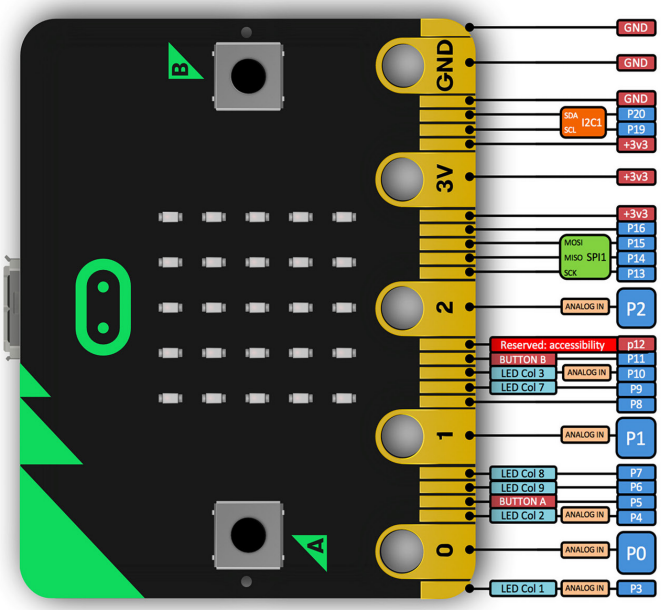
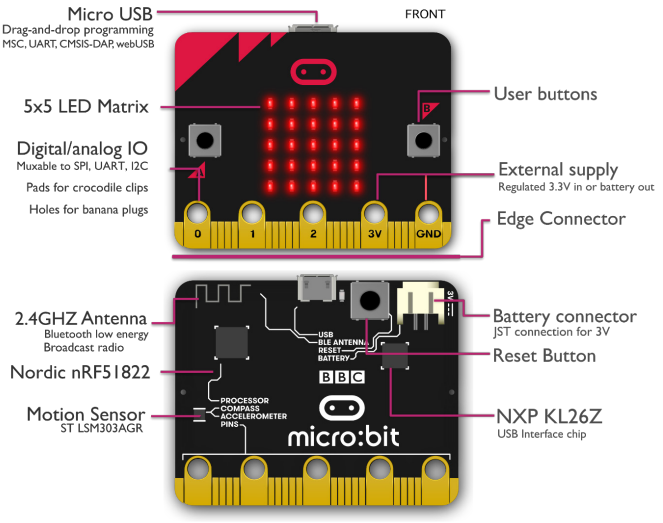


Pomiar pulsu z użyciem płytki micro:bit

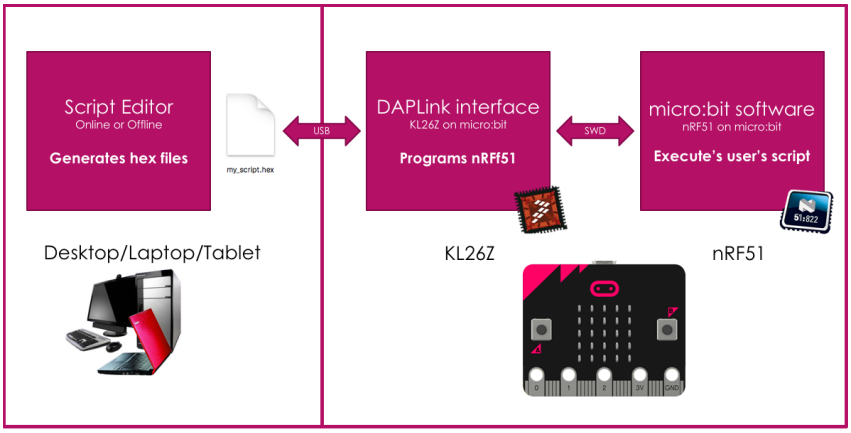
Micro:bit to niewielka, bogato wyposażona płytka ewaluacyjna, dzięki której każdy może szybko zaimplementować prototypy urządzeń wykonujących naprawdę zaawansowane funkcje, bez poświęcania wielu godzin na pisanie programu. W prezentowanym projekcie pokazano praktyczne zastosowanie płytki micro:bit – jako urządzenie do pomiaru pulsu.

