

Systemy Wbudowane

3 czerwca 2023

Kontroler MIDI

Dominik Grzesik, Marcin Mikuła

1. Opis projektu

Naszym zadaniem było skonfigurować mikrokontroler, tak aby po podłączeniu do komputera zgłosił się jako kontroler MIDI. Interfejs użytkownika powinien pozwalać na generowanie przynajmniej kilku dźwięków.

2. Wykorzystane platformy i sprzęt

Projekt początkowo rozwijany był na platformie STM32, na płytce F746ZG-Nucleo, a później na Raspberry Pi Pico W.

Do kodu wykorzystaliśmy:

1. STM32CubeIDE wraz z STM32CubeMX przy F746ZG-Nucleo
2. Mu Editor z CircuitPython przy Pico W

Dodatkowo wykorzystaliśmy moduł klawiatury matrycowej 4x4 od Waveshare.

3. STM32F746ZG-Nucleo

3.1 Przygotowanie

Na początku przygotowujemy projekt w **STM32CubeIDE**.

Należało:

1. Pobrać **STM32CubeMX** i wygenerować strukturę projektu dla naszej płytki - **F746ZG-Nucleo**.
2. Wejść w plik **.ioc**:
 - w **Connectivity** ustawić **USART3**
 - w **Middlewares and Software Packs** wybrać **FREERTOS** oraz **USB_DEVICE**, o którym więcej później.

3.2 Konfiguracja kontrolera MIDI

Na samym początku wybraliśmy piny, którymi łączyliśmy płytkę z modulem klawiatury. Te w kolumnach ustawiliśmy na **input** i nadaliśmy rezystorom własność **pull-up** w celu uniknięcia zakłóceń.

Tak gotowy projekt trzeba było przekształcić na **urządzenie MIDI**.

Ponieważ płytka, na której rozwijany był projekt nie wspierała klasy **MIDI Device Class**, naszym zadaniem było manualnie przekształcić na nią jedną z predefiniowanych klas. Aby to uzyskać należało ustawić odpowiednią klasę urządzenia w **USB_DEVICE** w pliku **.ioc** i zmodyfikować wygenerowane pliki:

1. Wybieramy klasę Audio Device Class, Human Interface Device Class albo Communication Device Class w **USB_DEVICE**.
2. Podmieniamy pliki **usbd_xxx.h** oraz **usbd_xxx.c** na **usbd_midi.c** oraz **usbd_midi.h** proponowane w przykładach aplikacji MIDI przez **STMicroelectronics**.
3. W plikach **usbd_device.c** oraz **usbd_desc.c** poprawić wszystkie wspomniane konfiguracje tak, aby wspierały urządzenie MIDI.

Mimo dokładnego wykonania powyższych kroków dla różnych klas urządzeń jak i ostatecznie dostawienia różnych samouczków i projektów innych osób do naszej płytki nie udało nam się skonfigurować mikrokontrolera - nie był widoczny jako urządzenie MIDI.

4. Raspberry Pi Pico W

4.1 Przygotowanie

Zanim zaczęliśmy pisać kod, należało przygotować środowisko oraz płytkę do pracy.

Należało:

1. Pobrać plik **UF2** pozwalający na programowanie Pico:
[Raspberry Pi Pico W UF2 File](#)
2. Przytrzymać przycisk **BOOTSEL**, aby zresetować płytkę i zezwolić na przekopiowanie na nią pobranego pliku
3. Pobrać bibliotekę **AdaFruit MIDI** i przekopiować folder **adafruit_midi** do folderu lib na **Pico W**
4. Jeśli w systemie plików nie było code.py, należało go utworzyć (u nas był)

