02

Geneza i zawartość Programu nauczania-uczenia się infotechniki

Krzysztof Wawrzyniak, Stanisław Ubermanowicz

U podstaw Projektu innowacyjno-testującego, mającego na celu poprawę jakości specjalistycznego kształcenia informatycznego i mechatronicznego (zwanego *infotechnicznym*, IT), legło szereg problemów wymagających interwencji systemowej. Nadrzędnym zadaniem projektowym było zwiększenie zainteresowania przyszłymi wyborami kierunków technicznych w ścieżkach rozwojowych uczniów. W realizacji tak specyficznego zadania, mającego dawać **efekty odroczone**, kluczowe było zdiagnozowanie przyczyn złego stanu rzeczy, określenie sposobów naprawy oraz opracowanie, przetestowanie, upowszechnienie i włączenie do polityki oświatowej narzędzi naprawczych. W rezultacie trwających ponad trzy lata działań wypracowano innowacyjną taktykę o nazwie „Strategia Wolnych i Otwartych Implementacji" (SWOI), która obejmuje całościowo aspekty merytoryczne, dydaktyczne i organizacyjne form zajęć pozalekcyjnych, specyficzną metodykę ich realizacji, środki służące osiąganiu i dokumentowaniu efektów oraz kwestie doskonalenia kompetencji kadr.

Zasadniczym narzędziem w Strategii SWOI jest niniejszy Program nauczania-uczenia się infotechniki. Zakłada on celowe, wczesne oddziaływanie na adolescentów poprzez ciekawe treści zajęć i silnie motywujące formy ćwiczeniowe na kołach zainteresowań IT. Z założenia ma to ukształtować trwałą chęć pogłębiania wiedzy i umiejętności poprzez pracę własną oraz wybór dalszych etapów kształcenia w kierunku infotechniki. O tym, jak bardzo ważne jest pobudzenie poprzez ciekawe treści i atrakcyjne formy, świadczy olbrzymie powodzenie oferty Centrum Nauki Kopernik czy oblegane prezentacje podczas Nocy Naukowców. Przyjrzyjmy się, jakie są przyczyny niechętnego nastawienia do tradycyjnej oferty proponowanej w szkole.

Konieczność zmian w edukacji infotechnicznej

Z pogłębionej diagnozy aktualnego sposobu realizacji materiału z dziedzin informatyki i technologii informacyjnych w szkolnictwie ogólnym wynika, że kształcenie na wszystkich etapach obraca się głównie w zakresie użytkowania gotowych, zamkniętych aplikacji. Jest to oczywiście ważne dla przygotowania ogółu adolescentów do posługiwania się komputerem jako narzędziem pracy. Jednakże potrzeby rynku pracy są coraz większe i nie wystarcza już umiejętność wykorzystania pakietu biurowego. Potrzebne są liczne

kadry o umiejętnościach programistycznego tworzenia implementacji, a przynajmniej dostosowywania aplikacji do potrzeb pracodawcy, np. pisanie makroprogramów czy formuł automatyzujących działania bądź wydobywanie optymalnych kwerend z baz danych.

Takich kompetencji powszechna oświata w zasadzie nie kształtuje. Jeszcze gorzej jest z wiedzą i umiejętnościami z obszarów mechatroniki, a przecież mikroprocesory są dziś w bardzo wielu urządzeniach powszechnego użytku i w maszynach produkcyjnych. Wysoce specjalistyczne kompetencje muszą być formowane odpowiednio **wcześnie i długofalowo**, aby rzeczywiście posiąść wiedzę i umiejętności, jakie są potrzebne w zinformatyzowanym świecie. Na wyuczenie się i osiąganie odpowiedniego poziomu kompetencji dopiero na studiach jest już za późno, wskutek czego tak wielu studentów odpada, a ci, którzy kończą wyższe szkoły techniczne, mają zdecydowanie niewystarczające kwalifikacje.

W okresie studiów jest za późno na nadganianie zaległości kompetencyjnych, których nie ukształtowano w systemie oświaty.

Aktualne wady edukacji komputerowej, wymagające naprawy to m.in.:
- nastawienie przede wszystkim na użytkowanie pakietów biurowych;
- nikła wiedza o alternatywnych systemach, programach i narzędziach;
- brak wczesnego kształtowania twórczych kompetencji infotechnicznych;
- brak ukierunkowywania uczniów na pożądane ścieżki specjalizacji;
- uczenie programowania w imperatywnej metodyce pisania kodu;
- pisanie kodu w trybie tekstowym, bez wsparcia narzędzi z widżetami;
- nieobecność problematyki mechatronicznej w programach szkolnych.

Skutkami tak niekorzystnego stanu rzeczy są następujące zjawiska:
- niska zdawalność egzaminów na stopień technika w obszarach IT;
- mała wybieralność i słaby poziom wyników z informatyki na maturze;
- niewielkie zainteresowanie kierunkami IT, zwłaszcza uczennic;
- niewystarczające przygotowanie do studiowania na kierunkach IT;
- liczne rezygnacje studentów nieradzących sobie z treściami studiów;
- niedobór odpowiednio wykształconej kadry do zawodów IT;
- brak umiejętności obsługi systemów WiOO używanych w firmach;
- bardzo niski poziom wynalazczości w Polsce.

Jak już wspomniano – środkiem zaradczym może stać się wypracowana i upowszechniona Strategia SWOI, zakładająca nie tylko modernizację treści, form i metod kształcenia, lecz również przyjęcie innych akcentów w podstawach ideowo-programowych, w zakładanych celach i efektach, a także stosowanie innej taktyki oddziaływań.

W pierwszym rzędzie chodzi o uspójnienie edukacji szkolnej i pozaszkolnej, o wykorzystanie kół zainteresowań jako najefektywniejszej formy specjalizacji, o zachęcenie do uczestnictwa uczniów/uczennic niepewnych co do swych możliwości, o formowanie trwalszych cech osobowościowych i umiejętności twórczych, a nie przekazywanie ulotnych wiadomości, o ocenianie rzeczywiste osiągnięć, a nie sprawdzające zapamiętanie.

Obszerne opisy taktyki racjonalizacyjnej, mechanizmów, uwarunkowań, metod i narzędzi realizacji zawarte są w obu tomach publikacji. Tutaj skoncentrujmy się na sposobie wykorzystania Programu nauczania-uczenia się, na jego celach, treściach i spodziewanych efektach.

Dostosowanie do potrzeb i dyspozycji uczniów

Program jest elastyczny pod względem doboru alternatywnych treści, wyznaczanego stopnia trudności zadań i oczekiwanych efektów.

Proponowany Program nauczania-uczenia się w swojej istocie zakłada aktywność i zaangażowanie w jego rzeczywistą realizację wszystkich uczestników – chodzi o prawdziwe upodmiotowienie trenerów i uczniów. To właśnie oni poprzez akt sprawczy praktycznego wykonywania czynności stają się współautorami programu, choćby poprzez wybory modułów, modyfikację zadań, dostosowywanie do własnych potrzeb, do osobistych preferencji i warunków pracy. Zyskana dzięki temu swoboda umożliwia efektywniejsze osiąganie celów kształcenia, pozostawiając trenerowi możliwość elastycznego podążania za uczniem i dostosowywania procesu kształcenia do indywidualnych oczekiwań wychowanków.

Program ten, jako swoisty wzorzec, ma również na celu zachęcanie nauczycieli do opracowywania różnych innych ofert edukacyjnych, z dopasowaniem do potrzeb i zainteresowań uczniów oraz włączania ich do planowania i realizacji zajęć. Takie wyjście naprzeciw wychowankom jest szansą na zwiększenie odpowiedzialności i zaangażowania uczniów do współtworzenia programów nauczania-uczenia się, a przez to – aktywniejszego uczestnictwa w zajęciach i większej motywacji do pracy, przekładających się na kształtowanie pasji i wiązanie dalszego kształcenia zgodnego z własnymi zainteresowaniami.

Realizacja Programu ma szansę być w pełni satysfakcjonująca dla trenera i uczniów pod warunkiem jego dynamicznego dostosowywania do konkretnych sytuacji i kontekstów edukacyjnych. Oznacza to pożytki z podjęcia przez trenera wskazanych poniżej działań:

- wybór tematyki adekwatnej do formowanych umiejętności i potrzeb uczniów;
- ułożenie sekwencji modułów w cykl spójny środowiskowo i tematycznie;
- modyfikowanie założonych celów i odpowiadających im osiągnięć ucznia;
- skonsultowanie propozycji z uczniami, zachęcenie ich do modyfikacji;
- dobranie form i metod pracy z uczniami stosownie do treści i faz realizacji.

W przypadkach daleko idących przeróbek warto pamiętać, że Program jest dorobkiem grona specjalistów, testowanym w licznych i zróżnicowanych środowiskach, optymalizowanym na podstawie wniosków formatywnych z obserwacji zajęć i ewaluacji efektów. Z tego względu stanowi on wart wykorzystania lub naśladowania wzorzec, a ewentualne modyfikacje zmierzające w stronę dotychczasowych przyzwyczajeń, sprzecznych z zaleceniami naprawczymi, zniweczyłoby cały wysiłek autorów i nie dałoby oczekiwanych efektów.