Micro:bit to programowalna „płytka rozwojowa" zaprojektowana przy udziale BBC i Nordic Semiconductor. Zawiera 32-bitowy procesor oraz wiele urządzeń peryferyjnych (**rysunek 1**): wyświetlacz z 25 diodami LED, przyciski, interfejsy analogowe i cyfrowe, czujnik temperatury i światła, akcelerometr, kompas, moduł transmisji radiowej, Bluetooth i interfejs USB. Jednak najważniejszy jest cel, jaki realizuje – micro:bit to moduł, z którego mogą korzystać dzieci i młodzież podczas nauki programowania i projektowania układów elektronicznych, nauki robotyki, budowy gier i prototypowania. Wraz z webowym i natywnym środowiskiem programistycznym może być obsługiwany nawet przez osoby, które nie mają pojęcia o programowaniu, a chcą od czegoś zacząć.

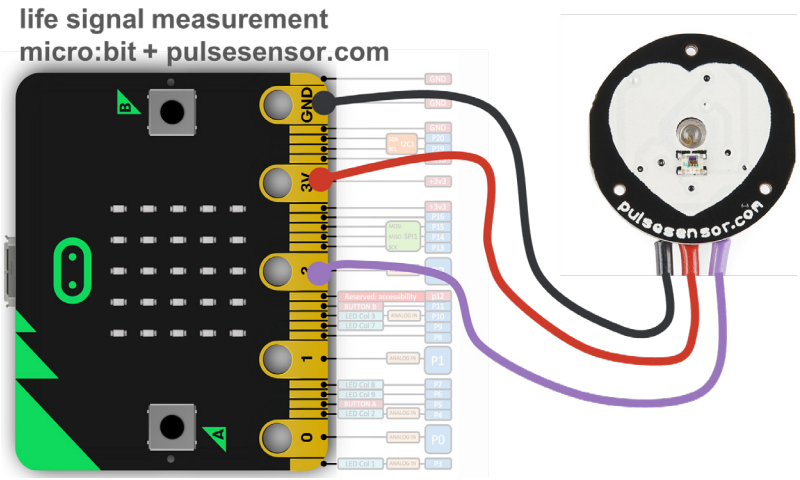
Podłączoną do komputera, za pomocą kabla USB, płytkę programujemy, dosłownie „wrzucając" napisane programy tak, jak na pamięć USB (**rysunek 2**). Program możemy napisać w przeglądarce internetowej lub w dedykowanej aplikacji, przy użyciu jednego z kilku języków programowania: graficznego blokowego języka programowania MakeCode (takiego jak Scratch), JavaScrip, microPython lub C++. Sceptycy powinni wiedzieć, że korzystając z MakeCode, każdy może szybko zaimplementować prototypy urządzeń wykonujących naprawdę zaawansowane funkcje, bez poświęcania wielu godzin na programowanie. Urządzenie doskonale nadaje się zarówno do nauki programowania dla najmłodszych, jak i do eksperymentowania w zaciszu domowych czy uniwersyteckich laboratoriów. Jeśli potrzebujesz kontrolera do laboratoryjnego prototypu i nie jesteś elektronikiem i programistą embedded, to micro:bit jest właśnie dla Ciebie.



Rysunek 1. Peryferia płytki micro:bit



Rysunek 2. Programator wbudowany w płytkę micro:bit



Rysunek 3. Sposób dołączenia czujnika pulsu

Pomiar pulsu

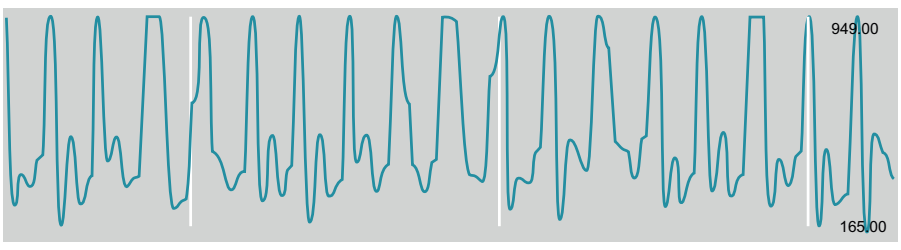
Do zbudowania monitora pulsu użyłem płytki micro:bit oraz optycznego detektora tętna pulsesensor.com. Lista wszystkich potrzebnych komponentów jest następująca:

1. Płytka micro:bit – ***https://bit.ly/2LW2cXV***,
2. Czujnik pulsu PulseSensor – ***https://bit.ly/2XrtRoN***,
3. Trzy przewody do płytek z zakończeniem żeńskim z jednej strony i haczykiem z drugiej,
4. Zasilacz z wtyczką USB micro-B (czyli zasilacz używany do ładowania wielu telefonów komórkowych),
5. Opcjonalnie adapter do układu micro:bit, by łatwiej było podłączyć przewody.

