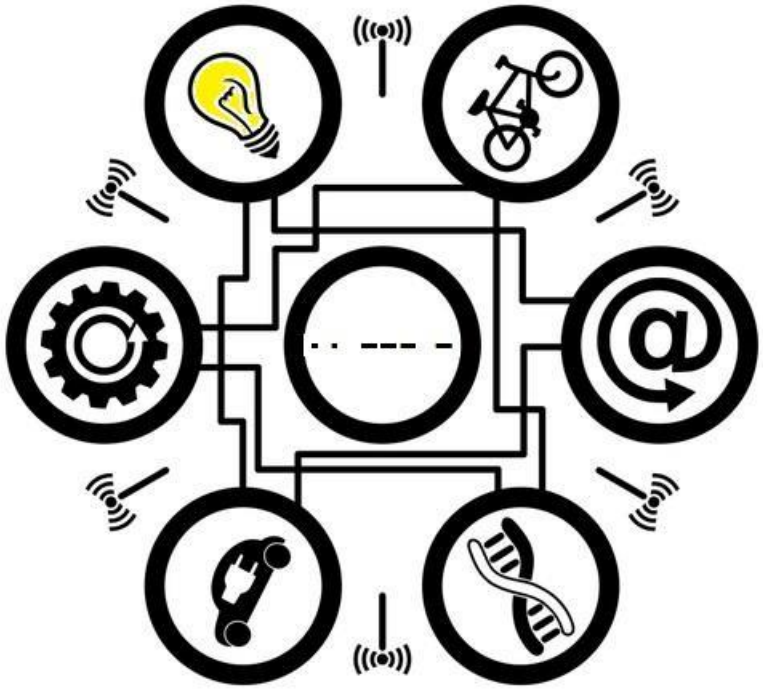
Scenariusz warsztatów



Współfinansowane przez Unię Europejską

Scenariusz warsztatów ·· --- -

Cele:
- zapoznanie uczestników z koncepcją Internetu Rzeczy
- rozbudzenie zainteresowania tematem oraz kreatywnego myślenia
- budowanie pozytywnych postaw w stosunku do nowych technologii
- zapoznanie uczestników z układami elektronicznymi i podstawami programowania

Materiały:
- film edukacyjny: https://www.youtube.com/watch?v=LlhmzVL5bm8
- zdjęcia i opis czujników (załącznik 1) na grupę
- alfabet Morse'a (załącznik 2) na grupę
- komponenty dla grup 2-3 uczestników: płytki ESP32, kable żeńsko-żeńskie, kabel USB, moduł mini sygnalizacji świetlnej, przekaźnik
- komputer na grupę z połączeniem do Internetu i zainstalowanym programem arduino1.8.13
- rzutnik
- konto na *https://appinventor.mit.edu/* (potrzebne będzie też konto Google)
- uprzednio przygotowana lampa:

Instrukcja przygotowania lampy:

Aby połączyć kabel 220V/230V do przekaźnika, najpierw przecinamy kabel. Można zauważyć, że składa się z 2 lub 3 drutów połączonych razem. Rozdziel druty i przetnij je tak jak na Rys. 1.

Rys. 1. Cięcie kabla 220/230V.

Instalacja Arduino1.8.13:
https://www.youtube.com/watch?v=Y0r5dXnl5hQ

Plan zajęć:

1. **Wprowadzenie:**

Film wprowadzający jest dostępny pod linkiem:
https://www.youtube.com/watch?v=LlhmzVL5bm8
Ustaw język narodowy i możliwie duże napisy.

Dyskusja na temat filmu. Poproś uczestników o podzielenie się swoimi odczuciami. Jakie wrażenia wywarł na nich film i sama koncepcja Internetu Rzeczy?

2. **Podstawy elektroniki i programowania:**

Podstawy elektroniki. Praca w grupach.
Rozdaj w grupach po jednej płytce ESP32 i zapytaj, czy ktoś wie, do czego służy. Wyjaśnij, że jest to układ elektroniczny, który służy do komunikacji różnych elementów elektronicznych. Zbiera informacje z czujników i przesyła je do komputera. Posiada także moduł WIFI, więc może łączyć się z komputerem nie tylko za pomocą kabla, ale i lokalnej sieci czy Internetu.
Rozdaj kable USB, kable żeńsko-żeńskie oraz diody każdej grupie. Wyjaśnij, że teraz będziemy łączyć poszczególne elementy elektroniczne. Kabel USB należy podłączyć do komputera do dowolnego portu. W innych portach nie powinny znajdować się inne podpięte urządzenia (np. telefony komórkowe). Mini USB podłączmy do płytki ESP32 (Rys. 2).
Diody podpinamy za pomocą kabli żeńsko-żeńskich. Uczestnicy powinni połączyć wybraną diodę LED (pin obok diody) z dowolnym pinem cyfrowym (zaczyna się od litery „D”).

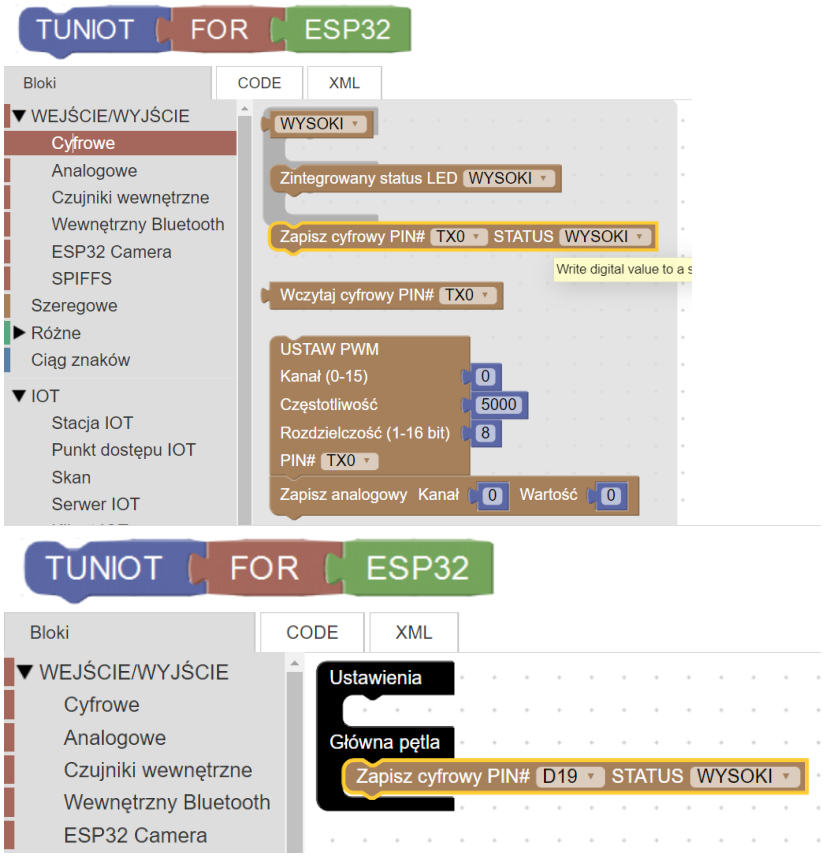
Rys. 2. Port do podpięcia kabla mini-USB na płytce ESP32.

Podstawy programowania. Praca w grupach.
Przedstawienie narzędzia TUNIOT. Narzędzie TUNIOT jest dostępne na stronie *easycoding.tn*. Z menu po lewej stronie wybierz odnośnik „TUNIOT FOR ESP32” i odpowiednią wersję językową (Rys. 3). Wytłumacz, na czy polega programowanie blokowe. Z menu po lewej stronie wybiera się odpowiednie kawałki kodu, które następnie umieszcza się w środkowej części narzędzia – głównym programie. Program podzielony jest na dwie części: **Ustawienia**, która będzie wykonywana tylko raz oraz **Główna pętla**, której zawartość będzie powtarzana.

