy gimnazjalne (po dostosowaniu: możliwość realizacji w młodszych klasach: I-III i IV-VI szkoły podstawowej). W młodszych klasach – możliwość wykorzystania programu mBlock (po przejściu scenariusza nr 18. *Programowanie Arduino z wykorzystaniem programu mBlock*) lub Scratch for Arduino (po przejściu scenariusza nr 1. *Wprowadzenie do Arduino*).

LICZBA UCZNIÓW/UCZENNIC W GRUPIE:

Liczba optymalna: 12, liczba maksymalna: 16

CZAS TRWANIA ZAJĘĆ:

90 min (lub 2 x 45 minut)

STOPIEŃ TRUDNOŚCI/SKOMPLIKOWANIA (w skali od 1 do 5 dla obszaru II. „Stwórz własnego robota”):

1

POTRZEBNY SPRZĘT I OPROGRAMOWANIE:

- komputer (przenośny lub stacjonarny),
- program Arduino IDE (do pobrania ze strony: http://www.arduino.org/downloads),
- (opcjonalnie) program mBlock (do pobrania ze strony: http://www.mblock.cc/download/) lub Scratch for Arduino (do pobrania ze strony: http://s4a.cat/),
- płytka Arduino UNO i kabel USB A-B (dla każdego uczestnika lub dla pary uczestników),
- płytka stykowa,
- oporniki 220 omów,
- joystick analogowy,
- przewody połączeniowe,
- diody LED w różnych kolorach,
- projektor i laptop (w części teoretycznej).

CO NALEŻY PRZYGOTOWAĆ PRZED ZAJĘCIAMI:

- zainstalować program Arduino IDE,
- (opcjonalnie): zainstalować program **mBlock** lub **Scratch for Arduino**,
- sprawdzić, czy wszystkie komputery wykrywają podłączone Arduino,
- przeczytać dokładnie scenariusz,

- zapoznać się z materiałami dodatkowymi (w części „Pigułka wiedzy i inspiracji”),
- wykonać samodzielnie zadania zawarte w scenariuszu,
- przy każdym stanowisku komputerowym rozłożyć elementy zestawu Arduino, które będą wykorzystywane na tych zajęciach,
- dopasować stopień trudności zadania do potrzeb i możliwości klasy, dla której organizowane są zajęcia według wskazówek zawartych w scenariuszu.

KOMPETENCJE OSOBY PROWADZĄCEJ:

- wie, czym jest projekt Arduino, zna podstawowe informacje o projekcie,
- potrafi przynajmniej w stopniu podstawowym obsługiwać Arduino,
- zna podstawowe pojęcia z zakresu elektroniki,
- zna podstawowe pojęcia programistyczne,
- wie, dlaczego warto uczyć się programowania i jakie korzyści daje posiadanie umiejętności programistycznych,
- potrafi zachęcić do nauki programowania zarówno chłopców, jak i dziewczynki.

PRZEBIEG ZAJĘĆ:

Podłączenie Arduino, uruchomienie programu Arduino IDE i przypomnienie podstawowych informacji – ok. 15 minut

Uwaga! Informacje o tym, jak podłączyć Arduino, uruchomić program Arduino IDE i Scratch for Arduino, a także podstawowe informacje niezbędne przy rozpoczynaniu pracy z Arduino zawierają scenariusze 1 i 2. Tę część zajęć warto powtarzać za każdym razem w takim zakresie, jaki jest potrzebny, do czasu aż podstawowy materiał zostanie utrwalony.