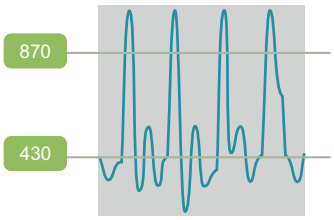
Optyczny detektor pulsesensor.com przymocowuje się do palca. Urządzenie prześwietla opuszek palca intensywnym światłem i monitoruje, jaka jego ilość powraca do detektora. Krew przepływająca w naczyniach krwionośnych pochłania światło. Możemy zaobserwować, że w jednej chwili do detektora powraca więcej światła, a w innej chwili mniej. Tak wygląda obserwacja przebiegu tętna.

Detektor ma 3 wyprowadzenia. Dwa służą do zasilania, a jeden jest wyjściem analogowym, na którym pojawia się sygnał napięciowy proporcjonalny do poziomu światła odbieranego przez detektor. Wyjście analogowe można podłączyć bezpośrednio do interfejsów analogowych mikrokontrolerów. Ja podłączyłem sygnał do płytki micro:bit tak jak na schemacie z **rysunku 3**.

Micro:bit może dzielić zasilanie z urządzeniami peryferyjnymi, więc poprzez przewody doprowadzone jest także zasilanie detektora optycznego. Interfejs analogowo-cyfrowy P2 płytki micro:bit odczytuje sygnał reprezentujący tętno.



Rysunek 4. Przebieg sygnału z czujnika pulsu



Rysunek 5. Wyznaczone wartości graniczne potrzebne do wykrywania impulsów

Uwaga: detektor pulsosensor.com jest bardzo wrażliwy na ruch (ruch generuje zakłócenia). Ważny jest też wybór miejsca pomiaru pulsu i nacisk przy mocowaniu detektora.

Działanie programu

Na stronie internetowej ***https://bit.ly/2ZwQhaV*** można znaleźć najprostszy algorytm odczytu sygnału tętna z czujnika, przeznaczony dla micro:bit. W prezentowanym projekcie pokazałem więcej możliwości tego układu. Dzięki połączeniu USB pomiędzy micro:bit a komputerem PC możliwa jest graficzna prezentacja sygnału pulsu. Środowisko programistyczne micro:bit umożliwia odbieranie danych z USB i prezentowanie ich w konsoli przeglądarki lub w dedykowanym programie w postaci liczb i wykresów. Możliwe jest też zapisywanie odebranych danych do plików CSV, dla dalszej analizy. W naszym przypadku na wykresie prezentowanym w konsoli można zaobserwować przebieg mierzonego pulsu. Natomiast wartość pulsu jest prezentowana na wyświetlaczu LED. Na **rysunku 4** pokazano wykres sygnału analogowego, który jest odbierany z czujnika. Wykres reprezentuje przebieg mierzonego tętna. Zaobserwowany przebieg nie jest tak precyzyjny jak na monitorach EKG, ale daje możliwość obserwacji i pomiaru pulsu.

Sygnał elektryczny na wejściu analogowym micro:bit może mieć wartość w zakresie 0...3 V. Natomiast zmienna reprezentująca wartość odczytywanego sygnału może mieć wartość całkowitą z przedziału od 0 do 1023. Dla przebiegu z rysunku 4 określono minimalną i maksymalną wartość zarejestrowanego sygnału. Sygnał ten obniżył się do minimalnej wartości 165 i osiągnął maksymalną wartość 949. Metodą prób i błędów zdefiniowano dwa decyzyjne poziomy sygnału odbieranego z pulsometru (**rysunek 5**):

1. Gdy poziom sygnału z pulsometru wzrośnie powyżej wartości 870, program zapisuje aktualny czas tego zdarzenia, uznając, że wykryty został impuls. Od tego ważnego momentu program nie będzie poszukiwał kolejnego wzrostu sygnału powyżej poziomu 870, aż do czasu, gdy poziom sygnału z pulsometru spadnie poniżej poziomu 430.