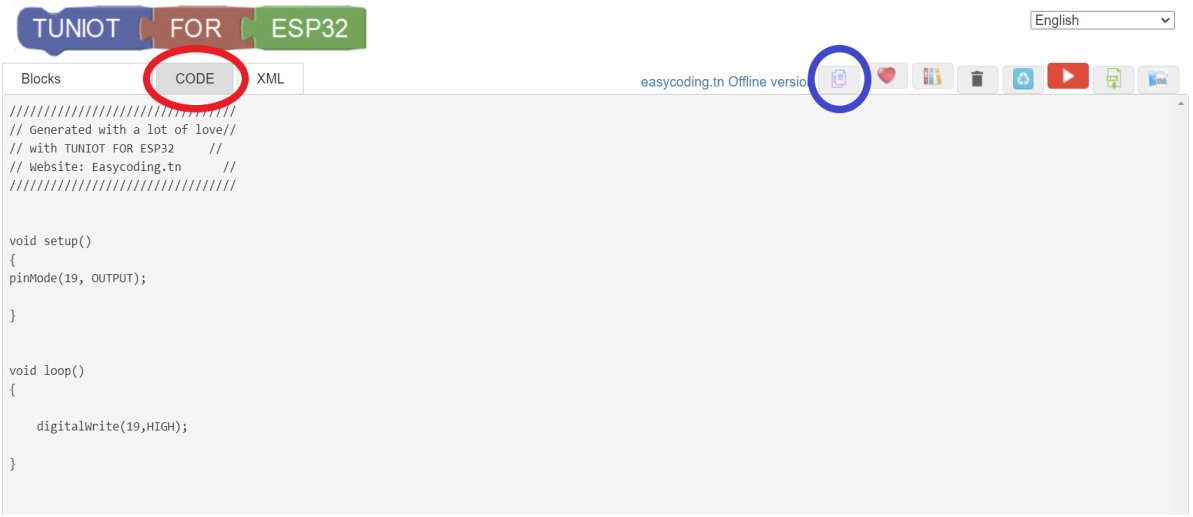
Rys. 3. Odnośnik do narzędzia TUNIOT na easycoding.tn

Ćwiczenie 1. Przejdźmy teraz do przykładu zapalenia diody LED. Z menu **WEJŚCIE/WYJŚCIE** wybierz opcję **Cyfrowe**, ponieważ nasza dioda podpięta jest do pinu cyfrowego. Wyjaśnij, czym różnią się piny cyfrowe od analogowych. Piny cyfrowe odbierają lub wysyłają sygnał zerojedynkowy na zasadzie włącz/wyłącz. Możliwe są tylko dwa stany: albo dioda jest zapalona, albo nie. Do pinów analogowych podpina się czujniki, które wysyłają szerszy zakres wiadomości, np. temperaturę, wilgotność czy naświetlenie możemy mierzyć w jakimś zakresie. Mamy wtedy sygnał bardzo zimny, zimny, letni, ciepły, gorący.

Następnie wybierz kawałek kodu **Zapisz cyfrowy PIN# D0 STATUS WYSOKI** i przeciągnij go do głównego programu pod napisem **Głównej pętli** (Rys. 4). Kawałek kodu powinien się „przykleić" do kodu głównego, wydając charakterystyczny dźwięk. Sprawdź, czy uczestnicy poradzili sobie z tym zadaniem. W nowym kawałku kodu można zmienić dwie rzeczy: numer pinu oraz jego stan w menu rozwijanego. W pierwszym przypadku wybieramy numer pinu, do którego podpięta jest dioda (np. **D19**), a w drugim pozostawiamy **WYSOKI**. **WYSOKI** oznacza, że dioda jest włączona, **NISKI** – że jest wyłączona. Kod zapalający diodę jest gotowy. Wytłumacz, że bloki są tłumaczone na język, który rozumieją komputery, w tym przypadku – język C. Jak on wygląda można podejrzeć w zakładce **CODE** (Rys. 5). Pierwsza ikona górnego menu pozwala na skopiowanie kodu (Rys. 5, niebieskie kółko).

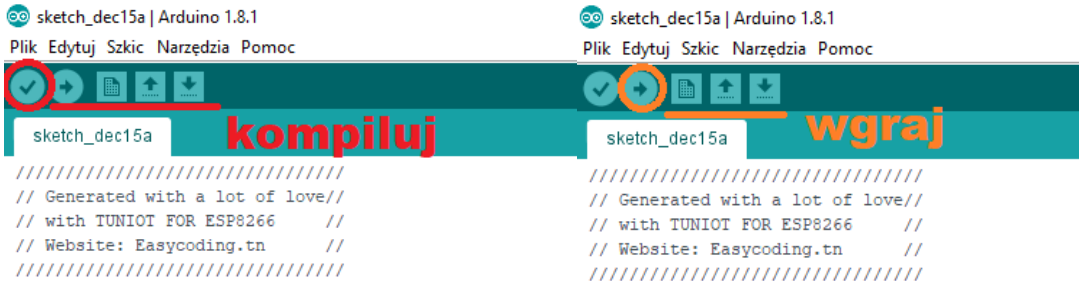


Rys 4. Umiejscowienie kodu pozwalającego wysłać sygnał do pinu cyfrowego.



Rys. 5. Widok narzędzia TUNIOT wraz z kodem zapisanym w języku C oraz zaznaczoną opcją kopiowania.

Kod należy wkleić do programu arduino1.8.13. Najłatwiej zaznaczyć stary kod kombinacją Ctrl+A, a następnie wkleić nowy – Ctrl+V. Program należy skompilować oraz wgrać na płytkę NODEMCU ESP32 za pomocą dwóch pierwszych ikon w menu głównym (Rys. 6). Po zakończeniu wgrywania, dioda powinna się zapalić. Jeśli tak się nie stanie, należy sprawdzić okablowanie, kod i spróbować wgrać jeszcze raz.

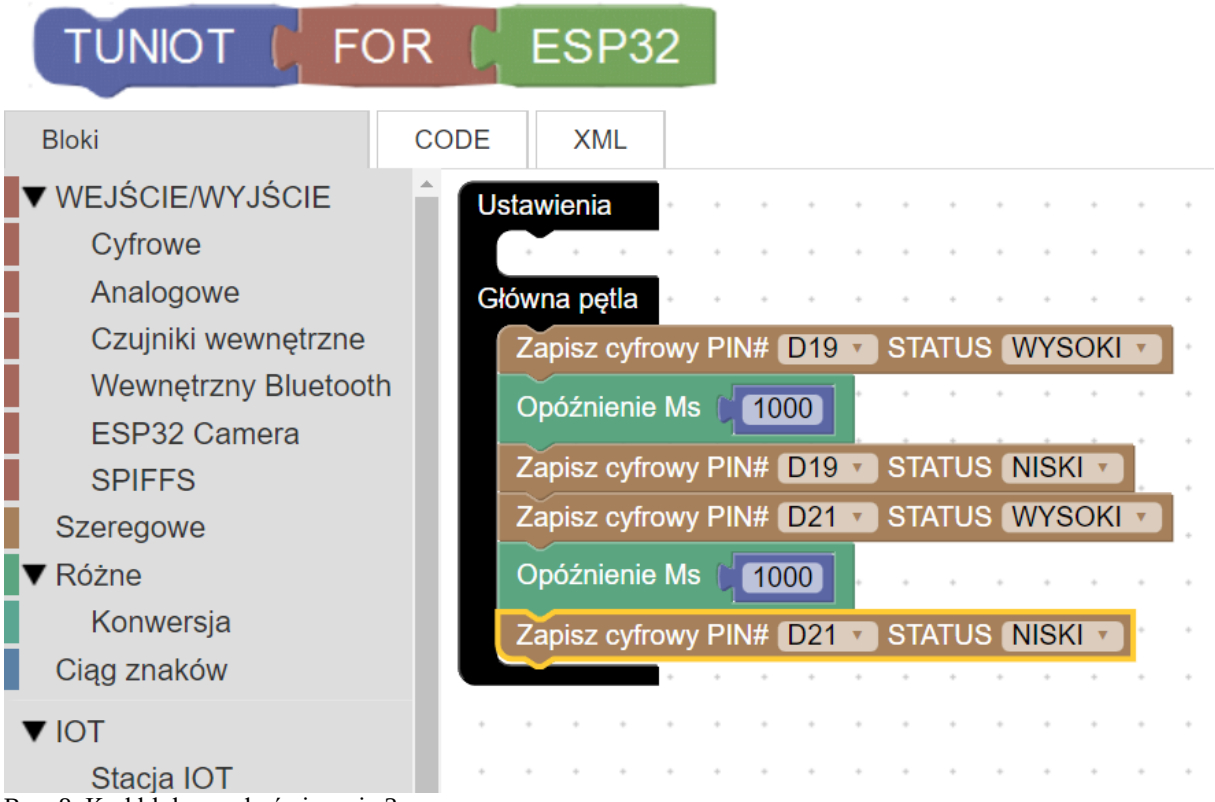


Rys. 6. Przyciski do kompilowania i wgrywania w programie arduino1.8.13.

Ćwiczenie 2. Sprawimy teraz, żeby dioda migotała (włączała się i wyłączała samoczynnie). Ćwiczenie do samodzielnego wykonania. Prawdopodobnie uczestnicy dodadzą tylko kawałek kodu wyłączającego diodę. Jest to poprawne rozumowanie, ale trzeba wyjaśnić, że interwał czasu pomiędzy kolejnymi błyśnięciami jest tak mały, że ludzkie oko nie jest tego w stanie wychwycić. Po komendach włączającej i wyłączającej diodę należy ustawić kawałek kodu z opóźnieniem: **Różne -> Opóźnienie Ms**. Opóźnienie liczone jest w milisekundach. Uczestnicy mogą wpisywać różne wartości, żeby sprawdzić, w jaki sposób migają diody. Przykład kodu jest przedstawiony na Rys. 7.

