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

# SCENARIUSZ 18. PROGRAMOWANIE ARDUINO Z WYKORZYSTANIEM PROGRAMU mBLOCK

*scenariusz zajęć pozalekcyjnych*

autor: Michał Podziomek

redakcja: Agnieszka Koszowska

## SŁOWA KLUCZOWE:

Arduino, programowanie, mBlock, Scratch4Arduino, S4a, elektronika

## KRÓTKI OPIS ZAJĘĆ:

Podczas zajęć uczniowie i uczennice uczą się podstaw programowania mikrokontrolera **Arduino** za pomocą programu **mBlock**. Poznają i/lub utrwalają wiedzę o Arduino i programują je korzystając z **Trybu Arduino** programu mBlock. Poznają i uczą się używać podstawowych pojęć programistycznych: funkcji, zmiennych. Tworzą i analizują prosty kod w języku **Arduino IDE**.

## WIEDZA I UMIEJĘTNOŚCI ZDOBYTE PRZEZ UCZNIA / UCZENNICĘ:

- wie, czym są mikrokontrolery i do czego służą,
- zna pojęcia: mikrokontroler, skrypt, program, algorytm, sterowanie, warunek, pętla,
- potrafi w podstawowym stopniu samodzielnie obsługiwać Arduino (podłączyć płytkę do komputera, wgrać prosty program),
- 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 interfejsu programu mBlock, wie, gdzie szukać potrzebnych bloków,
- potrafi samodzielnie uruchomić program mBlock,

- potrafi samodzielnie podłączyć Arduino do komputera z wykorzystaniem przewodu USB,
- potrafi włączać i wyłączać piny na Arduino z użyciem programu mBlock,
- 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,
- projektor i laptop (w części teoretycznej).

## CO NALEŻY PRZYGOTOWAĆ PRZED ZAJĘCIAMI:

- zapoznać się ze scenariuszem, upewnić się, że wszystko jest zrozumiałe i uzupełnić niezbędną wiedzę z wykorzystaniem pozostałych materiałów

szkoleniowych lub materiałów zewnętrznych,
- samodzielnie wykonać scenariusz zwracając uwagę na potencjalne problemy i zagrożenia,
- upewnić się że wymagane oprogramowanie jest zainstalowane na komputerach w miejscu przeprowadzania szkolenia - Arduino IDE, mBlock,
- upewnić się że każdy z komputerów jest w stanie rozpoznać podłączone Arduino,
- przy każdym stanowisku przygotować elementy zestawu Arduino, które będą wykorzystywane podczas zajęć,
- 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.

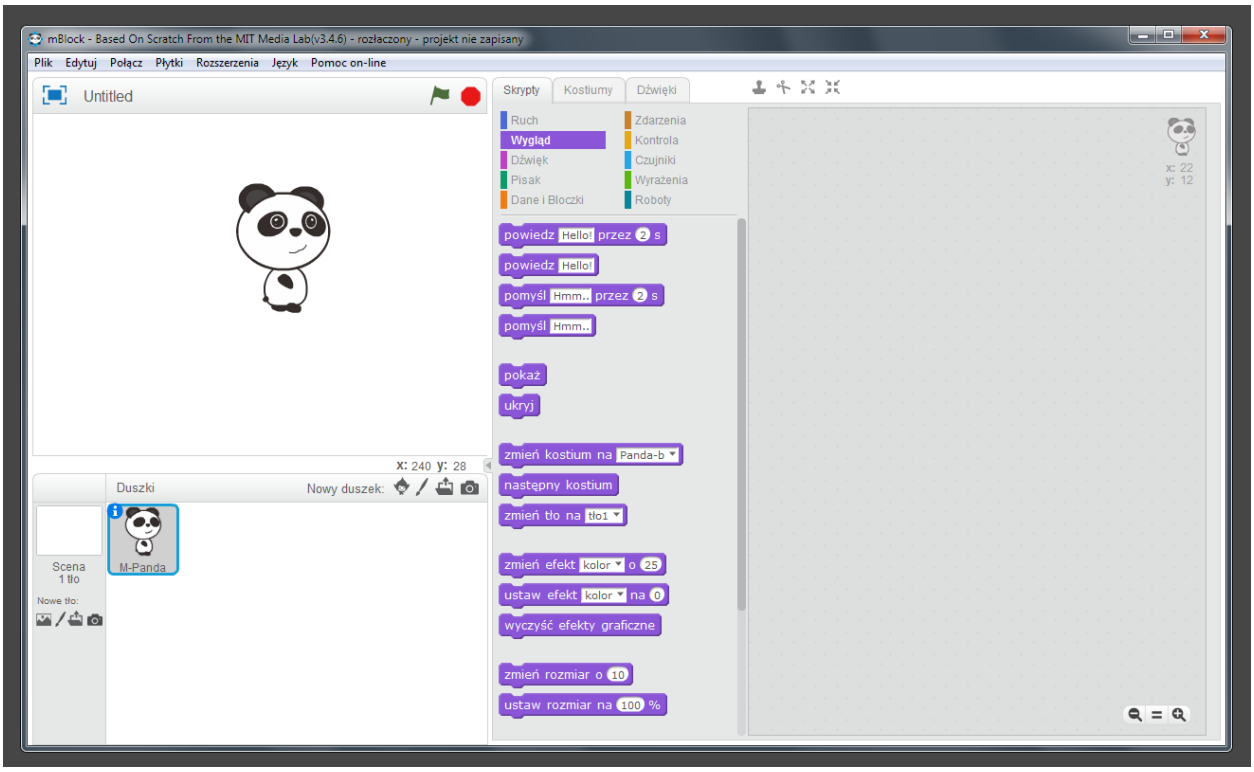
## PRZEBIEG ZAJĘĆ:

### Uruchamiamy program mBlock – 5 minut

Na początek włączamy program mBlock. Skrót do programu powinien być dostępny na pulpicie lub po wpisaniu jego nazwy w wyszukiwarkę systemową (zarówno Windows jak i Linux).

*Uwaga! Osoba prowadząca zajęcia powinna się upewnić wcześniej, czy program mBlock jest dostępny na komputerach, na których będą prowadzone zajęcia. Jeżeli jest problem ze znalezieniem, lokalizacją lub włączeniem programu, należy zgłosić problem opiekunowi sali informatycznej.*

