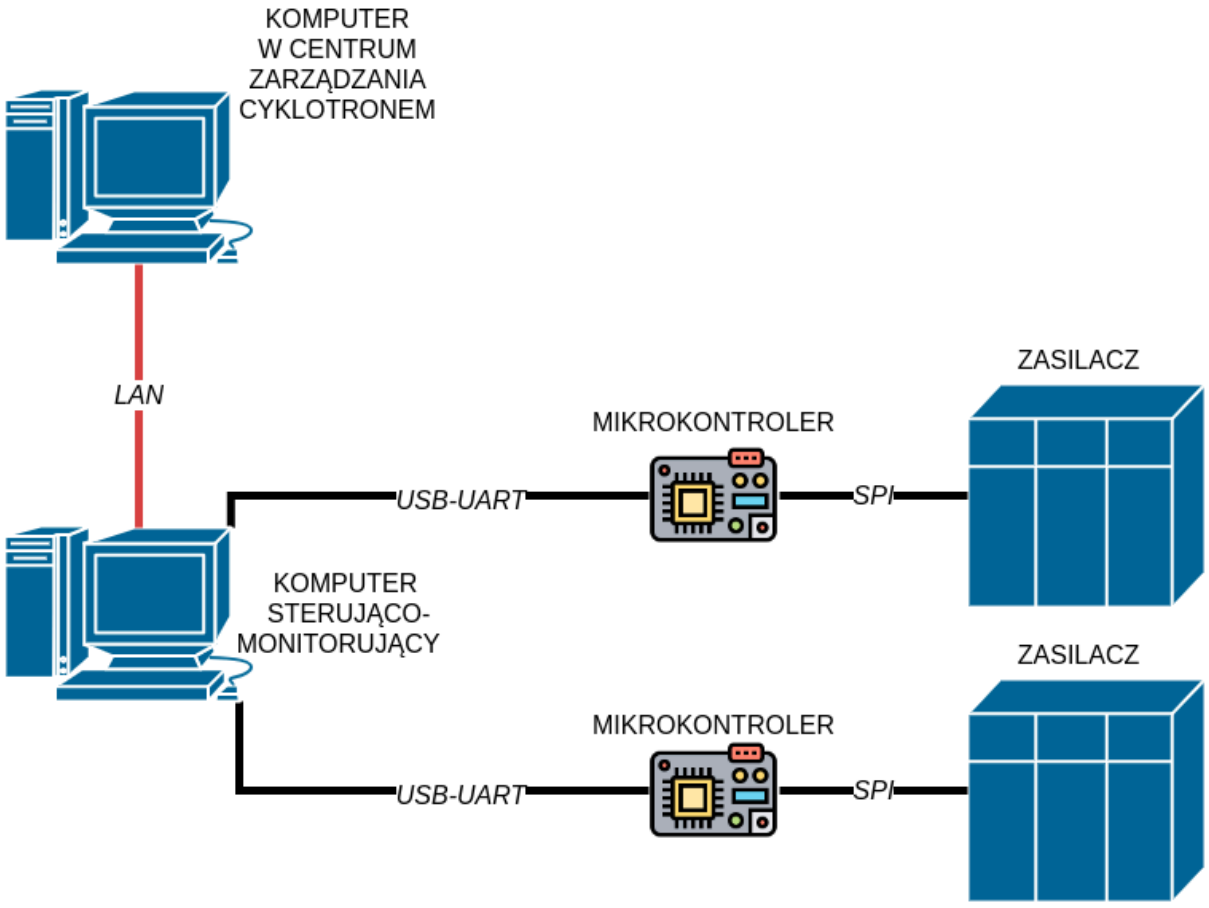
Aplikacja na komputer sterujący

Wstęp

Aplikacja służy do sterowania zasilaczami 200 A, pochodzącymi z Theodore Svedberg Laboratory Uniwersytetu w Uppsali. Umożliwia monitorowanie bieżącego stanu zasilacza (w tym informacji o błędach) oraz zmianę jego parametrów. Częścią aplikacji jest oprogramowanie na mikrokontroler Raspberry Pi Pico, komunikujące się bezpośrednio z zasilaczem za pomocą SPI, a także wykorzystujące protokół Modbus do porozumiewania się z komputerem sterującym.



