05

Implementacje do realizacji na kołach zainteresowań IT

Stanisław Ubermanowicz

W formowaniu kompetencji infotechnicznych kluczową rolę odgrywają *implementacje*. To tytułowe pojęcie w Strategii Wolnych i Otwartych Implementacji mieści w sobie całe spektrum znaczeń – od koncepcji rozwiązania problemu, poprzez proces urzeczywistniania pomysłu, aż do końcowego wytworu realizującego założoną funcjonalność. W dziedzinie dydaktyki implementacja służy jako środek-metoda uczenia się, na gruncie inżynierii oznacza proces projektowania, programowania lub konstruowania, a w obszarze infotechniki określa finalny wytwór informatyczny lub mechatroniczny.

Trafny dobór trudności zadań i zakresu samodzielnych czynności uczniów przy tworzeniu implementacji jest kluczowym zadaniem dla trenerów.

Wszystkie te specyficzne, cenne edukacyjnie walory implementowania wykorzystywane są podczas realizacji kół infotechnicznych. Implementacje mają jednak także tę właściwość, że możliwych rozwiązań wyznaczonego zadania jest bardzo wiele. Z tego względu zawarte w książce i na płycie opisy implementacji należy traktować jako przykłady do wykorzystania w starannie przemyślany sposób – nie jako gotowce do powielania, lecz jako materiał pomocniczy dla trenerów, którzy na tej podstawie wyznaczają zróżnicowany zakres ćwiczeń dopasowanych do możliwości uczniów.

W pierwszym podejściu trener wydobywa z opisu **istotę zadania implementacyjnego**. Zadaniem tym może być np. wykonanie prostej grafiki, wizualizacji, animacji, gry, interfejsu, układu pomiarowego lub sygnalizacyjnego. W modułach zaawansowanych zadaniem może być całościowe napisanie kodów realizujących procedury, funkcje lub złożone struktury w danym języku programowania. Mimo że implementacje opublikowano w pełnych wersjach, jako przykłady rozwiązań, to czynności uczniów nie powinny polegać na przepisywaniu bądź wgrywaniu kompletnych kodów źródłowych.

Na podstawie opisu każdej konkretnej implementacji, dedykowanej dla określonych grup wiekowych i rodzajów szkół, trener wyznacza na dane zajęcia adekwatne dla uczniów zadania, obejmujące albo tworzenie od podstaw wszystkich elementów, albo uzupełnianie celowo usuniętych obiektów lub fragmentów kodu.

Konieczne jest optymalne wypośrodkowanie pomiędzy zakresem dostarczonych uczniom niezbędnych półproduktów, a pozostawieniem pola do wykazania się przez nich własną inwencją.

Im bardziej dostarczona postać implementacji jest kompletna, tym bardziej zadanie powinno polegać na daleko idącej samodzielnej modyfikacji rozwiązania. Taka forma ma zastosowanie zwłaszcza przy tworzeniu modułów-interfejsów. Uczeń najpierw montuje układ wzorcowy według schematu, a następnie rozbudowuje elementy i modyfikuje oprogramowanie sterujące. Na zajęciach dotyczących projektowania graficznego należy podać wyłącznie treść zadania, określając to, co ma powstać, i pozostawiając pełną swobodę dla indywidualnej twórczości. W trudniejszych zadaniach programistycznych część kodów może być wczytywana, część uzupełniana, a ponadto – trener winien omawiać tylko takie fragmenty kodu, które są ściśle związane z tematem danej jednostki zajęć. Resztę warto pozostawić do **samodzielnego interpretowania** przez uczniów, gdyż umiejętność odczytywania znaczenia struktur i instrukcji w kodzie jest ważnym składnikiem kompetencji infotechnicznych.