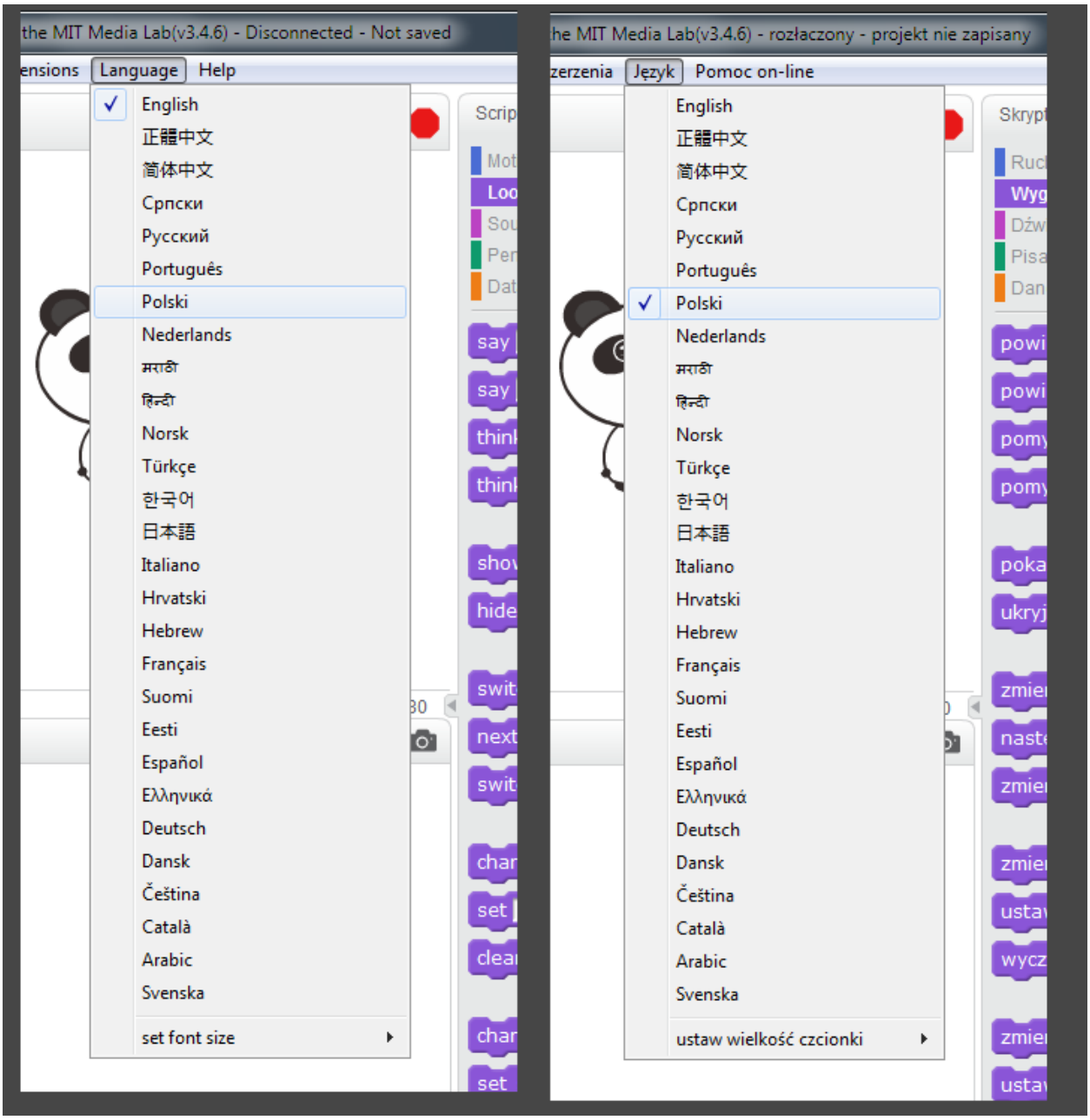
Po uruchomieniu przywita nas okno:




mBlock jest zmodyfikowanym programem Scratch, który umożliwia tworzenie skryptów do sterowania robotem mBot, ale posiada również moduł służący do programowania Arduino. Jednak, by móc to robić, należy przejść przez proces konfiguracji.

**Konfigurujemy mBlock do działania z Arduino – 15 minut**

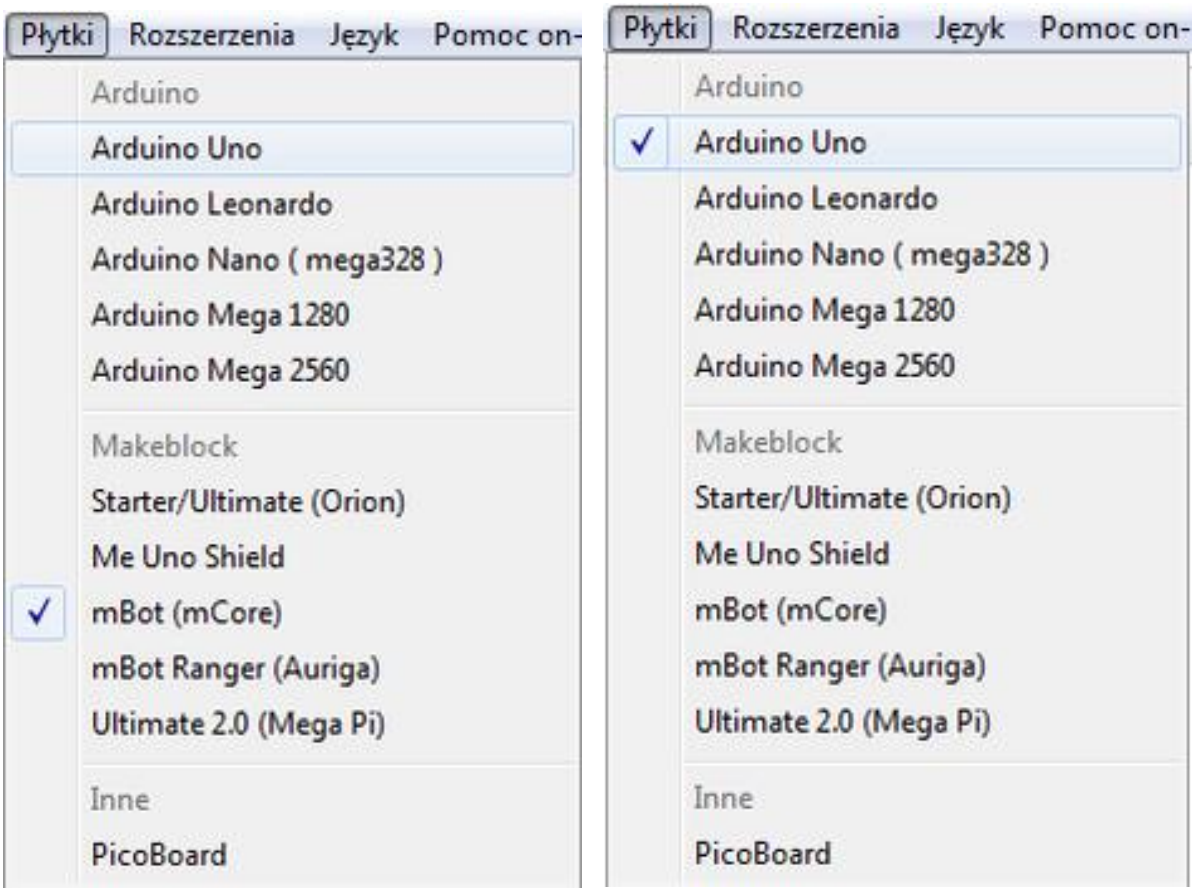
*Uwaga! Jeżeli mamy program w języku innym niż polski, możemy to zmienić u góry w menu (zakładka **Język** lub **Language**):*




By móc programować Arduino z wykorzystaniem programu mBlock, musimy wykonać dwie czynności: zmienić tryb programu z **mBot** na **Arduino** oraz włączyć **rozszerzenie Arduino**. Robimy to w następujący sposób:

*Krok pierwszy:*

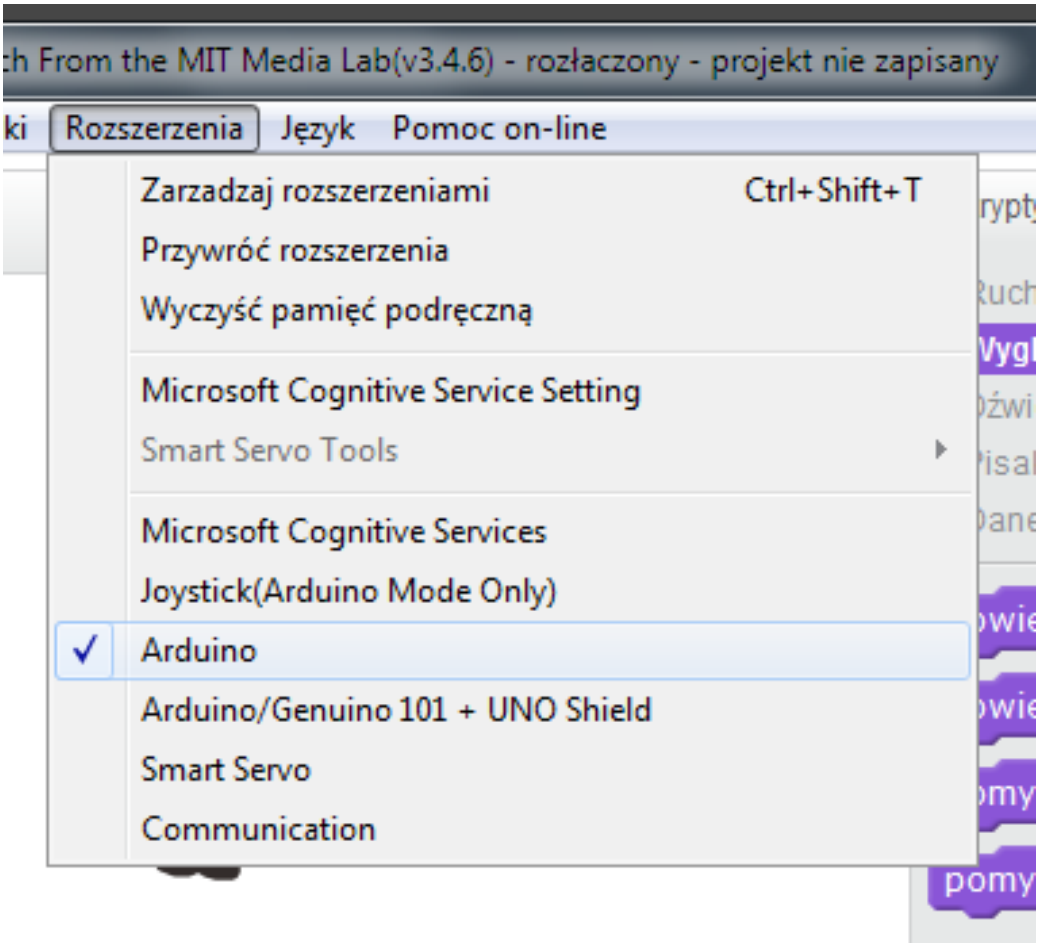
W menu **Płytki** zmieniamy zaznaczone pole **mBot (mCore)** na **Arduino Uno** (lub inne, jeżeli płytka, na której pracujemy, jest inna niż Arduino Uno).




To spowoduje, że zostanie włączony moduł z blokami Arduino. Jest on pod zakładką **Roboty**, ale do tego przejdziemy za chwilę, po zakończeniu konfiguracji.

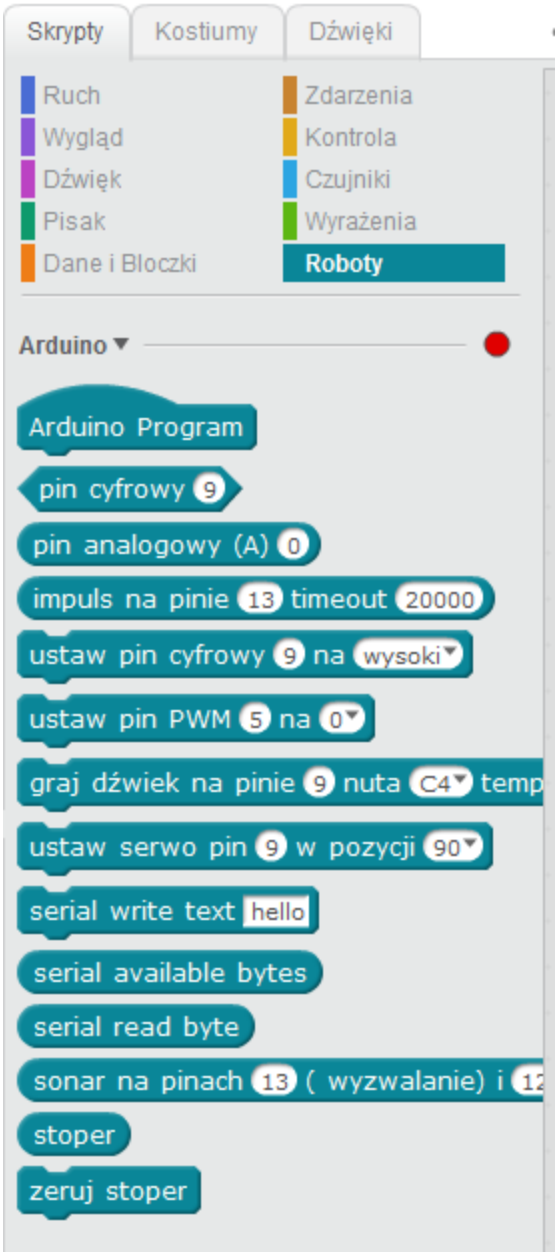
*Krok drugi:*

Z menu **Rozszerzenia** wybieramy Arduino:

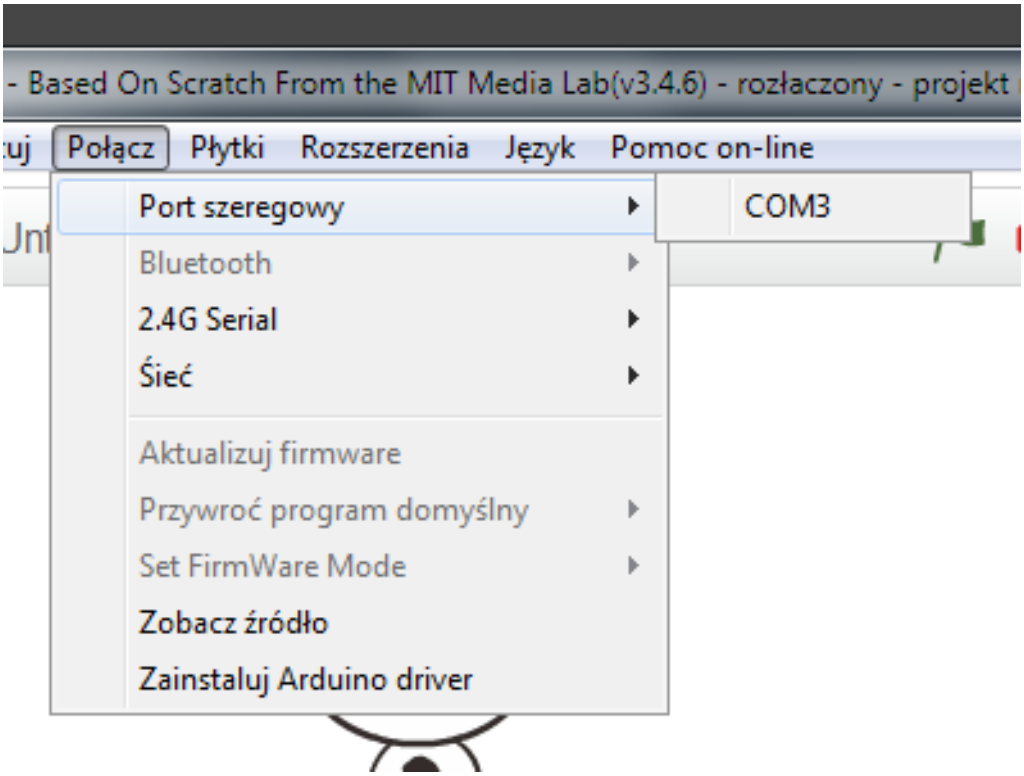



**Podłączamy płytkę Arduino – 10 minut**

Teraz możemy kliknąć w niebieski moduł **Roboty.** Naszym oczom powinien się ukazać następujący widok:



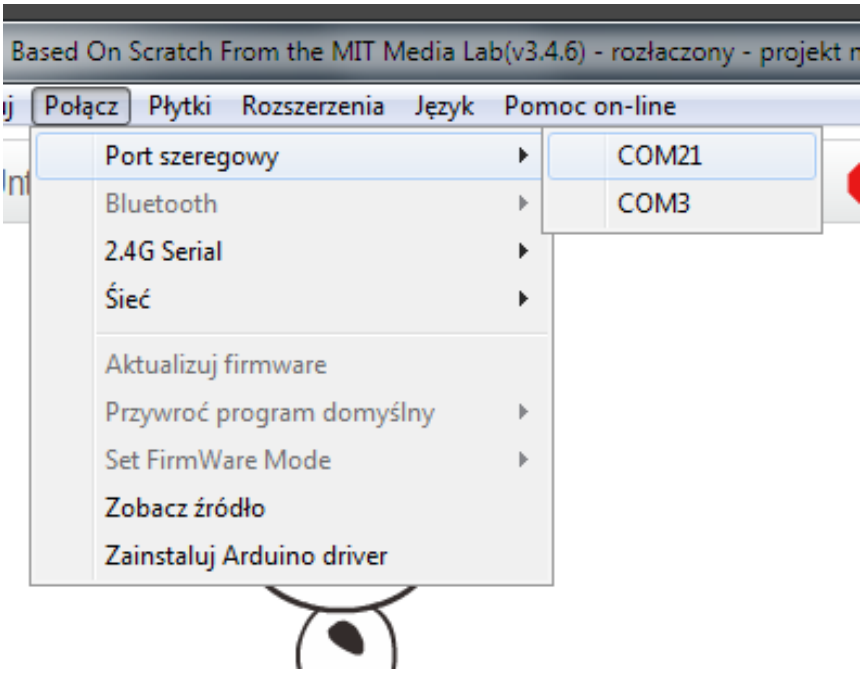

Tam gdzie do tej pory były moduły obsługujące robota mBot, widzimy nazwę Arduino, ze strzałką po lewej stronie i czerwoną kropką po prawej. Jest to rozszerzenie które przed chwilą włączyliśmy. Strzałka koło nazwy pozwala nam je schować lub rozwinąć – jest użyteczna jeśli stosujemy wiele rozszerzeń na raz. Po prawej widzimy czerwoną kropkę – oznacza ona brak połączenia z płytką. Jeżeli połączenie zostanie nawiązane, kropka zmieni kolor na zielony. Połączmy się zatem z płytką. Na początek rozwiniemy menu **Połącz>Port Szeregowy** i sprawdzimy jakie opcje zostaną wyświetlone. Na obrazku poniżej mamy dostępny port o nazwie COM3, natomiast wyświetlone porty mogą być różne dla każdego komputera:



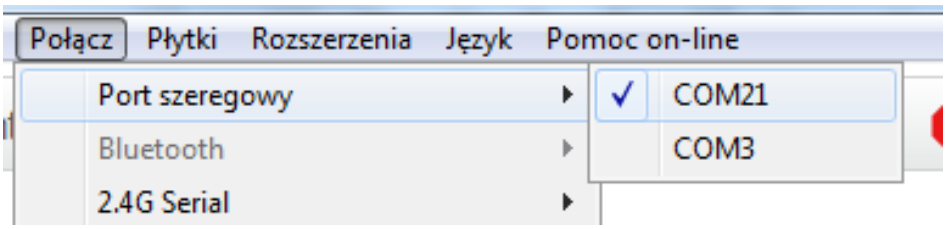

COM3, COM4, COM5... to są adresy urządzeń USB podłączonych do komputera. Może to być klawiatura, mysz, drukarka, dysk zewnętrzny – urządzeń może być wiele. Staramy się zapamiętać lub zapisać, które z portów się wyświetlają (czyli pod którymi mamy podpięte już jakieś urządzenie) i klikamy, żeby wyjść z tego menu (jest to konieczne!).

Następnie podłączamy przewód USB do naszej płytki Arduino pasującym końcem, a drugi koniec podłączamy do komputera. Ponownie klikamy w menu **Połącz>port szeregowy** i szukamy nowego dostępnego portu, który był na liście.

Na przykładzie poniżej widzimy port 21.




Klikamy w ten port, tak by został zaznaczony „ptaszkiem”.

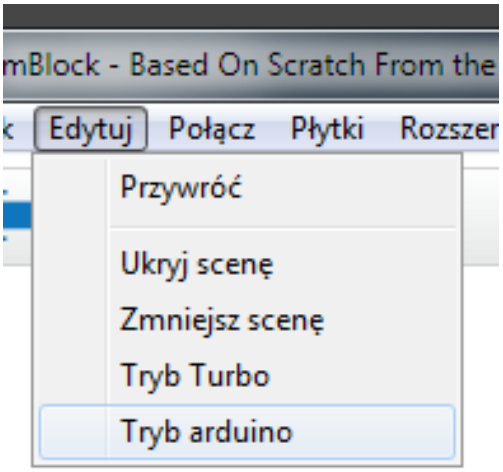



Jeżeli wszystko wykonaliśmy właściwie i nie wystąpił żaden błąd, czerwona kropka w module Roboty obok rozszerzenia Arduino powinna zmienić kolor na zielony. Oznacza to, że nawiązaliśmy połączenie z płytką i jesteśmy gotowi, by rozpocząć jej programowanie.