Interfejs graficzny jest przygotowany do połączenia aplikacji z komputerem w centrum zarządzania. Zawiera specjalny przycisk, który zmienia tryb z remote na local i odwrotnie. W obecnej wersji, przełączenie trybu pracy aplikacji na remote tym przyciskiem, jedynie blokuje możliwość wydawania poleceń zasilaczom przez

aplikację, bowiem sterowanie docelowo powinno odbywać się ze sterowni. Aplikacja natomiast dalej wyświetla informacje odczytane z zasilacza. **Nie jest wspierane sterowanie zasilaczami z komputera sterowni.**

W przyszłości aplikację będzie można wykorzystać także do obsługi zasilaczy o maksymalnym natężeniu 100 A. Interfejs graficzny uwzględnia parametr polaryzacji, używany w tym modelu zasilacza. Powstało także dedykowane mu oprogramowanie na mikrokontroler. Przed wdrożeniem tej funkcjonalności należy jednak gruntownie przetestować część kodu odpowiedzialną za komunikację z zasilaczem 100 A, gdyż twórcy nie mieli takiej możliwości.

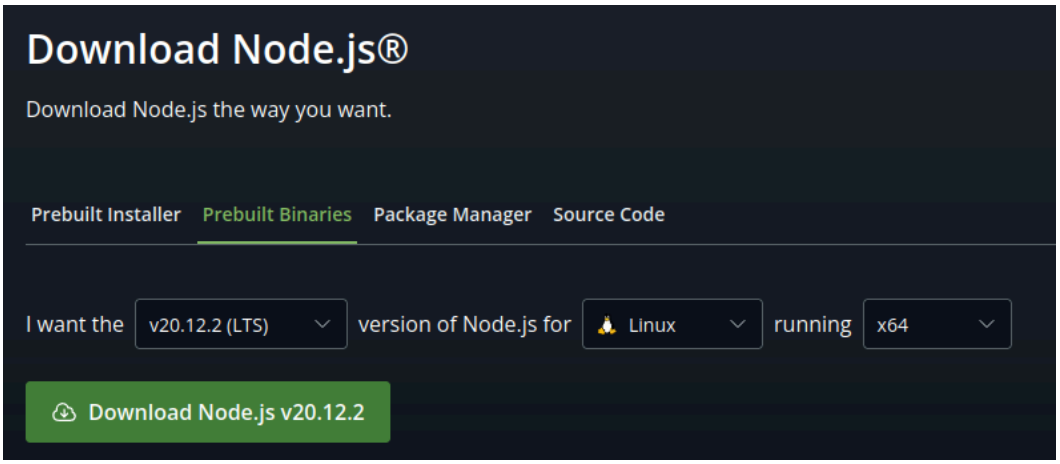
Repozytorium z kodem aplikacji:
https://github.com/ZPP-Cyclotron/ZPP-GatewayComputer
Repozytorium z kodem oprogramowania na mikrokontroler:
https://github.com/ZPP-Cyclotron/ZPP-RaspberryPiPico

Kompilacja kodu źródłowego aplikacji, tworzenie pliku instalacyjnego .deb

Aplikacja na komputer sterujący stworzona została przy użyciu frameworku Electron, bazującego na Nodejs. Poniżej opisano szczegółowo proces kompilacji kodu źródłowego, oraz tworzenie pliku instalacyjnego aplikacji, jak również instalowanie go.

Aby zbudować aplikację z kodu źródłowego, należy wykonać następujące kroki:

1. Zainstalować Node.js:
    a. wejść na stronę: https://nodejs.org/en/download/prebuilt-binaries
    b. wybrać wersję nodejs (najlepiej najnowszą wersję LTS)
    c. wybrać system operacyjny i architekturę (Linux x64)
    d. nacisnąć przycisk download, jak na rysunku poniżej:



    Zostanie pobrany plik o nazwie node-v20.12.2-linux-x64.tar.xz
    e. następnie należy wykonać następujące komendy (w folderze z plikiem node-v20.12.2-linux-x64.tar.xz):

```
tar xf node-v20.12.2-linux-x64.tar.xz
sudo mv ./node-v*-linux-x64/ /opt/
sudo ln -s /opt/node-v*-linux-x64/bin/node /usr/bin/node
sudo ln -s /opt/node-v*-linux-x64/bin/npm  /usr/bin/npm
```

    f. Teraz polecenia “node -v”, “npm -v” powinny wypisać wersje node i npm.
2. W katalogu **`ZPP-GATEWAYCOMPUTER/gateway-computer-app`** wykonać polecenia:
    a. **`npm install`** (zainstaluje wymagane biblioteki zewnętrzne)

b. **`npm run start`** (kompiluje i włącza aplikację)
c. jeśli chcemy stworzyć plik instalacyjny .deb, to należy wykonać **`npm run make`**, które stworzy plik instalacyjny w katalogu: **`ZPP-GATEWAYCOMPUTER/gateway-computer-app/out/make/deb/x64/gateway-computer-app_1.0.0_amd64.deb`**

Aby zainstalować plik .deb, należy wykonać skrypt **`ZPP-GATEWAYCOMPUTER/setup.sh`** Parametr destination oznacza folder w którym pojawi się plik wykonywalny. W tym samym folderze należy umieścić plik konfiguracyjny **`config.json`** aplikacji.

