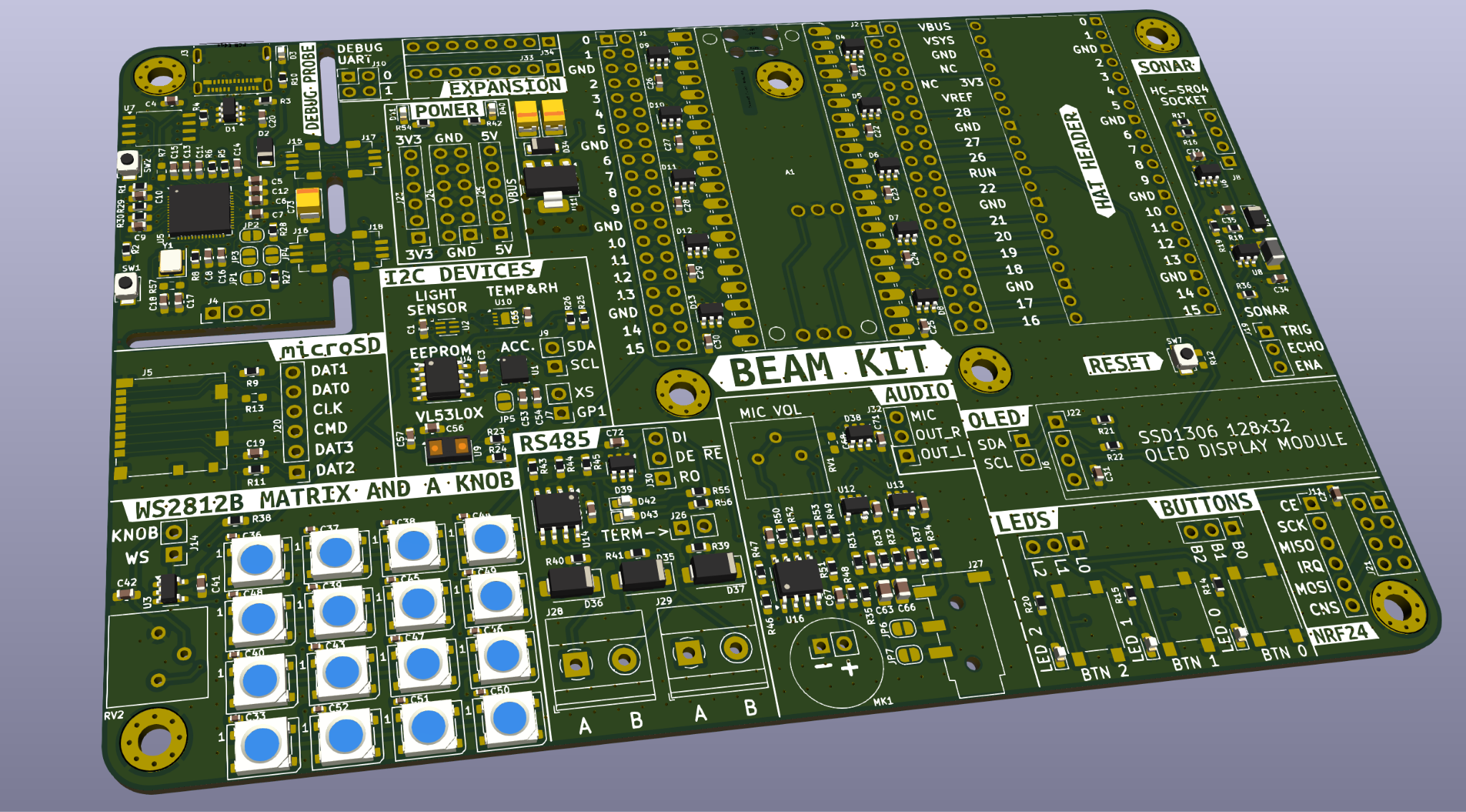
# Środowisko pracy: sprzęt

Zestaw dydaktyczny to płyta bazowa z miejscem na moduł [Pico 2 W](https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html) oraz wbudowanym debuggerem/programatorem. Do realizacji niektórych prostszych zadań możliwe jest używanie samego modułu Pico 2 W bez płyty bazowej, ale trudniej jest wtedy debugować projekty C/C++. Możliwe jest bezpośrednie połączenie kilkoma przewodami modułu Pico 2 W z oficjalnym programatorem lub innym układem Pico (np. podstawowym modułem Pico 1 RP2040) skonfigurowanym jako debugger/programator.