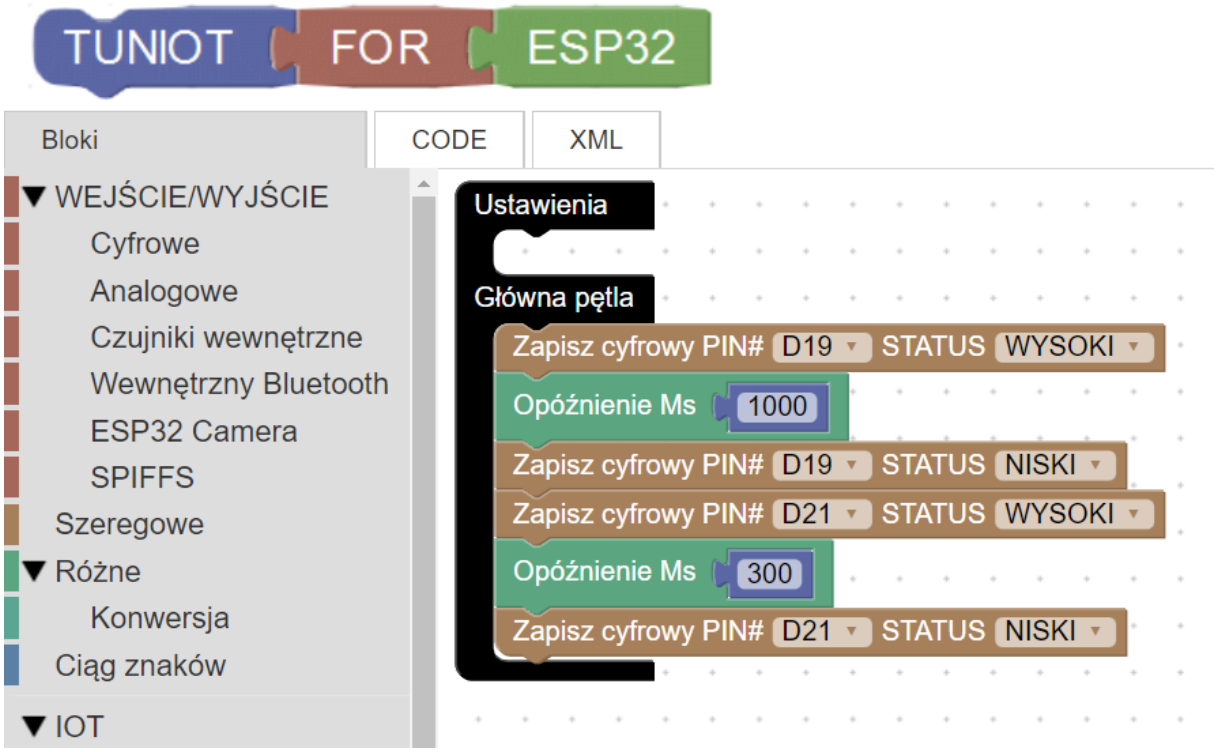
Rys. 7. Kod blokowy do ćwiczenia 2.

Ćwiczenie 3. Powtórzyć ćwiczenie 2, ale dla układu dwóch diod (np. czerwonej i zielonej), które mrugają synchronicznie (Rys. 8). Należy pamiętać o podpięciu odpowiedniej diody do pinu cyfrowego.



Rys. 8. Kod blokowy do ćwiczenia 3.

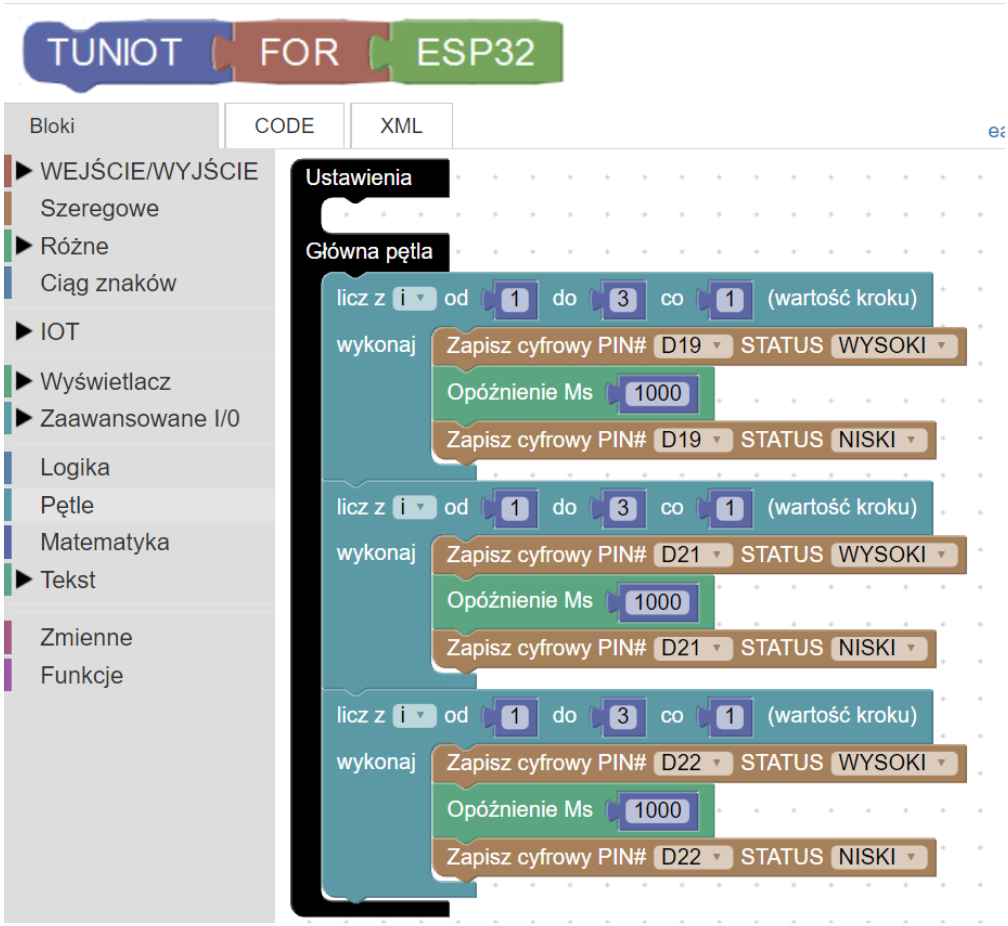
Ćwiczenie 4. Wariant ćwiczenia 3 – układ dwóch diod, z których jedna miga w innym tempie niż druga (Rys. 9).



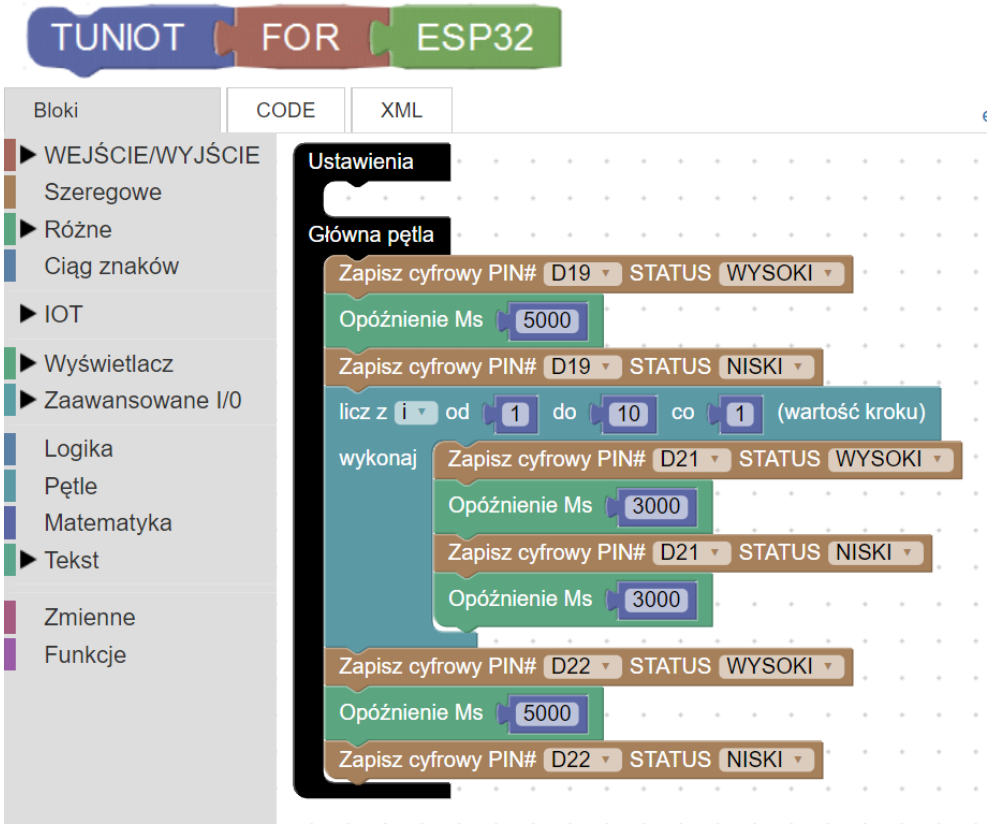
Rys. 9. Kod blokowy do ćwiczenia 4.

Ćwiczenie 5. Należy trzykrotnie zapalić jedną diodę, następnie trzykrotnie drugą i trzykrotnie trzecią (Rys. 10). Zaprezentowanie koncepcji pętli. Kod dopięty do pętli będzie wykonywany w zakresie od 1 do 3 z krokiem 1. Pętlę można znaleźć w: **Pętle -> licz z…**

Ćwiczenie 6. Zadanie do samodzielnego wykonania. Imitacja sygnalizacji świetlnej. Najpierw zapal zieloną diodę przez 5 sekund, później żółtą, która będzie migać 3 sekundy, a na koniec czerwoną, która powinna świecić nieprzerwanie 5 sekund (Rys. 11).