Komunikacja aplikacji z mikrokontrolerami

Poniżej umieszczono format ramek Modbus wysyłanych i odbieranych od mikrokontrolerów przez aplikację. Numer bitu w tabeli odpowiada adresowi bitu w protokole modbus (tzn. że zapis następuje na adresy 0,...,13).

Zapis do mikrokontrolera: funkcja Write Coils FC15

Zapis do bitów pod adresami 0,...,13 lub 0,..,17 (w zależności od kodu operacji).

numer bitu	0…1	2…13, lub 2…17 dla zasilacza 200A
znaczenie	kod operacji	argument operacji

Kody operacji są zdefiniowane następująco:

0x00	ON / OFF
0x01	Ustawienie polaryzacji[1]
0x11	Ustawienie natężenia prądu

[1] W momencie przekazywania aplikacji klientowi (kwiecień 2024) zmiana polaryzacji jest nieużywana w przypadku zasilacza 200A, oraz nieprzetestowana na zasilaczu 100A

Natomiast argumenty operacji to jeden bit w przypadku polecenia On/Off i ustawienia polaryzacji, albo 12 bitów dla zasilacza 100A lub 16 bitów dla zasilacza 200A reprezentujące liczbę amperów z dokładnością do 0.1A, przemnożoną

odpowiednio przez $\frac{2^{12}-1}{100}$ lub $\frac{2^{16}-1}{200}$.

W tej konwencji, wysłanie jako wartości prądu liczby binarnej 111111111111 do zasilacza 100A, oznaczać będzie wartość natężenia równą 100A (gdyż mnożymy 100A razy

$\frac{2^{12}-1}{100}$ i otrzymujemy wartość wysłaną do zasilacza). Liczba binarna przesyłana jest do mikrokontrolera w konwencji: bit o najmniejszym adresie jest najbardziej znaczący.

Odczyt stanu zasilacza z mikrokontrolera: funkcja Read Input Registers FC04

Stan zasilacza zapisany jest w 3 rejestrach mikrokontrolera, o adresach 0,1,2 odpowiednio:

ODCZYT - rejestr 0

numer bitu	0…11	12	13	14	15
znaczenie	Prąd odczytany	ON / OFF odczytany	Polarity	Reset[2]	Sterowanie zdalne / ręczne

ODCZYT - rejestr 1

numer bitu	0…11	12…13	14	15
znaczenie	Napięcie	Błędy	Polarity zadane	ON / OFF zadany

[2] W momencie przekazywania aplikacji klientowi (kwiecień 2024) reset zasilacza jest nieużywany.

