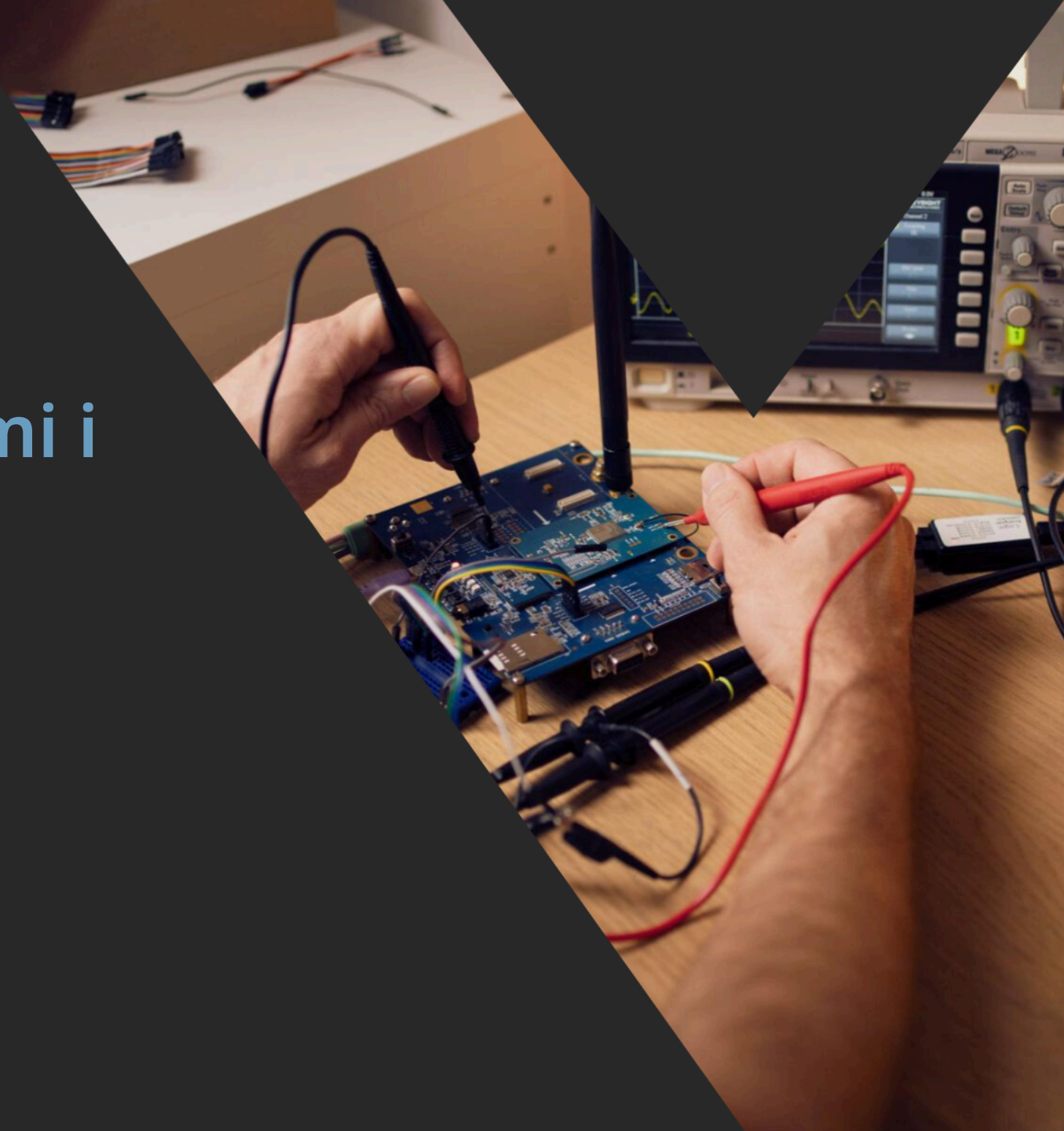


Zarządzanie narzędziami i projektami embedded

Dawid Marszałkiewicz

February 4, 2025

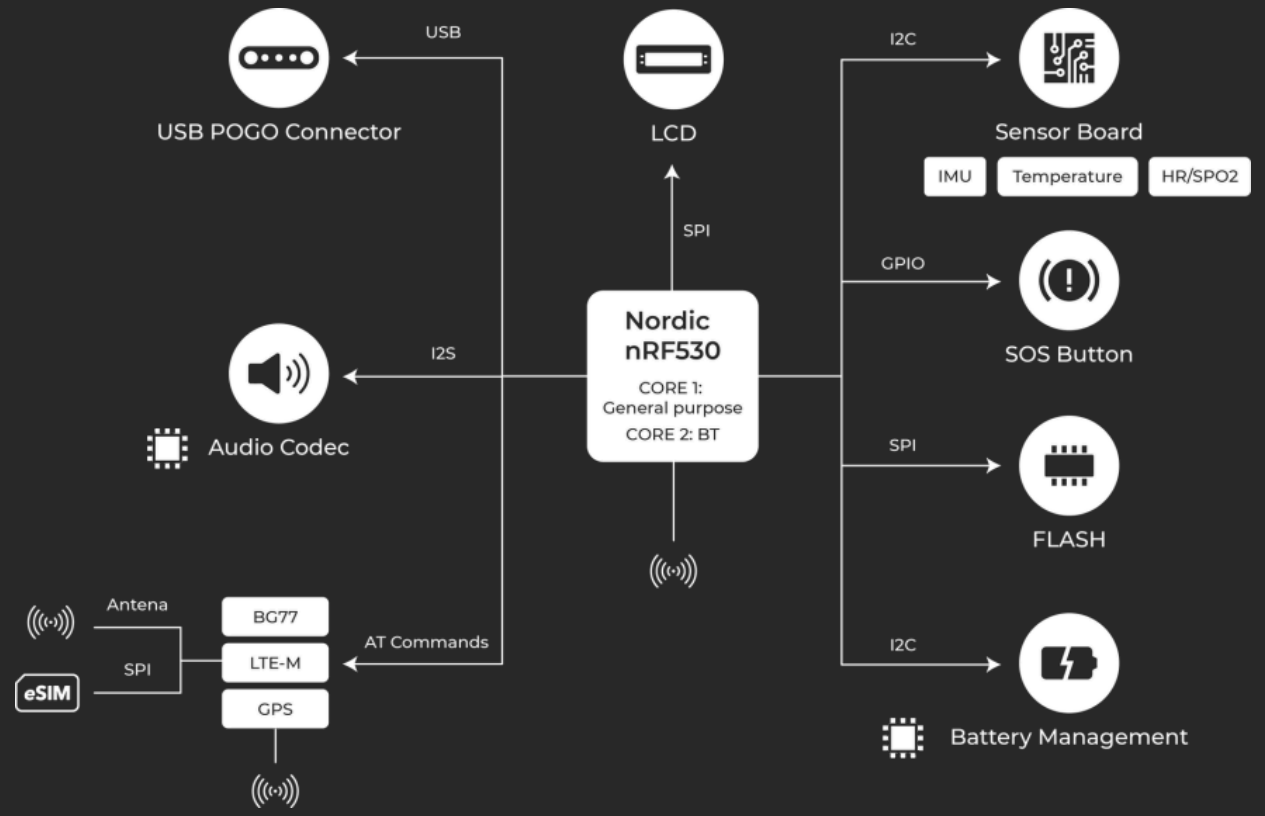
{ GoodByte
embedded software }

Kim jesteśmy, co robimy?

Audycja zawiera lokowanie produktu

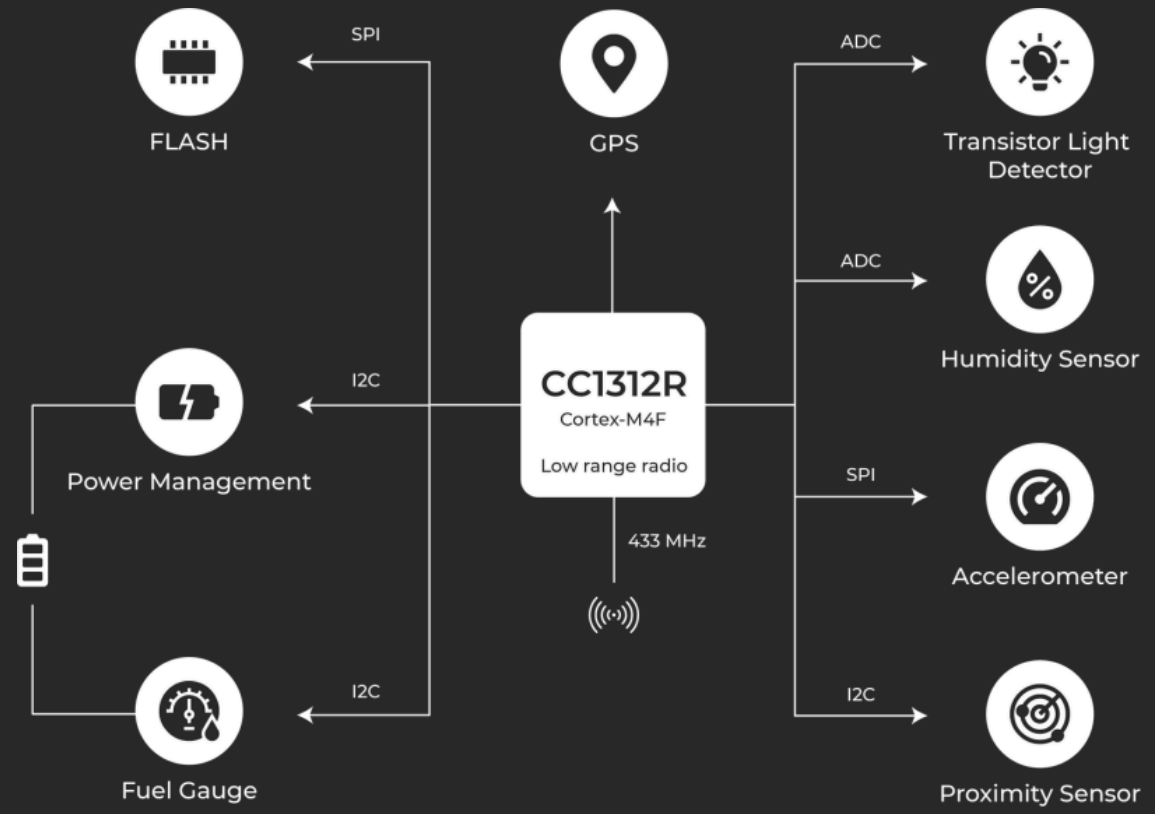
Kim jesteśmy, co robimy?

Kim jesteśmy, co robimy?