Cele ogólne kół zainteresowań infotechnicznych

W zajęciach pozalekcyjnych nie ma w zasadzie obowiązku, aby przygotować Program realizujący zapisy Podstawy programowej. Niemniej jednak możemy właśnie w niej znaleźć pewne wskazania co do celów, materiału nauczania i sposobów ich ujmowania. Analizując Podstawę programową kształcenia ogólnego, można trafić na zapis: „Celem kształcenia ogólnego na III i IV etapie edukacyjnym jest:

1. przyswojenie przez uczniów określonego zasobu wiadomości na temat faktów, zasad, teorii i praktyk;
2. zdobycie przez uczniów umiejętności wykorzystania posiadanych wiadomości podczas wykonywania zadań i rozwiązywania problemów;
3. kształtowanie u uczniów postaw warunkujących sprawne i odpowiedzialne funkcjonowanie we współczesnym świecie”.

Cele ogólne kół zainteresowań IT obejmują na równi formowanie wszystkich trzech sfer: poznawczą, wykonawczą i doznaniową.

Poniższe zestawienie celów ogólnych zawartych w proponowanym tu Programie nauczania-uczenia się dotyczy równomiernego formowania wszystkich trzech sfer: reprezentacji poznawczych (wiadomości), sfery wykonawczej (umiejętności) i doznaniowej sfery afektywnej (postawy). W konkretnym odniesieniu do obszarów infotechniki, są to:

Wiadomości

- Przyswojenie zagadnień z informatyki, algorytmiki, elektroniki i mechaniki;
- Znajomość kluczowych pojęć infotechnicznych i ich czynnościowe rozumienie;
- Znajomość istoty i pożytków wolnego systemu operacyjnego i otwartych aplikacji;
- Wiedza o złożonych i różnorodnych procesach tworzenia implementacji;
- Poznanie środowisk programowania i konstruowania układów mechatronicznych;
- Znajomość zasad i procedur osiągania bezpieczeństwa danych i szyfrowania.

Umiejętności

- Skuteczne i sprawne poruszanie się w systemie Linux i w środowisku Ubuntu;
- Umiejętności korzystania z narzędzi wolnego i otwartego oprogramowania;
- Umiejętności tworzenia implementacji i doskonalenia ich działania;
- Planowanie, podejmowanie i organizacja pracy w środowisku programistycznym;
- Projektowanie graficzne, obiektowo-skryptowe i wizualno-obiektowe;
- Programowanie imperatywne, wsadowe i obiektowo-zdarzeniowe;
- Konstruowanie i oprogramowywanie układów mechatronicznych.

Postawy

- Chęć ciągłego zdobywania i modyfikowania wiedzy i umiejętności;
- Przekonanie o potrzebie formowania cech logicznego myślenia i planowania;
- Motywacja do twórczego podejścia w implementowaniu i doskonaleniu dzieł;
- Uznanie dla znaczenia stylu partnerstwa z innymi uczniami oraz z trenerem;
- Przekonanie o znaczeniu interakcji w grupie podczas fazy projektowania;

- Wytrwałość w samodzielnym tworzeniu prawidłowo działających implementacji;
- Staranność działań w środowisku programistycznym i mechatronicznym.

Warto zaznaczyć, że cele ogólne Programu odzwierciedlają przede wszystkim dążenia do osiągnięcia stanu, który jest głównym zadaniem, czyli – **zwiększenia zainteresowania** wyborem kierunków infotechnicznych w dalszym kształceniu. Szczególny nacisk został położony na kształtowanie względnie trwałych postaw, przede wszystkim w warstwie wolicjonalnej.

Odniesienie Programu do Podstaw programowych

Analiza celów ogólnych Programu nauczania-uczenia się infotechniki ukazała, iż zasadniczo realizują one zapisy Podstawy programowej kształcenia ogólnego dla III i IV etapu edukacji, obowiązującej w polskim systemie oświaty. Również treści znajdują swoje odzwierciedlenie w szczegółowych podstawach przedmiotów szkolnych, co ukazuje poniższe zestawienie:

Podstawa programowa gimnazjum: informatyka	Program nauczania-uczenia się (moduły A)
Bezpieczne posługiwanie się komputerem i jego oprogramowaniem, korzystanie z sieci komputerowej. Uczeń: • posługuje się urządzeniami multimedialnymi, na przykład do nagrywania/odtwarzania obrazu i dźwięku; • wyszukuje i uruchamia programy, porządkuje i archiwizuje dane i programy; stosuje profilaktykę antywirusową; • samodzielnie i bezpiecznie pracuje w sieci lokalnej i globalnej;	Praca ze Szkolnym Remiksem Ubuntu – dedykowanym systemem operacyjnym i oprogramowaniem narzędziowym; praca z programami graficznymi Gimp i Inkscape, tworzenie grafiki komputerowej; praca w środowisku wizualno-skryptowym Scratch, projektowanie i programowanie, tworzenie animacji; praca w Serwisie e-Swoi, dedykowanym do zdalnej formy edukacji; poszukiwanie i pobieranie z Internetu wolnych i otwartych zasobów źródłowych na licencji Creative Commons
Wyszukiwanie i wykorzystywanie (gromadzenie, selekcjonowanie, przetwarzanie) informacji z różnych źródeł; współtworzenie zasobów w sieci. Uczeń: • pobiera informacje i dokumenty z różnych źródeł, w tym internetowych, ocenia pod względem treści i formy ich przydatność do wykorzystania w realizowanych zadaniach i projektach; • umieszcza informacje w odpowiednich serwisach internetowych.	Praca w Serwisie e-Swoi, dedykowanym do zdalnej formy edukacji; praca z e-Repozytorium przeznaczonym do gromadzenia, przechowywania i udostępniania zasobów źródłowych i własnych implementacji; praca z e-Portfolio przeznaczonym do opisywania działań i dokumentowania własnych osiągnięć ucznia
Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego. Uczeń: • wyjaśnia pojęcie algorytmu, podaje odpowiednie przykłady algorytmów rozwiązywania różnych problemów; • formułuje ścisły opis prostej sytuacji problemowej, analizuje ją i przedstawia rozwiązanie w postaci algorytmicznej; • wykonuje wybrane algorytmy za pomocą komputera.	Praca z algorytmem w programie Dia, opanowanie podstaw myślenia algorytmicznego, programowanie w środowisku wizualno-skryptowym Scratch – tworzenie skryptów obsługujących obiekty, budowanie kodu źródłowego, wizualizującego i realizującego algorytm w środowisku graficznym

Podstawa programowa gimnazjum: technika	Program nauczania-uczenia się (moduły A)
Opracowywanie koncepcji rozwiązań typowych problemów technicznych oraz przykładowych rozwiązań konstrukcyjnych	Praca z mikroprocesorowym modułem-interfejsem Arduino, montowanie układów mechatronicznych
Podstawa programowa liceum: informatyka	**Program nauczania-uczenia się (moduły B)**
Uczeń wykorzystuje technologie komunikacyjno-informacyjne do komunikacji i współpracy z nauczycielami i innymi uczniami, a także z innymi osobami, jak również w swoich działaniach kreatywnych. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego. Uczeń: • prowadzi dyskusje nad sytuacjami problemowymi; • formułuje specyfikacje dla wybranych sytuacji problemowych; • projektuje rozwiązanie: wybiera metodę rozwiązania, odpowiednio dobiera narzędzia komputerowe, tworzy projekt rozwiązania; • realizuje rozwiązanie na komputerze za pomocą oprogramowania aplikacyjnego lub języka programowania; • testuje otrzymane rozwiązanie, ocenia jego własności, w tym efektywność działania oraz zgodność ze specyfikacją;	Praca w Serwisie e-Swoi, praca nad implementacjami w zespołach projektowych, zadania konkursowe i zadania na kołach, praca w środowisku wizualno-skryptowym Scratch – tworzenie skryptów obsługujących obiekty, budowanie kodu źródłowego, wizualizującego i realizującego algorytm w środowisku graficznym; praca w środowisku programowania wizualno-obiektowego Lazarus (tworzenie, uzupełnianie i komentowanie kodu), mechatronika, programowanie i konstruowanie, projektowanie, tworzenie i testowanie implementacji
Podstawa programowa technikum: informatyka	**Program nauczania-uczenia się (moduły C)**
Komunikowanie się za pomocą komputera i technologii informacyjno-komunikacyjnych. Uczeń: • wykorzystuje zasoby i usługi sieci komputerowych w komunikacji z innymi użytkownikami, w tym do przesyłania i udostępniania danych; • bierze udział w dyskusjach w sieci (forum internetowe, czat). • sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu i uruchamianiu programów; • stosuje podstawowe konstrukcje programistyczne w wybranym języku programowania, instrukcje iteracyjne i warunkowe, rekurencję, funkcje i procedury, instrukcje wejścia i wyjścia • poprawnie tworzy strukturę programu • realizuje indywidualnie lub zespołowo projekt programistyczny	Zaawansowana praca w środowisku Linux i w Szkolnym Remiksie Ubuntu (dedykowany system operacyjny i oprogramowanie narzędziowe); edukacja zdalna i współpraca w społeczności sieciowej Serwis e-Swoi (dedykowana do zajęć platforma edukacyjna); programowanie w językach C, Python, HTML5 i JavaScript, montowanie układów mechatronicznych; praca w grupie nad zadaniami programistycznymi

Należy w tym miejscu jednak wyraźnie zaznaczyć, że Program, który jest przeznaczony na zajęcia pozalekcyjne w formie kół zainteresowań i aktywności pozaszkolnej, zasadniczo nie ma na celu realizacji zapisów Podstawy programowej. Nawiązując do zakresów tematycznych Podstawy, każdorazowo je przekracza, oferując treści spoza tych, jakie w szkolnictwie ogólnym obowiązują wszystkich uczniów. Jego celem nadrzędnym jest uzupełnienie kształcenia powszechnego o formowanie pogłębionych, twórczych umiejętności infotechnicznych, które są konieczne dla wczesnego ukierunkowania specjalistycznego.

