Magnetyczny sterownik MIDI

Granie na większości instrumentów nie należy do najprostszych. Na szczęście w sukurs przychodzą urządzenia elektroniczne, które pozwalają w prostszy sposób generować melodie. Niezależnie od tego czy jesteś kompozytorem muzycznym, melodystą, symfonikiem, czy też melomanem, który uwielbia tworzyć własne rytmy, jeśli znudziły Ci się tradycyjne kontrolery MIDI, to przedstawione poniżej urządzenie będzie ciekawą alternatywą do domowego studia.

Prezentowany system zawiera obracające się dyski z magnesami w formie sfer, które wyzwalają kolejne komunikaty MIDI. Zaprezentowane urządzenie pozwoli w kreatywny sposób syntezować nawet złożone sekwencje dźwiękowe, a dzięki prostej i intuicyjnej obsłudze, nadaje się nawet dla amatorów. Dodatkowo, zawarte poniżej opisy samego interfejsu MIDI i sposobu oprogramowania go na platformie Arduino, pozwolą na łatwiejsze konstruowanie samodzielnie wymyślonych interfejsów w przyszłości.

Zasada działania

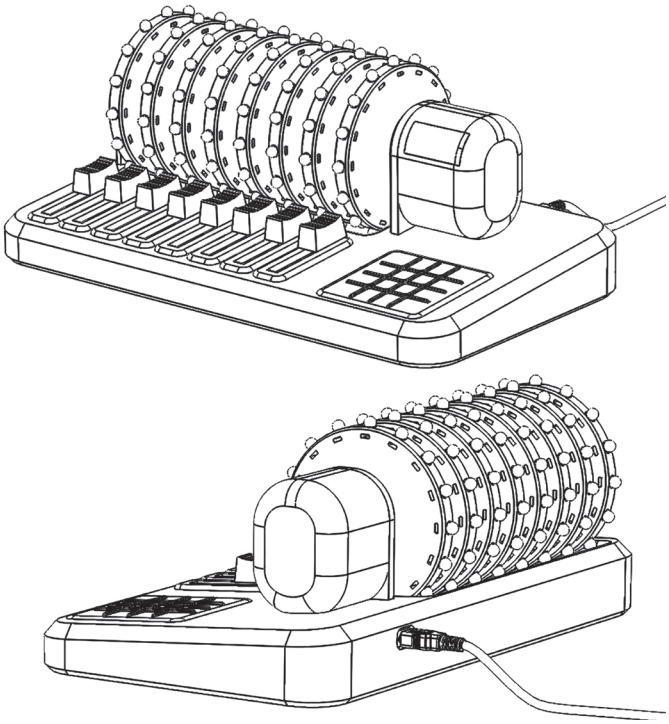
MIDI to interfejs cyfrowych instrumentów muzycznych, w ramach ekosystemu MIDI mieszczą się elektroniczne instrumenty muzyczne, oprogramowanie, komputery, a także inne cyfrowe kontrolery współdzielące pomiędzy sobą informacje muzyczne. Najlepszym sposobem na zrozumienie, czym jest MIDI, jest zrozumienie, czym nie jest:

- Sygnał MIDI to nie muzyka,
- MIDI nie zawiera w sobie żadnych rzeczywistych dźwięków,
- MIDI nie jest cyfrowym formatem plików muzycznych, takim jak są pliki MP3 czy WAV.

MIDI to nic innego jak strumień danych – zestaw instrukcji. Dane przesyłane poprzez interfejs MIDI zawierają w sobie listę zdarzeń oraz komunikatów, informujących urządzenia elektroniczne (instrument muzyczny, kartę dźwiękową w komputerze, telefon komórkowy, syntezator itp.), w jaki sposób wygenerowany ma być kolejny dźwięk. Oto kilka przykładów typowych komunikatów MIDI:

- **Naciśnięcie klawisza** – sygnalizuje umowne naciśnięcie klawisza, tj. zadanie pewnego dźwięku, niekoniecznie z fizycznej klawiatury. Komunikat ten może pochodzić z dowolnego instrumentu, sekwencera etc. Zawiera kompletne instrukcje dotyczące dźwięku – który klawisz został naciśnięty oraz jak mocno (parametr ten umownie nazywa się prędkością).
- **Puszczenie klawisza** – sygnalizuje zwolnienie klawisza, czyli zaprzestanie odtwarzania danego dźwięku.
- **Polifoniczne ciśnienie klawisza** – to miara siły nacisku na klawisz, który znajduje się już na samym dole i został naciśnięty. W przypadku niektórych urządzeń powoduje to dodanie dodatkowych efektów do odtwarzanej nuty, pozwala np. na stworzenie efektu vibrato.
- **Zmiana sterowania** – sygnał, który wskazuje, że kontroler – na przykład pedał nożny lub suwak na instrumencie – został naciśnięty lub zmienił położenie. Komunikat zmiany sterowania zawiera w sobie numer przypisany do danego kontrolera i wartość zmiany (w zakresie od 0 do 127).
- **Zmiana pozycji koła wysokości nuty** – sygnalizuje zmianę pozycji tzw. pitch wheel, które to pozwala na płynne przechodzenie pomiędzy wysokością dźwięku. Pozwala to na dodanie efektu tzw. portamento do granej sekwencji.