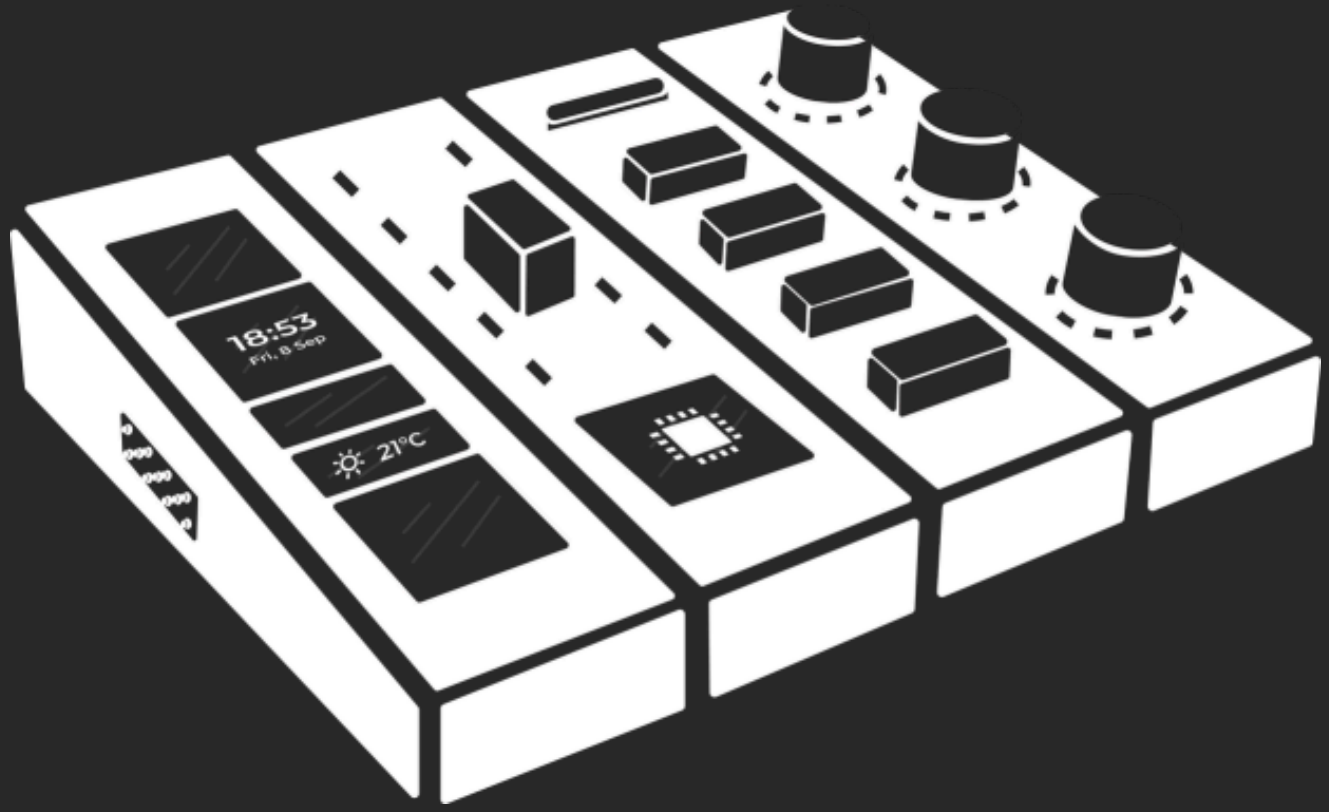


Kim jesteśmy, co robimy?

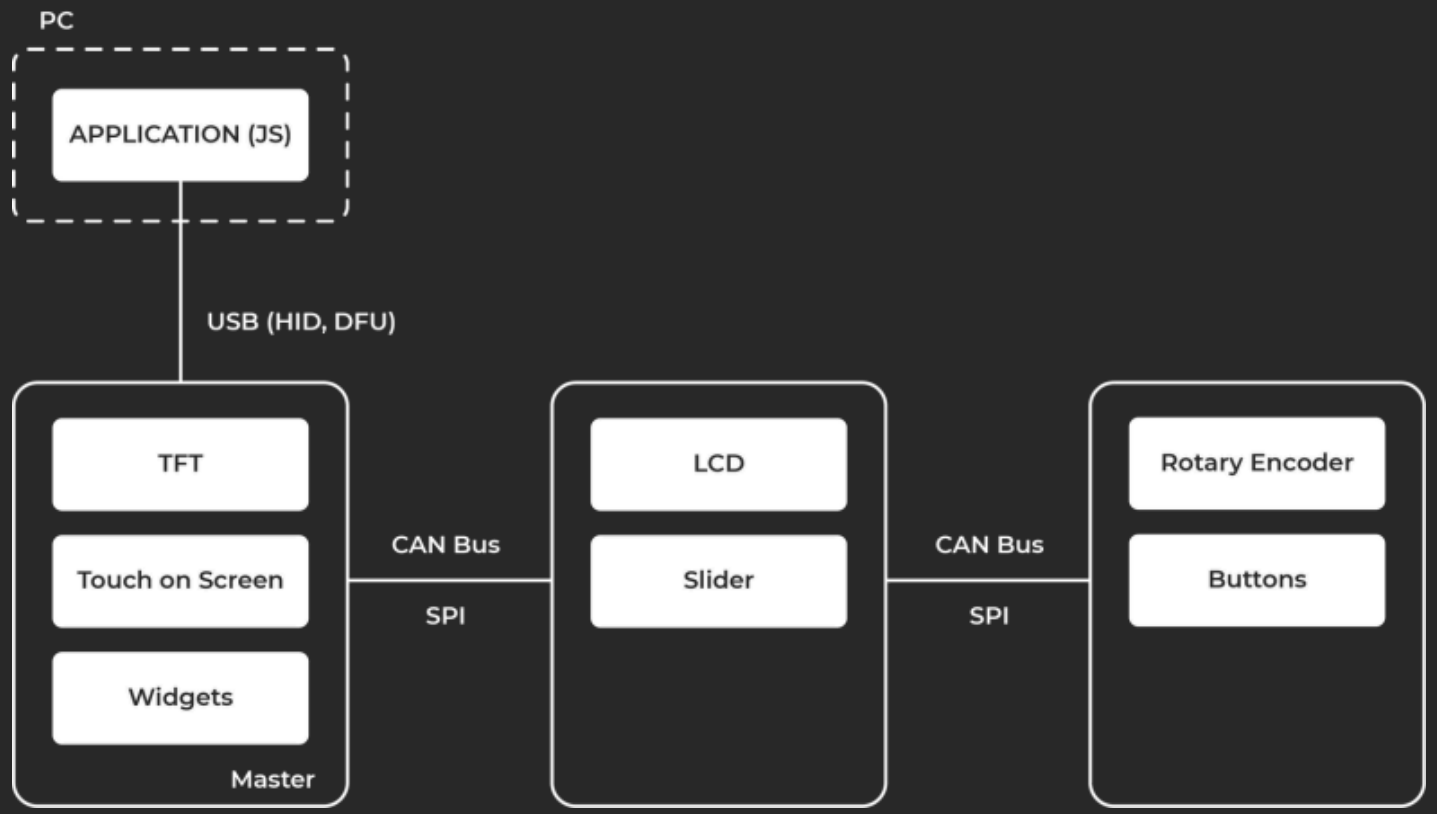
Kim jesteśmy, co robimy?



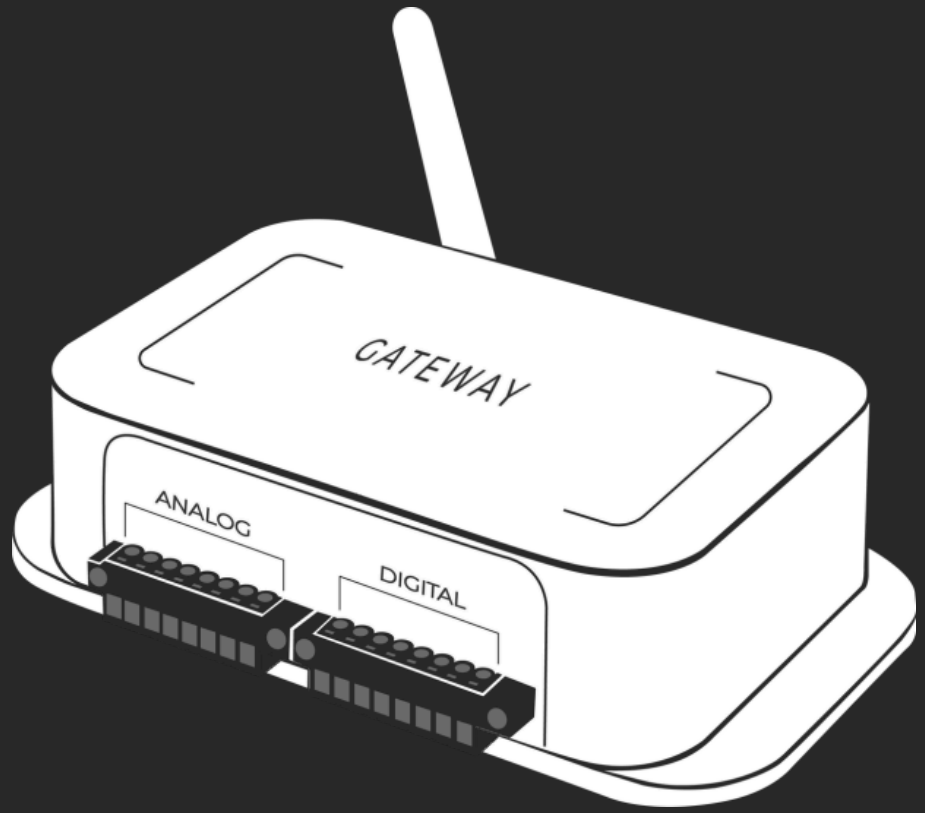
Kim jesteśmy, co robimy?



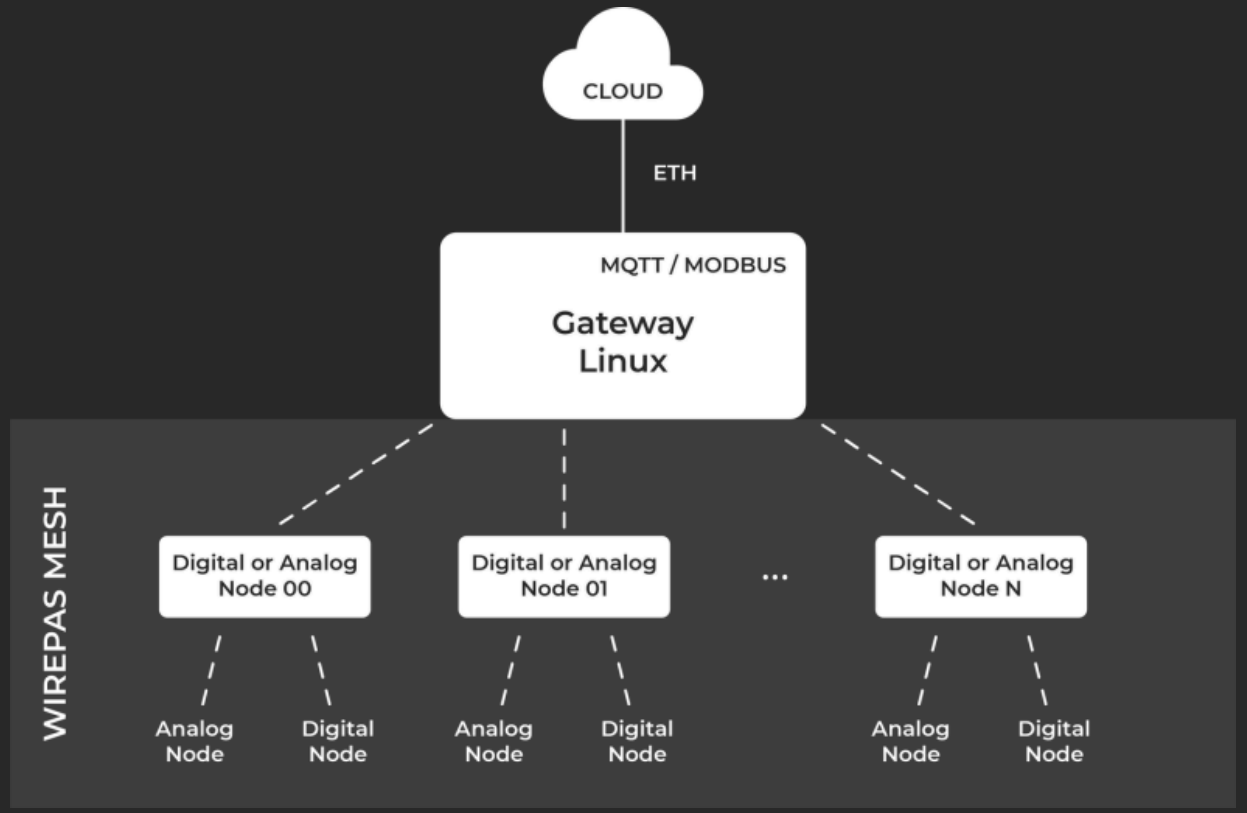
Kim jesteśmy, co robimy?