2. Gdy sygnał obniży się poniżej poziomu 430, program znów będzie czekał, aż poziom sygnału wzrośnie powyżej 870, by zapisać czas pojawienia się kolejnego impulsu. Na tej podstawie obliczany jest upływ czasu (delta) pomiędzy pierwszym wykrytym impulsem, a kolejnym. Wyliczona delta umożliwi, w następnym kroku, wyznaczenie pulsu.

W sygnale z pulsometru często obserwowane są zakłócenia. Po wzroście sygnału powyżej poziomu 870 sygnał może, w krótkim czasie, wielokrotnie opadać nieznacznie poniżej tego poziomu i znów wzrastać powyżej. Taki rodzaj zakłóceń mógłby powodować błędne wyniki. Dlatego niezbędne było określenie drugiego progu decyzyjnego o wartości 430, poniżej którego musi spaść sygnał, zanim rozpocznie się oczekiwanie na kolejny impuls tętna (**rysunek 6**).

Algorytm precyzyjnego obliczania tętna z pewnością może być bardziej dokładny. Zaprezentowany w projekcie sposób wybrano jako kompromis pomiędzy jakością pomiaru pulsu a stopniem skomplikowania algorytmu i liczbą warunków w programie. Metoda wykrywania impulsu i zapisywania czasu, w którym impuls został wykryty, umożliwia obliczenie upływu czasu pomiędzy dwoma następującymi po sobie impulsami (**rysunek 7**) z prostego wzoru:

delta_t = time2 – time1

gdzie:

time2 – to moment, w którym wykryto najnowszy impuls,

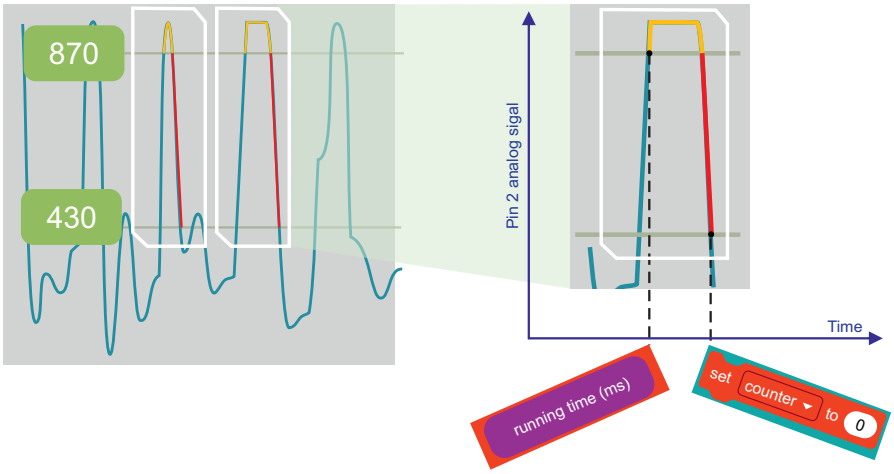
time1 – to moment, w którym wykryto impuls poprzedni.

Po obliczeniu upływu czasu pomiędzy impulsami możliwe jest obliczenie pulsu, który jest definowany jako liczba uderzeń serca podczas jednej minuty. Funkcja *Running time*, zastosowana w programie, zapisuje czas, w którym została wywołana, wyrażony w milisekundach. Obliczony upływ czasu również wyrażony jest w milisekundach. Dla poprawności obliczeń dzielimy liczbę 60 000 (jedna minuta to 60 000 milisekund) przez wyliczony upływ czasu, czyli *delta_t*. Na **rysunku 8**, który pokazuje algorytm blokowy, można zobaczyć sposób obliczania pulsu na podstawie upływu czasu. Zmienne w blokowym środowisku programowania micro:bit mogą przyjmować tylko wartości całkowite. Dlatego przy obliczaniu wyniku dzielenia uwzględniono poprawkę na resztę z dzielenia.