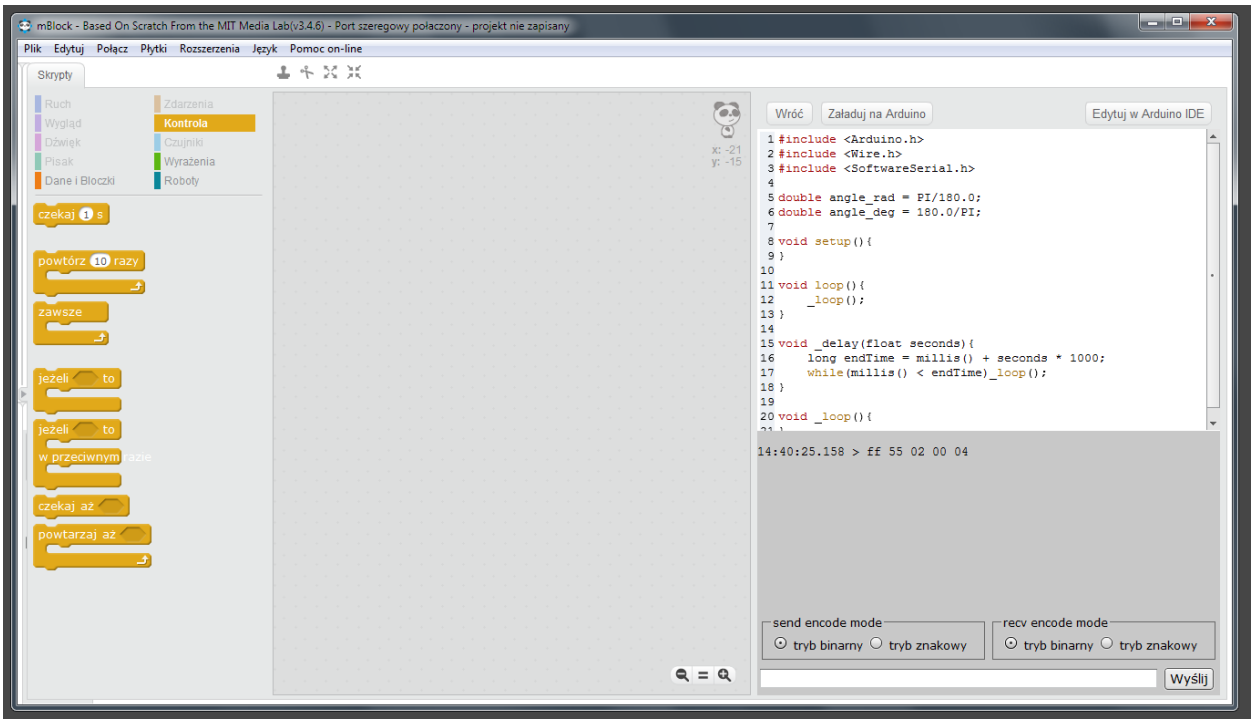

Zanim jednak to nastąpi, poznamy trochę bardziej zaawansowany widok, jaki udostępnia nam program mBlock. Pozwala on na więcej kontroli nad tym, co robimy. Ponadto będziemy mogli zobaczyć, jak wygląda prawdziwy kod programu, czyli taki, jakim posługują się na co dzień programiści, i jakiego używa się programując w języku Arduino IDE.

**Poznajemy tryb edycji Arduino – 15 minut**

By wejść do trybu Arduino, klikamy w menu **Edytuj>Tryb Arduino**.



Po kliknięciu w tę pozycję menu, zmieni się wygląd naszego okna. Elementy Sceny zostaną ukryte, a po prawej stronie pojawi się okno wyświetlające dodatkowe opcje:




Omówmy je zaczynając od góry.

Pasek na górze zawiera trzy przyciski: **Wróć**, **Załaduj na Arduino** i **Edytuj w Arduino IDE**.

Przycisk **Wróć** umożliwia powrót do normalnego widoku, w którym widzimy Scenę, a tryb Arduino jest ukryty.

Przycisk **Załaduj na Arduino** przygotowuje i wysyła nasz kod na płytkę Arduino. Przygotowanie to nazywamy **kompilacją** – jest to proces, w którym kod zrozumiały dla człowieka, pisany przy pomocy naszego języka (lub np. układanegp z „klocków” w języku Scratch), jest tłumaczony na kod zrozumiały dla programu, składający się z zer i jedynek (tzw. **zapis binarny**). Odwrócenie tego procesu jest często bardzo trudne i czasochłonne.

Przycisk **Edytuj w Arduino IDE** włączy program Arduino IDE służący do programowania Arduino (jeżeli jest on zainstalowany na tym komputerze) i otworzy skrypt ułożony wcześniej z „klocków” w mBlocku w formie poprawnego kodu programu. Widzimy go też w oknie poniżej. Skrót użyty w nazwie programu, czyli **IDE**, oznacza „integrated development environment”, czyli zintegrowane środowisko programistyczne. Jest to program do pisania programów, który daje programistom wiele narzędzi ułatwiających ich pracę. Więcej o tym czym jest IDE możemy poczytać w poniższym artykule:

https://pl.wikipedia.org/wiki/Zintegrowane_%C5%9Brodowisko_programistyczne

Jeżeli klikniemy w ten przycisk, pokaże się nam ekran startowy Arduino IDE, np taki:


Będą też widoczne ostatnio używane okna. Jeżeli pracujemy na komputerze publicznym, prawdopodobnie będą to programy, których używała osoba pracująca na Arduino przed nami. Możemy je obejrzeć. Okno Arduino IDE powinno wyglądać rak, jak na poniższym obrazku:


Zamknijmy teraz Arduino IDE (polecenien **File>Quit** lub kombinacją klawiszy **Ctrl+Q**) i wróćmy do mBlocka.

Okno z kodem znajduje się poniżej paska z przyciskami. Zawiera kod programu, który na razie nic nie robi. Dlaczego zatem jest tam wpisany? Są to jedynie opcje konfiguracyjne, których potrzebuje mBlock. Omówmy je szybko:

```
#include <Arduino.h>
```

Powyższa linia dołącza różne biblioteki, żeby nasze Arduino było „mądrzejsze", np. wiedziało, jaka jest wartość liczby pi lub potrafiło ją sobie policzyć z zadaną precyzją. **Biblioteki** to takie gotowe programy, z których możemy korzystać, ale żeby to zrobić, muszą one być na początku dołączone (ang. „included").

```
double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
```

To są dwie zmienne: jedna o nazwie **angle_rad** (czyli kąt_radiany), a druga o nazwie **angle_deg** (kąt_stopnie). Dzięki tym zmiennym Arduino będzie wiedziało, ile to jest radian, a ile to jest stopień kątowy. Jeżeli kiedyś będziemy używali Arduino z silniczkami elektrycznymi, to może być bardzo przydatne. Przed nazwami zmiennych jest słówko **double**. Ono mówi naszemu Arduino, co to jest za typ zmiennej, czyli czego może oczekiwać jako jej wartości. „Double" to po polsku „liczba podwójna". Typ to może być również **int** („integer", czyli liczba całkowita), **char** („characters", czyli znaki tekstowe), **float** („zmiennoprzecinkowe", czyli ułamkowe) itp.

Układając klocki w mBlocku nie mamy potrzeby definiowania typów zmiennych. Bardzo często możemy też ułożyć kod z klocków bez używania zmiennych w ogóle. Jednak w programowaniu „na poważnie" stosowanie zmiennych jest nieuniknione, często wręcz zbawienne, dlatego warto je opanować.

```
void setup(); void loop();
```

To są funkcje, w których będzie „mieszkał" kod, który napiszemy. Tutaj zaczną się pojawiać wpisy odpowiadające wybranym modułom, w miarę jak będziemy układać program z klocków Scratcha.