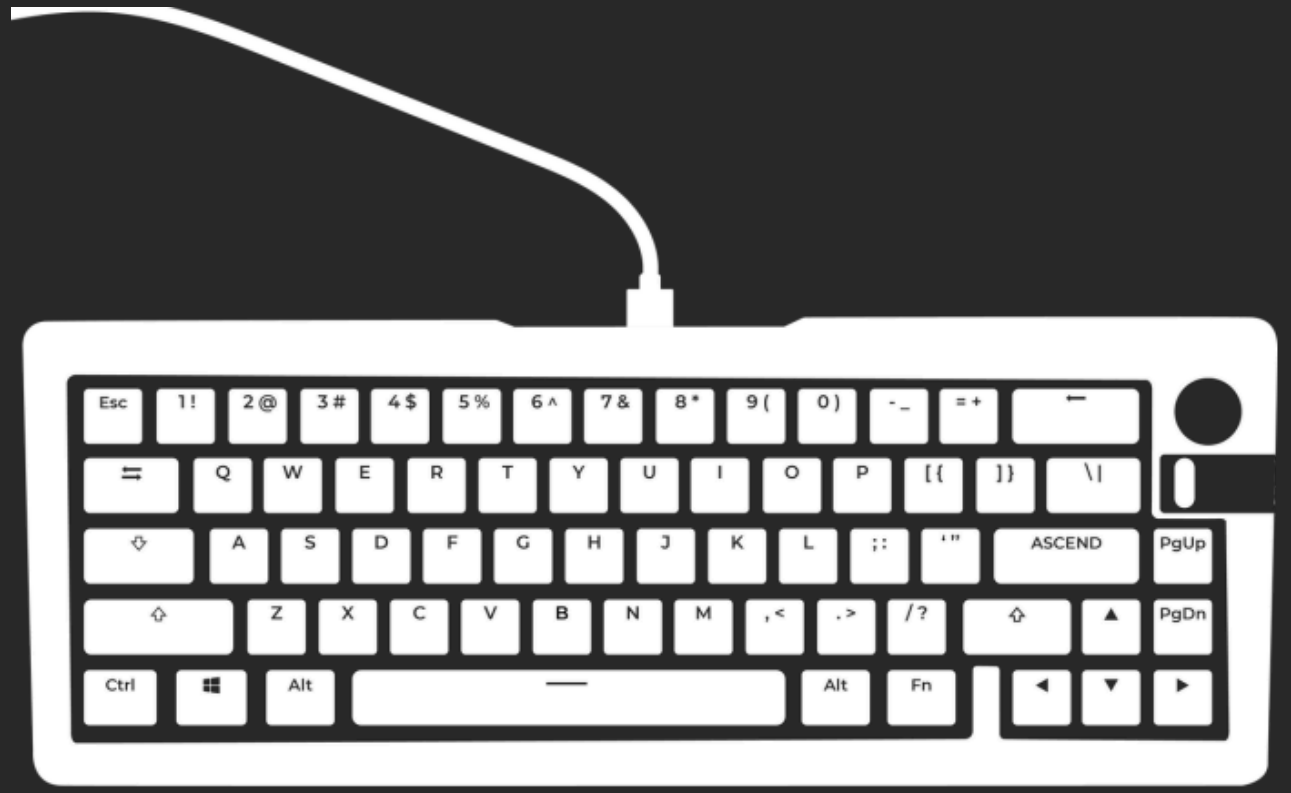
Kim jesteśmy, co robimy?



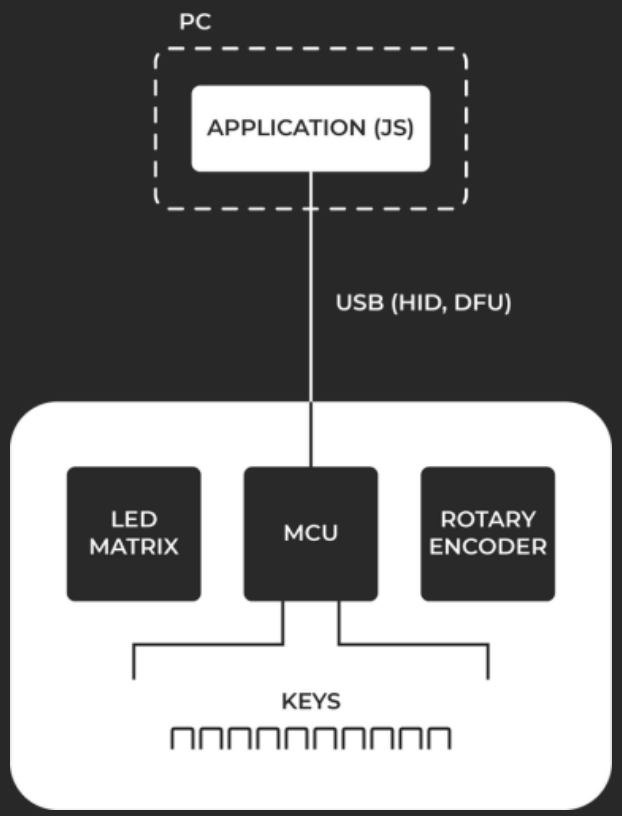
Kim jesteśmy, co robimy?



Kim jesteśmy, co robimy?

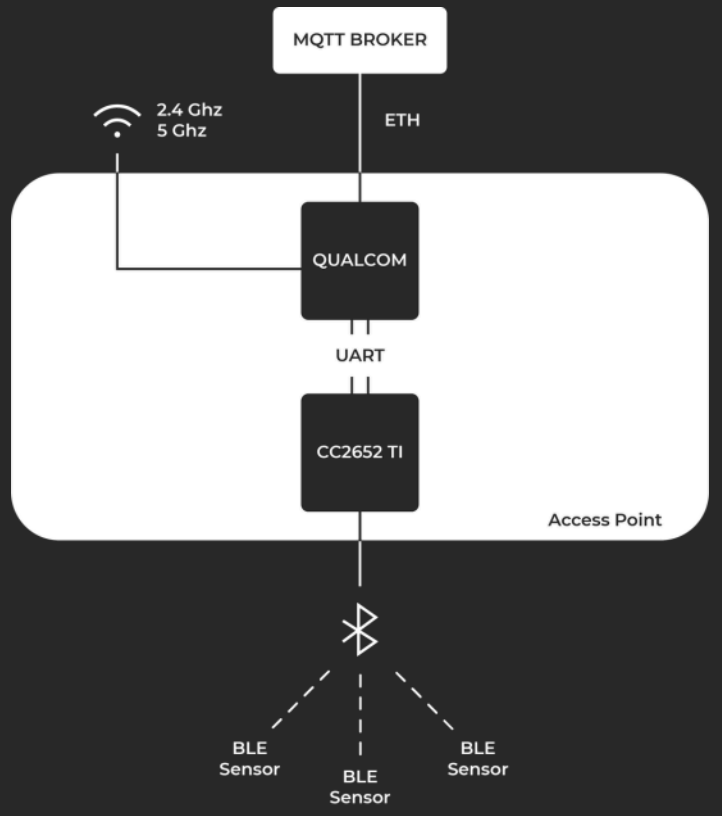


Kim jesteśmy, co robimy?



Kim jesteśmy, co robimy?

Kim jesteśmy, co robimy?

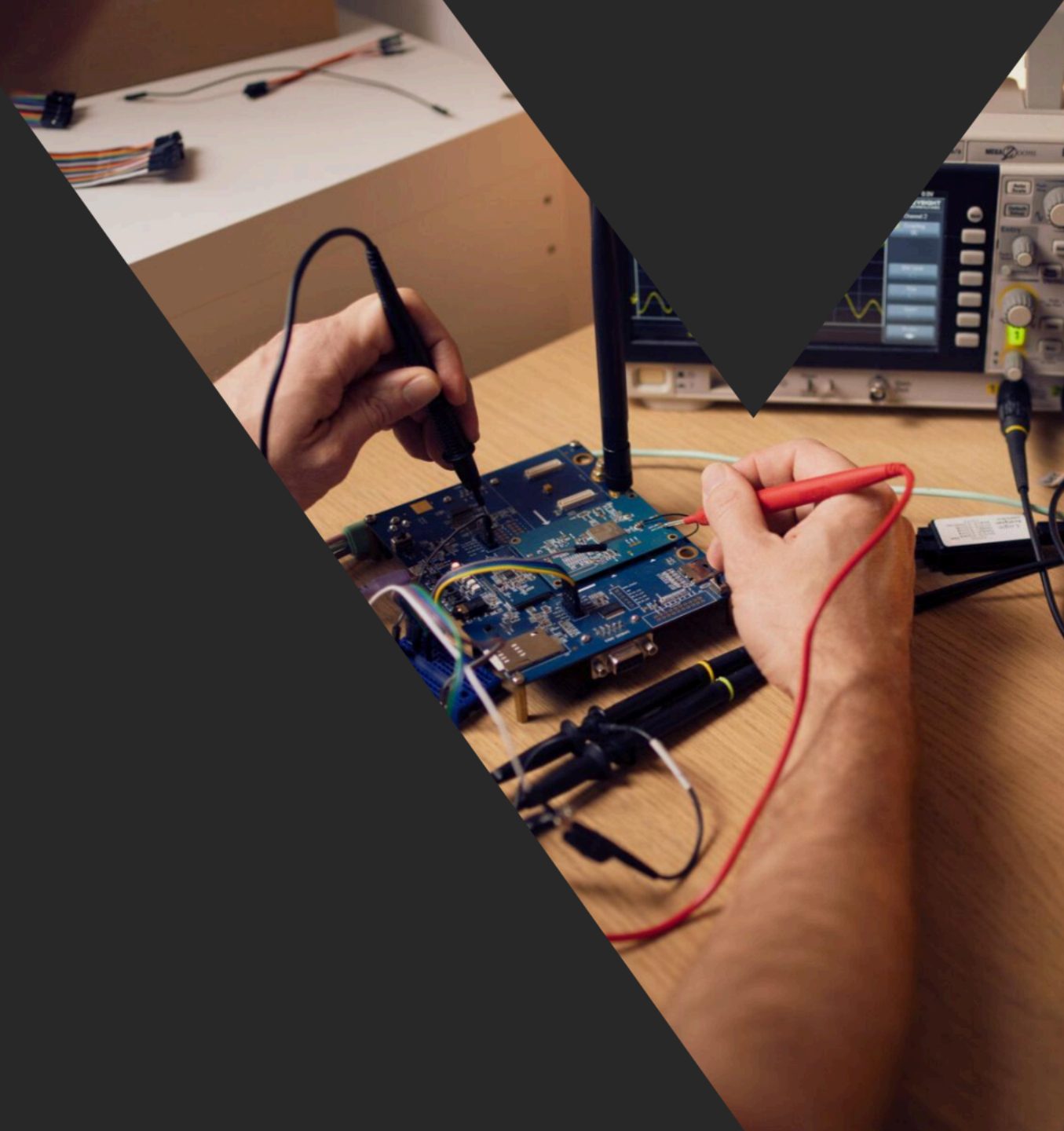


Do czego to prowadzi?