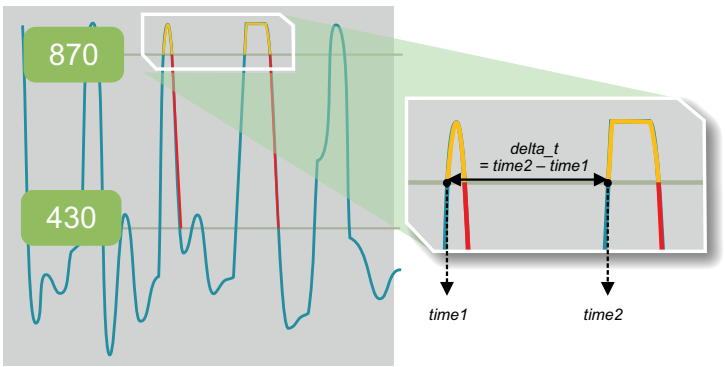
Na **rysunku 8** został pokazany program w postaci graficznej, podzielony na 5 głównych bloków. Jeden wykonywany jest przy starcie aplikacji. Cztery pozostałe to pętle, które wykonywane są równolegle i, co mało eleganckie, nie zawierają żadnych pauz. Na **rysunku 9** pokazano listę używanych w programie zmiennych. Choć zmienne w programie blokowym nie wymagają specjalnej formy deklaracji, to dobrze wcześniej przemyśleć ich liczbę i nazwy. Na **listingu 1** pokazany jest ten sam program w języku JavaScript.

Programiści mogą szybko przełączać się pomiędzy widokiem programowania blokowego a widokiem umożliwiającym podgląd i programowanie w języku JavaScrip. Oba tryby pracy można łączyć. Jeśli brakuje nam jakiegoś bloku i łatwiej taką funkcję napisać w JavaScript, to można właśnie tak zrobić. Dopisana funkcja czy linia kodu, po przełączeniu do trybu programowania blokowego, będzie zaprezentowana na oddzielnym bloku włączonym sekwencyjnie do całego programu.

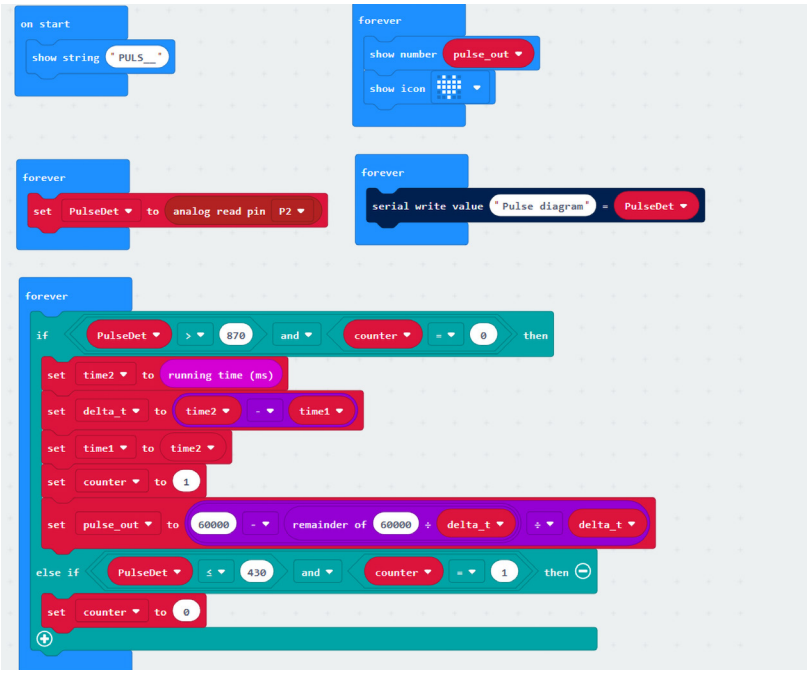
Blok „On Start" to fragment programu wykonywany tylko raz jako pierwszy zaraz po uruchomieniu urządzenia. Zawiera funkcję pokazującą na wyświetlaczu LED