Wstęp – co czego służy joystick? – 15 minut

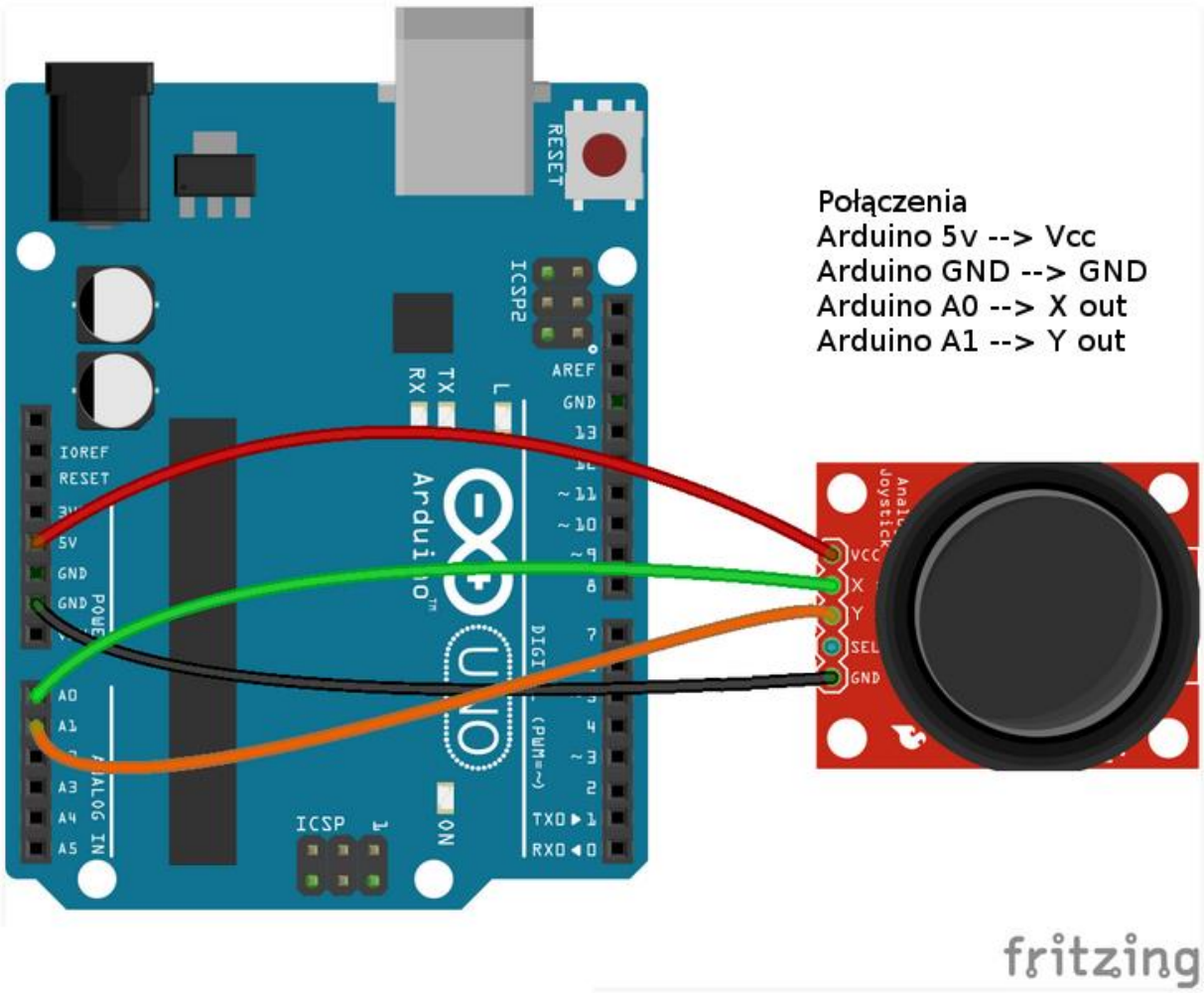
Joystick (wym. dżojstik) to urządzenie służące do sterowania ruchem, np. różnych obiektów na ekranie komputera, ale też zewnętrznych urządzeń (takich jak roboty). Joysticki są często wykorzystywane w grach, gdzie za ich pomocą steeruje się ruchem albo zachowaniem postaci. Joysticka używa się też w lotnictwie, np. za pomocą drążka kierowało się pierwszymi samolotami.

Możemy zadać uczniom pytania, takie jak:

- *czy korzystaliście kiedyś z joysticka?*
- *do czego służył używany przez Was joystick?*
- *w jaki sposób nim poruszaliście?*

Podłączamy joystick do Arduino – 15 minut

Montujemy układ z joystickiem analogowym wchodzącym w skład zestawu Arduino. Możemy skorzystać z poniższego schematu:



W tym miejscu możliwy jest podział zajęć na dwie części (kolejna część scenariusza będzie realizowana na następnych zajęciach).

Przypomnienie materiału, odtworzenie układu – 10 minut

Rozpoczynamy od krótkiego przypomnienia materiału z poprzedniej części zajęć i odtworzenia układu zbudowanego na poprzednich zajęciach.