- Wyzwania w przygotowaniu środowiska pracy dla ESP, STM, Nordic, TI
- Konieczność każdorazowego dostosowania środowisk do projektów
- Problem z przenoszeniem się między platformami
- Co z narzędziami upraszczającymi procesy?

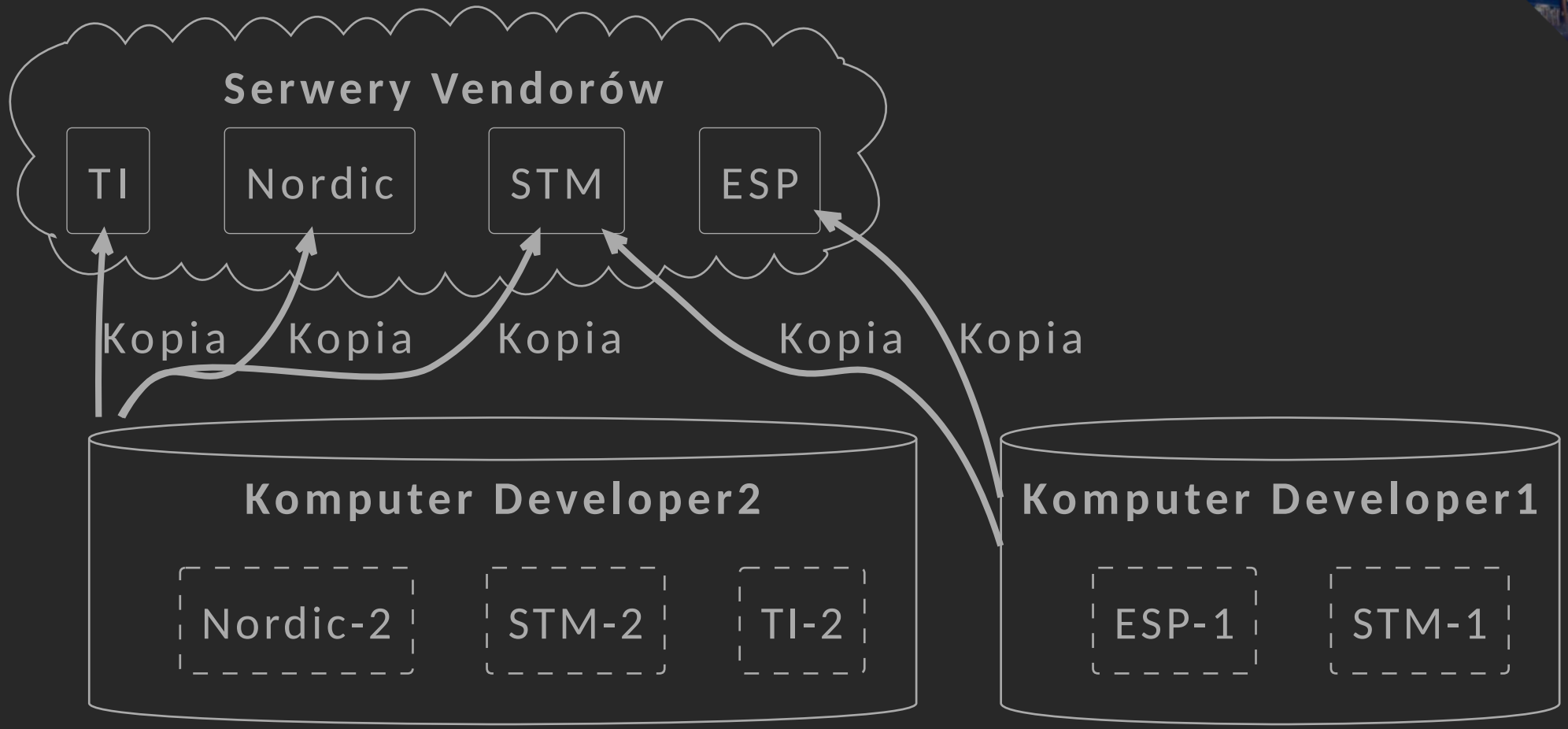
Czego byśmy chcieli?

- Możliwość wyboru narzędzi przez programistów
- Łatwe przełączanie między projektami
- Zapewnienie długotrwałego utrzymania
- Proste narzędzia zwiększające produktywność



Rozwiązanie I
Lokalna konfiguracja

Na czym to polega?



Jakie mamy dostępne IDE?

ESPRESSIF

Embedded Studio

STM32 CubeIDE

IAR SYSTEMS

KEIL™ Tools by ARM

A ile jest dodatkowych narzędzi?

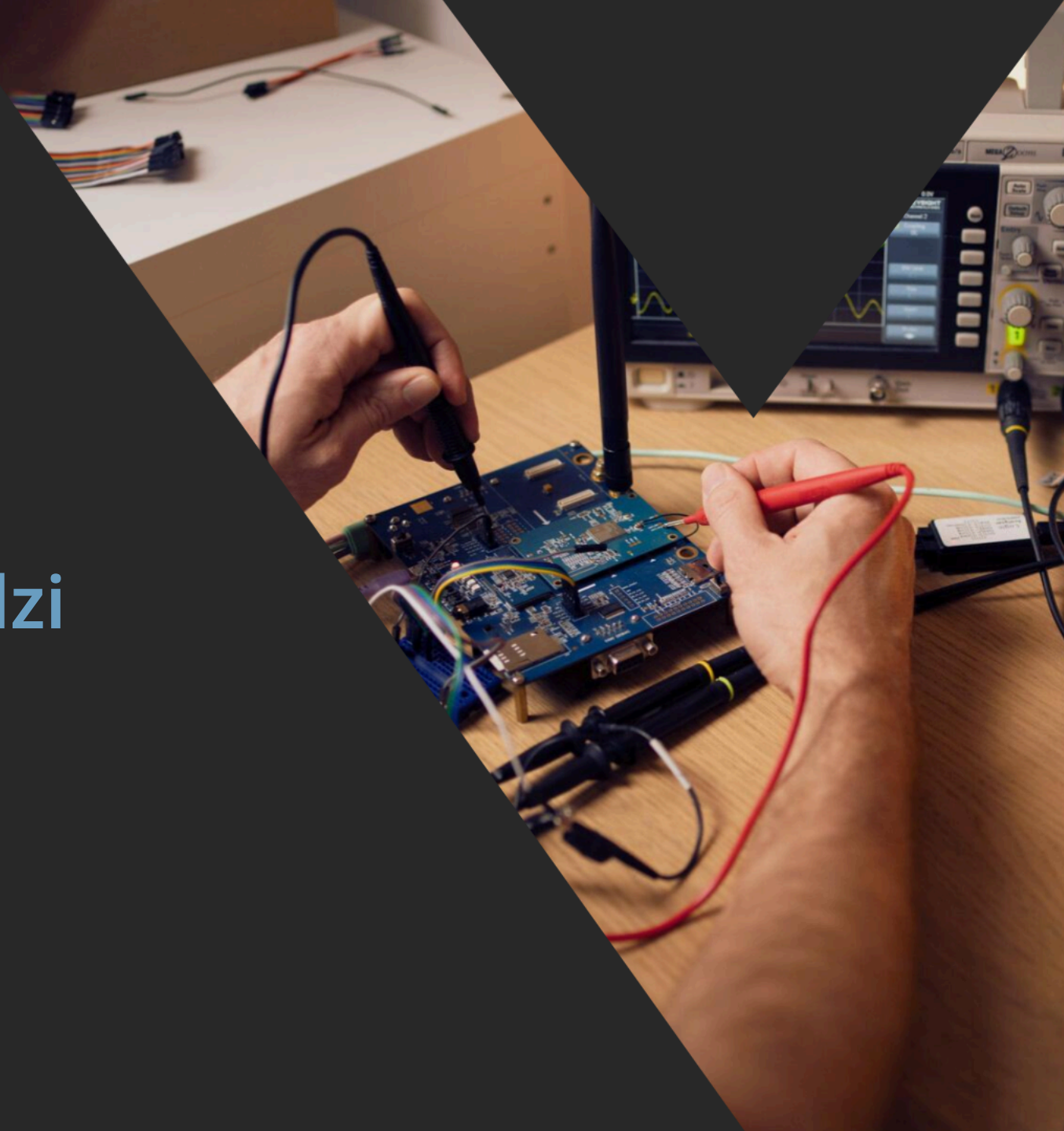
A ile jest dodatkowych narzędzi?

Plusy:

- Niezależność w wyborze narzędzi przez programistów
- Możliwość dostosowania środowiska do osobistych preferencji
- Brak centralnego punktu awarii

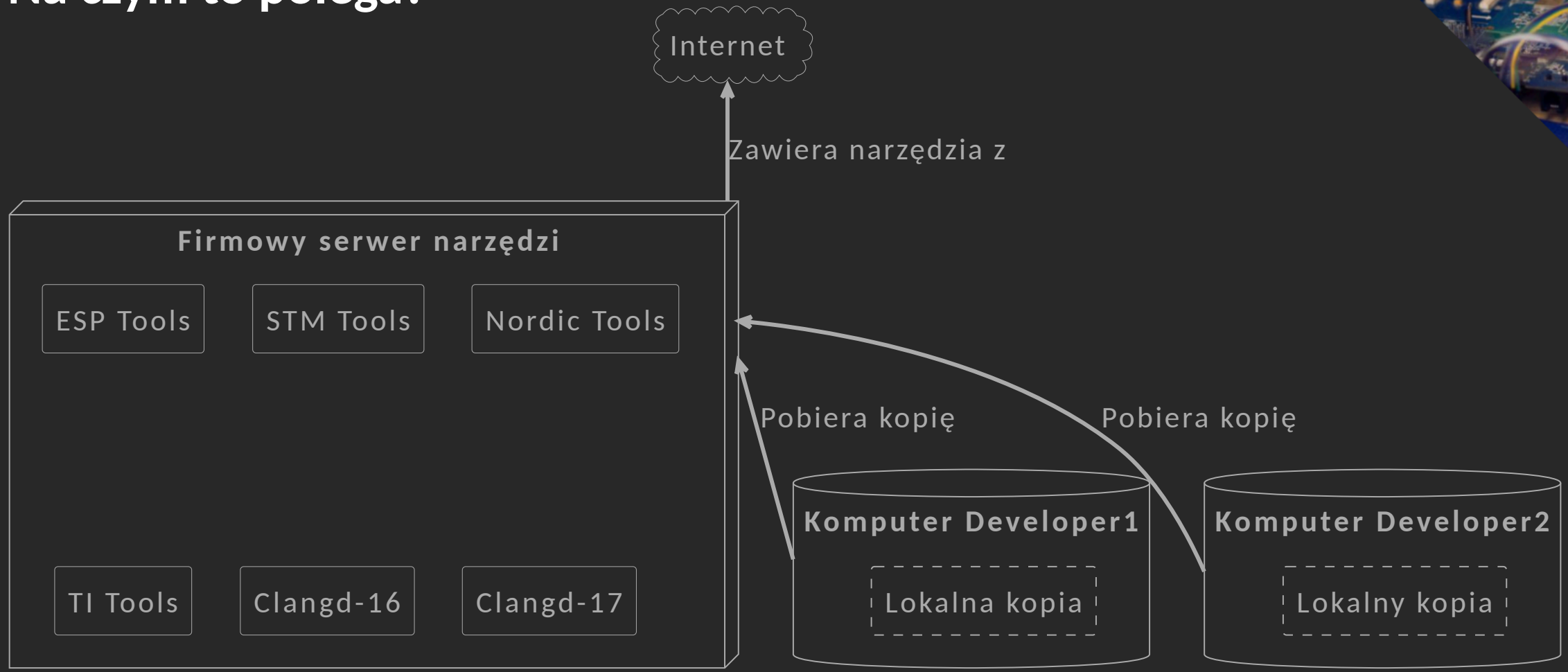
Minusy:

- Problemy z synchronizacją wersji narzędzi
- Trudność w powtarzalności środowisk między programistami
- Większy czas konfiguracji dla nowych członków zespołu
- Zależność od manualnych aktualizacji i konfiguracji narzędzi



Rozwiązanie II
Firmowy serwer narzędzi

Na czym to polega?