W zróżnicowanych postaciach opisów i rodzajach implementacji zawarte są pewne **wskazówki dla trenerów** co do sposobów i zakresów wyznaczania uczniom zadań. Jedną z form podpowiedzi jest przygotowanie do danej tematyki zajęć implementacji alternatywnych. Ma to miejsce w propozycjach zajęć mechatronicznych, kiedy to podobne tematycznie wersje różnią się stopniem rozbudowania układu. W modułach programistycznych zróżnicowany zakres ćwiczeń sugerowany jest poprzez zadania rozszerzające sformułowane w niektórych Konspektach-scenariuszach. Wówczas nie podaje się jednak gotowych rozwiązań, gdyż w większości są to przykłady formułowania zadań do wykonania już poza zajęciami stacjonarnymi. Inną formą zaleceń jest opracowanie kodów źródłowych w wersji kompletnej dla trenera oraz w wersji z celowo wyznaczonymi miejscami do uzupełnienia przez uczniów. Bez uzupełnienia taka implementacja nie działa, zatem uczeń chcący ją uruchomić musi najpierw prawidłowo wpisać brakujące instrukcje.

Kody źródłowe są głównie dla trenerów, którzy decydują, jaki ich zakres udostępnić uczniom. Natomiast instrukcje mechatroniczne są przeznaczone wprost dla uczniów.

Różnorodność implementacji opracowanych na potrzeby realizacji kół infotechnicznych wynika także z odmiennego podejścia do optymalnych metod i zagadnień tematycznych na danym etapie rozwoju uczniów. Otóż we wczesnej fazie wdrażania do problematyki projektowania najważniejsze jest **upoglądowienie**, a to uzyskuje się poprzez dominację reprezentacji wizualnych. Tak jest właśnie w propozycjach implementacji dla szkół podstawowych, gdzie zaleca się głównie zadania polegające na cyfrowej obróbce obrazów, na tworzeniu grafiki oraz na generowaniu i interpretacji wykresów. Umiejętność odczytywania wykresów jest m.in. weryfikowana na zewnętrznym sprawdzianie w klasach szóstych, stąd praktyczny walor takiej tematyki zajęć. Proponowane dla dzieci zajęcia z mechatroniki polegają na budowaniu układów, które także pełnią rolę poglądową, gdyż optycznie sygnalizują określone stany napięć. Ponadto elementarne oprogramowywanie mikrokontrolera odbywa się w graficznym środowisku wizualno-skryptowym Scratch for Arduino (S4A).

W początkowych etapach uczenia się programowania najważniejsze są metody poglądowe z wizualizacją efektów działania kodu źródłowego.

Na poziomie szkół gimnazjalnych proponuje się propedeutyczne wdrażanie do problematyki programowania, głównie poprzez **wizualizację procedur** obsługujących obiekty ekranowe. Punktem wyjścia jest tu także tworzenie obrazów, a następnie ożywianie obiektów za pomocą skryptów zmieniających ich położenie oraz właściwości. Znakomitym zintegrowanym środowiskiem narzędziowym do programowania wizualno-skryptowego jest Scratch, w którym obiekty zwane „duszkami" są sterowane przez automatycznie integrujące się polecenia, formułowane po polsku. A co najważniejsze – gotowe instrukcje języka programowania dostępne są i układane w graficznej formie dopasowujących się puzzli, ułatwiających budowę prawidłowej struktury kodu źródłowego. Moduły mechatroniczne dla gimnazjalistów realizują zagadnienia sterowania, odczytywania stanów i wskaźnikowania z użyciem środowiska graficznego Scratch for Arduino.

Dla szkół ponadgimnazjalnych niesprofilowanych adresowane są moduły wdrażające do programowania, wykorzystujące metodę **operacjonalizacji pojęć informatycznych**, będących w znacznej mierze kategoriami abstrakcyjnymi. Działania realizowane przez implementacje ułatwiają zrozumienie najważniejszych struktur języka programowania oraz zasad realizacji algorytmów, strategii wygranej i sztucznego intelektu. Także i tu kluczowe znaczenie ma wizualizacja, ale dochodzi ponadto silny środek motywujący, jakim jest tworzenie gier logicznych. Na tym poziomie – oprócz pracy w Scratchu – proponowane jest też programowanie wizualno-obiektowo-zdarzeniowe w zintegrowanym środowisku graficznym Lazarus z językiem FreePascal. Język ten jest dopuszczony do wykonywania zadań na maturze z informatyki, dlatego poznanie jego struktur i instrukcji ma praktyczne znaczenie w przygotowaniu się do zdania egzaminu z tej dziedziny i do dalszego studiowania infotechniki. Implementacje do modułów mechatronicznych wykorzystują dwie wersje środowisk współpracy z układem Arduino – jeden interfejs zorientowany jest na emulację interaktywnego terminala, a drugi na formę graficzno-skryptową. Polecane układy mechatroniczne służą do pomiarów światła i temperatury.