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

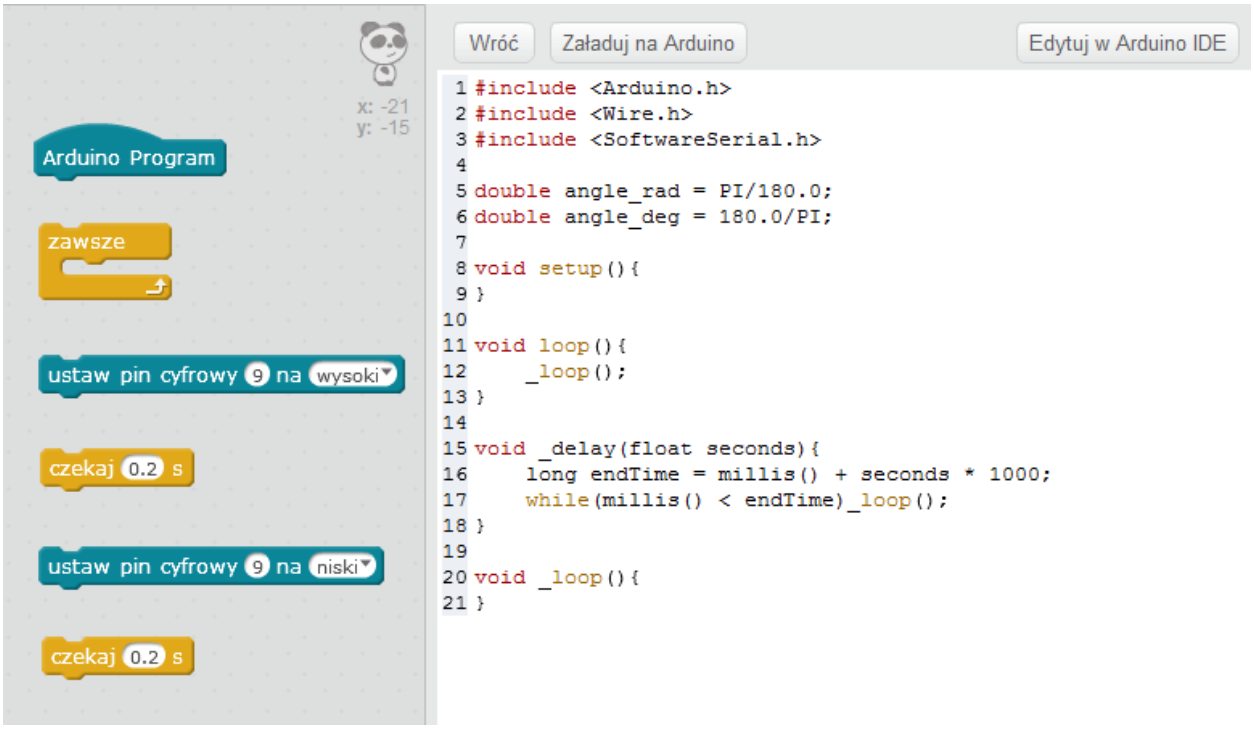
**Przypomnienie materiału z poprzedniej części zajęć – 10 minut**

Krótko przypominamy materiał z pierwszej części zajęć i uruchamiamy tryb edycji Arduino w programie mBlock.

**Przygotowanie programu na Arduino – 25 min.**

Nasze Arduino ma wbudowaną w płytkę małą lampkę – tzw. diodę LED. Dioda LED świeci, gdy przepływa przez nią prąd. Znajduje się ona na pinie numer 13. Pin jest to określenie styku elektronicznego – pochodzi od nazwy małych prostokątnych bolców umieszczanych na elektronice, do których możemy łatwo podłączać inne układy. Na Arduino Uno piny to są czarne listwy z otworkami na krawędziach płytki. Każdy z nich ma swój numer, niektóre mają opisy, jeszcze inne – symbole.

Ponieważ znamy „adres" naszej diody LED (pin 13), możemy z nią „porozmawiać" – mianowicie, możemy jej wydać sygnał zaświecenia lub zgaszenia poprzez włączenie lub wyłączenie napięcia na pinie. Ułóżmy to z klocków tak jak na sekwencji obrazków poniżej, jednocześnie obserwując jak zmienia się kod tekstowy Arduino w oknie po prawej stronie:




Wróć | Załaduj na Arduino | Edytuj w Arduino IDE

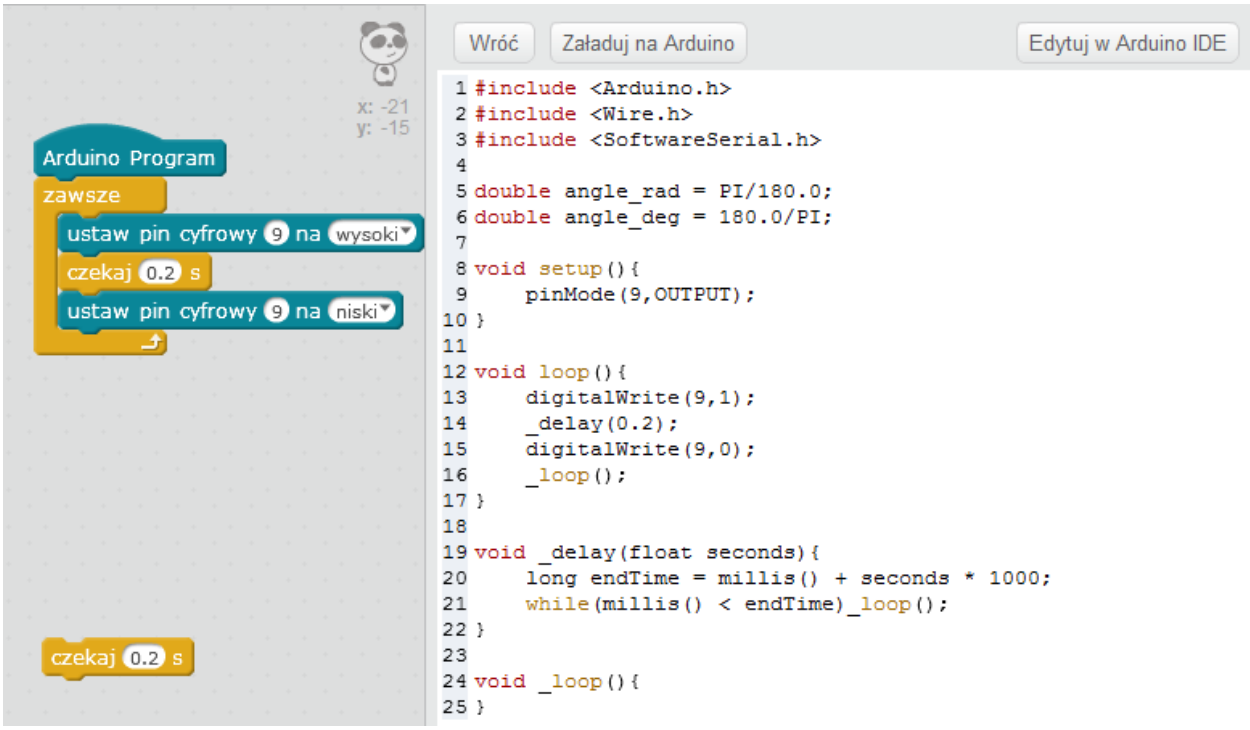
```
1 #include <Arduino.h>
2 #include <Wire.h>
3 #include <SoftwareSerial.h>
4
5 double angle_rad = PI/180.0;
6 double angle_deg = 180.0/PI;
7
8 void setup(){
9 }
10
11 void loop(){
12     _loop();
13 }
14
15 void _delay(float seconds){
16     long endTime = millis() + seconds * 1000;
17     while(millis() < endTime)_loop();
18 }
19
20 void _loop(){
21 }
```

Podłączamy klocek "zawsze". Brak zmiany w kodzie Arduino. Pamiętajmy o tym.