4.2 Kod

```
import board
import digitalio
import pwmio
import time
import usb_midi
import adafruit_midi
from adafruit_midi.note_on import NoteOn
from adafruit_midi.note_off import NoteOff

midi = adafruit_midi.MIDI(midi_out=usb_midi.ports[1], out_channel=0)

print("MacroPad MIDI Board")

button_pins = [board.GP0, board.GP1, board.GP2, board.GP3, board.GP4, board.GP5, board.GP6, board.GP7]
keys = [['ENT', '0', 'ESC', 'ONOFF'],
        ['7', '8', '9', 'LOCK'],
        ['4', '5', '6', 'GO'],
        ['1', '2', '3', 'STOP']]

rows = [board.GP4, board.GP5, board.GP6, board.GP7]
columns = [board.GP0, board.GP1, board.GP2, board.GP3]

keypad_rows = []
keypad_columns = []

for i in rows:
    keypad_rows.append(digitalio.DigitalInOut(i))

for i in columns:
    keypad_columns.append(digitalio.DigitalInOut(i))

col_pins = []
row_pins = []
```

```

for pin in keypad_rows:
    pin.direction = digitalio.Direction.OUTPUT
    pin.value = False
    row_pins.append(pin)

for pin in keypad_columns:
    pin.direction = digitalio.Direction.INPUT
    pin.pull = digitalio.Pull.UP
    col_pins.append(pin)

notes= [['F3', 'G3', 'A3', 'B3'],
        ['C3', 'D3', 'E3', 'D4'],
        ['G4', 'A5', 'B5', 'C5'],
        ['D5', 'E5', 'F6', 'G6']]

def fix_layout(to_fix):
    n=len(to_fix)
    fixed=[-1 for _ in range(n)]
    for i in range(n):
        fixed[(i+n-1)%n]=to_fix[i]
    return fixed
note_mapping=fix_layout(notes)
keys_pressed = [[False for _ in range(4)] for _ in range(4)]
def scankeys():
    for row in range(0, 4):
        for col in range(0, 4):
            row_pins[row].value = False
            key = None

            key_press = note_mapping[row][col]
            if col_pins[col].value == False and keys_pressed[row][col]==False:
                keys_pressed[row][col] = True
                print("You have pressed:", keys[row][col], key_press)
                midi.send(NoteOn(key_press, 60))

            if col_pins[col].value == True and keys_pressed[row][col] == True:
                keys_pressed[row][col] = False
                midi.send(NoteOff(key_press, 0))
            row_pins[row].value = True

while True:
    scankeys()

```

4.3 Krok po kroku:

- Wykorzystując funkcję **adafruit_midi.MIDI** przekształciliśmy nasze urządzenie na kontroler MIDI.
- Definiujemy listę pinów, którymi łączymy płytke z klawiaturą.
- Przy użyciu modułu **digitalio** możemy nadawać pinom odpowiednie stany i własności. W naszym przypadku na pinach w kolumnach ustawialiśmy **input** oraz rezystory w trybie **pull up**, aby uniknąć błędnych odczytów i mieć stały stan pinu w momencie kiedy

przypisany mu przycisk nie jest wciśnięty. Wybrane piny w wierszach ustawiamy na **output** oraz ich wartość na false.

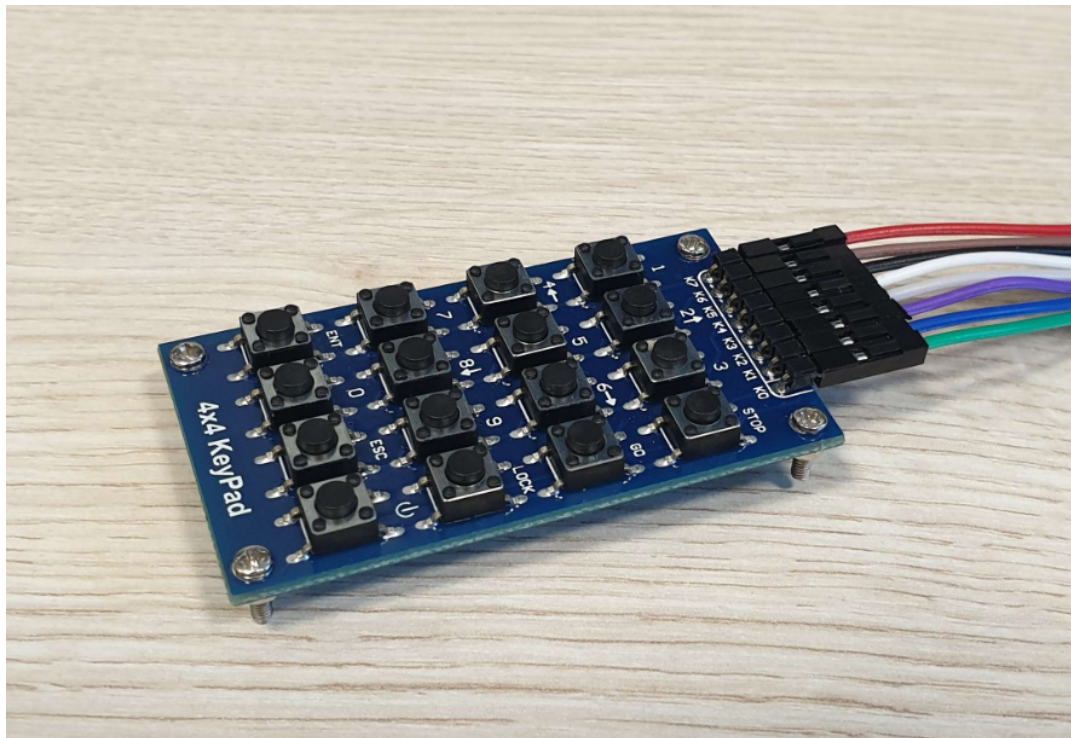
- Mapujemy klawisze klawiatury na wybrane przez nas nuty.
- Funkcja **fix_layout()** pomagała nam na przestawienie wierszy tak, aby bardziej intuicyjnie definiowało się mapę nut.
- **scankeys()** nasłuchuje na wciśnięcie klawisza i pozwala na wysłanie **sygnału MIDI** przy pomocy funkcji **midi.send**.
Uwzględniane jest również przytrzymanie klawisza przy wykorzystaniu tablicy **keys_pressed**.