W miarę nabywania umiejętności infotechnicznych przechodzi się coraz bardziej na zadania wymagające operowania na obiektach abstrakcyjnych.

Na potrzeby szkół sprofilowanych infotechnicznie opracowano implementacje oparte bardziej na **metodach wyobrażeniowych** niż poglądowych. Uczniowie z tych szkół wybrali już taki kierunek specjalizacji IT, dlatego dla nich ważniejsze jest doskonalenie umiejętności programowania bądź poznanie na kołach zainteresowań innego niż na lekcjach języka. Z tego względu implementowanie odbywa się tam na wyższym poziomie myślenia abstrakcyjnego, w metodykach programowania obiektowego i imperatywnego. Generowanie obiektów i struktur polega głównie na pisaniu kodu źródłowego, bez wsparcia graficznego, bez *widżetów* narzędziowych, dostarczających gotowe obiekty funkcjonalne. Taki sposób implementowania algorytmów wyłącznie z poziomu kodu wymagany jest m.in. na egzaminie maturalnym, a później na studiach. Proponowane tu do wyboru implementacje są zwięzłymi kursami konstruowania głównych struktur w języku C lub w języku Python oraz przykładami wzbogacania HTML o język JavaScript z biblioteką jQuery. Dla tych szkół opracowano też alter-

natywne moduły mechatroniczne, ilustrujące zaawansowane sposoby wykorzystania układów z mikrokontrolerem do budowy mierników różnych parametrów fizycznych, do konstruowania serwomechanizmów, sterowników prądowych i transmisyjnych.

Należy w tym miejscu podkreślić, że bloki Modułów A i B, dedykowanych na konkretne poziomy i rodzaje szkół niesprofilowanych, są przeznaczone głównie dla uczniów początkujących w dziedzinie programowania i konstruowania. Chodzi o poszerzenie rzeszy uczniów, których zaciekawiłaby ta problematyka. Z tego względu zasadniczy cykl zajęć służących uczniom do zapoznania się z zagadnieniami i sprawdzenia swego potencjału powinien składać się z minimum 12 jednostek dydaktycznych.

Warto jednakże realizować dalsze lub alternatywne Moduły dla tych uczniów, którzy chcą kontynuować udział w zajęciach dodatkowych. Z uczniami uzdolnionymi, przygotowanymi w cyklu zasadniczym, można z powodzeniem przeprowadzać także zajęcia dedykowane na wyższy etap szkolnictwa. Warto przy tym pamiętać, aby **przeplatać tematykę** programowania z mechatroniką. To właśnie konstruowanie interfejsów sprawia uczniom we wszystkich rodzajach szkół najwięcej satysfakcji, lecz uczenie się programowania jest tu równie ważne.

Trenerzy mają możliwość ułożenia własnego Programu, poprzez wybór Modułów do realizacji na kołach zainteresowań na podstawie analizy treści Konspektów-scenariuszy albo opisów zadań implementacyjnych i przykładowych rozwiązań. Wstępne rozpoznanie i dobór ułatwia tabela, zawierająca zestawienie oznaczeń kodowych, nazw implementacji i tematyki zajęć oraz wskazanie środowisk pracy i zagadnień infotechnicznych.