Program realizuje założenia Podstawy, lecz przekracza ją zakresem i poziomem trudności, dlatego nie powinien być realizowany dla wszystkich uczniów.

Opis modułów i oczekiwane efekty merytoryczne

Proponowane w Programie treści zostały podzielone na elementarne moduły, zgrupowane w bloki merytoryczne i ścieżki programowe o różnych poziomach trudności. **Moduły** to domknięte treściowo i organizacyjnie jednostki dydaktyczne kół zainteresowań IT, zaplanowane na 90 minut w szkołach gimnazjach i szkołach ponadgimnazjalnych, a w szkołach podstawowych na 45 minut. W tym czasie realizowane są pojedyncze Konspekty-scenariusze, połączone ściśle z wykonywaniem przypisanych im Implementacji – w pełni funkcjonalnych wytworów programistycznych lub mechatronicznych. Z tego powodu niedopuszczalne jest dzielenie modułów i przenoszenie części materiału na następne zajęcia. Całościowe wykonanie implementacji zaplanowanych na daną jednostkę jest konieczne i realne, co potwierdziło testowanie w 110 szkołach o różnych poziomach i profilach. Natomiast w niektórych Konspektach wskazano także część zadań rozszerzających, przeznaczonych do samodzielnego wykonania przez uczniów już poza zajęciami.

Dany moduł powinien być w całości zrealizowany na jednym spotkaniu, podczas którego musi powstać zaplanowana implementacja.

Cechą układu modułowego jest możliwość w miarę dowolnego układania Planu i doboru treści zajęć przez trenerów. Warto przy tym wybierać jako ciągłość treściową przynajmniej cztery moduły stanowiące spójny środowiskowo **blok merytoryczny**. W taki sposób pogrupowano w niniejszej publikacji Konspekty-scenariusze.

Na pełne ścieżki programowe kół zainteresowań IT – oznaczone adekwatnie do poziomu trudności kodami 0, A, B i C – składają się po trzy bloki merytoryczne, dając materiał dydaktyczny na minimum 12 spotkań w formie stacjonarnej. Realizowane zagadnienia powinny obejmować zarówno projektowanie i programowanie, jak też konstruowanie układów mechatronicznych.

Celowym zabiegiem motywująco-waloryzacyjnym jest taka sekwencja, w której najpierw wprowadza się tematykę użytkowania systemu i aplikacji Szkolnego Remiksu Ubuntu, w tym edycję i animację grafiki komputerowej (z wyjątkiem szkół sprofilowanych IT). Dopiero w drugiej części cyklu zajęć realizuje się najtrudniejsze dla początkujących treści, związane z uczeniem się programowania. Zwieńczeniem cyklu jest blok zajęć mechatronicznych, które sprawiają uczniom największą satysfakcję na wszystkich etapach edukacji.

Pakiet Szkolnego Remiksu Ubuntu wraz z kompletnym oprogramowaniem potrzebnym do zajęć, znajduje się na dołączonej do książki płycie.

Przeznaczona na poziom **szkół podstawowych** ścieżka 0 zawiera głównie podstawy użytkowania aplikacji w środowisku Ubuntu, natomiast wprowadzona tam namiastka budowy kodów źródłowych w języku jawnym (po polsku), polegająca na układaniu skryptów w formie puzzli w środowisku graficznym Scratch S4A, służy najmłodszym uczniom wyłącznie do oprogramowywania modułu-interfejsu Arduino. Ze względu na bardzo wysokie walory edukacyjne środowiska Scratch, jest ono zalecane również dla starszych uczniów w niektórych modułach

Do mechatroniki potrzebny jest układ Arduino z mikrokontrolerem Atmega oraz

A, B i C. W przypadkach uczniów o szczególnych zdolnościach w podstawówkach, trener może podjąć z nimi próbę realizacji także wybranych modułów ze ścieżek A1, A2 i A3.

zestaw montażowy z podzespołami elektronicznymi.

Na poziom **szkół gimnazjalnych** zaleca się realizację wszystkich dość łatwych modułów ze ścieżki A. Środowiskiem projektowania są tam aplikacje do tworzenia grafiki komputerowej w systemie Linux z Ubuntu, a narzędziem programowania wizualno-skryptowego jest Scratch. Ta aplikacja narzędziowa, wzbogacona o bibliotekę S4A, służy także do oprogramowywania mikrokontrolera i sterowania podzespołami tworzącymi układ elektroniczny na płytce montażowej. Dla zaawansowanych gimnazjalistów zaleca się próbę realizacji modułów średnio trudnych B1 i B3, bądź też nieco trudniejszych B2 w zintegrowanym środowisku Lazarus, opartym na języku FreePascal.

Na poziom **szkół ponadgimnazjalnych** niesprofilowanych, czyli z wyłączeniem liceów i techników o profilu zbieżnym z infotechniką, polecana jest ścieżka z wybranymi blokami modułów B. Część z tych modułów dotyczy pracy w systemie Linux i w aplikacjach narzędziowych SRU, alternatywne moduły wykorzystują środowisko Scratch, a zasadniczym narzędziem programowania obiektowo-zdarzeniowego jest Lazarus z językiem FreePascal. Język ten (bez GUI) jest dopuszczony na egzaminie maturalnym, dlatego pisanie w nim kodów źródłowych przydaje się jako wprawka do matury z informatyki. Ponadto zarówno Scratch, jak i Lazarus dają implementacje przenośne na inne platformy systemowe. W bloku B są też alternatywne moduły B3 – jedne wykorzystują do oprogramowywania Arduino prosty interfejs IDE, a drugie wizualno-skryptowy interfejs S4A.

Dla **szkół sprofilowanych** infotechnicznie zaplanowano trzy alternatywne bloki modułów C – z językami programowania C, Phyton lub JavaScript. Ten ostatni powiązany jest z rozszerzaniem możliwości stron w HTML, tworzeniem arkuszy stylów CSS i wykorzystaniem biblioteki jQuery. Trudną ścieżkę programistyczną C można realizować dla uczniów, którzy mają w programie szkolnym dany język lub jako rozszerzenie w sytuacji, gdy w szkole taki język bądź treści nie są nauczane. Ze względu na to, że moduły z języków C i Python oparte są na stylu programowania imperatywnego, nie poleca się tych modułów dla szkół o innych profilach niż informatyczne (bądź pochodne). Natomiast z tego poziomu bloki mechatroniczne C3, choć dość trudne, to jednak są najbardziej interesujące, zatem można je polecić także do klas niesprofilowanych w liceach ogólnokształcących.

Prezentowany w poniższej tabeli **opis modułów** ma na celu uzyskanie wstępnej orientacji w zakresie tematyki modułów i realizowanych treści oraz założonych efektów merytorycznych. *Efektami merytorycznymi* nazywamy tu zbiór obserwowanych na kołach i ocenianych bezpośrednio wskaźników osiągania celów szczegółowych, dotyczących przedmiotowej wiedzy i umiejętności z zakresu szeroko rozumianej infotechniki. Są one zaliczane do grupy tzw. *efektów twardych*, przy czym pamiętać należy, że na kołach zainteresowań równie ważne jest formowanie kompetencji społecznych i osiąganie tzw. *efektów miękkich* (zmian świadomości, postaw, cech wolicjonalnych), co jest mierzone w procesie ewaluacji.

Paradoksalnie efekty merytoryczne, zwane „twardymi", szybciej się dezaktualizują i są mniej trwałe niż uformowane efekty „miękkie", oznaczające pożądane cechy osobowości.

Warto podkreślić, że sam proces dochodzenia do wiedzy i umiejętności na kołach zainteresowań powinien być postrzegany jako ważniejszy od określania precyzyjnych efektów kształcenia i nastawienia na ich osiąganie. Stąd tak ważna jest rola trenera – bacznego obserwatora pracy uczniów – dla którego najistotniejsze będzie podążanie za każdym indywidualnie uczniem i za jego potrzebami.