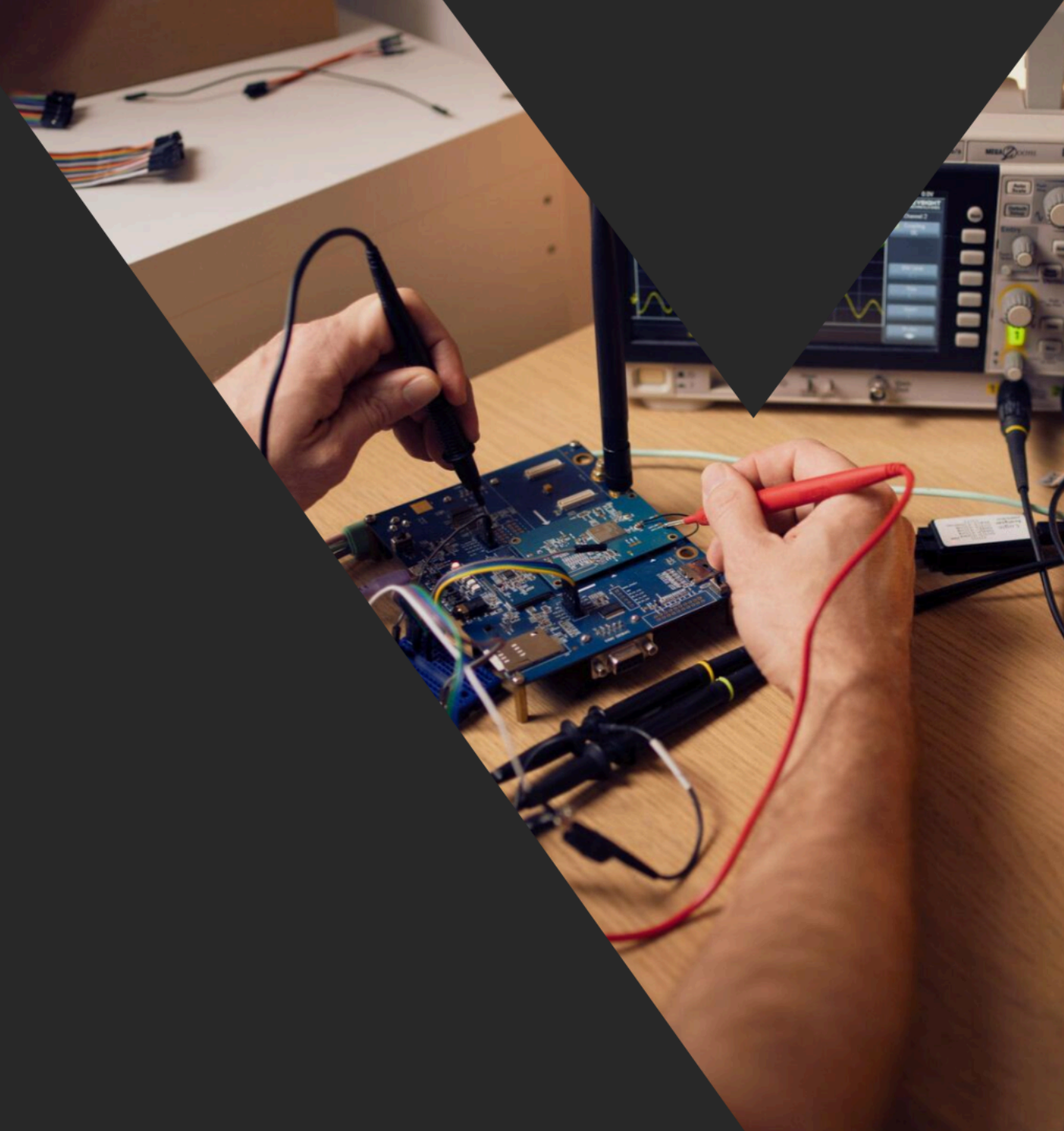
Plusy:

- Centralne zarządzanie narzędziami
- Powtarzalność środowisk
- Kontrola wersji narzędzi
- Szybsze wdrożenie nowych osób

Minusy:

- Złożoność utrzymania serwera
- Zależność od serwera (brak offline) lub lokalnie kopie narzędzi
- Ukryta zależność do narzędzi
- Jak współpracować z innymi firmami?

Rozwiązanie III
Docker



Czym jest Docker?

Definicja:

- Platforma do uruchamiania aplikacji w kontenerach.

Podstawowe pojęcia:

- **Obraz (Image):** Szablon aplikacji z zależnościami.
- **Kontener (Container):** Uruchomione środowisko na bazie obrazu.

Czym jest Docker?

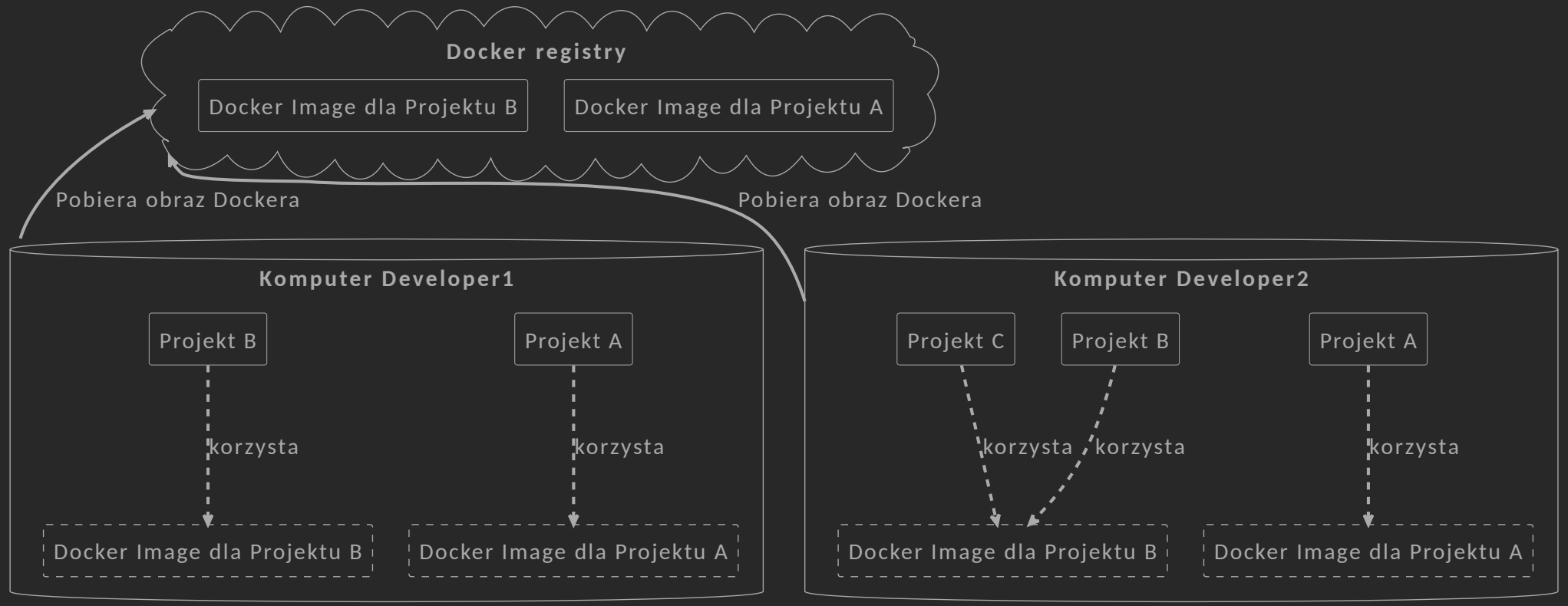
Podstawowe pojęcia (cd.):

- **Dockerfile:** Przepis na stworzenie obrazu.
- **Registry:** Repozytorium obrazów (np. Docker Hub).
- **Volume:** Przechowywanie danych dla kontenerów.

Zalety:

- Izolacja środowisk.
- Powtarzalność konfiguracji.
- Szybkie wdrożenie aplikacji.

Na czym to polega?



Przykład implementacji - Dockerfile

```
FROM ubuntu:22.04
ARG USER=developer
ARG UID=1000
ARG GID=1000
USER root

RUN apt-get install -y gcc-arm-none-eabi

ARG JLINK_DEB=JLink_Linux_V766_x86_64.deb
RUN download_jlink.sh $JLINK_DEB \
    && apt-get install -y ./$JLINK_DEB

#...
```

Przykład implementacji - Dockerfile

```
FROM ubuntu:22.04
ARG USER=developer
ARG UID=1000
ARG GID=1000
USER root

RUN apt-get install -y gcc-arm-none-eabi

ARG JLINK_DEB=JLink_Linux_V766_x86_64.deb
RUN download_jlink.sh $JLINK_DEB \
    && apt-get install -y ./$JLINK_DEB

#...
```

Przykład implementacji - Dockerfile

```
FROM ubuntu:22.04
ARG USER=developer
ARG UID=1000
ARG GID=1000
USER root

RUN apt-get install -y gcc-arm-none-eabi

ARG JLINK_DEB=JLink_Linux_V766_x86_64.deb
RUN download_jlink.sh $JLINK_DEB \
    && apt-get install -y ./$JLINK_DEB

#...
```

Przykład implementacji - Dockerfile

```
FROM ubuntu:22.04
ARG USER=developer
ARG UID=1000
ARG GID=1000
USER root

RUN apt-get install -y gcc-arm-none-eabi

ARG JLINK_DEB=JLink_Linux_V766_x86_64.deb
RUN download_jlink.sh $JLINK_DEB \
    && apt-get install -y ./$JLINK_DEB

#...
```

Przykład implementacji - Dockerfile

```
#...
RUN apt-get update && apt-get install -y clangd-17

RUN apt-get install -y protobuf-compiler=3.12
RUN python3 -m pip install protobuf==4.21.12

USER ${UID}:${GID}
```

Przykład implementacji - Dockerfile

```
#...
RUN apt-get update && apt-get install -y clangd-17

RUN apt-get install -y protobuf-compiler=3.12
RUN python3 -m pip install protobuf==4.21.12

USER ${UID}:${GID}
```

Przykład implementacji - Dockerfile

```
#...
RUN apt-get update && apt-get install -y clangd-17

RUN apt-get install -y protobuf-compiler=3.12
RUN python3 -m pip install protobuf==4.21.12

USER ${UID}:${GID}
```