Wykaz implementacji, środowisk i zagadnień

Kod	Nazwa Implementacji	Temat Konspektu-scenariusza	Środowisko: zagadnienia
Moduły dla szkół podstawowych			
01.1	Instrukcje	Wprowadzenie do środowiska SRU	01 - SRU, JAlbum, TuxPaint: obróbka zdjęć, grafika
01.2	Album zdjęć	Album - wspomnienia z wakacji	
01.3	Reklama	Tworzenie reklamy ośrodka wczasowego	
02.1	Diagramy kolumnowe	Oszczędzaj energię elektryczną	02 - Calc, Firefox, Google: tabele, diagramy, Internet
02.2	Diagramy kołowe	Zdrowy tryb życia	
02.3	Internet	Wyszukiwanie informacji w sieci	
03.1	LED	Programowa sygnalizacja diodą LED	03 - S4A, Arduino: sygnalizacja, sterowanie, losowanie
03.2	Sterowanie klawiaturą w S4A	Sterowanie diody LED z klawiatury	
03.3	Wybór losowy	Losowy czas świecenia diody LED	
03.4	Button	Wyświetlanie stanu przycisku Button	03 - S4A, Arduino: wyświetlanie, przełączanie, regulacja
03.5	Button i LED	Przełączanie diody LED przyciskiem	
03.6	Jasno i ciemno	Programowa regulacja jasności diody	

Moduły dla szkół gimnazjalnych

A1.1	Poszukiwanie skarbów	Wprowadzenie do środowiska SRU	A1 - Linux, SRU: awatary, graffiti, algorytm
A1.2	Obraz wektorowy	Tworzenie awatarów	
A1.3	Graffiti	Implementacja Graffiti	
A1.4	Stworzenie algorytmu map myśli	Algorytm na mapie myśli	
A2.1	Robot biedronka	Animacja biedronki	A2 - Scratch: animacja, gra, labirynt, konwersja
A2.2	Animacja, gra logiczna - zgadywanka	Moja własna gra komputerowa	
A2.3	Labirynt	Labirynt interaktywny	
A2.4	Liczyć jak komputer	Różne systemy liczbowe	
A3.1	Środowisko Scratch S4A i moduł-interfejs	Scratch i obsługa modułu-interfejsu	A3 - S4A, Arduino: interfejs, czujnik, wskaźnik, gra
A3.2	Pomiar temperatury	Pomiar temperatury i wizualizacja RGB	
A3.3	Układ pomiarowy Arduino	Odczytywanie stanu czujników	
A3.4	Rozszerzenie możliwości S4A	Interakcja modułu-interfejsu i środowiska S4A	

Moduły dla szkół ponadgimnazjalnych niesprofilowanych

B1.1	Szyfrowanie	SRU - Jak chronić ważne informacje	B1 - Linux, SRU, Scratch: szyfrowanie, wizualizacja
B1.2	Wybór drogi	Wizualizacja instrukcji warunkowej	
B1.3	Zakręcona mrówka	Wizualizacja pętli	
B1.4	Magiczny operator	Wizualizacja operatorów przypisania	
B1.5	Zamieszanie z kolorami	Ewaluacja prawej strony wyrażeń	B1 - Scratch: skrypty, wyrażenia, gry, sortowanie
B1.6	Kółko i krzyżyk - Scratch	Rozbudowa gry o strategię blokowania	
B1.7	Gra Pong	Rozbudowa gry Pong o nowe elementy	
B1.8	Animacja – sortowanie	Wybrane algorytmy sortowania	
B2.1	Kółko i krzyżyk - Lazarus	Wprowadzenie do środowiska Lazarus	B2 - Lazarus, Free Pascal: gry, automat, zegar
B2.2	Gra w światełka	Gra w zapalone światełka	
B2.3	Gra w życie	Automaty	
B2.4	Zegar binarny	Zegar	
B2.5	Gra „Papier-kamień-nożyce”	Losowanie i porównywanie	B2 - Lazarus, Free Pascal: gry, algorytmy, strategie
B2.6	Układanka alfabetyczna	Rozrzucanie i porządkowanie	
B2.7	Wieże Hanoi	Odkrywanie i stosowanie algorytmu	
B2.8	Gra logiczna NIM	Namiastka sztucznej inteligencji	
B3.1	Środowisko mikrokontrolera	Funkcjonalność modułu-interfejsu	B3 - Arduino IDE: terminal, przetwornik, pomiar, gra
B3.2	Pomiar oświetlenia	Sterowanie z użyciem fotorezystora	
B3.3	Termometr cyfrowy	Pomiar temperatury i wyświetlacz LCD	
B3.4	Pamięć i zręczność	Interakcja człowiek - moduł-interfejsu	
B3.5	Środowisko Scratch S4A i moduł-interfejs	Możliwości S4A i modułu-interfejsu	B3 - S4A, Arduino: interfejs, regulacja, sygnalizacja
B3.6	Przetwornik analogowo-cyfrowy	Działanie fotorezystora i potencjometru	
B3.7	Funkcjonalność modułu-interfejsu	Środowisko mikrokontrolera	
B3.8	Obsługa wyświetlacza	Termometr cyfrowy	