Treści dla szkół podstawowych

MODUŁY 0		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Wprowadzenie do środowiska Szkolnego Remiksu Ubuntu	Podstawowe informacje o komputerach i systemie Linux, operacje na aplikacjach i plikach; higiena pracy z komputerem.	• omawia pojęcia: *komputer, system, urządzenia peryferyjne;* • wykorzystuje podstawowe funkcje obsługi Linuksa; • rozpoznaje funkcje ikon związanych z oprogramowaniem; • korzysta z różnych sposobów bezpiecznego kopiowania plików i zakładania katalogów w Linuksie.
Album zdjęć – wspomnienia z wakacji	Wybór zdjęć do reportażu, wykadrowane i retusz w programie Gimp. Opisanie zdjęć i utworzenie albumu jako strony WWW w programie JAlbum.	• wskazuje i opisuje podstawowe sposoby i metody obróbki fotografii cyfrowej; • wykorzystuje podstawowe narzędzia do grafiki rastrowej; • dokonuje świadomej selekcji zdjęć i porządkuje zbiór fotografii cyfrowych w formie albumu zdjęć; • publikuje zdjęcia jako album na stronie WWW.
Tworzenie reklamy ośrodka wczasowego	Praca z edytorem graficznym TuxPaint i tekstowym LibreOffice Writer, tworzenie napisów w FontWork, łączenie grafiki z tekstem.	• omawia podstawowe pojęcia związane z edytorami tekstu i grafiki, a także formatami plików; • wykorzystuje podstawowe funkcje edytora tekstów; • wstawia do edytora tekstów elementy graficzne; • pozyskuje ozdobne czcionki i wykorzystuje FontWork.
Prezentacja danych do kampanii „Oszczędzaj energię elektryczną”	Tworzenie formuł w arkuszu kalkulacyjnym i diagramów słupkowych na podstawie danych o miesięcznym zużyciu prądu w rodzinach.	• omawia sposoby gromadzenia danych; • obsługuje arkusz kalkulacyjny Calc LibreOffice; • wpisuje poprawne formuły obliczeniowe; • generuje odpowiedni wykres kolumnowy; • prawidłowo formatuje wykres.
Prezentacja danych do kampanii „Zdrowy tryb życia”	Tworzenie formuł w arkuszu kalkulacyjnym i diagramów kołowych na podstawie danych z Ankiety nt.: Co pijesz na śniadanie?	• zbiera dane ankietowe i wprowadza je do odpowiedniej tabeli; • wpisuje formuły w arkuszu kalkulacyjnym Calc; • generuje wykresy z wykorzystaniem kreatora; • prawidłowo formatuje wykresy; • analizuje informacje zawarte na wykresie.

Wyszukiwanie informacji w sieci	Praca z przeglądarkami Internetu i serwisami wyszukującymi: Mozilla Firefox + Web Of Trust, Midori, Chromium, Google, IxQuick, Wikipedia	• obsługuje przeglądarkę WWW; • omawia pojęcia dotyczące sieci komputerowych, związane z wyszukiwaniem i śladami w Internecie; • potrafi wskazać i użyć różne wyszukiwarki; • odróżnia reklamę od wartościowych wyników wyszukiwania.
Programowa sygnalizacja diodą LED	Tworzenie kodu do włączania diody LED na płytce Arduino w środowisku wizualno-skryptowym Scratch S4A.	• omawia podstawowe pojęcia: *mikrokontroler, dioda elektroluminescencyjna*; • podłącza i obsługuje moduł-interfejs Arduino; • korzysta z funkcji obsługi wyjść w programie Scratch S4A; • tworzy kod włączający sygnalizację diodą LED.
Sterowanie diody LED z klawiatury	Tworzenie w środowisku Scratch S4A kodu do sterowania diodą za pomocą klawiatury.	• omawia pojęcia: *symbole matematyczne, sterowanie*; • montuje układ z wykorzystaniem płytki Arduino; • korzysta z funkcji obsługi wejść w programie Scratch S4A; • tworzy kod do sterowania diodą LED z klawiatury.
Losowy czas świecenia diody LED	Tworzenie w Scratch S4A kodu do losowego ustawiania czasu świecenia diody LED w Arduino.	• omawia pojęcia: *kod, zmienne, losowanie*; • wykorzystuje funkcje losowania z programu Scratch S4A; • deklaruje typy zmiennych, definiuje i przypisuje im wartości; • tworzy kod sterujący zmiennym świeceniem diody LED.
Wyświetlanie stanu przycisku Button	Tworzenie w Scratch S4A kodu ukazującego na ekranie tekst zależny do zmiany stanu na wejściu cyfrowym Arduino.	• omawia pojęcia: *button, opornik, wejście/wyjście cyfrowe*; • korzysta z funkcji obsługi *wej-wyj* programu Scratch S4A; • montuje i uruchamia układy na podstawie schematów; • tworzy kod do obsługi i wizualizacji stanu włącznika.
Przełączanie diody LED przyciskiem	Montowanie układu i tworzenie w Scratch S4A kodu sterującego diodą LED za pomocą przycisku.	• omawia pojęcia związane z przełączaniem stanu; • korzysta z funkcji obsługi *wej-wyj* programu Scratch S4A; • montuje układy na podstawie schematów i modyfikuje je; • tworzy kod do sterowania diodą LED poprzez button.
Programowa regulacja jasności diody LED	Wykorzystanie wyjścia PWM do pulsacyjnego sterowania diodą LED w środowisku Scratch S4A.	• omawia pojęcia związane ze zmianą szerokości impulsów; • korzysta z wyjścia PWM i funkcji programu Scratch S4A; • montuje i uruchamia układ na podstawie schematu; • tworzy kod do sterowania jasnością diody LED.

Treści dla szkół gimnazjalnych

MODUŁY A		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Wprowadzenie do środowiska SRU – poszukiwanie skarbu	Ukształtowanie wiedzy o Szkolnym Remiksie Ubuntu, rozwijanie umiejętności poruszania się w środowisku Linux, zdobycie wiedzy o licencjach Creative Commons (CC)	• omawia pojęcia: *dysk, katalog, plik*; • wykorzysta podstawowe funkcje obsługi Linuksa; • omawia licencje Creative Commons; • korzysta z różnych sposobów kopiowania plików, zakładania katalogów w Linuksie;

		• znajduje w Internecie ciekawy obrazek z zachowaniem licencjonowania; • znajduje ciekawe zdjęcie w Internecie i czcionkę w systemie do dalszego wykorzystania.
Tworzenie awatarów	Tworzenie obiektów wektorowych, porównanie grafiki rastrowej i wektorowej	• omawia podstawowe pojęcia grafiki wektorowej; • wskazuje różnice pomiędzy grafiką wektorową a rastrową; • wykorzystuje podstawowe narzędzia programu do grafiki wektorowej; • tworzy określone obiekty wektorowe (awatar, postać z gry komputerowej etc.); • rozpoznaje własne cechy osobowościowe i odzwierciedla je w awatarze.
Tworzenie graffiti	Tworzenie infografiki komputerowej na potrzeby graffiti. Wyszukiwanie i wykorzystanie czcionek oraz zdjęć na licencjach CC	• omawia podstawowe pojęcia grafiki rastrowej; • wskazuje różnice pomiędzy grafiką wektorową a rastrową; • wykorzystuje podstawowe narzędzia programu do obróbki graficznej; • tworzy określone obiekty wektorowe i łączy je z obrazem rastrowym.
Algorytm na mapie myśli	Praca z mapami myśli, zapoznanie z algorytmami i graficznymi formami ich reprezentacji	• tworzy cztery proste algorytmy w postaci przykładów wizualnych, • omawia na konkretnych przykładach pojęcia: *algorytm, instrukcja warunkowa, pętla, system pozycyjny binarny a dziesiętny, sortowanie bąbelkowe*, • wykorzystuje podstawowe funkcje programu FreeMind i Dia, • angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.
Stworzenie animacji biedronki poruszającej się po linii	Poznanie zasad tworzenia programów komputerowych i animacji za pomocą instrukcji języka programowania	• omawia podstawowe pojęcia algorytmiczne: *skrypt, wyrażenia, algorytm, pętla, wyrażenie warunkowe, zmienna, czujnik, instrukcja*; • wykorzystuje podstawowe narzędzia programu Scratch; • tworzy animację interaktywną; • współpracuje z innymi uczniami oraz z nauczycielem.
Moja własna gra komputerowa	Stworzenie własnej animacji oraz algorytmu wyszukiwania binarnego w programie Scratch	• omawia podstawowe pojęcia *algorytmiczne: skrypt, wyrażenia, algorytm, pętla, wyrażenie warunkowe, zmienna, czujnik, instrukcja*; • wykorzystuje podstawowe narzędzia programu Scratch; • tworzy animację interaktywną; • współpracuje z innymi uczniami oraz z nauczycielem.
Labirynt interaktywny	Stworzenie interaktywnej gry w programie Scratch ze skryptami obsługującymi animację obiektów	• omawia podstawowe pojęcia, takie jak: *pętla, wyrażenie warunkowe, zmienna*; • wykorzystuje podstawowe narzędzia programu Scratch; • korzysta z wcześniej przygotowanego obrazu (labiryntu) w przygotowywanej pracy; • tworzy animację interaktywną; • współpracuje z innymi uczennicami i uczniami oraz z nauczycielem.