Przykład implementacji - run.sh

```
source .env

docker run \
    --rm --interactive \
    --volume /dev/bus/usb/:/dev/bus/usb \
    --volume $workdir:$user_home/workdir \
    # ...
    --name $image_name \
    $image_name:$image_tag
```

Przykład implementacji - run.sh

```
source .env

docker run \
    --rm --interactive \
    --volume /dev/bus/usb/:/dev/bus/usb \
    --volume $workdir:$user_home/workdir \
    # ...
    --name $image_name \
    $image_name:$image_tag
```

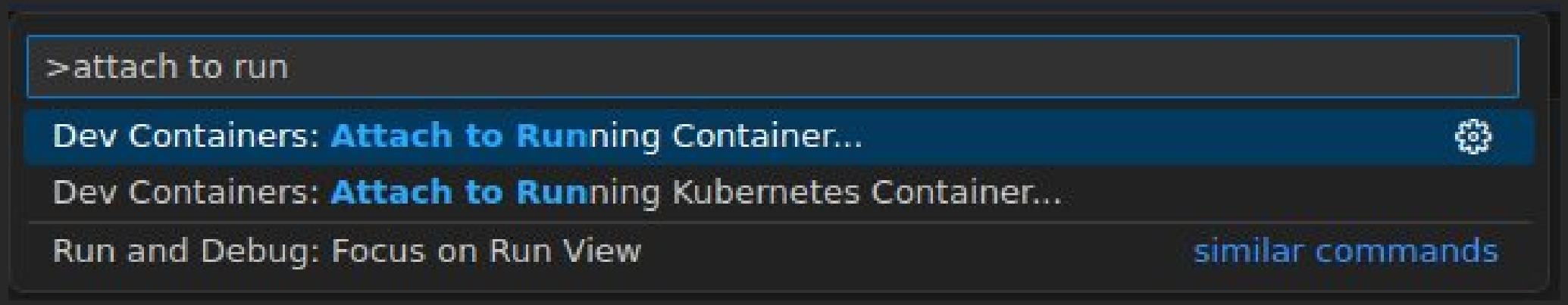
Przykład implementacji - run.sh

```
source .env

docker run \
    --rm --interactive \
    --volume /dev/bus/usb/:/dev/bus/usb \
    --volume $workdir:$user_home/workdir \
    # ...
    --name $image_name \
    $image_name:$image_tag
```

Przykład implementacji - run.sh

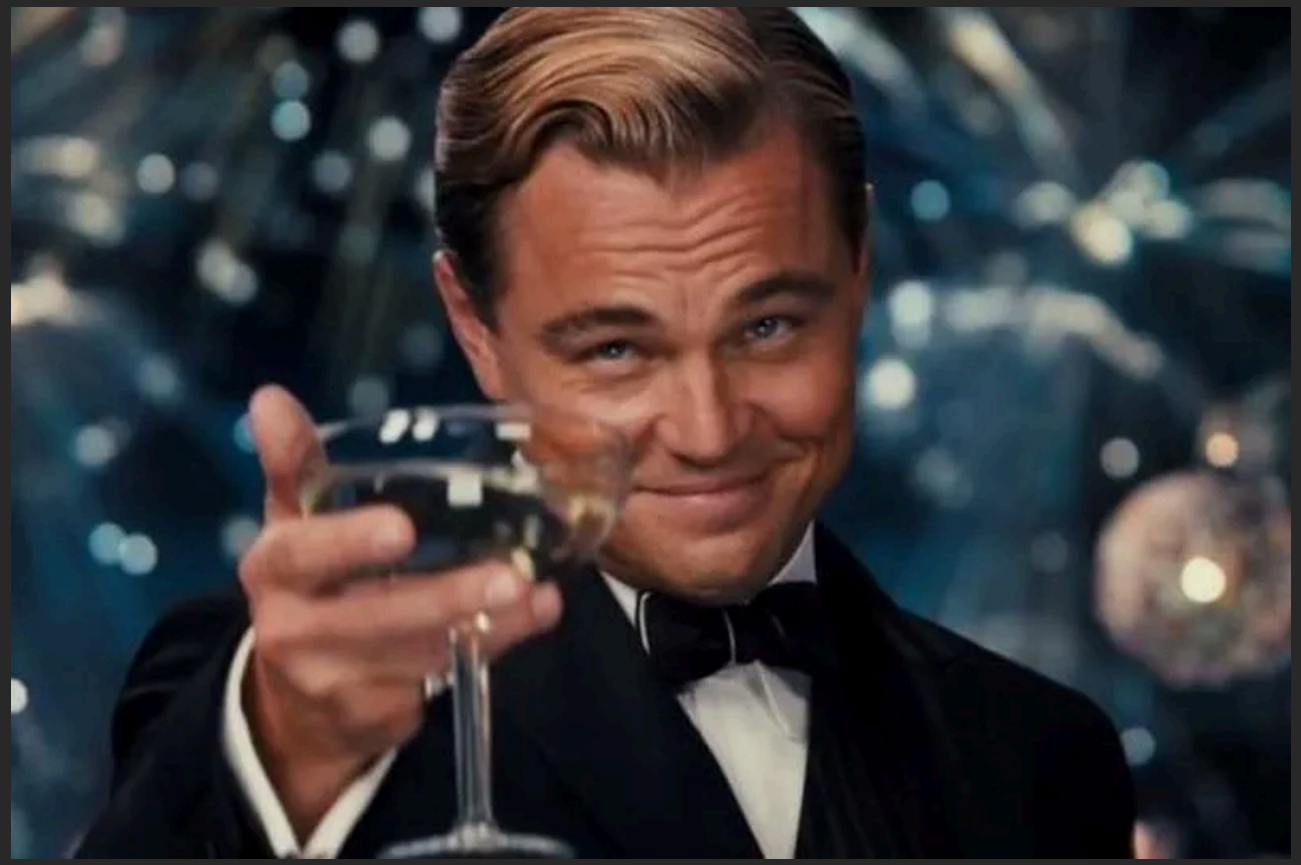
```
source .env

docker run \
    --rm --interactive \
    --volume /dev/bus/usb/:/dev/bus/usb \
    --volume $workdir:$user_home/workdir \
    # ...
    --name $image_name \
    $image_name:$image_tag
```

Przykład implementacji - VS Code



Przykład implementacji - ostatni krok

Przykład implementacji - ostatni krok



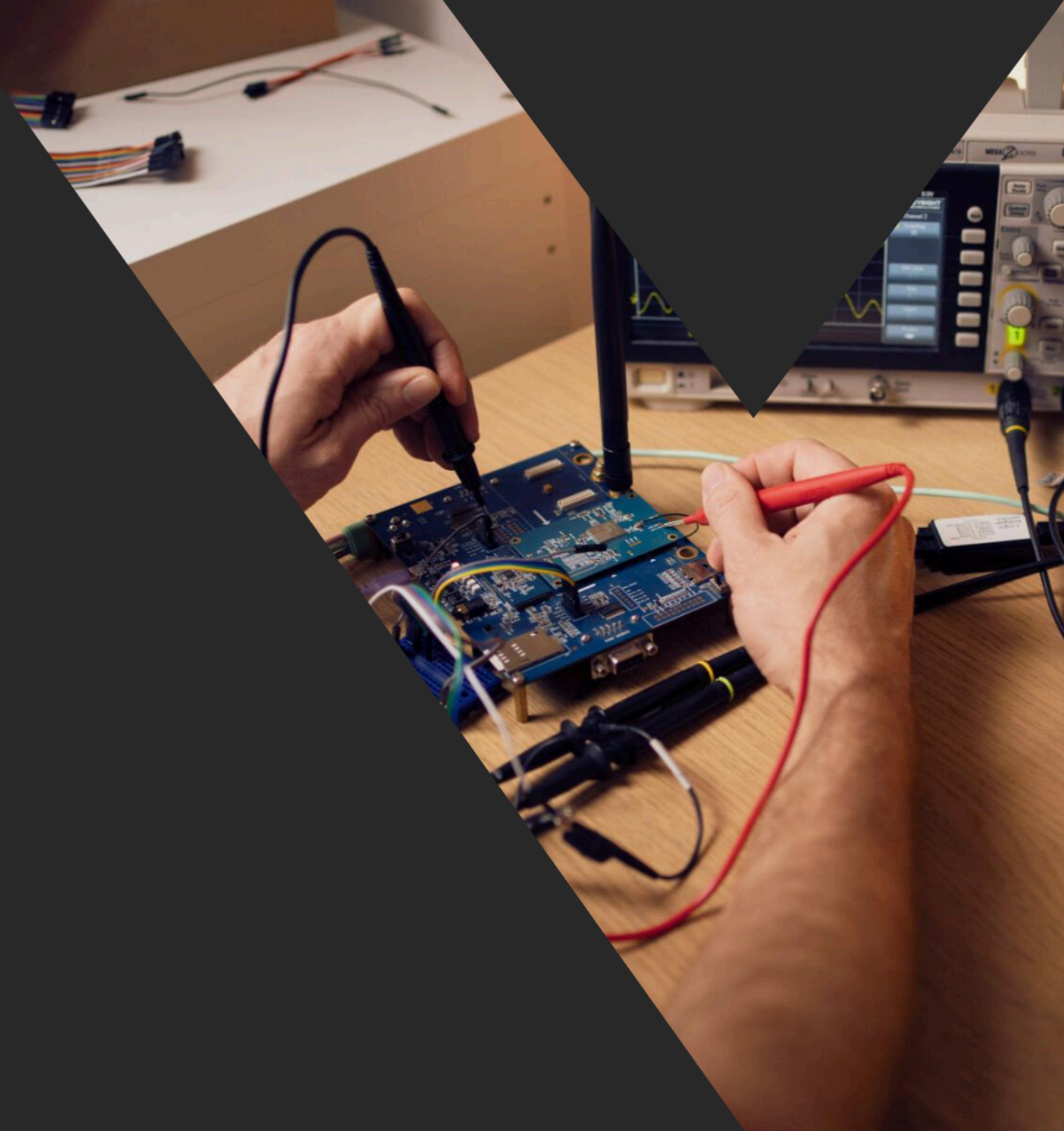
Plusy:

- Izolacja środowisk dla każdego projektu.
- Szybkie wdrażanie nowych członków zespołu.
- Szybkie przełączanie się między projektami.
- Powtarzalność środowisk.

Minusy:

- Wymaga konfiguracji dla każdego projektu.
- Potrzebna znajomość Dockera.
- Większe zużycie zasobów.