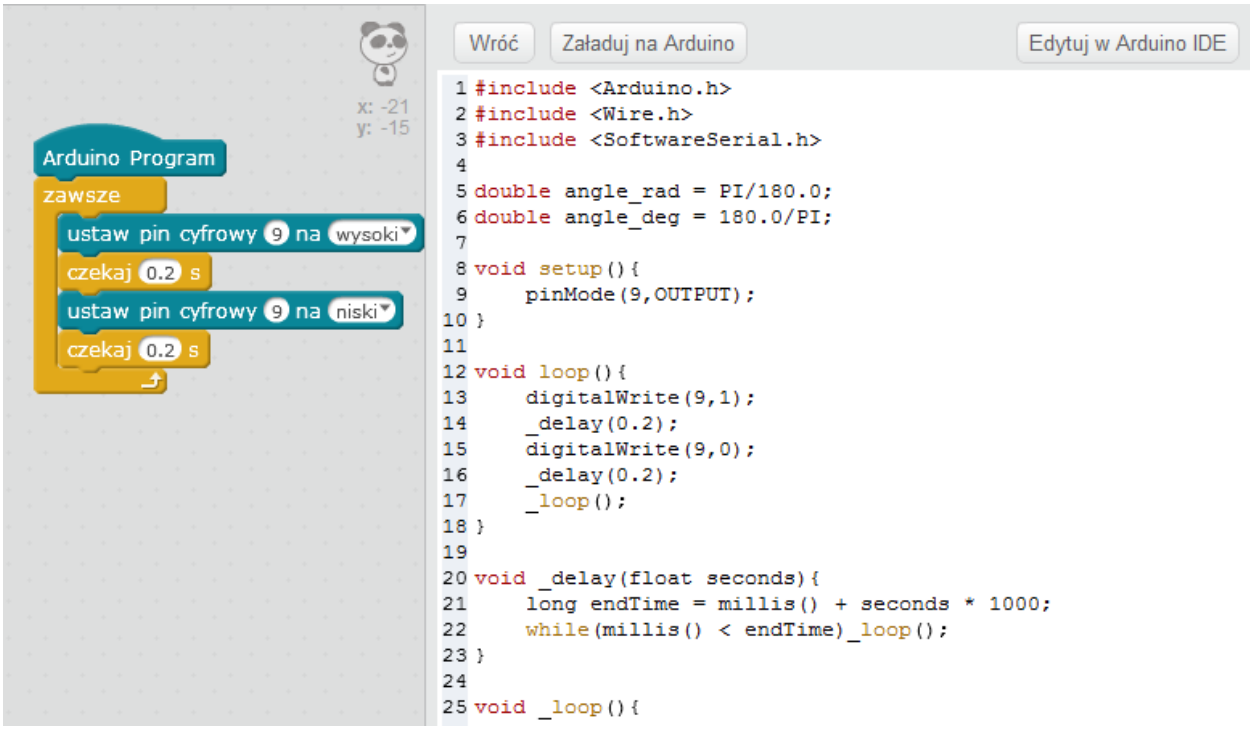

Wróć | Załaduj na Arduino | Edytuj w Arduino IDE

```
1 #include <Arduino.h>
2 #include <Wire.h>
3 #include <SoftwareSerial.h>
4
5 double angle_rad = PI/180.0;
6 double angle_deg = 180.0/PI;
7
8 void setup(){
9     pinMode(9,OUTPUT);
10 }
11
12 void loop(){
13     digitalWrite(9,1);
14     _loop();
15 }
16
17 void _delay(float seconds){
18     long endTime = millis() + seconds * 1000;
19     while(millis() < endTime)_loop();
20 }
21
22 void _loop(){
23 }
```

Podłączamy klocek „ustaw pin cyfrowy”. Pojawia się linia 9 – **pinMode**, oraz 13 – **digitalWrite**.

Dodajemy „ustaw pin cyfrowy” – pojawia się linia 15, **digitalWrite**. Natomiast nie pojawia się pinMode, tak jak to się stało poprzednim razem.




Dodajemy „czekaj” - pojawia się linia 16, **_delay**.

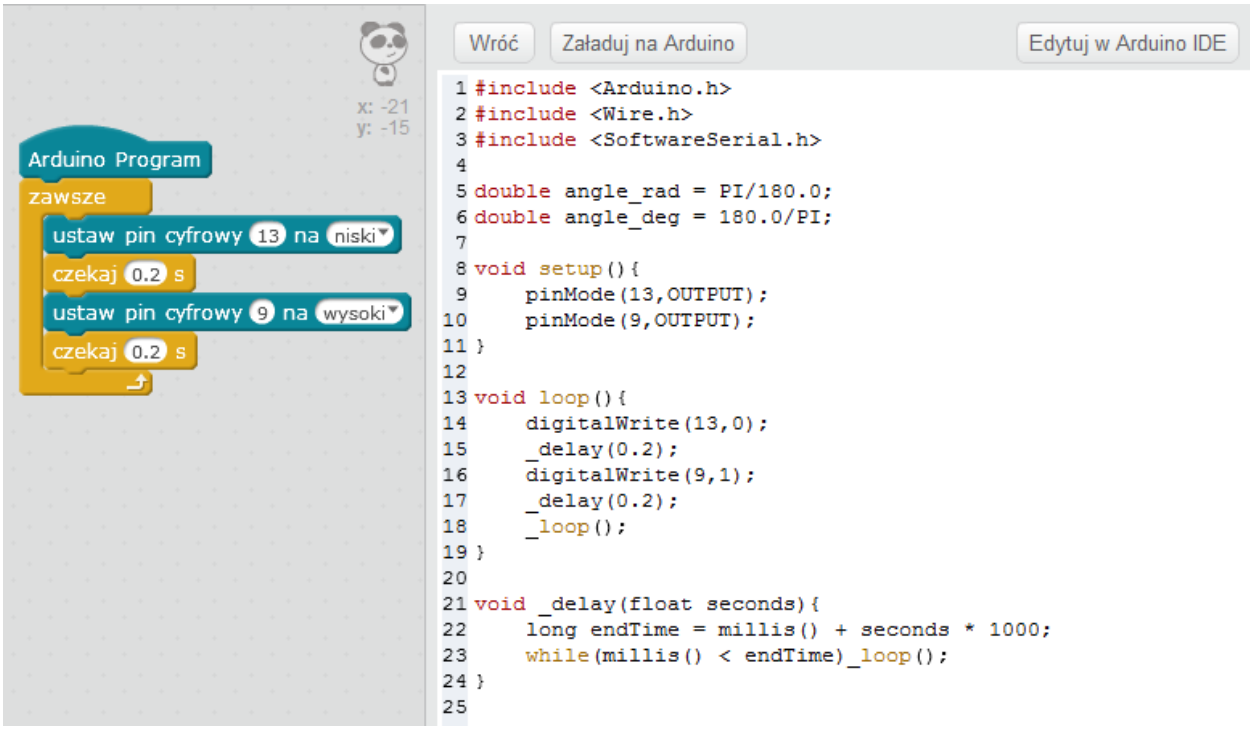
Łatwo możemy zauważyć następującą zależność: po dołożeniu klocka w mBlock, pojawia się nowa linijka w kodzie Arduino. Dla „Ustaw pin cyfrowy 9 na wysoki” - zostaje dodana linijka wewnątrz funkcji **void loop()** (na ostatnim obrazku jest to linijka 13), w której widzimy: **digitalWrite(9,1);**.

Ta linijka zapisuje na pinie 9 stan **1**, czyli wysoki (ang. **HIGH**). Stan niski, czyli **LOW**, widzimy w linijce 15, gdzie mamy **digitalWrite(9,0);** czekaj 0.2 s widzimy w linijkach 14 i 16: **_delay(0.2);**.

Uwaga! Pamiętajmy o średnikach! Starajmy się zaobserwować, w których miejscach są stawiane średniki w kodzie Arduino. Będzie to bardzo ważne na późniejszym etapie naszych prac programistycznych.