Programowanie układu – 20 minut

Tworzymy nowy projekt. Z menu **File** wybieramy **New**, albo klikamy w przycisk New w interfejsie. Zobaczymy nowe okno z następującym tekstem:

```
void setup() {
  // put your setup code here, to run once:
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

Jest to podstawowa struktura programu Arduino. Zgodnie z opisem w języku angielskim, część „setup" jest wykonywana tylko raz, na początku uruchomienia programu, po czym wykonywana jest część „loop" na okrągło, czyli w tzw. pętli. Jeżeli uczestnicy brali udział w warsztatach ze Scratcha, termin pętli powinien być im znany. Tekst który następuje po **//** to komentarze – możemy po tym znaku wpisać co chcemy. Zazwyczaj komentarze informują o tym, co wykonuje dana część programu. Komentarz musi być w tej samej linii co znak **//**.

Piszemy program, który będzie nam wyświetlał informację w Serial Monitor, jaka wartość pochodzi od modułu Joystick. Będą to dwie współrzędne położenia gałki Joystick X i Y.

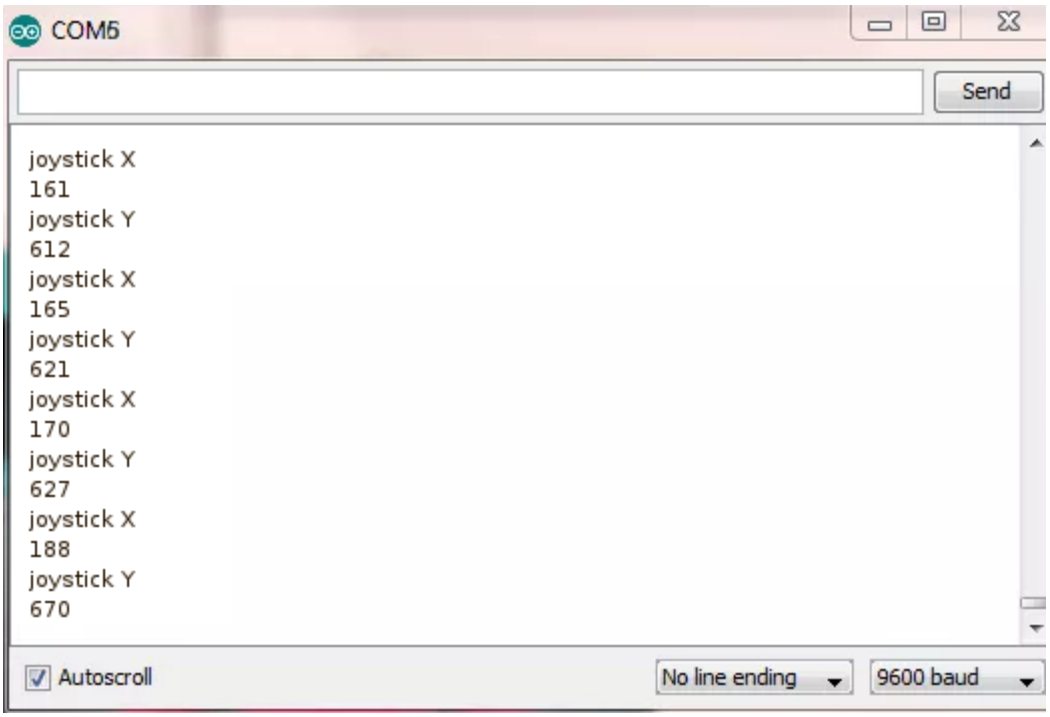
Robimy to wpisując w okno następujący kod (wyjaśnienia znajdują się w komentarzach).

```
// mówimy Arduino o stworzonych zmiennych które będą przechowywać wartości z joysticka
int JoystickX = 0;  // wartość która będzie pochodzić z analog pin A0
```

```
int JoystickY = 0;  // wartość która będzie pochodzić z analog pin A1

void setup() {
  // mówimy Arduino że będzie się komunikowało z komputerem za pomocą przewodu USB
  Serial.begin(9600);
}
void loop() {
JoystickX = analogRead(A0); // Arduino pobiera wartości z pinu analogowego A0
JoystickY = analogRead(A1); // Arduino pobiera wartości z pinu analogowego A1

    Serial.println("joystick X ");
    Serial.println("JoystickX"); // wyświetlamy wartości na serial monitorze
    Serial.println("joystick Y ");
    Serial.println("JoystickY"); // wyświetlamy wartości na serial monitorze