Rozwiązanie IV
Devcontainer

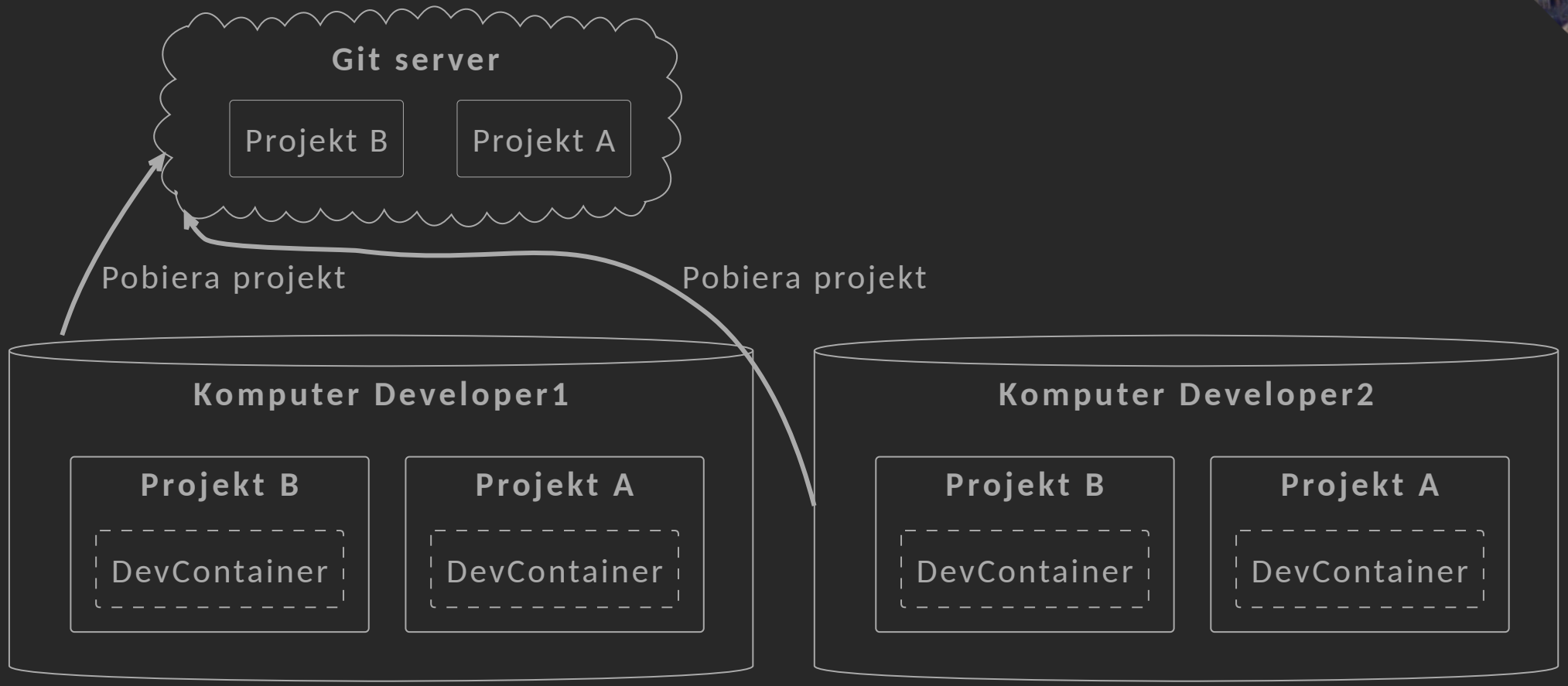


Jak działają devcontainers

Podstawy:

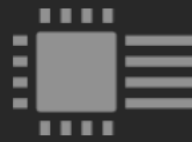
- Pliki konfiguracyjne dla kontenera Dockera.
- Automatyczne uruchamianie środowiska projektu.
- Integracja z Visual Studio Code, JetBrains IDE.
- Posiada własne CLI.

Na czym to polega?



Devcontainers - jak wykorzystać istniejące obrazy?

```
# docker-compose.yml
services:
  dev_container:
    image: dev_super_secret
    build:
      dockerfile: Dockerfile
      context: .
    #...
    privileged: true
    volumes:
      - /dev:/dev
      - ../:/repos/${REPO_NAME}
```



Devcontainers - jak wykorzystać istniejące obrazy?

```
# docker-compose.yml
services:
  dev_container:
    image: dev_super_secret
    build:
      dockerfile: Dockerfile
      context: .
    #...
    privileged: true
    volumes:
      - /dev:/dev
      - ../:/repos/${REPO_NAME}
```

Devcontainers - jak wykorzystać istniejące obrazy?

```
# docker-compose.yml
services:
  dev_container:
    image: dev_super_secret
    build:
      dockerfile: Dockerfile
      context: .
    #...
    privileged: true
    volumes:
      - /dev:/dev
      - ../:/repos/${REPO_NAME}
```

Devcontainers - jak wykorzystać istniejące obrazy?

```
// devcontainer.json
{
  "dockerComposeFile": ["./docker-compose.yml"],
  "service": "dev_container",
  "workspaceFolder": "/repos/${localWorkspaceFolderBasename}",
  "shutdownAction": "stopCompose",
  "customizations": {
      "vscode": {
          "extensions":
            ["llvm-vs-code-extensions.vscode-clangd"]
      }
  }
}
```

Devcontainers - jak wykorzystać istniejące obrazy?

```
// devcontainer.json
{
    "dockerComposeFile": ["./docker-compose.yml"],
    "service": "dev_container",
    "workspaceFolder": "/repos/${localWorkspaceFolderBasename}",
    "shutdownAction": "stopCompose",
    "customizations": {
        "vscode": {
            "extensions":
              ["llvm-vs-code-extensions.vscode-clangd"]
        }
    }
}
```

Użytkownik CI lubi to 👍

```yaml
jobs:
  build:
    steps:
        - name: Run make ci-build in dev container
          uses: devcontainers/ci@v0.3
          with:
            cacheFrom: ghcr.io/example/example-devcontainer
            push: never
            runCmd: make ci-build
```

Ale...

Ale...

Przykład bardziej skomplikowanego Dockerfile'a

```
# nRF Connect SDK + CodeChecker
RUN source ~/.zephyrrc && \
    mkdir /root/ncs/v${NRF_SDK_VERSION} && \
    cd /root/ncs/v${NRF_SDK_VERSION} && \
    west init -m https://github.com/nrfconnect/sdk-nrf --mr v${NRF_SDK_VERSION} && \
    west update && \
    west zephyr-export && \
    pip install CodeChecker
```

Ale...

Przykład bardziej skomplikowanego Dockerfile'a

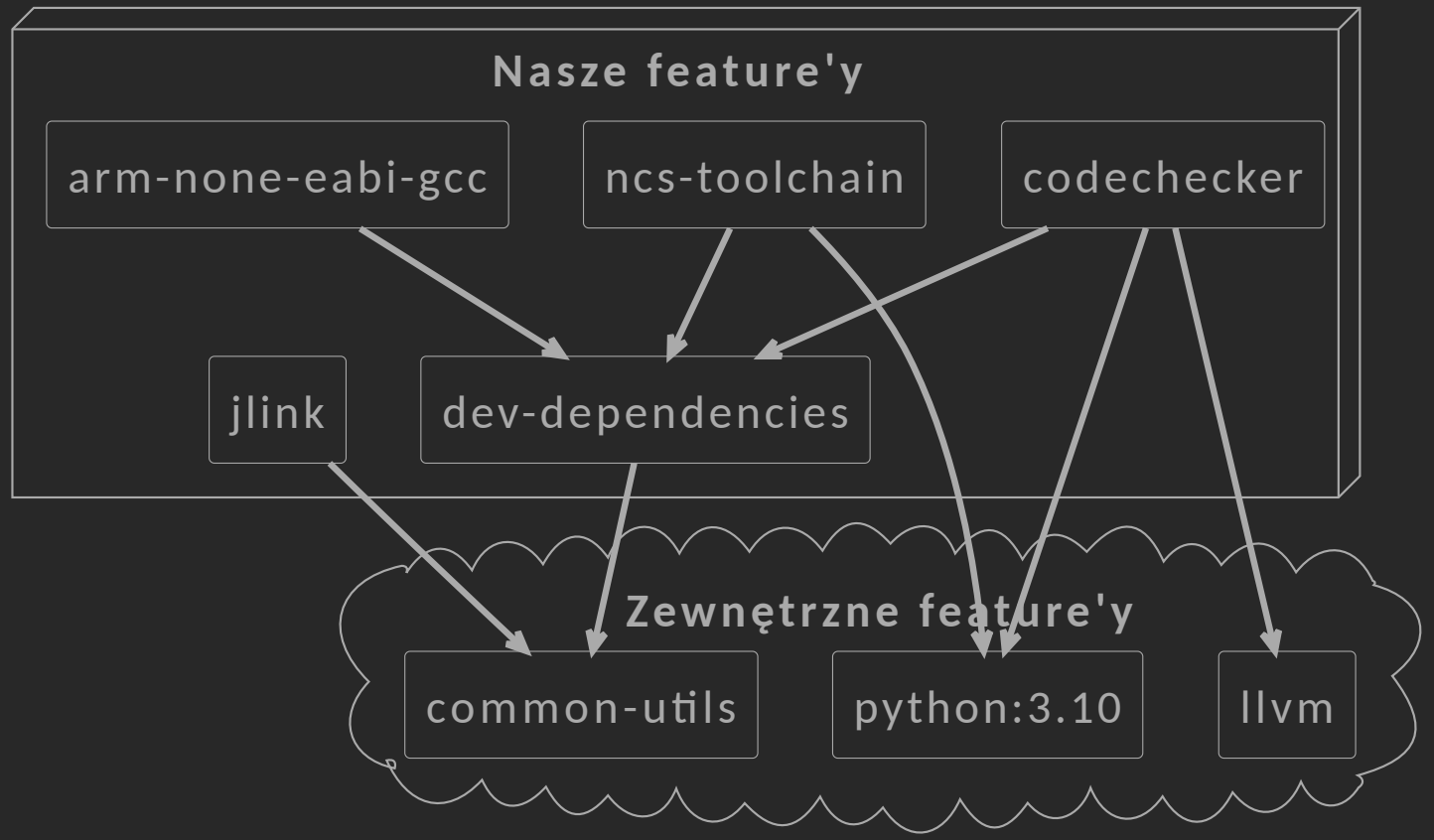
```
# nRF Connect SDK + CodeChecker
RUN source ~/.zephyrrc && \
    mkdir /root/ncs/v${NRF_SDK_VERSION} && \
    cd /root/ncs/v${NRF_SDK_VERSION} && \
    west init -m https://github.com/nrfconnect/sdk-nrf --mr v${NRF_SDK_VERSION} && \
    west update && \
    west zephyr-export && \
    pip install CodeChecker
```

No i gdzie zależności?

Dev Container Features

- **Features** dodają funkcjonalności do kontenerów DevContainer.
- Automatyzują instalację narzędzi i konfiguracji w środowisku.
- Minimalizują ilość ręcznych konfiguracji.