Rys. 10. Kod blokowy do ćwiczenia 5.

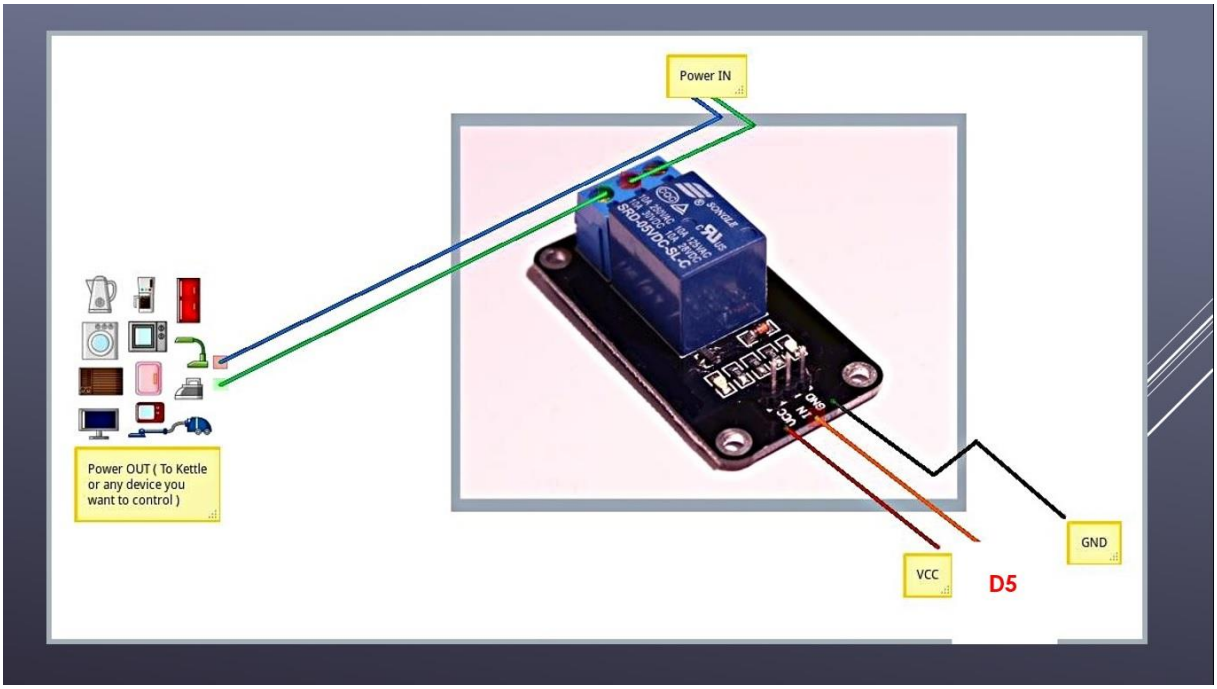


Rys. 11. Kod blokowy do ćwiczenia 6.

Włączanie/wyłączanie lampy. Prezentacja.

UWAGA! Ta część warsztatu wymaga podłączenia do gniazdka elektrycznego 230V. Uczniowie nie powinni wykonywać jej samodzielnie, bez nadzoru osoby dorosłej.

Pokaż uczniom, jak wygląda przekaźnik i wyjaśnij, że służy on do włączania/wyłączania urządzenia elektrycznego (np. czajnika, lampy czy mikrofalówki). Połącz go z płytą ESP32 i lampą tak jak na Rys. 12. W rezultacie powinniśmy otrzymać urządzenie wyglądające jak na zdjęciu poniżej (Rys. 13).



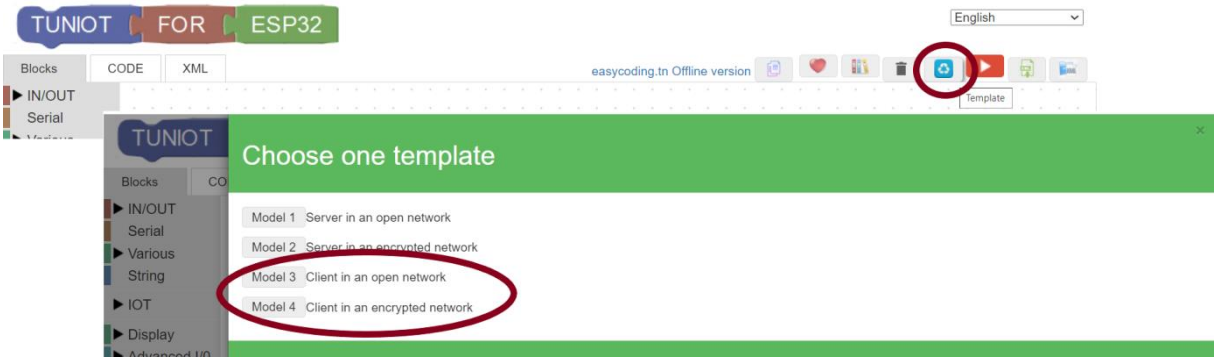
Rys. 12. Sposób podłączenia przekaźnika. Górna część (kable włączania i wyłączania lampy). Dolną część podłącz do płytki ESP32 (VCC, dowolny pin cyfrowy i pin uziemienia).

Rys. 13. Podłączenie przekaźnika.

Użyj kodu blokowego z ćwiczenia 2 do zaprogramowania lampy. Można ustawić większe opóźnienie. Wyjaśnij uczestnikom, że w ten sposób mogą kontrolować dowolne urządzenie, używając prostego programu w ramach Internetu Rzeczy. Zapytaj ich, co chcieliby kontrolować za pomocą przekaźnika (tylko urządzenia, które można włączyć i wyłączyć).

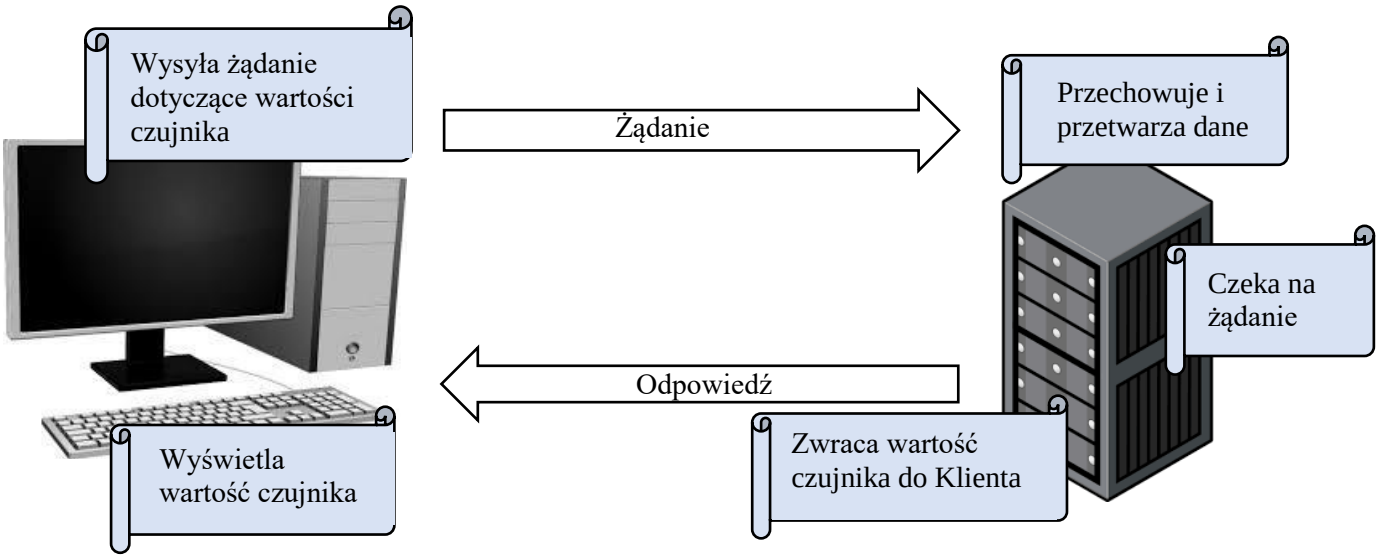
Część zaawansowana – kontrolowanie lampy przez Internet

Jeżeli nie masz czasu na wykonanie zaawansowanej części warsztatu, możesz użyć gotowego kody z zakładki SZABLONY w narzędziu TUNIOT (Rys. 14) i przejść do ćwiczenia 9.



Rys. 14. Importowanie szablonu TUNIOT. Wybierz Model 3, jeśli Twoja sieć jest niezabezpieczona hasłem lub Model 4, jeśli musisz wpisać hasło, by się połączyć.