   // czekamy 100 milisekund, czyli jedną dziesiątą sekundy
  delay (100);
}
```

Następnie klikamy przycisk **Upload**. Jeżeli wystąpiły błędy, należy je poprawić. Program powinien być przepisany bardzo starannie i bez błędów.

Klikamy w przycisk **Serial Monitor** lub wybieramy z menu Tools > Serial Monitor. Powinniśmy zobaczyć okno w którym co 1/10 sekundy pojawia się napis „joystick X" i jego wartość oraz „joystick Y” i jego wartość – tak, jak na poniższym rysunku:



Skąd wiedzieliśmy, co wpisać, by uzyskać żądany efekt? Otóż każdy język programowania zawiera słowa o konkretnym znaczeniu. Lista słów dostępnych w Arduino jest dostępna pod poniższym adresem:
https://www.arduino.cc/en/Reference/HomePage

O wiele wygodniej na początek korzystać z przykładów programów dołączonych do Arduino (**File>Examples**). Języka programowania uczymy się w bardzo podobny sposób jak języka obcego – musimy poznać jego słowa, składnię, gramatykę, ortografię.

I tak, bardzo upraszczając: **„Serial”** jest to **funkcja**. Ma ona swoje podfunkcje, które potrafią wykonywać różne operacje, np. **„println” drukuje linię**, a **„begin” rozpoczyna komunikację z komputerem**. Te podfunkcje mają swoje **parametry**, które wpisujemy w nawiasie. „println” musi wiedzeć, co konkretnie ma wydrukować („hello world” czy wartość zmiennej „JoystickX”). „begin” musi wiedzieć, z jaką częstotliwością ma rozmawiać z komputerem (w tym wypadku - 9600 sygnałów na sekundę).

Pętla programu działa bardzo szybko. Na przykład procesor płytki Arduino Mega ma częstotliwość pracy 16 MHz. **Hz**, czyli Hertz, to jednostka **częstotliwości** – określa jeden **cykl** (czyli jedno zdarzenie) na sekundę. *MHz* to MegaHertz – czyli jeden

milion Hertzów. Tak więc przykładowy moduł Arduino Mega może pracować z prędkością 16 milionów zdarzeń na sekundę.

Zadania dla uczniów – 15 minut

Zadanie 1

Zmodyfikuj wiadomość, którą Arduino wysyła do komputera i załaduj kod ponownie na płytkę:

- gdy gałka joysticka jest skierowana ku górze, to zamiast wartości liczbowej w serial monitorze powinien pojawić się napis „góra”,
- gdy gałka joysticka jest skierowana ku dołowi, to zamiast wartości liczbowej w serial monitorze powinien pojawić się napis „dol”,
- gdy gałka joysticka jest skierowana w prawo, to zamiast wartości liczbowej w serial monitorze powinien pojawić się napis „prawo”,
- gdy gałka joysticka jest skierowana w lewo, to zamiast wartości liczbowej w serial monitorze powinien pojawić się napis „lewo”.

Zadanie 2

Należy dodać do układu 4 diody. Nowy układ ma działać w ten sposób, że:

- gdy gałka joysticka jest skierowana ku górze, to zamiast wartości liczbowej w serial monitorze powinien pojawić się napis „góra”, zapala się dioda nr 1,