Systemy liczbowe – zasady konwersji liczb	Stworzenie interaktywnej prezentacji, pokazującej jak liczone są wartości między różnymi pozycyjnymi systemami liczbowymi, ze szczególnym uwzględnieniem systemu binarnego.	• omawia pojęcia: *podstawa, pozycja, cyfra i liczba, pętla, instrukcja warunkowa, zmienna* • dostrzega, w jaki sposób liczby są prezentowane wewnątrz współczesnych maszyn cyfrowych • omawia różnice i podobieństwa między systemem binarnym a dziesiętnym • tworzy interaktywną prezentację z zakresu liczenia wartości w różnych liczbowych systemach pozycyjnych.
Środowisko Scratch z obsługą modułu-interfejsu	Wprowadzenie w świat mikrokontrolerów na przykładzie modułu-interfejsu Arduino oraz jego obsługa w środowisku Scratch for Arduino (S4A)	• omawia pojęcia: *button, wejście cyfrowe; dioda LED; opornik; pętla; wyrażenie warunkowe; zmienna; mikrokontroler; port USB i RS232*; • montuje i uruchamia przykładowe układy na podstawie schematów; • deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • obsługuje środowisko Scratch S4A i zna jego funkcje; • wykorzystuje podstawowe narzędzia programu Scratch S4A; • prawidłowo podłącza i steruje podzespołami: dioda LED, dioda RGB, przycisk; • angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.
Pomiar temperatury i wizualizacja za pomocą diody RGB	Budowa układu do wizualizacji regulacji jasności i kolorów diody RGB. Pomiar temperatury i prezentacja pomiarów przy wykorzystaniu trzech kolorów diody.	• prawidłowo obsługuje środowisko programowania graficznego Scratch S4A; • samodzielnie montuje i uruchamia przykładowe układy na podstawie schematów; • deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo łączy i odczytuje wskazania czujnika temperatury; • trafnie używa zwrotów: *czujnik, przetwornik A/D, dioda RGB, buzzer, modulacja szerokości impulsu, wejście analogowe*; • angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.
Odczytywanie stanu czujników przez układ pomiarowy z fotorezystorem	Konstruowanie i oprogramowanie układów do odczytu stanu czujników na przykładzie interfejsów z układem Arduino do pomiaru natężenia światła	• zgodnie z zasadami działania podłącza czujnik pomiarowy: fotorezystor; • prawidłowo buduje i oprogramowuje moduł-interfejs wskazujący poziomy oświetlenia; • uruchamia ukazywanie odczytów w środowisku Linux; • modyfikuje i rozbudowuje pomiarowe układy elektroniczne oraz kody źródłowe; • trafnie używa sformułowań: *czujnik, czułość, wejście analogowe, przetwornik A/D.*
Interakcja modułu-interfejsu ze środowiskiem S4A	Budowa interfejsu z wykorzystaniem zestawu Arduino i programu w środowisku Scratch S4A w celu stworzenia interaktywnej gry ping-pong.	• prawidłowo obsługuje środowisko programowania graficznego Scratch S4A; • samodzielnie montuje i uruchamia przykładowe układy na podstawie schematów; • deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • programuje mikrokontroler Atmega oraz steruje nim z poziomu komputera; • prawidłowo łączy i steruje przyciskami z zestawu modułu-interfejsu; • trafnie używa zwrotów: *button, wejście cyfrowe*; • angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem.

Treści dla szkół ponadgimnazjalnych niesprofilowanych

MODUŁY B – programowanie		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Szyfrowanie – jak chronić ważne informacje	Zaawansowane użytkowanie Linuksa, podstawowe informacje o bezpieczeństwie danych i szyfrowaniu	• wskaże wagę bezpieczeństwa danych oraz zalety podpisu elektronicznego i szyfrowania; • omawia sposób zdobywania informacji z manuali systemowych; • omawia obsługę konsoli; • wykorzystuje podstawowe funkcje obsługi Linuksa, aplikacji gpg, Kgpg; • tworzy własne klucze pgp, obsługuje gpg z linii poleceń.
Wizualizacja instrukcji warunkowej – „Wybór drogi”	Prezentacja za pomocą Scratcha sposobu podejmowania decyzji przez program wykonujący instrukcję warunkową.	• omawia pojęcia: *instrukcja warunkowa, zmienna, algorytm*; • wykorzystuje Scratch do stworzenia wizualizacji działania instrukcji warunkowej.
Wizualizacja działania pętli – „Zakręcona mrówka”	Prezentacja za pomocą Scratcha zasady realizacji powtórzeń przez program wykonujący instrukcję pętli.	• omawia pojęcia: *iteracja, instrukcja pętli, procedura*; • wykorzystuje Scratcha do stworzenia wizualizacji działania pętli.
Wizualizacja operatorów przypisania – „Magiczny operator”	Prezentacja funkcji operatora przypisania wartości do zmiennej w programie Scratch.	• omawia pojęcia: *zmienna, zmienna pusta, deklaracja zmiennej, przypisanie, wyrażenie, ewaluacja*; • wskazuje, w jaki sposób maszyny wyliczają wartości wyrażeń; • wykorzystuje Scratcha do stworzenia wizualizacji przypisania wartości.
Ewaluacja prawej strony wyrażeń – „Zamieszanie z kolorami”	Prezentacja metody ewaluacji prawej strony wyrażenia na przykładzie mieszania kolorów.	• omawia pojęcia: *przypisanie, wyrażenie, ewaluacja*; • wskazuje, w jaki sposób program sprawdzają wartości wyrażeń; • wykorzystuje Scratcha do przedstawienia zasady ewaluacji.
Gra w „Kółko i krzyżyk” ze strategią blokowania przeciwnika	Odnalezienie w istniejącym kodzie miejsca, które należy rozbudować o sprawdzanie istnienia kombinacji figur w grze „Kółko i krzyżyk”, które w następnym ruchu pozwolą przeciwnikowi na wygranie i zablokowanie ruchu przeciwnika	• posługuje się naturalnym językiem przy opisie algorytmów • wskazuje miejsca w algorytmie realizujące dane zadania; • poprzez dokonanie analizy algorytmu proponuje rozwiązanie rozszerzające jego działanie o założone funkcje; • prezentuje dane w macierzy dwuwymiarowej za pomocą liniowej adresacji, wykorzystuje tablice; • dzieli algorytm na podprogramy; • realizuje grę z interfejsem graficznym oraz algorytmem komputerowego przeciwnika
Gra Pong o zmiennej strategii	Zmiany w skryptach gry Pong, dodające kolejne elementy i skrypty zmieniające jej zasady	• analizuje kod i tłumaczy zasadę działania programu; • posługuje się naturalnym językiem przy opisie algorytmów; • wskazuje miejsca w kodzie realizujące poszczególne zadania; • omawia podstawowe pojęcia: *zmienna globalna, zmienna lokalna, lista, kolejka*; • prezentuje dane w strukturach typu lista i kolejka danych; • wskazuje podprogramy w algorytmie; • programuje skrypty obsługujące animację obiektów; • prawidłowo implementuje zmianę w grze.

Wybrane algorytmy sortowania	Wizualizacja sposobu sortowania elementów zbioru w porządku rosnącym, na przykładzie zbioru liczbowego o zmiennej liczbie elementów. Związek ilości kroków algorytmu z liczbą elementów w zbiorze.	• omawia podstawowe pojęcia: *algorytm, sortowanie*; • charakteryzuje sposób działania algorytmów: *bubble sort, inserting sort* oraz *selection sort* oraz uporządkuje 5-elementowy zbiór tymi metodami; • narysuje schemat blokowy algorytmu realizującego sortowanie bąbelkowe; • analizuje ilość kroków algorytmu w zależności od liczby elementów zbioru; • zaimplementuje algorytmy w języku Scratch.
Wprowadzenie do środowiska Lazarus – „Kółko i krzyżyk"	Prezentacja zintegrowanego, graficznego środowiska programowania Lazarus. Tworzenie prostego programu w języku FreePascal.	• wyjaśnia zasadę programowania obiektowo-zdarzeniowego; • określa, do czego służą składowe środowiska Lazarus: *Inspektor obiektów, Właściwości, Zdarzenia, Favorites, Komponenty, Komunikaty, Edytor źródeł - Kod programu, Menu Edytora, Formularz*; • prawidłowo porusza się w środowisku Lazarus; • stosuje podstawowe konstrukcje języka FreePascal; • wskazuje korzyści płynące z umiejętności programowania
Gra w zapalane światełka	Prosta gra w światełka – praca w Lazarusie. Przenoszenie stanów poprzedzających do parametrów bieżących.	• analizuje kod i tłumaczy zasadę działania programu; • programuje w stopniu pozwalającym na zaimplementowanie większej planszy gry; • programuje w stopniu pozwalającym na uzupełnienie brakujących fragmentów kodu • tworzy program, w którym następny krok zależy od bieżącego stanu gry
Automaty – „Gra w życie"	Implementacja „Gry w życie" jako przykład prostej symulacji opartej na automatach	• omawia, czym są automaty i do czego służą • określa zasady działania automatów • konstruuje prosty automat
Zegar binarny	Przedstawienie zasady działania zegara decybinarnego	• omawia zasady działania zegara opartego na systemie binarnym • odczytuje i zapisuje cyfry przy użyciu systemu binarnego • przekształca zapis z systemu dziesiętnego na dwójkowy • wyjaśnia zasadę funkcjonowania zegara binarnego • omawia działanie operatora logicznego *and* i *or* na zmiennych binarnych
Losowanie i porównywanie – gra „Kamień-nożyce-papier"	Realizacja gry losowej, polegającej na równoczesnym wyborze dwóch przedmiotów, z których jeden wygrywa według zasady gry „Kamień-nożyce-papier"	• obsługuje zintegrowane środowisko programowania wspieranego interfejsem graficznym; • tworzy potrzebne obiekty za pomocą GUI i odpowiednio modyfikuje ich właściwości; • nazywa główne bloki struktury kodu źródłowego i elementarne instrukcje języka FreePascal; • trafnie operacjonalizuje i objaśnia pojęcia: *obiekty, atrybuty, kod źródłowy, instrukcje*; • prawidłowo wprowadza kod źródłowy i doprowadza do pełnego działania implementacji; • próbuje wygrać z implementacją sztucznej inteligencji.