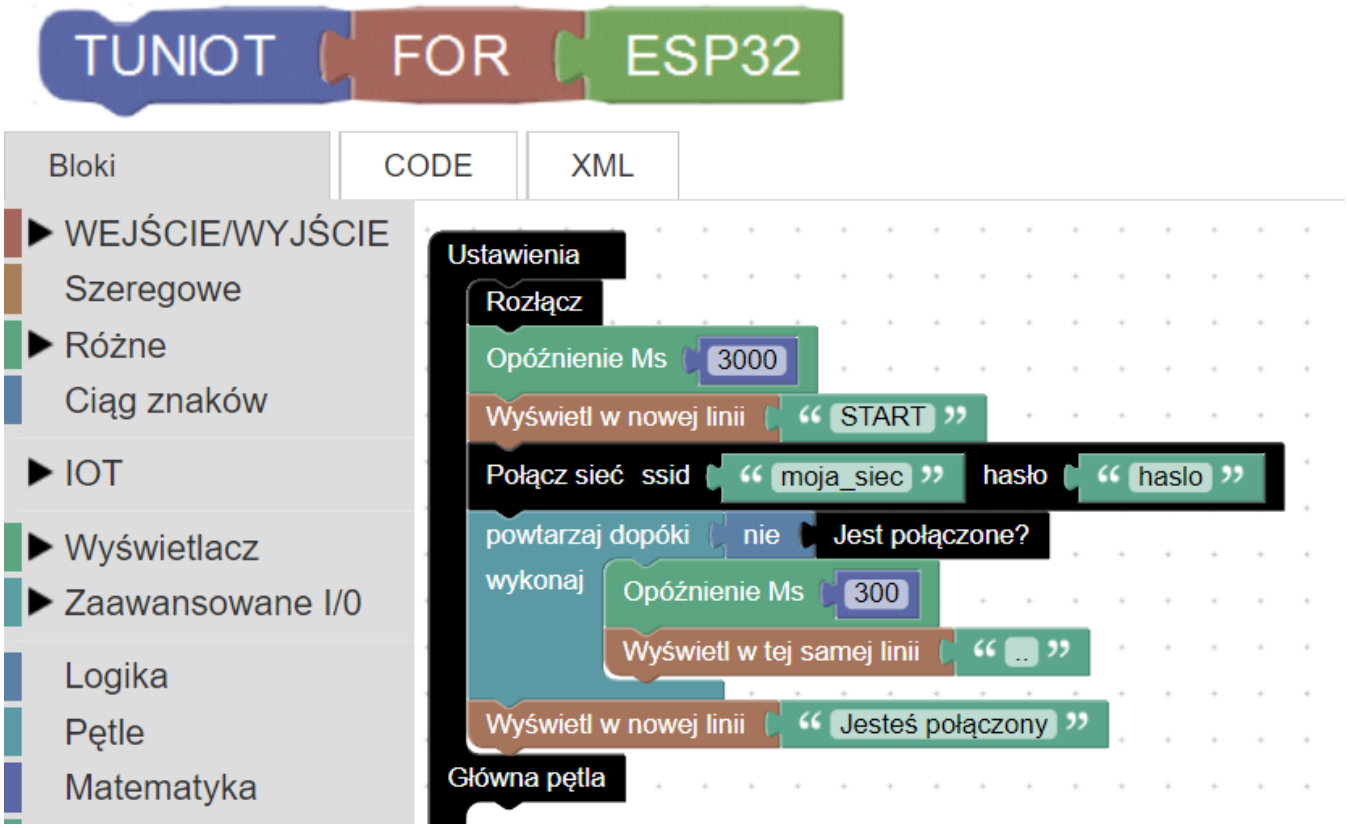
Komunikacja Klient-Serwer. Dyskusja.
Wyjaśnij, czym jest komunikacja klient-serwer. Możesz użyć analogii sprzedawcy sklepowego, który czeka na informację od klienta. Przydatny może być też poniższy schemat.



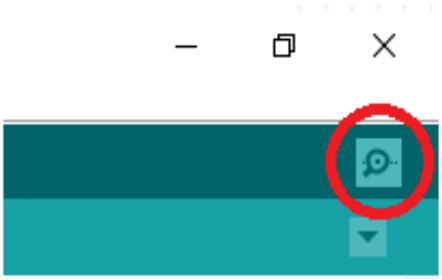
Co w naszym przypadku (komputer czy płytka NODEMCU) jest klientem, a co serwerem?
Podaj przykłady architektury klient-serwer:
- poczta e-mail: klientem poczty elektronicznej jest program umożliwiający tworzyć maile, przesyłać je do serwera poczty, pobierać maile z serwera. Na serwerze przechowywana jest poczta

- strony WWW: za pomocą przeglądarki klient wysyła do serwera WWW zapytania o wyświetlenie strony. Serwer w odpowiedzi wysyła wybrany plik strony WWW (plik HTML). Klient (przeglądarka) odkodowuje plik HTML.

Połączenie z siecią lokalną. Praca w grupach.
Ćwiczenie 7. Utwórz hot spot ze swojego komputera lub telefonu. Można skorzystać również z lokalnej sieci WIFI pod warunkiem, że nie jest wymagane logowanie dwustopniowe (np. poprzez przeglądarkę). W narzędziu TUNIOT do pola **Ustawienia** przenieś blok **Rozłącz** z menu **IOT -> Stacja IOT**. Polecenie to rozłącza płytkę z poprzednią siecią (jeśli jakaś była ustawiona). Następnie ustaw opóźnienie ok. 3-5 sekund. Będzie ono potrzebne do otworzenia **Monitora portu szeregowego**. Wypiszmy w **Monitorze portu szeregowego** informację „START", żeby sprawdzić, czy program działa. Następnie z menu **IOT -> Stacja IOT** wybierz blok **Połącz sieć SSID** z opcją hasła lub bez. W pierwszym polu należy wpisać nazwę sieci, a w drugiej hasło, jeśli sieć jest zabezpieczona. Kolejnym krokiem będzie utworzenie pętli „dopóki" (while), w której będziemy sprawdzać, czy udało się nawiązać połączenie. Wytłumacz, jak działa pętla „dopóki" (while): dopóki płytka nie zostanie połączona z siecią lokalną, wypisz na ekranie np. „ …". Po ukończeniu pętli (nawiązaniu połączenia) wypisz komunikat o tym fakcie (Rys. 14). Wklej kod do programu arduino1.8.13 i wgraj go w standardowy sposób. Sprawdź w **Monitorze portu szeregowego**, czy udało się połączyć z siecią. Monitorze portu szeregowego otwieramy za pomocą lupki w prawym górnym rogu programu (Rys. 15).

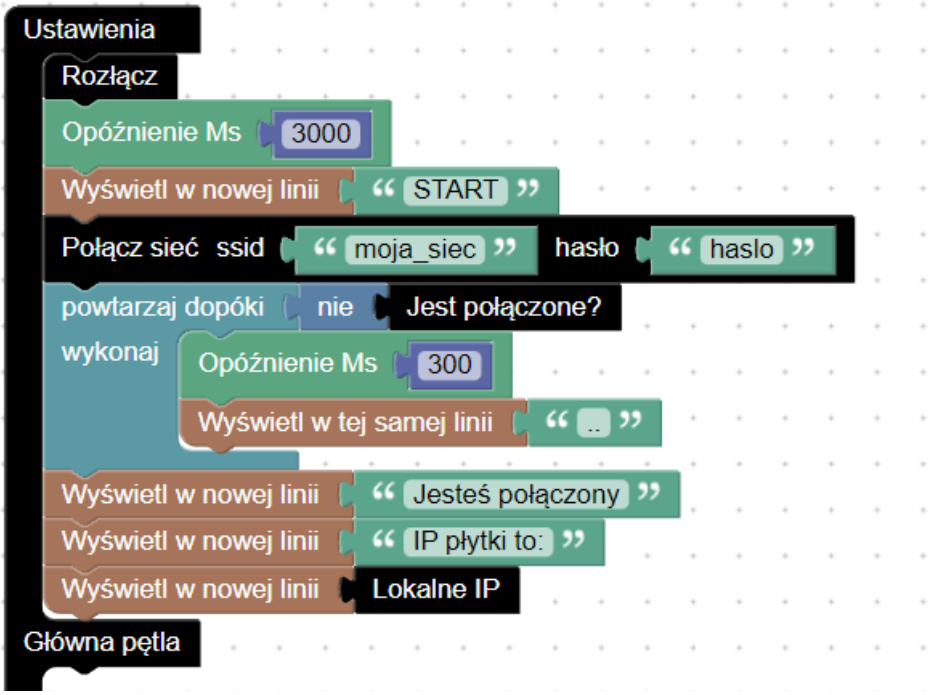


Rys. 14. Kod blokowy do ćwiczenia 7.



Rys. 15. Ikona monitoru portu szeregowego w programie arduino1.8.13.