- gdy gałka joysticka jest skierowana ku dołowi, zapala się dioda nr 2,
- gdy gałka joysticka jest skierowana w prawo, zapala się dioda nr 3,
- gdy gałka joysticka jest skierowana w lewo, zapala się dioda nr 4.

MOŻLIWE MODYFIKACJE DLA MŁODSZYCH KLAS:

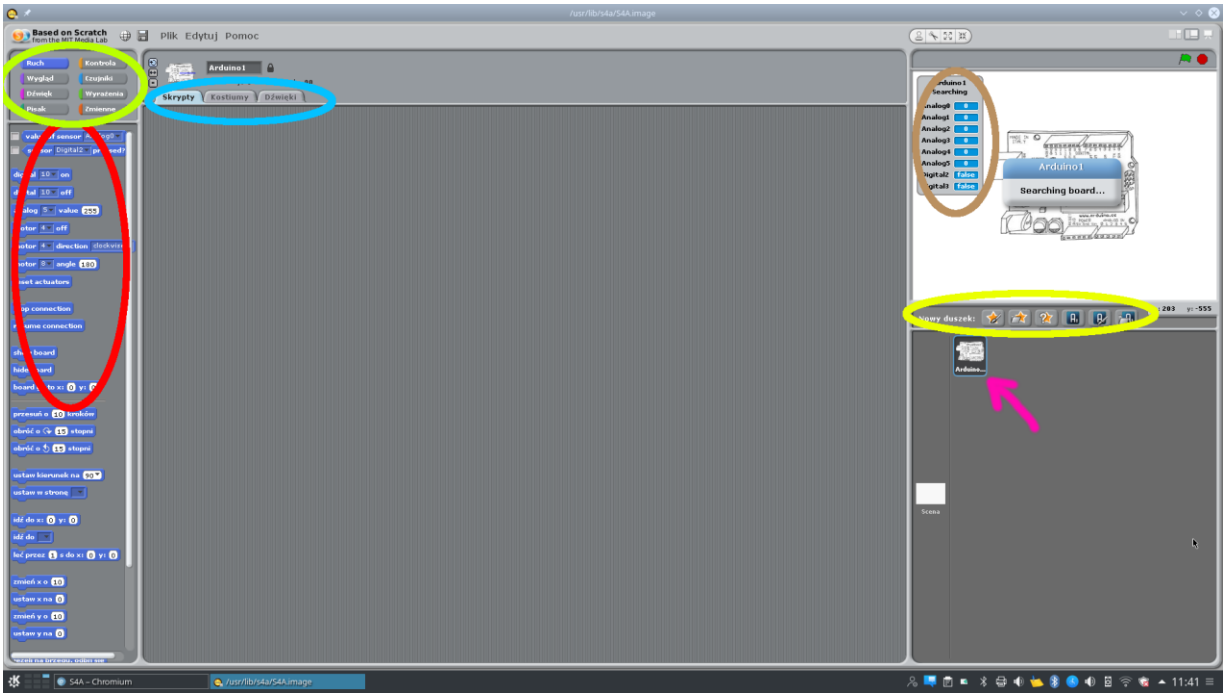
Pracując z uczniami w młodszych klasach można wykorzystać zamiast Arduino IDE program S4A (Arduino for Scratch). W przypadku zajęć z młodszymi dziećmi warto zwrócić uwagę na ewentualne problemy z dokładnym podłączaniem przewodów.

Wprowadzenie do programu S4A – Scratch for Arduino (na przykładzie z modułem joystick)

Program Scratch for Arduino został stworzony po to, by można było łatwiej rozpocząć swoją przygodę z programowaniem. Program możemy pobrać ze strony: http://s4a.cat/.

Scratch for Arduino to modyfikacja oprogramowania Scratch (języka i środowiska programistycznego), przygotowana z myślą o osobach, które chcą w prosty sposób uczyć się podstaw programowania za pomocą Arduino i projektowania własnych urządzeń. Program jest oparty na środowsku Scratch, dlatego jego wygląd i funkcjonalności są podobne.

Na poniższym obrazku zostały zamieszczone kolorowe zaznaczenia, po to by precyzyjnie wyjaśnić podobieństwa i różnice.



Zieloną elipsą zostały zaznaczone kategorie bloków służących do budowania skryptów. Są one takie same jak w języku Scratch, z tą różnicą, że w kategorii „Ruch” jest dostępnych więcej bloków.

Bloki znajdujące się w czerwonej elipsie są blokami dedykowanymi dla Arduino, ale nimi zajmiemy się później. Niebieska elipsa oznacza zakładki aktywnego duszka, czyli skrypty, kostiumy i dźwięki. Żółta, natomiast, pokazuje opcje dodawania nowego duszka oraz nowego duszka Arduino.

Po prawej stronie na białej scenie znajduje się rysunek Arduino oraz komunikat „Arduino1 Searching board…”. Oznacza on, iż program S4A nie nawiązał połączenia z płytką. Aby nawiązać połączenie z Arduino, musimy wgrać na nie odpowiedni program, który będzie działał w sposób ciagły. Jego zadaniem jest tłumaczenie skryptów w języku Scratch (czyli bloków, które ułożyliśmy) na język zrozumiały dla płytki Arduino (czyli na odpowiedni kod). Taki program możemy pobrać z strony twórców programu S4A lub z innych miejsc w sieci – jest to program Open Source, dostępny na otwartych licencjach. Powstało też wiele modyfikacji programu, z których również można legalnie korzystać.

Tutaj można przeczytać o licencji MIT http://opensource.org/licenses/MIT.

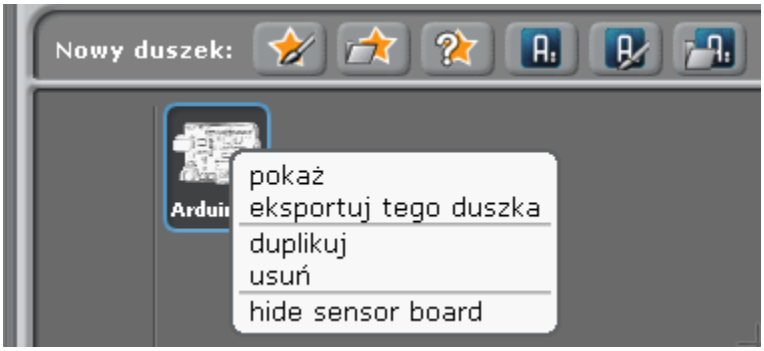
Program do pobrania ze strony twórców Scratch for Arduino:
http://vps34736.ovh.net/S4A/S4AFirmware16.ino