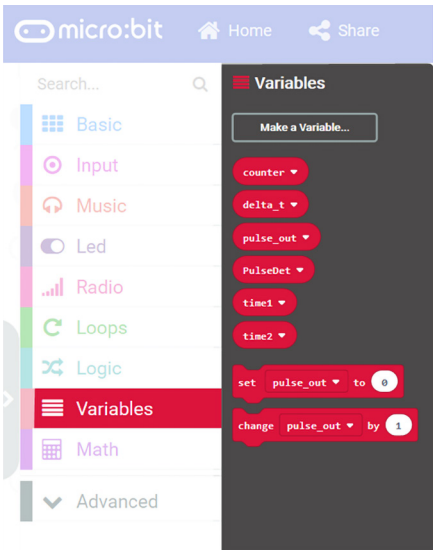
Rysunek 6. Działanie metody niwelowania zakłóceń



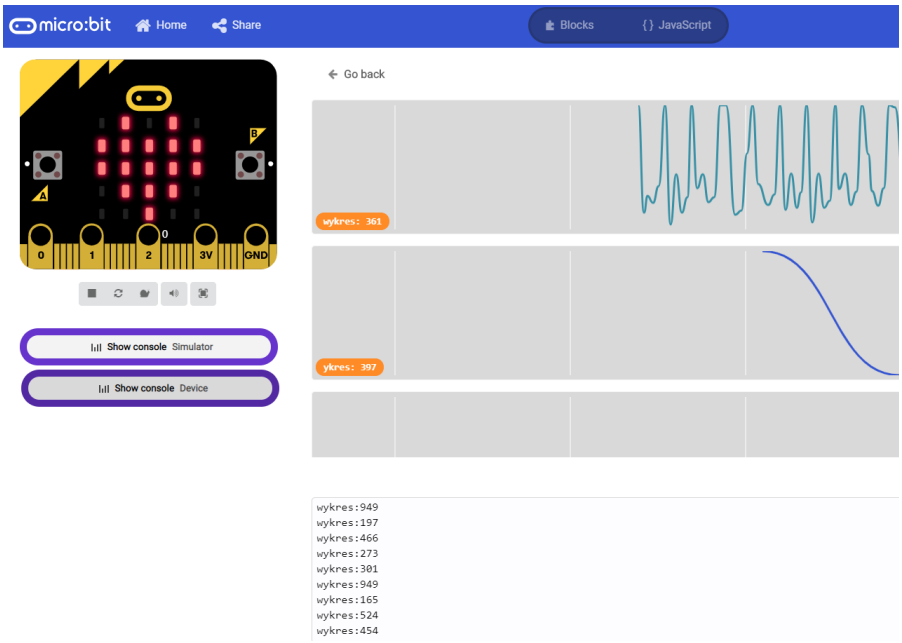
Rysunek 7. Sposób obliczania czasu pomiędzy impulsami



Rysunek 8. Graficzna postać programu sterującego



Rysunek 9. Zmienne użyte w programie



Rysunek 10. Konsola środowiska programistycznego

napis powitalny „Puls____”. Pozostałe cztery bloki programu to pętle, które wykonywane są „równolegle”.

Pierwsza pętla odpowiada za odczyt wartości sygnału z pulsometru. Jest skonstruowana z bloku „forever”, czyli jest wykonywana nieprzerwanie. Czujnik jest podłączony do interfejsu analogowo-cyfrowego na pinie P2 układu micro:bit. Funkcja „analog read pin P2” umożliwia odczyt wartości tego sygnału. Odczytana wartość zapisywana jest w zmiennej *PulseDet*.

Druga pętla jest wykonywana jako jeden z trzech scenariuszy. Scenariusze są wybierane w zależności od wyniku instrukcji warunkowej *if*. Pierwszy scenariusz jest wykonywany przy jednoczesnym spełnieniu dwóch warunków:

- wartość zmiennej *counter* jest równa 0,
- wartość sygnału z pulsometru zapisana w zmiennej *PulseDet* jest większa od 870.

Spełnienie obu warunków oznacza, że wykryto rosnące zbocze impulsu. Czas wystąpienia tego zdarzenia jest zapisywany w zmiennej *time2*, a czas poprzedniego impulsu jest przenoszony do zmiennej *time1*. W kolejnym kroku wyliczana jest *delta_t*, upływ czasu pomiędzy impulsami. Warto zauważyć, że po uruchomieniu programu pierwszy wyliczony puls obarczony będzie błędem, ponieważ zmienna *time1* będzie zawierała wartość 0. W kolejnym kroku zmienna *counter* uzyskuje wartość 1. Dzięki temu upływ czasu pomiędzy impulsami i puls nie będą obliczane, dopóki wartość sygnału nie spadnie poniżej 430. Dopiero wtedy wartość zmiennej *counter* znów przyjmie 0.