* <https://www.raspberrypi.com/products/debug-probe/>
* <https://datasheets.raspberrypi.com/pico/pico-2-product-brief.pdf>

Do typowej pracy na zajęciach z językiem C/C++ wystarczy jeden przewód USB-C - wtedy ze złącza programatora zasilana jest cała płyta bazowa, a dodatkowo programator ma wbudowany konwerter USB-UART.  
Z językiem micropython wygodniej będzie użyć złącza microUSB umieszczonego na module Pico2W, bo dodatkowy programator/debuger nie jest wtedy niezbędny.   
Jest więc kilka możliwości do wyboru - jeśli chcesz potrenować w domu niewielkim kosztem, to za 20 PLN kupisz Pico1 (będzie służył jako debugger), a na Pico 2 W wydasz 36 PLN. Dodatkowe kilkanaście PLN to płytka stykowa, goldpiny i przewody połączeniowe - koszt całości powinien się zamknąć w 70 PLN.

Planowaliśmy używać gotowych zestawów <https://www.waveshare.com/wiki/Pico-Sensor-Kit-B> na zajęciach, ale pod koniec lutego dystrybutor anulował całe zamówienie i dlatego zdecydowaliśmy wspólnie z głównym wykonawcą Jakubem Mnichem zaprojektować od podstaw i zlecić wykonanie autorskich zestawów [BEAM KIT](https://github.com/jmnich/beam_kit_1) z wbudowanym programatorem:



BEAM KIT jest szyty konkretnie pod kursy embedded na naszym wydziale, i jeśli nie popełniliśmy krytycznych błędów w trakcie weekendowego maratonu projektowego, to można będzie używać takich bajerów jak:

* **wbudowany programator ze złączem USB-C**, które od razu zasili płytę bazową i Pico2W,
* gniazdo na moduł Pico1 lub Pico2,
* **interfejs RS485 (do modbus RTU),**
* mikrofon + DAC (wyjście liniowe),
* OLED na I2C,
* sonar ultradźwiękowy,
* laserowy czujnik odległości VL53L0X,
* cyfrowy czujnik natężenia światła,
* cyfrowy termometr i wilgotnościomierz,
* akcelerometr i żyroskop (6DoF),
* czytnik microSD,
* 16 adresowalnych RGB LED WS2812B w układzie 4x4,
* potencjometr,
* 3 x user LED, 3 x user BUTTON
* gniazdo popularnego modułu radiowego NRF24 (ale także niektórych modułów LoRa),
* osobny header do podłączania dodatkowych nakładek (HAT) dedykowanych dla Pico,
* gniazdo 1x8 w rastrze 2.54 mm do podłączania czegokolwiek (czujniki, klawiatury matrycowe, …),
* zabezpieczenia ESD (TVS) na większości linii sygnałowych.

No i to co jest ważne w modułach W to układ do WiFi, z którego też będziemy korzystać w dalszej części kursu.

Staraliśmy się wyposażyć te zestawy w najważniejsze peryferia tak, aby w trakcie zajęć ograniczyć ilość dodatkowego i często zawodnego okablowania. Wszędzie na PCB już są doprowadzone linie zasilające peryferia, więc zmniejszamy ryzyko popełnienia krytycznego błędu w trakcie podłączania wbudowanych peryferiów do MCU.

Z wszystkimi wymienionymi peryferiami (oraz wielu innymi) możesz też samodzielnie eksperymentować w domowym zaciszu. Typowy koszt dodatkowego układu peryferyjnego, który można podłączyć do Pico2W kilkoma przewodami, wynosi 5-10 PLN, i są to te same układy które mogą być podłączane do Arduino czy ESP32.

Celowo nie używamy egzotycznych układów, więc znalezienie gotowych rozwiązań związanych z ich uruchomieniem i użyciem we własnym projekcie jest dość łatwe.

Skopiowanie gotowego projektu z github jest na dobry początek, ale postaram się tak zdefiniować problemy do rozwiązania, żebyście łatwo trafiali na gotowce z błędami, które trzeba będzie znaleźć i poprawić.