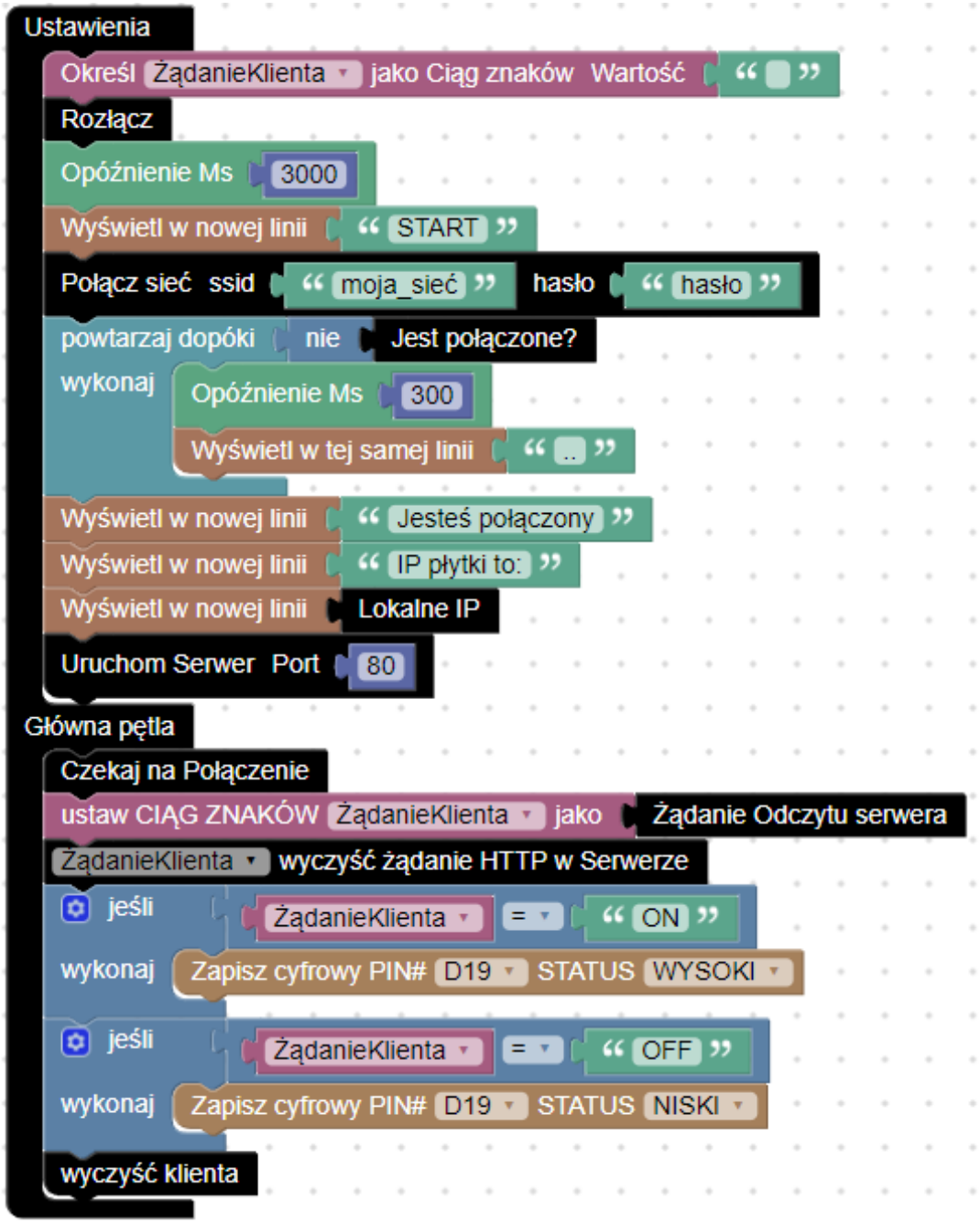
Ćwiczenie 8. Do połączenia płytki NODEMCU do sieci będziemy potrzebować jej adresu IP. Wytłumacz, że jest to parametr, po którym rozpoznaje się urządzenie w sieci. Do kodu z ćwiczenia 7 wystarczy dodać blok **Wypisz w nowej linii** wraz z podpiętymi **Lokalne IP** (Rys. 16). Dane te zostaną wyświetlone w **Monitorze portu szeregowego**.



Rys. 16. Kod blokowy do ćwiczenia 8.

Ćwiczenie 9. Nasza płytka-serwer będzie teraz oczekiwać na żądanie włączenia/wyłączenia lampy/diody. Należy zadeklarować zmienną tekstową (“string”) wiadomości klienta. Zróbmy to na początku kodu (w pętli **Ustawienia**) poprzez dodanie do kodu bloku **Określ “i” jako Ciąg znaków Wartość** z menu **Ciąg znaków**. Możemy zmienić „i” na inną nazwę np. „ŻądanieKlienta”. Dodaj pusty blok tekstowy z menu **Tekst**. W ten sposób zdefiniowaliśmy pustą zmienną.
W **Głównej pętli** dodaj blok **Czekaj na Połączenie** z menu **IOT -> Serwer IOT**. Poniżej dodaj blok **Ustaw CIĄG ZNAKÓW „ŻądanieKlienta” jako** z menu **Ciąg znaków**. Program automatycznie zmieni nazwę zmiennej na tę już zdefiniowaną. Wróć do menu **IOT -> Serwer IOT** i dodaj blok **Żądanie Odczytu serwera**. Nasza zmienna jest teraz pusta. Zostanie zastąpiona odczytem z serwera. W menu **Ciąg znaków** wybierz blok **wyczyść żądanie HTTP w Serwerze** i zamieść poniżej.

Z menu **Logika** wybierz pierwszy blok i zamieść poniżej. Wypełnij go następującymi blokami. W pierwszej linijce: blok = (z menu **Logika**). Przed znakiem '=' dodaj blok 'ŻądanieKlienta' (z menu **Ciąg znaków**), a po nim – pusty blok tekstowy z menu **Tekst**. Zmień tekst na 'ON'. W drugiej linijce (po 'wykonaj') dodaj blok **Zapisz cyfrowy PIN# D19 STATUS WYSOKI** z menu **WEJŚCIE/WYJŚCIE -> Cyfrowe**. Skopiuj pętlę „jeśli" i przyklej poniżej. Tym razem zmienimy tekst 'ON' na 'OFF' oraz status 'WYSOKI' na 'NISKI'. Ta część kodu sprawdza, co jest żądaniem klienta. Jeśli jest to 'ON', płytka (serwer) włączy diodę. Jeśli jest to 'OFF', dioda zostanie wyłączona. Ostatnią rzeczą jest dodanie bloku **wyczyść klienta** z menu **IOT -> Serwer IOT** (Rys. 17). Wyczyści on żądanie klienta, żeby program mógł zostać uruchomiony jeszcze raz.



Rys. 17. Kod blokowy do ćwiczenia 9.

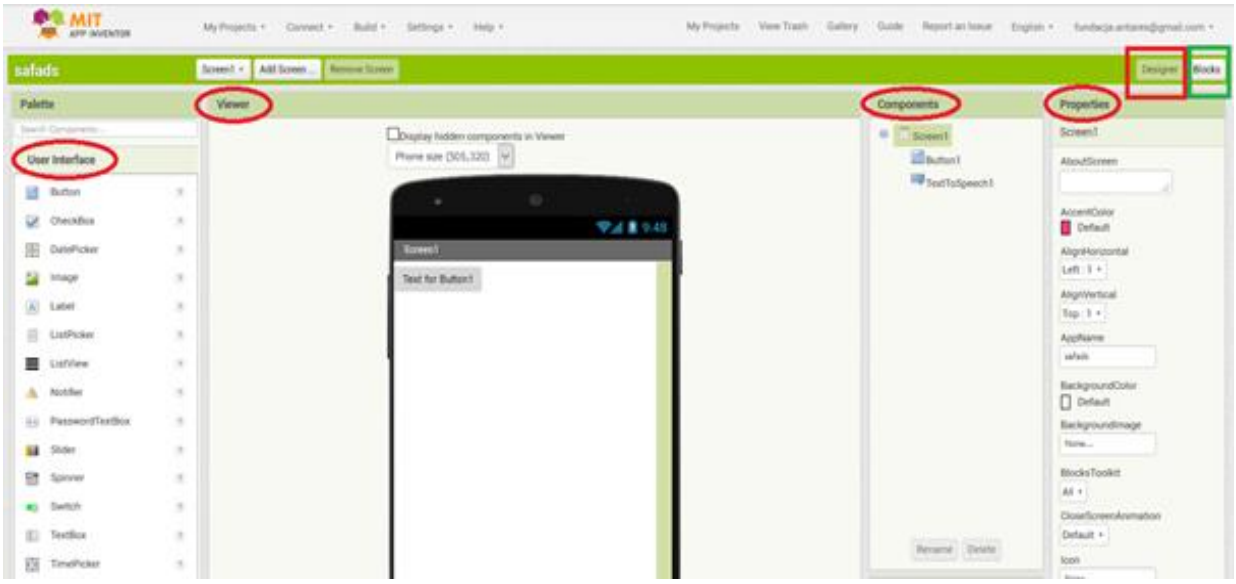
3. Kontrolowanie lampy za pomocą aplikacji mobilnej. Alfabet Morse'a:

Opowiedz uczestnikom o historii telekomunikacji. Możesz wykorzystać załączoną infografikę (załącznik 3).