Rozrzucanie i porządkowanie – gra „Układanka”	Realizacja gry logicznej typu puzzle, polegającej na porządkowaniu 24 liter alfabetu (A÷X) ułożonych w tablicy 5x5, poprzez przemieszczanie liter z pól stycznych do pola pustego.	• trafnie operacjonalizuje i objaśnia pojęcia: *kody liter, tablica, menu, obsługa zdarzeń*; • tworzy foremną macierz widżetów TPanel i modyfikuje ich atrybuty z poziomu kodu; • prawidłowo uzupełnia fragmenty kodu źródłowego i doprowadza do działania implementacji; • samodzielnie stosuje w praktyce strategię porządkowania w układance alfabetycznej; • chętnie rozwiązuje zadanie uporządkowania całej macierzy o rozmiarach 5x5.
Odkrywanie i stosowanie algorytmu – gra „Wieże Hanoi”	Wizualizacja strategii wygranej w grze decyzyjnej, polegającej na przenoszeniu obiektów o różnej wielkości między 3 cokołami tak, aby obiektu większego nie stawiać na mniejszym.	• trafnie operacjonalizuje i objaśnia pojęcia: *strategia, algorytm, animacja, wizualizacja*; • wpisuje właściwe parametry umiejscawiające obiekty na ekranie w odpowiednim miejscu; • prawidłowo uzupełnia fragmenty kodu źródłowego i doprowadza do działania implementacji; • opisuje werbalnie i optymalnie realizuje strategię wygranej w łamigłówce „Wieże Hanoi”; • chętnie rozwiązuje utrudnione zadanie uporządkowania z pozycji pośredniej nieregularnej.
Namiastka sztucznej inteligencja – gra NIM	Realizacja gry logicznej NIM, w której chodzi o to, aby podczas naprzemiennego pobierania obiektów z jednego rzędu na planszy uniknąć konieczności zabrania obiektu ostatniego. Losowane są różne układy do 10 obiektów w każdym z trzech rzędów. Komputer ma zaprogramowaną strategię wygranej, dlatego rozpoczyna gracz.	• trafnie operacjonalizuje i objaśnia pojęcia: *warunki, operatory logiczne, sterowanie, indeksy*; • z poziomu kodu prawidłowo tworzy obiekty TImage lub modyfikuje ich atrybuty; • prawidłowo uzupełnia fragmenty kodu źródłowego i doprowadza do działania implementacji; • opisuje werbalnie i optymalnie realizuje strategię wygranej w końcowej fazie gry NIM; • chętnie stosuje w praktyce strategię wygranej poprzez analizę parzystości grup binarnych.

MODUŁY B – mechatronika		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Funkcjonalność modułu-interfejsu. Środowisko mikrokontrolera	Zastosowanie modułu-interfejsu Arduino oraz obsługa interaktywnego terminala Arduino IDE, służącego do programowania mikrokontrolera. Zestawianie połączeń na podstawie instrukcji montażu układów. Zaimplementowanie kodu do wyświetlania tekstów oraz do sterowania diodą wbudowaną w moduł-interfejs.	• trafnie objaśnia pojęcia: *mikrokontroler; port USB; dioda LED; button; opornik*; • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo podłącza diody LED oraz RGB; • steruje diodami i modyfikuje treść wyświetlanych komunikatów.

Sterowanie elementami wykonawczymi z wykorzystaniem czujnika światła	Wizualizacja działania elementów modułu-interfejsu. Wykorzystanie funkcji przetwornika analogowo-cyfrowego do układów pomiarowych. Istota funkcjonowania fotorezystora. Konstruowanie i oprogramowanie układów do odczytu stanu czujnika na przykładzie interfejsu do pomiaru natężenia światła. Prezentacja wyników z wykorzystaniem diody RGB.	• zgodnie z zasadami działania podłącza czujnik pomiarowy: fotorezystor; • prawidłowo buduje i oprogramowuje moduł-interfejs wskazujący poziomy oświetlenia; • uruchamia ukazywanie odczytów na wyświetlaczu LCD lub w środowisku Linux; • modyfikuje i rozbudowuje pomiarowe układy elektroniczne oraz kody źródłowe; • trafnie używa sformułowań: *czujnik, wejście analogowe, przetwornik A/D*.
Pomiar temperatury i obsługa wyświetlacza LCD 2x16	Wizualizacja działania zestawu modułu-interfejsu z układem Arduino. Wykorzystanie funkcji przetwornika analogowo-cyfrowego do budowy układu pomiarowego. Istota funkcjonowania i zastosowania termistora. Podłączenie i sterowanie LCD do wyświetlania tekstów. Konstruowanie i oprogramowanie układu do odczytu stanu czujnika. Prezentacja odczytu temperatury i skrajnych wartości na LCD oraz na ekranie monitora.	• zgodnie z zasadami działania podłącza czujnik pomiarowy termistor; • prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru temperatury; • uruchamia ukazywanie odczytów na wyświetlaczu LCD lub w środowisku Linux; • modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy; • dokonuje przeliczenia wartości pomiarów temperatury do innych skal; • trafnie używa sformułowań: *czujnik, stopnie Celsjusza, stopnie Fahrenheita, Kelvin, czułość, wejście analogowe, przetwornik A/D*.
Interakcja człowieka z modułem-interfejsem	Budowa układu i programu do symulacji losowania jednej z sześciu liczb jak w kostce do gry. Prezentacja wyniku losowania z wykorzystaniem diod LED. Poszerzanie wiedzy o zastosowaniu modułu-interfejsu poprzez rozbudowę o kolejne elementy. Efektem jest opracowanie implementacji pozwalającej ćwiczyć pamięć i zręczność.	• samodzielnie projektuje i oprogramowuje układy na platformie Arduino; • implementuje układy interfejsu z diodami LED i przyciskami button; • mechanicznie i programowo steruje wejściami i wyjściami cyfrowymi; • implementuje grę zręcznościowo-pamięciową z losowaniem liczb; • prawidłowo analizuje i pisze objaśnienia kodu źródłowego implementacji; • rozbudowuje moduł-interfejs, dodając wyświetlacz LCD i buzzer; • trafnie używa sformułowań: *PULLUP, wejście/wyjście cyfrowe, typ logiczny (boolean)*.
Możliwości środowiska Scratch S4A i modułu-interfejsu Arduino	Wprowadzenie w świat mikrokontrolerów na przykładzie modułu-interfejsu Arduino oraz jego obsługa w środowisku Scratch (S4A). Prezentacja i wyjaśnienie sposobu zestawiania połączeń na podstawie instrukcji ilustrującej montaż układów.	• omawia pojęcia: *button, wejście cyfrowe; dioda RGB; opornik; pętla; wyrażenie warunkowe; zmienna; mikrokontroler; port USB i RS232*; • montuje i uruchamia przykładowe układy na podstawie schematów; • deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • obsługuje środowisko Scratch S4A, wykorzystując jego funkcje; • używa podstawowych narzędzi programu Scratch S4A; • omawia istotę działania oraz sposób podłączania i sterowania podzespołami: dioda RGB, przycisk; • angażuje się we współpracę z innymi uczennicami i uczniami oraz z nauczycielem;

Działanie fotorezystora i potencjometru – przetwornik analogowo-cyfrowy	Istota funkcjonowania i zastosowania fotorezystora i potencjometru. Konstruowanie i oprogramowanie układów do odczytu stanu potencjometru i wartości fotorezystora	• zgodnie z zasadami działania podłącza czujnik pomiarowy: fotorezystor; • prawidłowo buduje i oprogramowuje moduł-interfejs wskazujący odczyty z wejścia analogowego; • modyfikuje i rozbudowuje pomiarowe układy elektroniczne oraz kody źródłowe; • trafnie używa sformułowań: *czujnik, czułość, wejście analogowe, przetwornik A/D*; • wskazuje zastosowania modułów-interfejsów
Środowisko mikrokontrolera – rozbudowa funkcjonalności	Podłączenie i sterowanie diodami LED na przykładzie sygnalizacji świetlnej. Obsługa przycisków i sterowanie dziękowe buzzera. Zaimplementowanie kodu do sterowania diodą.	• trafnie objaśnia pojęcia: *mikrokontroler; port USB; dioda LED; button; opornik; buzzer*; • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo podłącza diodę LED i steruje jasnością diody; • modyfikuje treść wyświetlanych komunikatów;
Termometr cyfrowy – obsługa wyświetlacza	Podłączenie i sterowanie wyświetlaczem LCD z wykorzystaniem płytki stykowej. Zaimplementowanie kodu do wyświetlania tekstów.	• zgodnie z zasadami działania podłącza czujnik pomiarowy termistor; • prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru temperatury; • uruchamia ukazywanie odczytów na wyświetlaczu LCD lub w środowisku Linux; • modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy; • przedstawia skale i jednostki temperatury oraz omawia zależności między nimi • prawidłowo dokonuje przeliczenia wartości pomiarów temperatury do innych skal; • trafnie używa sformułowań: *czujnik, stopnie Celsjusza, stopnie Fahrenheita, Kelvin, czułość, wejście analogowe, przetwornik A/D*;