Moduły dla szkół sprofilowanych infotechnicznie - język C

C1.1	Nauka języka C - zmienne	Struktura programu, zmienne oraz typy danych	C1 - język C: zmienne, wyrażenia, pętle
C1.2	Nauka języka C - wyrażenia *if-else*	Wyrażenia warunkowe *if-else*	
C1.3	Nauka języka C - wyrażenie *switch*	Wrażenie warunkowe *switch{case:. . .;}*	
C1.4	Nauka języka C - pętle *while*	Pętle - *while()* i *do {} while()*	
C2.1	Nauka języka C - tablice i pętla *for*	Tablice i pętla *for()*	C2 - język C: tablice, funkcje, struktury, wskaźniki
C2.2	Nauka języka C - funkcje	Funkcje	
C2.3	Nauka języka C - struktury i unie	Struktury i unie	
C2.4	Nauka języka C - wskaźniki	Wskaźniki	
C3.1	Budowa układów i programowanie modułu	Środowisko mikrokontrolera	C3 - Arduino: tranzystor, hallotron, zegar
C3.2	Zwiększenie możliwości wyjścia cyfrowego	Tranzystor NPN	
C3.3	Budowa, działanie i zastosowanie hallotronu	Pole magnetyczne	
C3.4	Pomiar czasu i wyświetlacz LCD	Zegar	

Moduły dla szkół sprofilowanych infotechnicznie - język Python

C1.1	Nauka języka Python - zmienne	Zmienne, ich typy i dane	C1 - język Python: zmienne, wyrażenia, pętle, dane
C1.2	Nauka języka Python - wyrażenia warunkowe	Wyrażenia warunkowe	
C1.3	Nauka języka Python - pętle *while*	Pętle - *while*	
C1.4	Nauka języka Python - typy danych	Zaawansowane typy danych	
C2.1	Nauka języka Python - pętle *for*	Pętle - *for*	C2 - język Python: pętle, funkcje, wyjątki, klasy
C2.2	Nauka języka Python - funkcje	Funkcje	
C2.3	Nauka języka Python - wyjątki	Wyjątki	
C2.4	Nauka języka Python - klasy	Klasy	
C3.1	Wprowadzenie do środowiska mikrokontrolera	Wykorzystanie wejścia analogowego	C3 - Arduino: omomierz, sterowanie, termometr
C3.2	Budowa prostego omomierza	Pomiar wartości opornika	
C3.3	Sterowanie układem z interfejsu Arduino IDE	Kontrola diody RGB	
C3.4	Prezentacja pomiarów	Prezentacja pomiaru temperatury na RGB	

Moduły dla szkół sprofilowanych infotechnicznie - HTML, JavaScript

C1.1	Podstawy HTML	Podstawy HTML	C1 - HTML, CSS, JavaScript, formularz
C1.2	CSS i box model	CSS i box model	
C1.3	Formularze HTML	Formularze HTML	
C1.4	JavaScript	JavaScript	
C2.1	Podstawy jQuery	Podstawy jQuery	C2 – jQuery, lista, baza danych, Canvas
C2.2	jQuery - zdarzenia i efekty	jQuery - zdarzenia i efekty	
C2.3	Przechowywanie danych	Przechowywanie danych	
C2.4	Canvas	Canvas	
C3.1	Wykorzystanie wejścia analogowego	Środowisko mikrokontrolera	C3 - Arduino: sterowanie, dalmierz, serwo
C3.2	Protokół komunikacyjny 1-wire	Protokół komunikacyjny 1-wire	
C3.3	Ultradźwiękowy pomiar odległości	Pomiar i odczyt odległości na LCD	
C3.4	Sterowanie serwomechanizmem	Sterowanie elementami wykonawczymi	