ODCZYT - rejestr 2

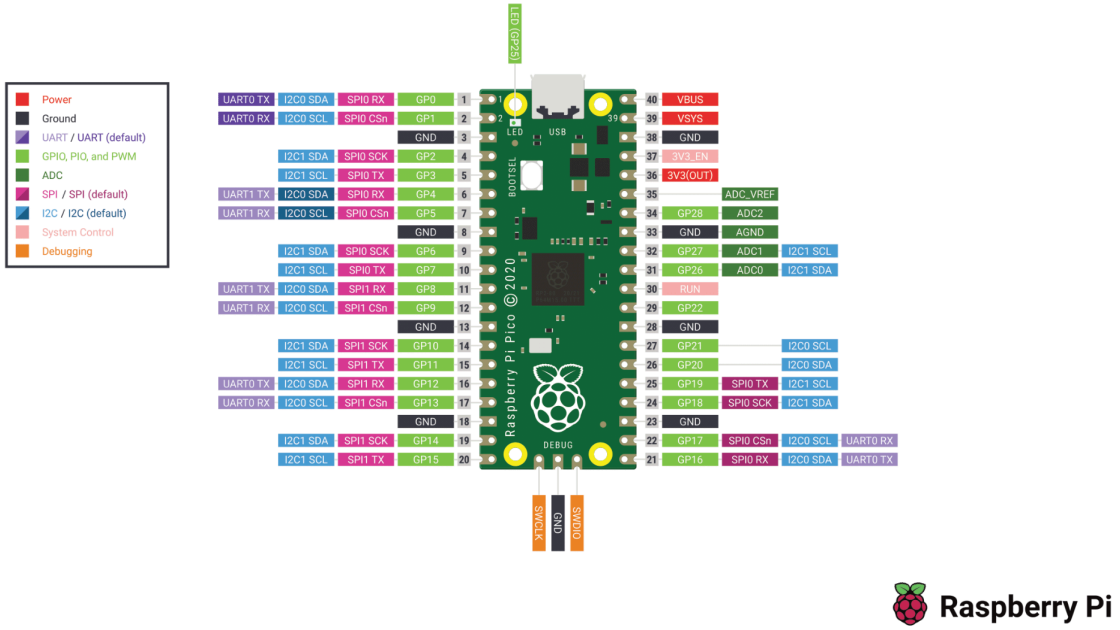
numer bitu	0…15
znaczenie	Prąd zadany

Przy czym w rejestrach zastosowano konwencję: bit o najmniejszym adresie jest najmniej znaczący, na przykład, wartość rejestru 0 równa 40956 oznaczać będzie przy zasilaczu 200A co następuje:
40956 = 1001111111111100 w systemie binarnym.

111111111100	1	0	0	1
Prąd odczytany. 111111111100 = 4092. Maksymalna wartość to 4095, po przeskalowaniu prąd to: 4092/4095 * (200A) = 199.853479853, co zaokrąglamy do 199.9A	ON/OFF odczytany	Polarity	Reset	Sterowanie zdalne / ręczne

Połączenie komputera sterującego z mikrokontrolerami

W obecnym stanie komputer sterujący komunikuje się z Raspberry Pi Pico po kablu USB 2.0, który łączy się z przejściówką USB-UART. Do przejściówki podpięte są kable, łączące się z pinami 1, 2, 3 Raspberry Pi Pico.



Źródło: https://datasheets.raspberrypi.com

Obsługa aplikacji przez użytkownika

Plik konfiguracyjny

W pliku konfiguracyjnym o nazwie config.json są następujące modyfikowalne parametry:

- refreshInterval (wspólny dla wszystkich zasilaczy) - określa, co ile milisekund aplikacja odczytuje dane z zasilacza,
- modbusTimeout (wspólny dla wszystkich zasilaczy) - określa, ile milisekund aplikacja będzie czekać na odpowiedź od mikrokontrolera,
- no - numer zasilacza,
- port - port USB, przez który aplikacja będzie przesyłać i odbierać komunikaty Modbus,
- polarity - przyjmuje wartość “const” lub “mutable” (dla zasilacza 200 A zawsze “const”),
- maxCurrent - maksymalne natężenie w zasilaczu,
- x, y - umiejscowienie parametrów zasilacza w interfejsie graficznym.

Poniżej przedstawiono przykładową zawartość pliku konfiguracyjnego:

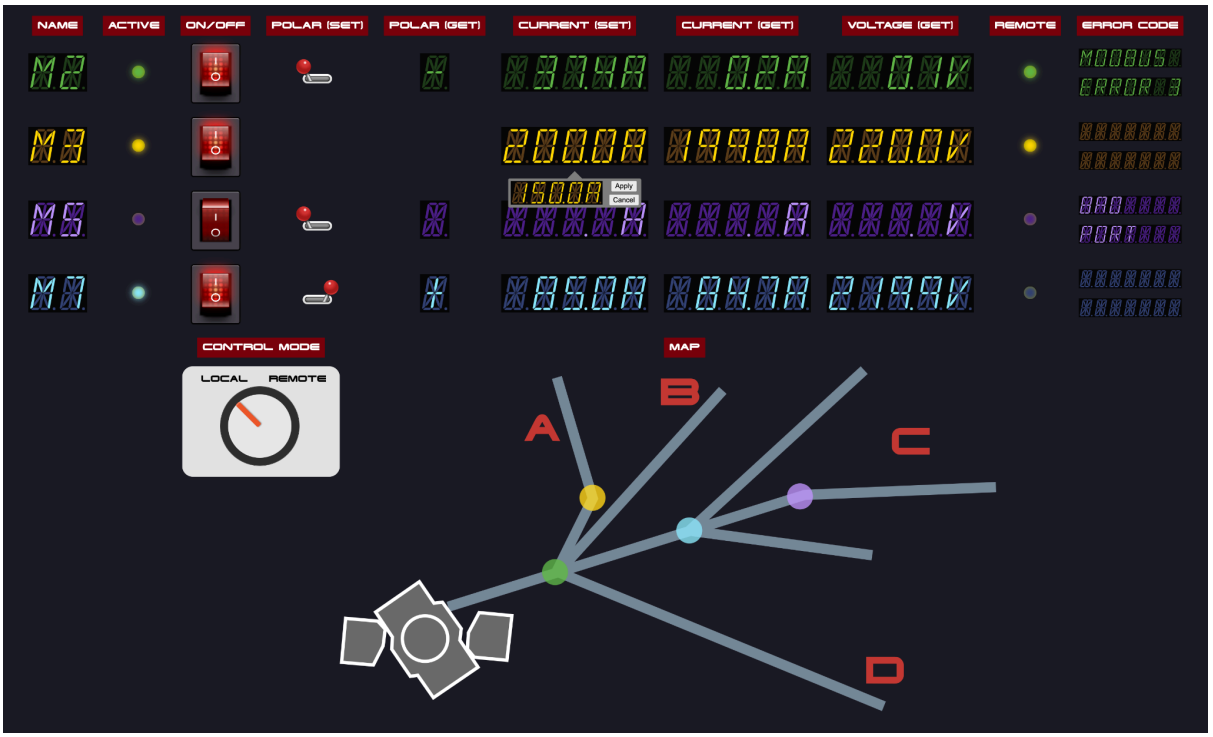

```
{
  "suppliers": [
    {
      "no": "88",
      "port": "/dev/pts/3",
      "polarity": "const",
      "maxCurrent": "200",
      "baudRate": 9600,
      "x": "-2403.3091",
      "y": "-705.22424"
    }
  ],

  "refreshInterval": 2000,
  "modbusTimeout": 500
}
```