Następnie wyliczany jest puls. Do zmiennej *pulse_out* zapisywana jest wartość reprezentująca puls, wyliczona jako liczba czasów pomiędzy dwoma impulsami, jaka zmieści się w jednej minucie. Wynikiem operacji dzielenia może nie być liczba całkowita. Dlatego w programie dodano poprawkę, która pozwala uniknąć błędu zapisu liczby zmiennoprzecinkowej do zmiennej *pulse_out* Pozbycie się ułamków było możliwe z wykorzystaniem funkcji *remainder of* wyliczającej resztę z dzielenia. Wartość ta jest odejmowana od 60000 milisekund a następnie wynik jest dzielony przez zmienną *delta_t*. Dzięki temu do zmiennej *pulse_out* zawsze trafi wartość całkowita. Po wyliczeniu pulsu program kończy wykonywanie pierwszego scenariusza.

Drugi możliwy scenariusz dla pętli nr 2 będzie wykonany przy spełnieniu warunków:

- wartość zmiennej *PulseDet* reprezentującej poziom sygnału z pulsometru jest mniejsza lub równa 430,
- wartość zmiennej *counter* wynosi 1.

Wartość zmiennej *counter* ustalana jest na 1 w chwili, w której sygnał z pulsometru przekroczył wartość 870. Dla wyeliminowania zakłóceń czas kolejnego impulsu nie będzie zapisywany, dopóki wartość sygnału nie obniży się do 430. Przy spełnieniu tych warunków wartość zmiennej *couter* ustawiana jest na 0. Od tego momentu znów możliwa jest rejestracja czasu wystąpienia impulsu, którego poziom wzrośnie powyżej 870.

Trzeci scenariusz wystąpi, gdy żaden z wcześniejszych scenariuszy nie zostanie wykonany, a jego zadanie to ponowne wykonanie pętli od początku.

Kolejny blok programu to trzecia pętla odpowiedzialna za wyświetlanie wartości pulsu na wyświetlaczu LED płytki micro:bit. Na ekranie prezentowana jest wartość zmiennej *pulse_out* na przemian z ikoną obrazującą serce.

Ostatni blok programu realizuje przesyłanie wartości sygnału pulsometru do komputera PC lub innego urządzenia podłączonego do USB płytki micro:bit. W nieskończonej pętli funkcja *serial write value* wysyła tekst zawierający ciąg znaków składający się ze słów: „Pulse diagram”, oraz wartości ze zmiennej *PulseDet*. Dzięki temu w konsoli środowiska programistycznego można obserwować wykres prezentujący zmienność poziomu sygnału z pulsometru w funkcji czasu. Widok konsoli można uruchomić, klikając na przycisk Show console device (**rysunek 10**). Przycisk ten pojawia się tylko wtedy, gdy program korzysta z funkcji *serial write value* oraz gdy bicro:bit podłączony jest do komputera. Pełny kod programu dostępu jest pod adresem ***https://bit.ly/2zjGXN1***.

Jan Ekiel
jan.ekiel@likims.com

Listing 1. Program realizujący pomiar pulsu, zapisany w języku JavaScript

```
let time1 = 0
let delta_t = 0
let time2 = 0
let PulseDet = 0
let pulse_out = 0
basic.showString(„PULS__”)
let counter = 0
basic.forever(function () {
    basic.showNumber(pulse_out)
    basic.showIcon(IconNames.Heart)
})
basic.forever(function () {
    PulseDet = pins.analogReadPin(AnalogPin.P2)
})
basic.forever(function () {
    serial.writeValue(„Pulse diagram”, PulseDet)
})
basic.forever(function () {
    if (PulseDet > 870 && counter == 0) {
        time2 = input.runningTime()
        delta_t = time2 - time1
        time1 = time2
        counter = 1
        pulse_out = (60000 - 60000 % delta_t) / delta_t
    } else if (PulseDet <= 430 && counter == 1) {
        counter = 0
    }
})
```