Narzędzie potrzebne do warsztatów znajduje się na stronie https://appinventor.mit.edu/. Do zalogowania będzie potrzebne konto Google. Można wcześniej przygotować kilka kont dla uczestników warsztatów. Klikając w przycisk **Create Apps!** możemy się zalogować do konta, a następnie stworzyć nowy projekt (**Start new project**) pod dowolną nazwą. Poproś uczestników o zainstalowanie aplikacji MIT App Inventor 2 na ich smartfonach (z Androidem).

Ukaże nam się widok podzielony na kilka sekcji (Rys. 17): **Panel Użytkownika** (User Interface), **Podgląd** (Viewer), **Komponenty** (Components) i **Właściwości** (Properties). Wszystkie powyższe wchodzą w skład części do projektowania aplikacji (**Designer**). Z **Panelu Użytkownika** będziemy wybierać elementy aplikacji, które należy przenieść do **Podglądu**. Tutaj też będziemy widzieć, jak zmienia się aplikacja. Komponenty mogą być widoczne w aplikacji (np. przyciski) lub ukryte (np. wiadomość głosowa po kliknięciu w przycisk). W części **Właściwości** można zmieniać opcje poszczególnych komponentów.

Gdy przejdziemy do zakładki **Bloki** (Blocks) będziemy mogli tworzyć kod blokowy podobny do tego, na którym pracowaliśmy w narzędziu TUNIOT.



Rys. 17. Interfejs programu MITapp.

Ćwiczenie 10 (opcjonalne): Pierwsza aplikacja będzie się składać z przycisku, po naciśnięciu którego będzie można usłyszeć: „Hej, jestem programem". W tym celu z **Panelu Użytkownika** należy wybrać komponent **Przycisk** (Button) i przeciągnąć go do **Podglądu**. Z zakładki **Media** wybierz **TekstDoMowy** (TextToSpeech) i również przeciągnij. Ten element nie pojawi się na ekranie. Jest jednym z niewidocznych komponentów, których lista widnieje pod rysunkiem telefonu. W zakładce **Bloki** należy przygotować kod do obsługi wybranych

komponentów. Z menu **Button1** wybierz blok **when Button1.Click do**, a następnie doczep do niego **call TextToSpeech1.Speak message** z menu **TextToSpeech1**. Wiadomość należy wpisać w blok tekstowy (pierwszy blok w menu **Text**) i doczepić do kodu (Rys. 18).

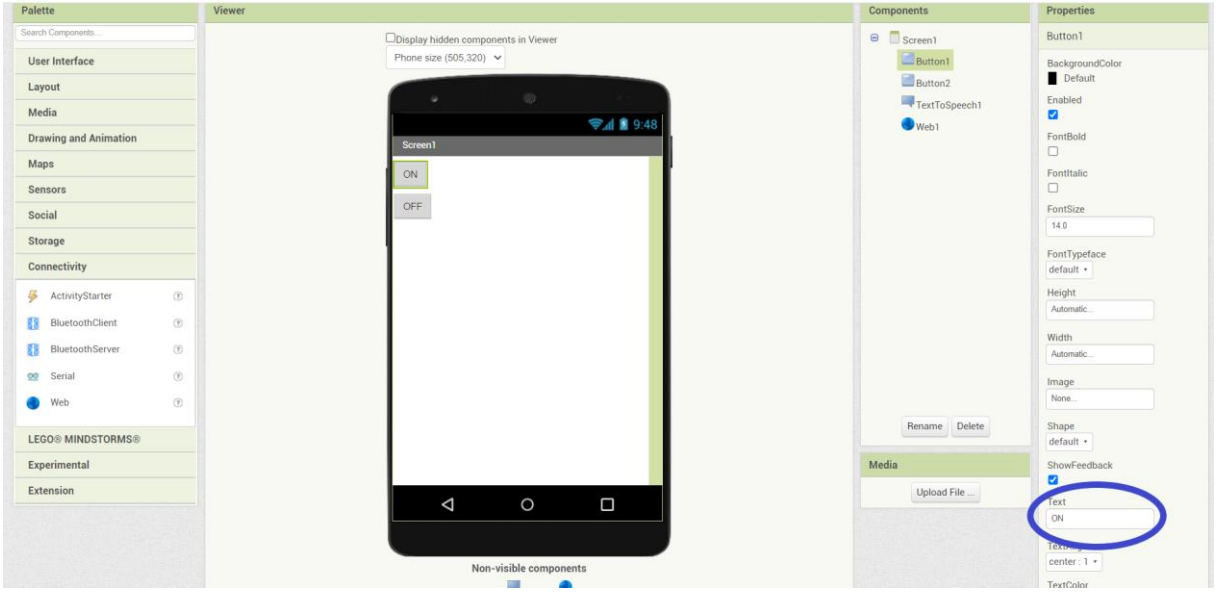
Po ukończeniu ćwiczenia na stronie internetowej z menu głównego należy wybrać **Połącz** (Connect), a następnie **AI Companion**. Na ekranie pojawi się kod QR, który można zeskanować pobraną aplikacją. Jeżeli skanowanie jest utrudnione, można również ręcznie wpisać kod składający się z 6 liter i przetestować aplikację. Należy pamiętać, że zarówno telefon jak i komputer powinny znajdować się w jednej sieci.

Ćwiczenie 11: Kolejne ćwiczenie polega kontrolowaniu diod za pomocą smartfona. Powróć do widoku Projektowania (**Designer**). Dodaj drugi przycisk i zmień nazwy przycisków na 'ON' i 'OFF'. Możesz to zrobić, wybierając **Button1** w menu **Components** oraz zmieniając pole **Text** w menu **Properties** (zaznaczone niebieskim kółkiem na Rys. 19). Dodaj kolejny ukryty component: **Web** z menu **Connectivity**. Twój widok projektowania (**Designer**) powinien wyglądać tak jak na Rys. 19.

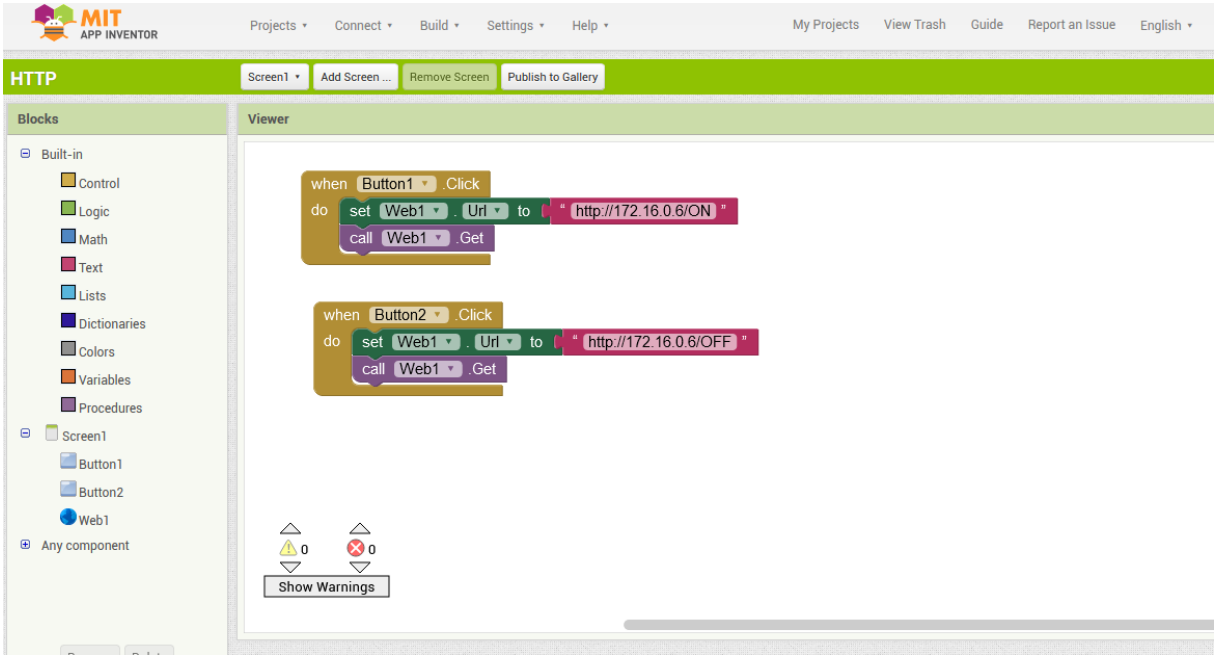
Rys. 18. Kod blokowy do ćwiczenia 10.

W widoku **Blocks** usuń fioletowy i czerwony blok, jeśli udało się wykonać ćwiczenie 10. Jeżeli pominięto ćwiczenie 10, dodaj blok **when Button1.Click do** z menu **Button1**. Przyklej do niego blok **set Web1 Url to** (z menu **Web1**) i pusty blok tekstowy z menu **Text** (to jest pierwszy blok w tym menu). Skopiuj swój adres IP to pustego bloku tekstowego. Po adresie wpisz '/ON'. Tak więc tekst w bloku powinien wyglądać przykładowo tak: http://192.169.136.80/ON. Wybierz blok **call Web1.Get** z menu **Web1** i przyklej go poniżej. Ta część kodu zmienia tekst w adresie IP na 'ON' po naciśnięciu przycisku.