Treści dla szkół sprofilowanych infotechnicznie

MODUŁY C – programowanie, język C		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Struktura programu, zmienne oraz typy danych	Poznanie struktury programu, pojęcia zmiennych oraz podstawowych typów danych, jakie zmienne mogą przybierać w języku C.	• omawia podstawowe pojęcia programowania: *zmienna, typy zmiennych* • omawia strukturę programu w języku C • pisze prosty program zawierający kilka zmiennych, który kompiluje się bez błędów oraz poprawnie działa • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux

Wyrażenia warunkowe *if – else*	Poznanie struktury oraz zastosowania wyrażeń warunkowych *if – else* w języku C	• omawia podstawowe pojęcia programowania: *wyrażenia warunkowe if – else* i wskazuje, do czego służą • pisze prosty program zawierający *wyrażenia warunkowe if – else* i zachowujący się różnie, zależnie od wprowadzonych danych • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux
Wyrażenie warunkowe *switch{ case: . . .;}*	Poznanie struktury oraz zastosowania wyrażenia warunkowego *switch {case: . . .;}* w języku C.	• omawia podstawowe pojęcia programowania: *wyrażenia warunkowe switch {case: . . .;}* i wskazuje, do czego służą • pisze prosty program zawierający wyrażenie warunkowe *switch {case: . . .;}* i zachowujący się różnie, zależnie od wprowadzonych danych • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux
Pętle – *while()* i *do {} while()*	Poznanie, czym są pętle i jak korzystać z pętli *while* oraz *do{} while()* w języku C.	• omawia podstawowe pojęcia programowania: *pętle – while()* i *do {} while()* i wskazuje, do czego służą • pisze prosty program korzystający z pętli do wykonywania powtarzających się czynności • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux
Tablice i pętla *for()*	Poznanie, czym są tablice oraz pętla *for()* w języku C	• omawia podstawowe pojęcia programowania: *tablice, pętle for()* i wskazuje, do czego służą • pisze prosty program korzystający z tablic oraz wykorzystuje pętlę *for()* do operowania na nich • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux
Funkcje	Tworzenie oraz używanie funkcji w języku C.	• wyjaśnia, czym są funkcje i kiedy kod należy dzielić na mniejsze funkcje, a także jak się tworzy funkcje oraz jak ich używać • pisze prosty program podzielony na funkcje realizujące pojedyncze zadania • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux
Struktury i unie	Tworzenie oraz praktyczne korzystanie ze struktur oraz unii w języku C	• wyjaśnia, czym są: *struktury i unie* oraz dlaczego są ważne i potrzebne w programowaniu • wyjaśnia, jak się deklaruje struktury i unie oraz do czego się ich używa • pisze program korzystający ze struktur porządkowania danych • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux
Wskaźniki	Poznanie zasad funkcjonowania pamięci w programach oraz wskaźników jako metody bezpośredniego dostępu do niej	• wyjaśnia, jak działa pamięć, jak dane są w niej umieszczane oraz jak można się do nich dostać, używając wskaźników • omawia, jak zarządzana jest pamięć oraz jak zmienne są w niej umieszczane • wyjaśnia, jak się deklaruje i na różne sposoby używa wskaźników • deklaruje i wykorzystuje wskaźniki • obsługuje kompilator w podstawowym zakresie • uruchamia program z konsoli systemu Linux

MODUŁY C – mechatronika, wersja I		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Budowa i programowanie modułu-interfejsu	Zastosowanie modułu-interfejsu Arduino oraz obsługa interaktywnego terminala Arduino IDE, służącego do programowania mikrokontrolera.	• trafnie objaśnia pojęcia: *mikrokontroler; port USB; dioda LED; button; opornik; potencjometr, modulacja szerokości impulsu – PWM* • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • przesyła wyniki z układu do komputera; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo podłącza diodę LED i steruje jasnością diody; • modyfikuje treść wyświetlanych komunikatów;
Tranzystor NPN	Zastosowanie tranzystora do sterowania pięcioma diodami LED połączonymi równolegle. Wykorzystanie modulacji szerokości impulsów do sterowania jasnością świecenia diod.	• trafnie objaśnia pojęcia: *tranzystor NPN, tranzystor PNP, button, modulacja szerokości imppulsu* • wymienia rodzaje tranzystorów, podaje parametry i rozpoznaje oznaczenia wyprowadzeń; • wykorzystuje tranzystor jako przełącznik; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo wykonuje pomiary i przelicza wartości napięcia, prądu i oporności; • trafnie używa słowa: *klucz tranzystorowy, stan nasycenia i zatkania, Volt, Amper*;
Pole magnetyczne	Wykorzystanie efektu Halla do sygnalizacji przy pomocy diody RGB oraz występowania pola magnetycznego. Zastosowanie, budowa i działanie hallotronu.	• trafnie objaśnia pojęcia: *pole magnetyczne, histereza, półprzewodnik, dioda RGB, efekt Halla*; • objaśnia zasadę działania hallotronu i prawidłowo stosuje go w układach; • podaje parametry i rozpoznaje oznaczenia wyprowadzeń; • prawidłowo rozbudowuje układy o dodatkowe, niezbędne do pracy elementy; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
Zegar	Stworzenie prostego stopera i zegara przy użyciu modułu-interfejsu. Wykorzystanie podstawowych funkcji do sterowania i prezentacji czasu na wyświetlaczu LCD oraz na ekranie monitora. Zasada działania i używania bibliotek.	• prawidłowo buduje i oprogramowuje moduł-interfejs wskazujący aktualny czas; • prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru czasu; • uruchamia ukazywanie teksu na wyświetlaczu LCD; • modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy; • trafnie używa sformułowań: *czas, sekunda, milisekunda, opóźnienie, pętle, LCD, biblioteka*;

MODUŁY C – programowanie, język Python		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Zmienne, typy i pobieranie danych	Poznanie pojęcia zmiennych, ich typów oraz sposoby wczytywania danych z klawiatury w języku Python	• omawia podstawowe pojęcia programowania: *zmienna, interpreter* • pisze prosty program zawierający kilka zmiennych oraz pobiera ich wartości od użytkownika • wykorzystuje interpreter do interaktywnego testowania kawałków kodu przed włączeniem ich do programu • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python
Wyrażenia warunkowe	Poznanie wyrażeń warunkowych *if – elif – else* w języku Python	• omawia podstawowe pojęcia programowania: *wyrażenia warunkowe, struktura wyrażeń warunkowych* oraz docenia ich istotność w programowaniu • pisze programy, które proszą użytkownika o wprowadzenie danych oraz podejmują odpowiednią akcję • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python
Pętle – *while*	Poznanie pętli *while()* w języku Python	• omawia podstawowe pojęcia programowania: *pętla while* • pisze programy, które powtarzają pewne czynności określoną liczbę razy • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python
Zaawansowane typy danych	Poznanie zaawansowanych typów danych oraz nauczenie się, do czego i jak je wykorzystywać w języku Python	• omawia zaawansowane typy danych i ich przydatność w programowaniu • pisze programy, które korzystają z list, krotek i słowników oraz wie, czym są zbiory i jak ich używać • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python
Pętle – *for*	Poznanie innego rodzaju pętli, jaką jest pętla for w języku Python	• objaśnia podstawowe pojęcia programowania: *pętla for*, omawia różnice pomiędzy pętlami • pisze programy, które używają pętli for do operacji na poszczególnych elementach większych zbiorów • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python
Funkcje	Poznanie tego, jak w Pythonie tworzyć własne funkcje oraz kiedy należy program dzielić na funkcje	• omawia podstawowe pojęcia programowania: *funkcje* • pisze kod realizujący własne funkcje • dzieli długie programy na funkcje, aby były czytelniejsze i efektywniejsze • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python
Wyjątki	Poznanie tego, czym są wyjątki i jak ich używać w języku Python	• omawia, czym są *wyjątki* i jak pomagają programiście • przewiduje, kiedy program może wyrzucić wyjątek • programowo wyłapuje i obsługuje wyjątek w programie • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python

Klasy	Poznanie tego, czym są klasy i jak dzięki nim tworzyć efektywniejsze programy w języku Python	• omawia podstawowe pojęcia programowania: *klasy, programowanie obiektowe* • uzasadnia używanie klas dla ułatwienia pisania dużych programów • rozumie paradygmat programowania obiektowego oraz pisze programy korzystające z niego • tworzy własne klasy i korzysta z gotowych bibliotek Phytona • uruchamia program z konsoli systemu Linux przy użyciu interpretera języka Python