Program napisany przez blogera profesora Garcia, który uwzględnia działanie czujnika odległości
https://www.dropbox.com/s/czuegm2u5z22zvd/S4A_firmware_Profe_Garcia.txt?dl=0

Jest to zaawansowany program, dlatego nie będziemy go analizować, a ograniczymy się do wgrania go na płytkę za pomocą Arduino, tak samo jak wygrywaliśmy np. program Blink.

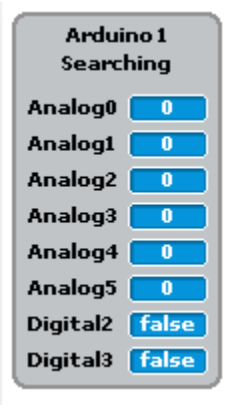
Gdy program będzie wgrany, diody RX i TX na Arduino powinny zacząć migać bardzo szybko, co świadczy o ciągłym przesyłaniu informacji między Arduino a S4A. Aby korzystać z S4A, najczęściej potrzebne jest odświeżenie programu S4A i przypomnienie mu, aby jeszcze raz sprawdził połączenie i poszukał płytki Arduino (dlatego przeprowadzamy taką operację).

W miejscu, w którym na rysunku zaznaczona jest różowa strzałka, znajdują się miniatury duszków. Gdy klikniemy prawym przyciskiem myszy duszka, który wygląda tak jak Arduino i nazywa się Arduino1, rozwinie się menu z opcjami. Wybieramy z niego opcję „usuń”. Wówczas nasz duszek zniknie.



Dodajemy nowego duszka Arduino, klikając w ikonę z dużą literą „A”:

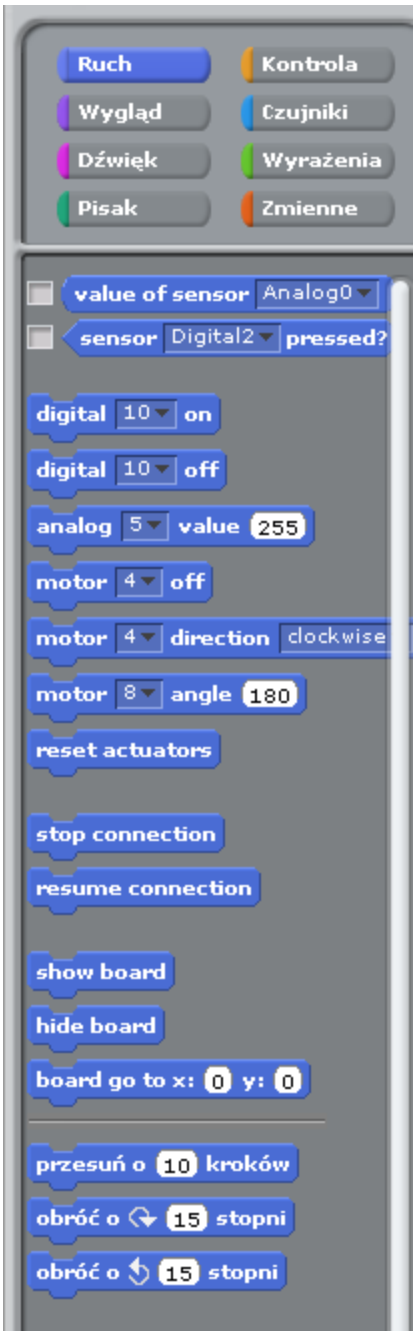
Połączenie powinno zostać nawiązane automatycznie, może się zdarzyć, że trzeba chwilę poczekać. Sprawdzamy, czy Arduino jest poprawnie podłączone obserwując poniższy rysunek znajdujący się przy duszku Arduino:



Rysunek przedstawia tablicę z wartościami odbieranymi z Arduino. Jeżeli te wartości się nie poruszają lub wynoszą zero, płytka nie jest podłączona poprawnie. Jeżeli wartości się zmieniają na dodatnie, oznacza to, że mamy połączenie.

Jeżeli nie ma połączenia, sprawdzamy, czy płytka jest podłączona do komputera - często pomaga wyjęcie i ponowne włożenie kabla USB.

Gdy połączenie jest poprawne, przechodzimy do omówienia pierwszych bloków z kategorii Ruch.

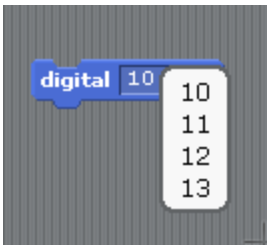

Ruch
Kontrola
Wygląd
Czujniki
Dźwięk
Wyrażenia
Pisak
Zmienne
value of sensor Analog0
sensor Digital2 pressed?
digital 10 on
digital 10 off
analog 5 value 255
motor 4 off
motor 4 direction clockwise
motor 8 angle 180
reset actuators
stop connection
resume connection
show board
hide board
board go to x: 0 y: 0
przesuń o 10 kroków
obróć o 15 stopni
obróć o 15 stopni

Omawiamy poszczególne bloki na przykładzie „digital… on” i „digital… off”.

Blok digital (10) ON (włączony) ustawia na Arduino stan wysoki, czyli wysyła napięcie 5V na pin o numerze 10.

Blok digital (10) OFF (wyłączony) ustawia na Arduino stan niski, czyli wysyła napięcie 0V na pin o numerze 10.

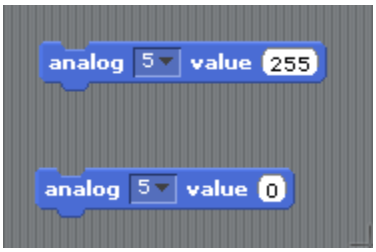
Piny wybieramy z listy rozwijanej, która otworzy się, gdy kliniemy strzałkę w polu z liczbą 10 – dostępne piny to: 10, 11, 12, 13.