Jest w naszym kodzie jednak jeden problem: „wołamy” do pinu 9, na którym obecnie nic nie ma, ponieważ nic tam nie podłączyliśmy. Powinniśmy wołać do pinu

13. Zmieńmy zatem wartości w kodzie mBlock. Musimy to wykonać dwa razy - za każdym razem obserwujemy, co się zmienia w kodzie Arduino po prawej:




Jeżeli dobrze obserwowaliśmy, to zobaczymy, że po wpisaniu „13” zamiast pierwszej liczby „9” zmieniła się linijka 9 i została dodana zawartość linijki 10. Następnie, po wpisaniu drugiej „13”, linijka 10 została usunięta.

Byliśmy właśnie świadkami deklarowania **pinów cyfrowych**, czyli takich, na których możemy włączać i wyłączać prąd. Ich przeciwieństwem są piny analogowe, na których możemy regulować napięcie prądu. Piny cyfrowe, których chcemy użyć na Arduino muszą być **deklarowane**, czyli musimy wcześniej wysłać do Arduino informację o tym, że będziemy korzystać z danego pinu w naszym programie. Jeżeli tego nie zrobimy, pin będzie przez cały czas trwania programu wyłączony.

Podczas wpisywania wartości w kodzie mBlock, w kodzie Arduino pin 13 pojawił się przed pinem 9. Stało się tak dlatego, że piny są deklarowane w programie w kolejności wystąpienia. Tak samo linijki z **digitalWrite** oraz **delay** były wpisywane w kolejności, w jakiej ułożylismy klocki. Natomiast deklaracje pinów zostały dodane w zupełnie innym miejscu, a mianowicie w **void setup()**. Void setup() zawiera kod, który na naszej płytce zostanie wykonany tylko raz, zanim zacznie być wykonywana pętla **void loop()**. Jest to więc właściwe miejsce na zadeklarowanie, których pinów będziemy używać. Pamiętajmy jednak obserwować funkcję void setup() podczas układania klocków mBlocka – dowiemy się wtedy, co jeszcze powinno być w niej umieszczane.

**void loop()**to pętla, w której działa nasz program. Czyli: po wykonaniu zawartości tej funkcji, Arduino wraca na jej początek i wykonuje ją ponownie. Zaobserwowaliśmy to na samym początku, kiedy układaliśmy nasz program. Po dołączeniu klocka **zawsze** do klocka Arduino nie nastąpiła żadna zmiana w kodzie po prawej stronie. Stało się tak dlatego, że klocek **zawsze** odpowiada funkcji **void loop()**, która jest (w większości przypadków) niezbędną, podstawową częścią programu Arduino.

Załadujemy teraz kod na płytkę Arduino. Upewniamy się, że w oknie z dostępnymi modułami mBlock po lewej stronie jest zapalone zielone światełko koło tytułu Arduino (tak jak podłączaliśmy Arduino na początku ćwiczeń). Jeżeli światełko jest czerwone, musimy ponownie wybrać port, na którym jest podłączone Arduino **(Połącz>Port szeregowy)**. Jeżeli światełko jest zielone, klikamy przycisk z prawej strony nad kodem Arduino „Załaduj na Arduino”. Po dłuższej chwili kod znajdzie się na płytce, a ta zacznie mrugać diodą LED. W oknie na samym dole po prawej stronie będziemy otrzymywali komunikaty o postępie procesu kompilacji

i załadowywania na Arduino. Jeżeli wystąpią jakiekolwiek błędy, zostaną tam też wyświetlone.

**Zadania – 10 minut**

*Zadanie 1:*

Pin 13, na którym jest dioda, na naszym Arduino (UNO) jest obok linii z oznaczeniem **PWM**. PWM oznacza piny, które potrafią płynnie sterować prądem przez nie płynącym, a nie tylko go włączać i wyłączać. To oznacza, że możemy diodę płynnie ściemniać lub rozjaśniać. Zadanie polega na:

- odpięciu kodu od turkusowego bloczka Arduino, by nie został przypadkiem załadowany na płytkę Arduino (zaobserwujmy zmianę w kodzie gdy to zrobimy),
- podpięciu do turkusowego bloczka pętli **zawsze**,
- ułożeniu w pętli **zawsze** wielu (przynajmniej 5) bloczków **ustaw pin PWM 13 na *(wartość)***, i zmianie każdej wartości na inną w przedziale 0-255 (można rosnąco, malejąco, lub dowolnie),
- dodanie między każdą z nich bloczka **czekaj**, z dowolną wartością mniejszą od 0.5,
- załadowanie kodu na Arduino. Program powinien zmieniać natężenie świecenia lampki zgodnie z ustalonymi przez nas wartościami.

*Zadanie 2:*

Po wykonaniu zadania 1, klikamy w przycisk „Edytuj w Arduino”. Powinno zostać uruchomione Arduino IDE i nasz kod otwarty jako projekt. Jeżeli tak się nie stanie, należy uruchomić Arduino IDE manualnie, utworzyć nowy projekt z **File>New**, usunąć z niego wszystko, a następnie skopiować i wkleić kod z mBlock. Następnie spróbować podłączyć się do płytki Arduino UNO w sposób analogiczny do mBlocka (wybrać nazwę płytki z **Tools>Board**, wybrać port na którym jest podłączona z Tools>Port) i załadować na nią kod poprzez kliknięcie ikonki strzałka w prawo na górze panelu.

*Zadanie 3:*

Zamienić wartości **analogWrite** na inne korzystając z Arduino IDE.

*Zadanie dodatkowe:*

Nazwa funkcji pauzującej program w Arduino nazywa się **delay**. mBlock stosuje inną funkcję, ma ona podkreślnik na początku, czyli **_delay**. Wskaż, czy i gdzie jest ona definiowana w kodzie pochodzącym z mBlocka. Jeżeli się to uda, zamień wszystkie użycia **_delay** w funkcji **void loop** na zwykłe **delay(200);** (delay(1000) jest równe jednej sekundzie).

## MOŻLIWE MODYFIKACJE DLA MŁODSZYCH KLAS:

Zajęcia w oparciu o ten scenariusz można przeprowadzić w klasach młodszych. W przypadku zajęć z młodszymi dziećmi warto zwrócić uwagę na ewentualne problemy z dokładnym podłączaniem przewodów. Uczniowie starsi mogą wykonywać zadania samodzielnie, młodszym warto podpowiadać i w większym stopniu podawać gotowe rozwiązania. Można zmniejszyć liczbę zadań i/lub je uprościć.

## ZADANIE SPRAWDZAJĄCE UMIEJĘTNOŚCI ZDOBYTE PODCZAS ZAJĘĆ:

Uczeń / uczennica, pracując samodzielnie albo w dwu- lub trzyosobowym zespole ma za zadanie samodzielnie ułożyć program wykorzystujący wbudowaną diodę LED na płytce Arduino, oraz pokazać gdzie w kodzie Arduino znajdują się poszczególne klocki, oraz wyjaśnić funkcje kodu (co jest w nawiasach, w którym miejscu, co jest w void loop, co w void setup, dlaczego).

## 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

Przydatne mogą też okazać się scenariusze 1-18 dotyczące nauki podstaw programowania z wykorzystaniem środowiska mBlock i robota mBot, opracowane w ramach projektu „MoboLab – roboty i tablety w Twojej szkole”.

*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).*