MODUŁY C – mechatronika, wersja II		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Sterowanie diodami – wykorzystanie wejścia analogowego	Zastosowanie modułu-interfejsu Arduino oraz obsługa interaktywnego terminala Arduino IDE, służącego do programowania mikrokontrolera.	• trafnie objaśnia pojęcia: *mikrokontroler; port USB; dioda LED; button; opornik; czujnik nachylenia* • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo podłącza diodę LED i steruje jasnością diody; • modyfikuje parametry wyświetlanych komunikatów;
Pomiar wartości opornika	Zastosowanie modułu-interfejsu Arduino jako narzędzia do pomocy w oszacowaniu wartości oporników. Budowa i sposoby rozpoznawania oznaczeń oporników. Oszacowanie wartości rezystorów połączonych szeregowo i równolegle.	• trafnie objaśnia pojęcia: *dzielnik napięcia, opornik*, • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • stosuje metodę przesyłania wyników z układu do komputera; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • odczytuje wartość opornika na podstawie kodu barwnego lub z użyciem modułu-interfejsu; • oblicza wartość rezystancji w przypadku łączenia rezystorów równolegle lub szeregowo • dokonuje przekształceń wzoru prawa Ohma, aby wyznaczyć wartość rezystancji;

Kontrola diody RGB	Komunikacja modułu-interfejsu z komputerem PC na przykładzie sterowania jasnością i kolorami diody RGB	• trafnie objaśnia pojęcia: *półprzewodnik, dioda RGB, port szeregowy, komunikacja, transmisja danych*; • podaje parametry i rozpoznaje oznaczenia wyprowadzeń diody RGB; • prawidłowo rozbudowuje układy o dodatkowe, niezbędne do pracy elementy; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości;
Prezentacja temperatury z wykorzystaniem diody RGB i buzzera	Zastosowanie modułu-interfejsu Arduino do odczytu temperatury. Przełożenie odczytów wartości z wejścia analogowego na sterowanie jasnością i barwą diody RGB. Definiowanie wartości progowych temperatury i sygnalizowanie alarmem w przypadku ich przekroczenia.	• zgodnie z zasadami działania podłącza czujnik pomiarowy termistor; buzzer, diodę RGB; • prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru temperatury; • uruchamia ukazywanie odczytów w środowisku Linux; • modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy; • dokonuje przeliczenia wartości pomiarów temperatury do innych skal; • trafnie używa sformułowań: *czujnik, stopnie Celsjusza, Fahrenheita i Kelvina, czułość, wejście analogowe, przetwornik A/D*;

MODUŁY C – HTML i JavaScript		Efekty – wskaźniki osiągnięcia celów
Tematyka modułów	**Opis realizowanych treści**	**Uczeń:**
Podstawy HTML	Tworzenie szkieletu strony internetowej, zawierającej nagłówek, stopkę, nawigację i treść. Walidacja prawidłowości kodu	• objaśnia zastosowanie podstawowych tagów HTML i sposobów rozmieszczania ich na stronie; • tworzy prosty dokument HTML i dopasowuje jego wygląd; • weryfikuje poprawność dokumentu, korzystając z jednego z walidatorów • analizuje strukturę dokumentu, korzystając z narzędzi dostępnych w przeglądarce
CSS i box model	Poznanie podstaw CSS. Planowanie typografii i modyfikacja rozmieszczeń elementów na stronie.	• rozmieszcza elementy HTML na stronie za pomocą CSS; • dopasowuje wygląd dokumentu HTML, korzystając z pliku CSS; • swobodnie rozmieszcza elementy dokumentu na stronie za pomocą różnych technik; • analizuje strukturę dokumentu, korzystając z narzędzi dostępnych w przeglądarce.
Formularze HTML	Tworzenie dokumentu z formularzem umożliwiającym wprowadzanie informacji przez użytkownika, z wykorzystaniem odpowiednich elementów w zależności od rodzaju informacji	• objaśnia zastosowanie tagów HTML służących do budowania formularzy • tworzy rozbudowany formularz HTML • dobiera odpowiednie elementy formularza do typu danych wprowadzanych przez użytkownika • weryfikuje poprawność stworzonego dokumentu HTML

JavaScript	Poznanie podstawy korzystania z języka JavaScript oraz stworzenie implementacji wykorzystującej poznane elementy języka	• dokonuje operacji na zmiennych • tworzy proste struktury danych • definiuje, przekazuje i uruchamia funkcje • tworzy prostą aplikację dokonującą interakcji z użytkownikiem
Podstawy jQuery	Napisanie programu tak, żeby komunikował się z użytkownikiem poprzez formularz i dokument HTML	• wyszukuje dowolne elementy HTML z wykorzystaniem selektorów CSS • pobiera informacje wprowadzone do formularza HTML • modyfikuje dokument HTML poprzez zmianę i dodawanie nowych elementów
jQuery – zdarzenia i efekty	Dodatkowe informacje o tym, jak reagować na różne operacje dokonywane przez użytkownika oraz prezentować wynik działań poprzez zmiany efektów w dokumencie HTML	• dodaje do aplikacji obsługę zdarzeń • tworzy obsługę zdarzenia dla elementów tworzonych dynamicznie przez aplikację • modyfikuje dokument HTML, dokonując zmian z drobnymi animacjami • łączy animacje w sekwencje następujące jedna po drugiej • korzysta z efektów jQuery do modyfikowania dokumentu HTML
Przechowywanie danych	Uzupełnienie aplikacji ToDo o możliwość trwałego zapisania wprowadzonych informacji i ich odczytu przy ponownym uruchomieniu strony	• wymienia sposoby na trwały zapis danych oraz omawia ich wady i zalety • zapisuje informacje na czas działania sesji przeglądarki • trwale zapisuje informacje, niezależnie od czasu działania sesji • pobiera i modyfikuje całość lub część przechowywanych danych • dobiera najlepsze narzędzie w zależności od typu informacji do przechowania
Canvas	Tworzenie implementacji pozwalającej na rysowanie prostych figur geometrycznych	• wskazuje sposoby tworzenia animacji obiektów i omawia różnice między nimi • tworzy implementację i rysuje proste figury geometryczne z wykorzystaniem różnych stylów • tworzy interfejs właściwie reagujący na akcje wykonywane przez użytkownika z użyciem myszki

MODUŁY C – mechatronika, wersja III

Efekty – wskaźniki osiągnięcia celów

Tematyka modułów	Opis realizowanych treści	Uczeń:
Wykorzystanie wejścia analogowego do sterowania diodami	Zastosowanie modułu-interfejsu Arduino oraz obsługa interaktywnego terminala Arduino IDE, służącego do programowania mikrokontrolera.	• trafnie objaśnia pojęcia: *mikrokontroler; port USB; dioda elektroluminescencyjna; button; opornik; potencjometr,* • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • stosuje metodę przesłania wyników z układu do komputera; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • steruje diodą elektroluminescencyjną oraz modyfikuje treść wyświetlanych komunikatów;

Protokół komunikacyjny *1-wire*	Budowa układu i programu do odczytu danych, wykorzystując interfejs 1-wire na przykładzie czujnika Dallas DS18B20. Rozszerzenie wiedzy dotyczącej adresowania czujników ich sposobów zasilania i wykorzystywania bibliotek w celu sterowania.	• zgodnie z zasadami działania podłącza czujnik pomiarowy; • prawidłowo buduje i oprogramowuje moduł-interfejs służący do pomiaru temperatury; • uruchamia ukazywanie odczytów na wyświetlaczu LCD lub w środowisku Linux; • modyfikuje i rozbudowuje pomiarowy układ elektroniczny oraz kod źródłowy; • przedstawia skale i jednostki temperatury oraz zależności między nimi; • dokonuje odczytu adresu czujników; • trafnie używa sformułowań: *czujnik, stopnie Celsjusza, 1-wire, czułość, wejście cyfrowe, biblioteka*;
Ultradźwiękowy pomiar odległości	Pomiar odległości z wykorzystaniem czujnika ultradźwiękowego HS-SR04. Prezentacja odczytów na wyświetlaczu LCD oraz sygnalizacja wizualna wykorzystująca diodę RGB, gdy obiekt znajduje się w zadanej odległości.	• trafnie objaśnia pojęcia: *czujnik ultradźwiękowy, wyświetlacz ciekłokrystaliczny, RGB* • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje metodę przesyłania wyników z układu do komputera; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • prawidłowo podłącza serwomechanizm i dokonuje pomiarów odległości; • steruje diodą RGB;
Sterowanie elementami wykonawczymi	Budowa, działanie i sposoby sterowania serwomechanizmem. Programowa oraz mechaniczna zmiana parametrów sterujących.	• trafnie objaśnia pojęcia: *serwomechanizm, biblioteka, PWM* • poprawnie obsługuje terminal do pisania kodu sterującego i wgrywa kod do Arduino; • stosuje elementy kodu do tworzenia i modyfikacji programów sterujących moduł-interfejs; • poprawnie deklaruje podstawowe typy zmiennych, definiuje i przypisuje im wartości; • steruje kątem osi serwomechanizmu; • podłącza i steruje serwomechanizm programowo oraz wykorzystaniem buttonów i potencjometru;