Nasze feature'y



Przykładowa konfiguracja z feature'ami

```
{
    "image": "ubuntu:22.04",
    "features": {
        "ghcr.io/goodbyte-software/goodbyte-features/ncs-toolchain:latest": {
            "version": "0.16.8",
            "architecture": "arm"
        },
        "ghcr.io/goodbyte-software/goodbyte-features/codechecker:latest": {},
        "ghcr.io/goodbyte-software/goodbyte-features/jlink:latest": {
            "version": "7.94e"
        }
    }
}
```

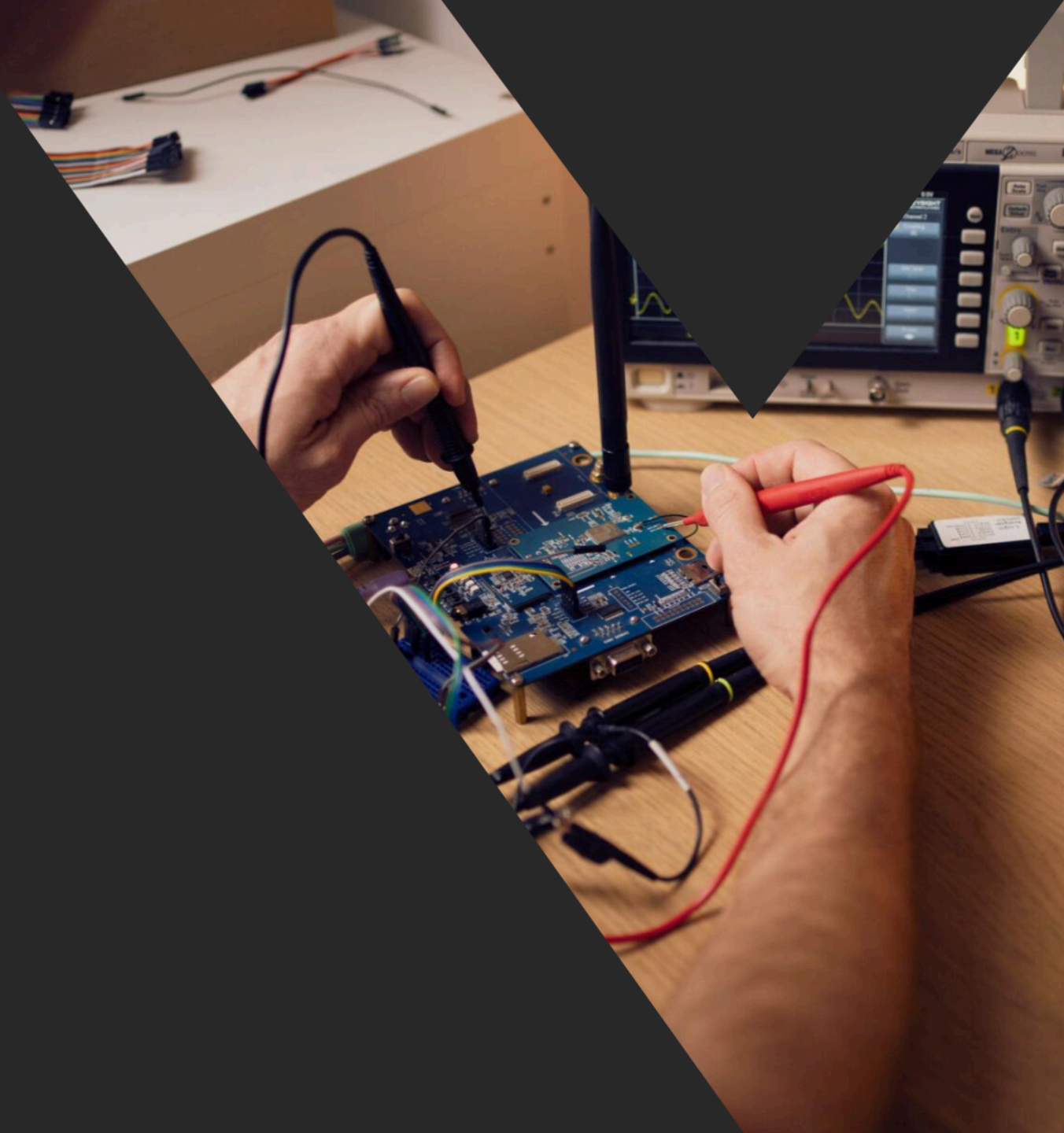
Dev Container Templates

- **Templates** to predefiniowane konfiguracje środowisk deweloperskich.
- Skracają czas konfiguracji, standaryzują środowiska i ułatwiają wdrożenie nowych projektów.

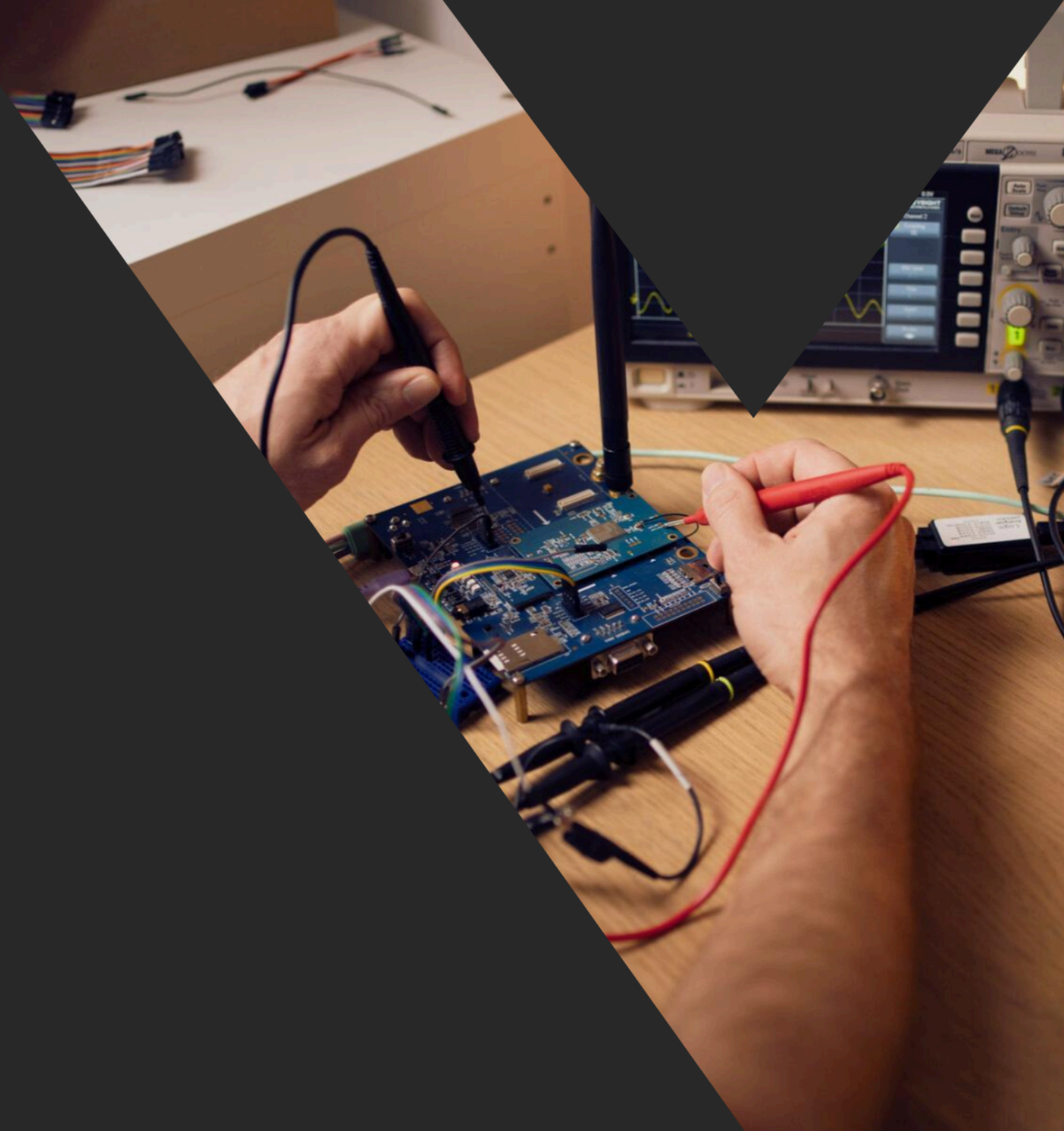
Przykładowy template

```
├── .devcontainer
│   ├── devcontainer.json
│   └── onCreateCommand.sh
├── .vscode
│   └── settings.json
└── app
    ├── west.yml
    └── prj.conf
```

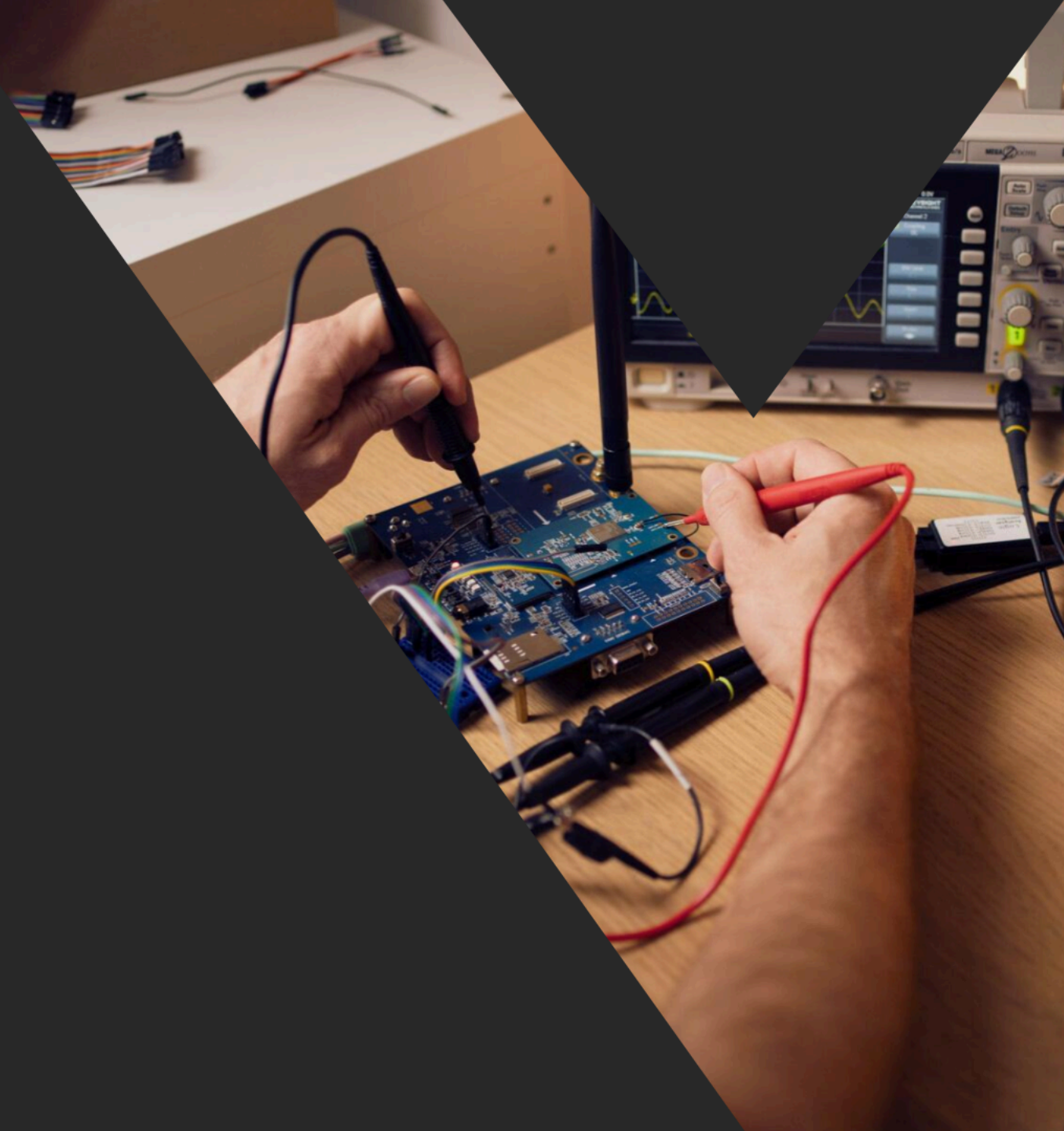
Demo



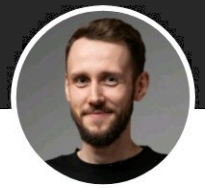
To już jest koniec (?)



Czas na pytania



Dziękuję za uwagę i zapraszam do kontaktu!



Dawid Marszalkiewicz

Lack of software quality is
the biggest cost 🔷 Zephyr enthu...