Punktowane będą też zadania związane z integracją - o ile np. nie jest wyzwaniem wyświetlenie w terminalu czy nawet na OLED wartości temperatury i wilgotności z SHT30, to wysłanie tych wartości do serwera Apache oraz odpowiedniego rozkazu magistralą modbus RTU będzie wymagało pewnego wysiłku większego niż użycie Ctrl+C/Ctrl+V.

Zadania dotyczące komunikacji UDP, ModBus RTU, LoRa, czy integracji z Home Assistant mogą być realizowane we własnym zakresie, ale prawdopodobnie wygodniej i szybciej będzie skorzystać z gotowych środowisk deweloperskich, które udostępnię na zajęciach.

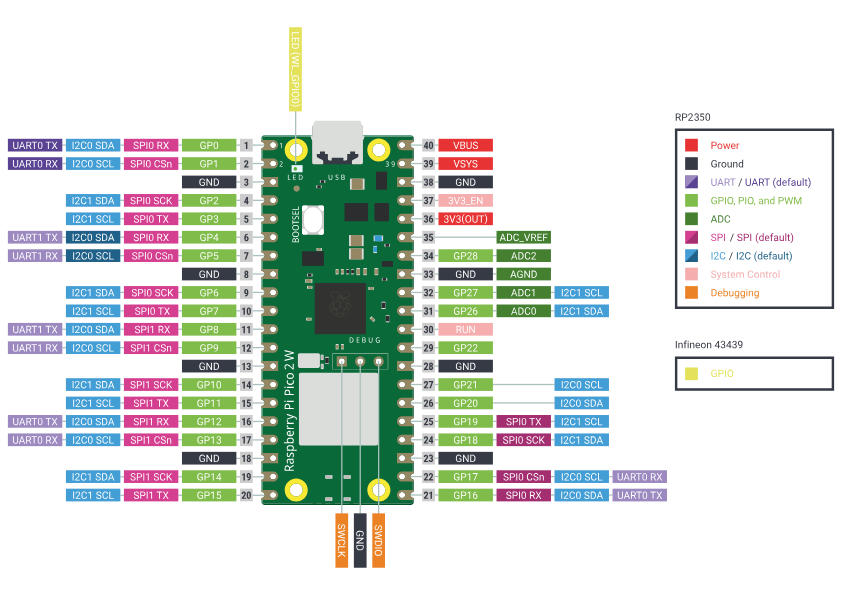
## Możliwe niespodzianki

Jest kilka różnych oficjalnych wersji Raspberry Pi Pico (np. 1, 1W, 2, 2W), do tego dochodzą wersje innych producentów. **Na zajęciach używamy Pico2W**. Przegląd dostępnych oficjalnych wersji jest np. tutaj:   
<https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html>

* Możliwą niespodzianką jest to, że wersje “W” z wbudowanym modułem bezprzewodowym nie mają łatwo dostępnej diody świecącej na pokładzie, zamiast tego dioda jest sterowana pośrednio przez CYW43 - nie zadziałają więc proste przykłady migania diodą przygotowane dla Pico1.
* Jest też kilka niespodzianek spowodowanych użyciem dość nowego MCU RP2350 zamiast starszego RP2040. Niektóre przykłady przeznaczone dla RP2040 (Pico1) oraz niektóre funkcje oficjalnego SDK nie działają z nowszym rdzeniem, nie jest ich wiele, ale warto na to zwracać uwagę.
* Niektóre chińskie zamienniki oficjalnego PiPico mają identyczny footprint, ale inny układ wyprowadzeń na pinach, więc można się nieprzyjemnie zdziwić!
* Złącza programatora różnią się zależności od wersji modułu PiPico, i są umieszczane w różnych miejscach PCB

Poniższy obrazek warto wydrukować i mieć zawsze pod ręką, pozwoli to uniknąć błędów wyboru kierunków transmisji (np. Tx/Rx) czy ID modułu (np. SPI0/SPI1).

Pin RUN ma szczególne znaczenie i jest tak naprawdę sygnałem nRESET (sam moduł nie ma fizycznego przycisku RESET).



Rys. Pinout modułu Pico 2 W