4.3 Efekt

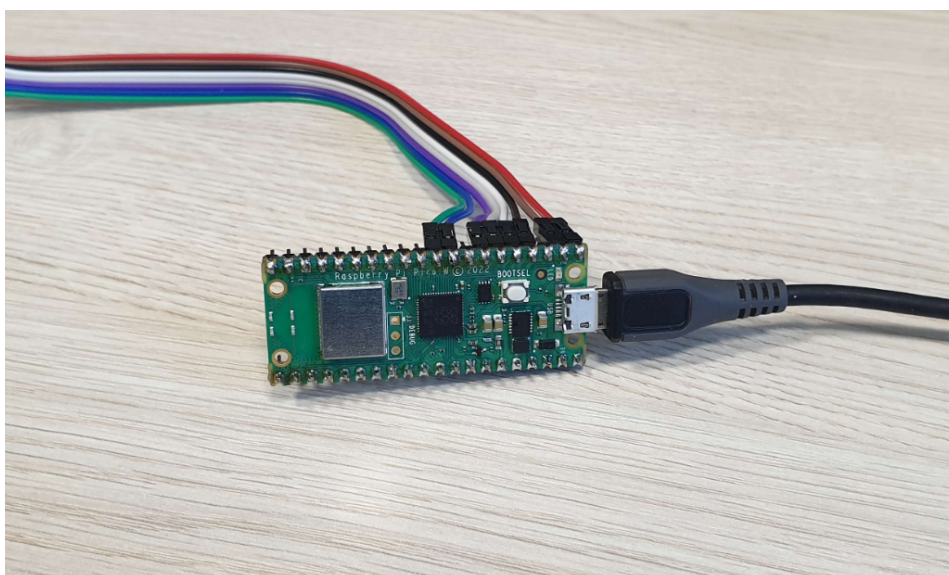
Urządzenie po podpięciu do komputera zostaje wykryte jako kontroler MIDI.

Mikrokontroler wraz z klawiaturą pozwala na granie na 16 nutach.

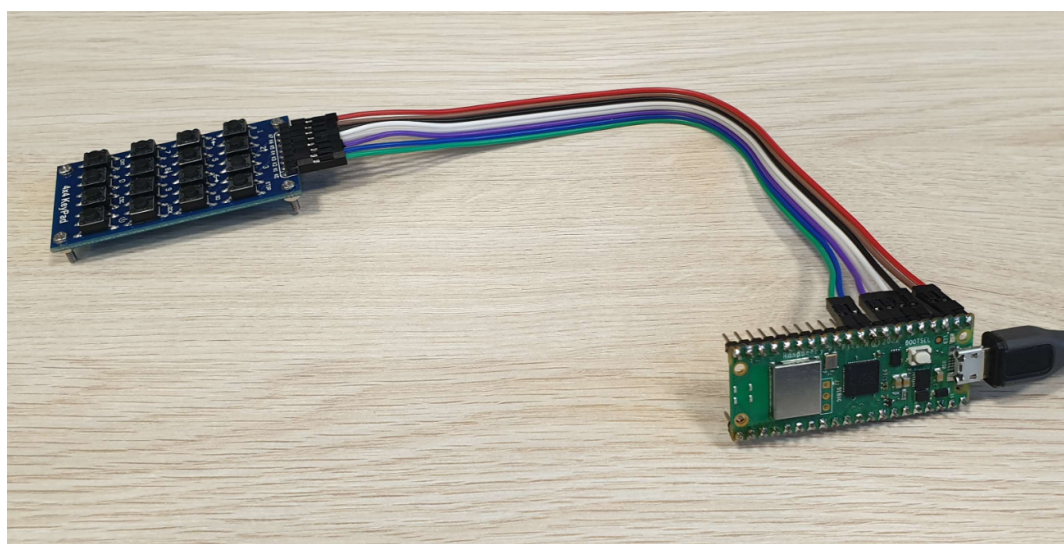
4.4 Zdjęcia



1. Klawiatura 4x4 firmy Waveshare



2. Płytki Raspberry Pi Pico W



3. Zbudowany kontroler MIDI

5. Podsumowanie

Ostatecznie w wyniku naszej pracy udało się stworzyć kontroler MIDI na platformie **Raspberry Pi Pico W**, a jego działanie przetestowaliśmy i zaprezentowaliśmy na zajęciach.