Zadajemy uczniom pytanie: co stanie się, jeśli do pinu 10 zostanie podłączona dioda? I tworzymy skrypt jak na poniższym obrazku:

Po naciśnięciu flagi dioda zaświeci się (bo wysyłamy 5V stan wysoki na pin 10), świeci się przez 1 sekundę, po czym gaśnie, (bo wysyłamy 0V stan niski na pin 10).

Następnie omawiamy blok „analog… value…”. Wysyła on na pin 5 wartość od 0 do 255 (jest to proporcjonalne napięcie od 0V do 5V). Możemy wybrać inne piny niż 5 analogicznie jak w przypadku „digital… on” – piny wybieramy z listy rozwijanej, która pojawi się po kliknięciu strzałki. Dostępne wartości (piny) to: 5, 6, 9, 10.



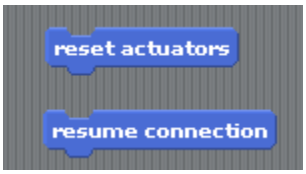
Te bloki możemy wykorzystać, jeśli chcemy w sposób płynny zmienić jasność świecącej diody, używając np. zmiennej czy precyzyjnego sterowania servem lub częstotliwością buzzera.

Jeśli podłączymy do Arduino silnik krokowy pod pin numer 4, to możemy nim sterować za pomocą poniższych bloków: motor… direction…” i „motor… off”.
Za pomocą bloku „motor ... direction…” ustawiamy kierunek ruchu silnika (zgodny z ruchem zegara lub przeciwny do ruchu zegara), a za pomocą „motor… off” wyłączamy silnik (czyli wysyłamy stan niski 0V).

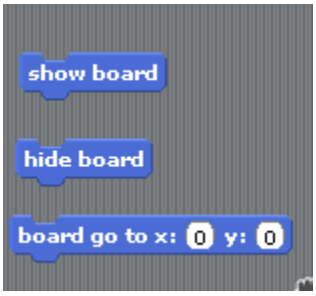
Następny blok „motor… angle…” jest przeznaczony do sterowania servem. Podpinamy sygnał serva pod pin numer 8. Angle oznacza tu kąt położenia orczyka serva.

W poniższym skrypcie po naciśnięciu flagi orczyk serva ustawi się pod kątem 0 stopni, poczeka 1 sekundę, a następnie ustawi orczyk serva pod kątem 180 stopni.

Dwa bloki „reset actuators” i „resume connection” odpowiadają za połączenie z Arduino: możemy je zresetować lub przywrócić. Te bloki są używane do bardziej zaawansowanych projektów.



Kolejne bloki: „show board”, „hide board” oraz „board go to x:… y:…” odpowiadają za wygląd i położenie duszka Arduino. Możemy duszka pokazać, ukryć lub ustawić jego dowolną pozycję na scenie za pomocą układu współrzędnych.

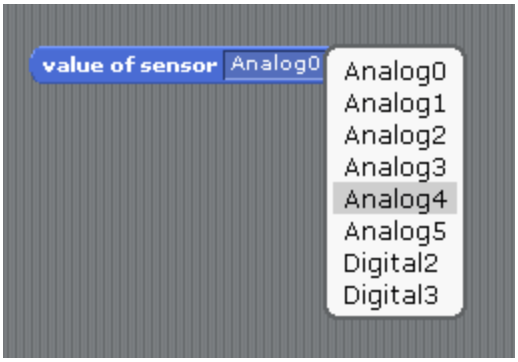


Teraz omówimy pierwsze dwa bloki i zaprezentujemy je na przykładzie z Joystick

Value of sensor to blok z zaokrąglonymi krawędziami. Dlatego wiemy, że znajdują się w nim najczęściej liczby.

Ten blok pobiera wartości z Arduino z pinów oznaczonych jako Analog IN.

Schemat do podłączenia Joystik do Arduino jest taki sam, jak wyżej. A nas interesuje wejście A0 oraz A1, ponieważ są one podłączone i zmieniają się na nich wartości.



A więc:

vale of sensor Analog0 to liczba 213
vale of sensor Analog1 to liczba 211

Dlatego teraz możemy uzależnić tę wartość w warunku.

I gdy vale of sensor Analog0 będzie mniejsza niż 500, to warunek jest spełniony i daje komunikat TRUE/prawda.