Interfejs graficzny

W interfejsie graficznym mamy następujące elementy:

		OPIS	NAZWA ELEMENTU
		Numer zasilacza (z pliku konfiguracyjnego).	NAME
		Świeci się wtw. włączone są obwody prądowe zasilacza.	ACTIVE

OPCJE ZASILACZA	ODCZYT	W przypadku zasilacza 200 A nie występuje. Oznacza polaryzację.	POLAR (GET)
		Oznacza aktualne natężenie w zasilaczu. Natężenie wyliczane jest jako średnia z 5 następujących jeden po drugim odpytań z mikrokontrolera do zasilacza.	CURRENT (GET)
		Oznacza aktualne napięcie w zasilaczu. Podobnie jak natężenie wyliczane jest jako średnia z 5 następujących jeden po drugim odpytań z mikrokontrolera do zasilacza.	VOLTAGE (GET)
		Świeci się wtw. zasilacz pracuje w trybie zdalnym i można nim sterować w aplikacji.	REMOTE
		Wypisuje zarówno błędy połączenia, jak i samego zasilacza. Opisy poszczególnych kodów błędów są poniżej..	ERROR CODE
	ZAPIS	Włącza/wyłącza obwody prądowe. Jeśli wyłączy się obwody przy niezerowym natężeniu, najpierw stopniowo zmniejszy natężenie do 0. Jeśli zasilacz nie jest w trybie zdalnym, pokazuje, stan przycisku na zasilaczu. Wciśnięte “1” i podświetlenie oznacza, że ustawione zostało włączenie zasilacza. Wciśnięte “0” oznacza ustawione wyłączenie zasilacza.	ON/OFF
		W przypadku zasilacza 200 A nie występuje. Pozwala na zmianę polaryzacji. Dźwignia po lewej stronie oznacza, że polaryzacja będzie ujemna, po prawej - dodatnia.	POLAR (SET)
		Pozwala na wprowadzenie natężenia w jednostce Amper.	CURRENT (SET)
INNE		W obecnej wersji po wybraniu remote, blokuje możliwość wydawania poleceń zasilaczom z aplikacji, natomiast wyświetla informacje odczytane z zasilaczy. **W obecnej wersji nie umożliwia sterowania przy pomocy komputera ze sterowni.** W przyszłości będzie służył do zmiany trybu sterowania z aplikacji na komputer w centrum zarządzania i odwrotnie.	CONTROL MODE
		Schemat cyklotronu.	MAP



Kody błędów

- **`MODBUS ERROR X`**, błąd komunikacji z mikrokontrolerem, X oznacza kod błędu (Exception Code), zgodnie z protokołem modbus
- **`BAD PORT`**, port podany w pliku konfiguracyjnym nie istnieje, albo aplikacja nie ma do niego dostępu.
- **`SPLR ERR`**, błąd zasilacza
- **`CRC error`**, oznacza błędną ramkę modbus otrzymaną od mikrokontrolera
- **`Unknown`**, w pozostałych przypadkach