Skopiuj cały kod i wklej go poniżej, zmieniając ‘Button1’ na ‘Button2’ w najbardziej zewnętrznym bloku. Zmień również ‘ON’ po adresie IP na ‘OFF’. Zapytaj uczestników, czy domyślają się, jak działa druga część kodu. Wgraj kod do aplikacji MIT App Inventor 2 tak jak w ćwiczeniu 10 (ciemnoczerwony tekst) i przetestuj.



Rys. 19. Interfejs MITapp podczas przygotowania ćwiczenia 11.



Rys. 20. Kod blokowy do ćwiczenia 11.

4. **Zakończenie:**

Użycie czujników. Praca w grupach.
Każda grupa otrzymuje obrazki z kilkoma czujnikami (wycięte z załącznika 1). Poproś uczestników, by przedyskutowali w grupach możliwości użycia tych czujników w życiu codziennym/przemyśle/komunikacji. Pod koniec ćwiczenia każda z grup prezentuje swoje pomysły.

Ewaluacja warsztatów. Dyskusja.
Zapytaj uczestników o ich wrażenia dotyczące warsztatów. Co było największą trudnością/wyzwaniem? Co pamiętają o Internecie Rzeczy? Czy potrafią wytłumaczyć ideę Internetu Rzeczy?

Rozwiązywanie problemów:

- Problem z kompilacją kodu w Adruino: za długa ścieżka do folderu z Arduino (przenieś na Pulpit lub Dysk C:)
- Zły port: w arduino-1.8.13 wybierz Narzędzia -> Port -> COMx (zazwyczaj ostatni)
- Nie podłączaj innych urządzeń (np. telefonu, myszki) do innych portów USB
- Komputer powinien być podłączony do prądu
- Sprawdź komponenty, czy są sprawne
- Sprawdź kabel USB – tanie kable szybko się psują
- Sprawdź podpięcie kabli: czy pin uziemienia jest podpięty
- Sprawdź kod: czy wybrano właściwy pin, czy nie zapomniano o opóźnieniu (częsty błąd)
- Nie zapomnij skasować wszystkich starych bloków w kodzie
- Monitor portu szeregowego powinien zostać zamknięty przed uruchomieniem kolejnego programu

Załącznik 1:


czujnik ciśnienia powietrza


czujnik nachylenia (kąta)


czujnik nacisku


czujnik ph


czujnik przepływu cieczy


czujnik ruchu

 czujnik podczerwieni

 czujnik wilgotności

 czytnik linii papilarnych

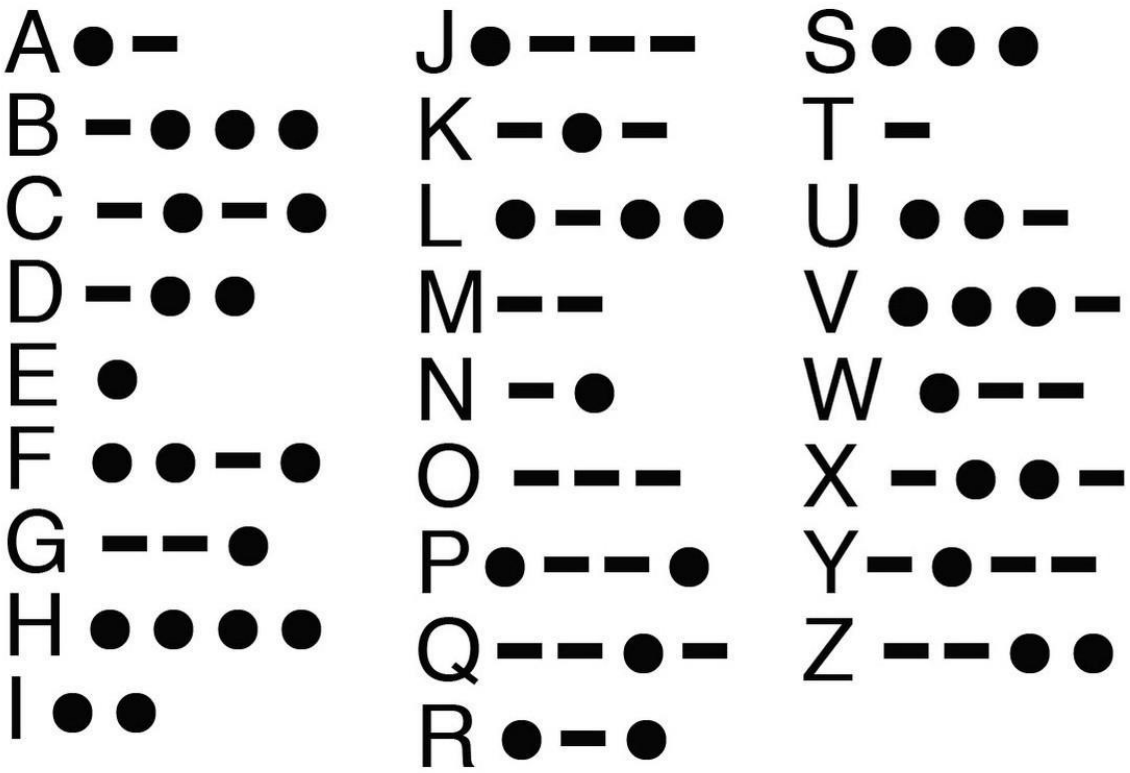
 detektor dźwięku

 czujnik światła i koloru

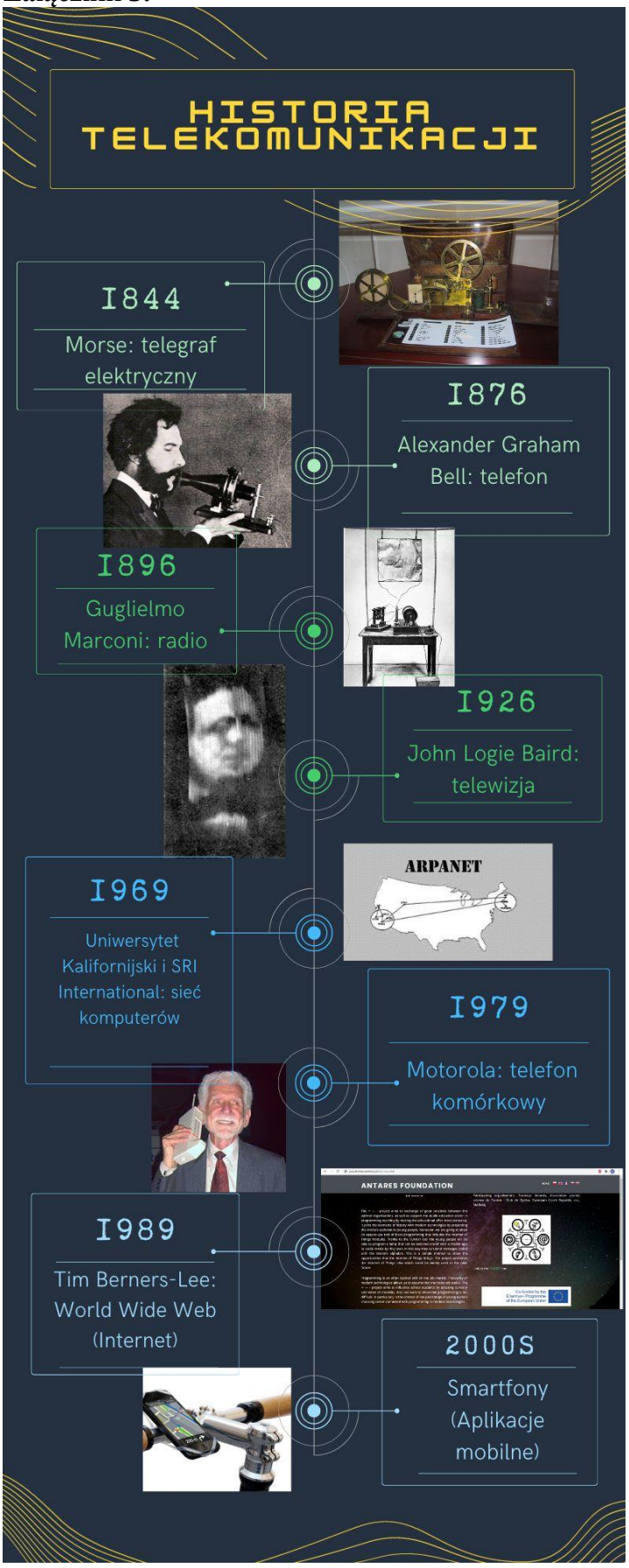
 czujnik temperatury

 czujnik wibracji

Załącznik 2:



Załącznik 3:


HISTORIA TELEKOMUNIKACJI
1844
Morse: telegraf elektryczny
1876
Alexander Graham Bell: telefon
1896
Guglielmo Marconi: radio
1926
John Logie Baird: telewizja
ARPANET
1969
Uniwersytet Kalifornijski i SRI International: sieć komputerów
1979
Motorola: telefon komórkowy
ANTARES FOUNDATION
1989
Tim Berners-Lee: World Wide Web (Internet)
2000S
Smartfony (Aplikacje mobilne)