W tym przypadku użyliśmy 500, ale na każdej płytce jest inaczej i trzeba zaobserwować moment zmiany – gdy przesuwamy gałkę na joystiku, wartość się zmienia.

Dzięki temu blokowi otrzymujemy funkcjonalność wpływania sensorów Arduino na duszki w programie, ale też na wyjścia na Arduino.

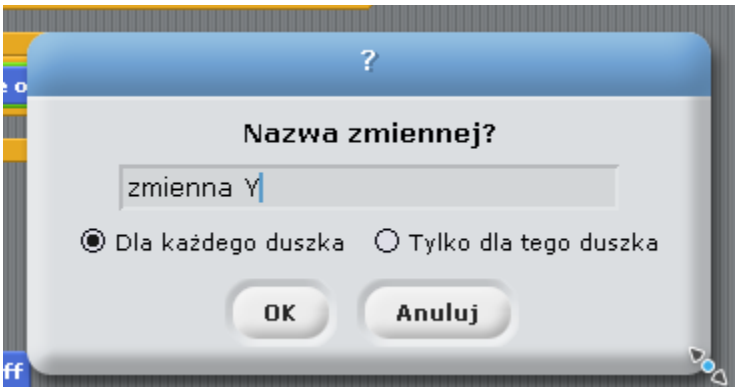
Gdy ułożymy taki skrypt, będzie on odbierał informacje z Arduino z Joystik z A0 i A1, i wpływał na pin numer 13.

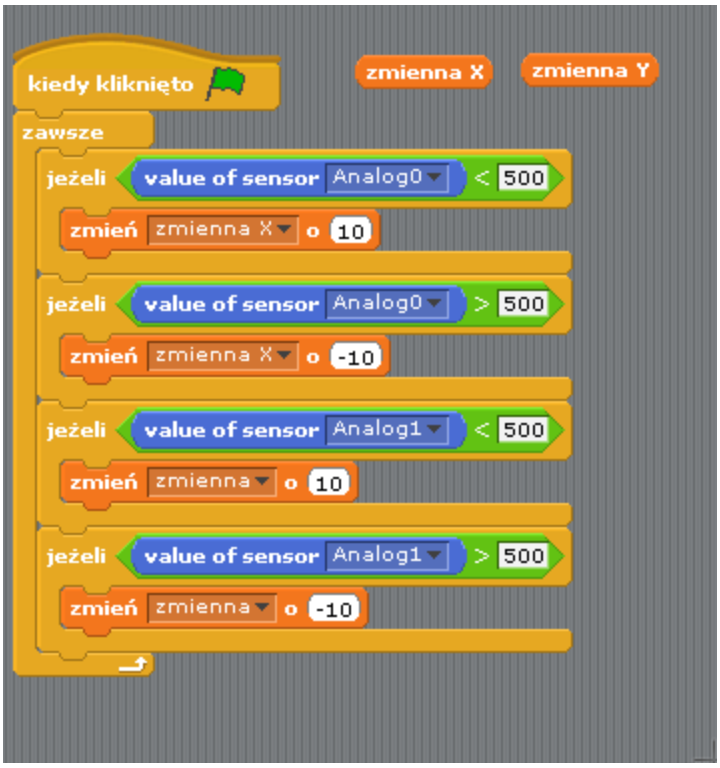
Testując ten skrypt możemy zaobserwować jak wbudowana dioda na Arduino pod 13 z Literą L jest zależna od pozycji joystika.

Zadanie dla bardziej zaawansowanych

Należy sprawić, by joystik sterował duszkiem - kotem. Do tego zadania trzeba stworzyć nowe zmienne dla wszystkich duszków, ponieważ duszki (np. kot) nie mają bloków odpowiedzialnych za sterowanie Arduino.

Przykładowe rozwiązanie:





I w nowo stworzonym duszku:

ZADANIE SPRAWDZAJĄCE UMIEJĘTNOŚCI ZDOBYTE PODCZAS ZAJĘĆ:

Uczeń / uczennica, pracując samodzielnie albo w dwu- lub trzyosobowym zespole buduje układ z wykorzystaniem Arduino i joysticka. Zadanie polega na podłączeniu płytki z joystickiem, wytłumaczeniu, gdzie biegną piny zasilające i sygnałowe, oraz wyjaśnieniu zasady działania programu i tego, skąd się biorą wartości X oraz Y.

PIGUŁKA WIEDZY I INSPIRACJI DLA OSÓB PROWADZĄCYCH:

Kurs programowania Arduino Forbot:
http://forbot.pl/blog/artykuly/programowanie/kurs-arduino-w-robotyce-1-wstep-id936

Podstawowe informacje na temat prądu elektrycznego:
http://forbot.pl/blog/artykuly/podstawy/podstawy-elektroniki-1-napiecie-prad-opor-zasilanie-id3947

Jak działa płytka stykowa (prototypowa):
https://pl.wikipedia.org/wiki/P%C5%82ytka_prototypowa

Czym jest joystick:
https://pl.wikipedia.org/wiki/D%C5%BCojstik

Strona programu Arduino For Scratch
http://s4a.cat/

Zasady bezpieczeństwa w postępowaniu z modułami Arduino:
https://www.rugged-circuits.com/10-ways-to-destroy-an-arduino/

Scenariusz został opracowany na potrzeby projektu „MoboLab – roboty i tablety w Twojej szkole". Celem projektu jest zwiększenie kompetencji informatycznych z zakresu programowania i wykorzystywania technologii mobilnych w uczeniu się, a także kreatywności, innowacyjności i umiejętności współpracy w zespole z wykorzystaniem TIK, uczniów / uczennic z (UCZ) z 6 szkół podnadgimnazjalnych i 4 gimnazjów Wołomina i Zielonki. Projekt dofinansowany jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego (Regionalny Program Operacyjny Województwa Mazowieckiego na lata 2014-2020, Oś Priorytetowa X. Edukacja dla rozwoju regionu, Działanie 10.1. Edukacja ogólna i przedszkolna, Poddziałanie 10.1.2. Edukacja ogólna w ramach ZIT).