Elektrycznie rzecz biorąc interfejs MIDI jest szeregowym interfejsem *half-duplex*, który pracuje w pętli prądowej (typowo 5 mA dla stanu „0” i ok. 0 mA dla stanu „1” – układ pracuje w logice odwrotnej). Nominalna prędkość działania interfejsu wynosi 31 250 bitów na sekundę. W przypadku tego urządzenia wykorzystana zostanie emulacja interfejsu MIDI poprzez interfejs USB na jednym z zastosowanych modułów Arduino.

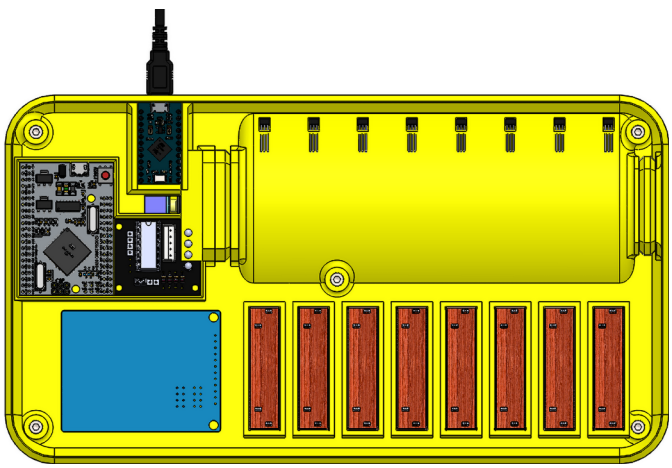


Rysunek 1. Widok projektu sterownika

Zaprezentowany sterownik, do generowania komunikatów MIDI wykorzystuje sensory magnetyczne i suwaki. Nad sensorami Halla, znajdującymi się w dolnej części urządzenia, obracają się koła z tworzywa, które posiadają po 16 zagłębień. Zagłębienia te rozmieszczone są na obwodzie koła i służą do przymocowania malutkich, sferycznych magnesów neodymowych. Takich kół w systemie jest 8. Każdy z ośmiu kanałów jest niezależnie sterowany za pomocą suwaka, który odpowiada za wysokość nuty, przesyłanej przy każdym wyzwoleniu kanału do interfejsu MIDI. Za wyzwalanie kanału odpowiadają sensory Halla – za każdym razem, gdy nad sensorem przesunie się magnes na kole, dany kanał zostanie wyzwolony. Umieszczając na kole magnesy, sterujemy konfiguracją dźwięków, a regulując prędkość obrotową kół, zmieniamy tempo naszej melodii.

Projekt, budowa i montaż części mechanicznej

Cały projekt urządzenia wykonany został z pomocą SolidWorksa. Autor projektu zadbał, by gotowe urządzenie było kompaktowe, łatwe w obsłudze i estetyczne. Duży nacisk położono na wykorzystanie dostępnych komercyjnie elementów. Obudowa, jak i większość elementów mechanicznych sterownika, wykonane zostały techniką druku 3D. Wykorzystano drukarki FDM oraz DLP. Autor udostępnia pliki STL z poszczególnymi elementami wraz z użytymi parametrami druku, na stronie swojego projektu na portalu Instructables (link na końcu projektu). Na **rysunku 1** pokazano projekt całego urządzenia.



Rysunek 2. Model 3D podstawy sterownika MIDI wraz z zainstalowanymi komponentami elektronicznymi

Podstawa

Podstawa sterownika jest miejscem, gdzie znajduje się cała elektronika urządzenia. Model 3D złożonej podstawy układu pokazano na **rysunku 2**. Do jej montażu, oprócz wydrukowanego elementu, potrzebne będą:

- moduł Arduino Micro,
- moduł Arduino Mega Pro Mini,
- klawiatura dotykowa TTP229 z 16 przyciskami,
- osiem sensorów Halla A3144,
- osiem potencjometrów suwakowych 30 mm o rezystancji maksymalnej 22 kΩ,
- moduł ze sterownikiem mocy ULN2003 do sterowania silnikiem krokowym.

Wszystkie elementy instalowane są w podstawie sterownika na wcisk, bez konieczności stosowania śrub czy kleju. Zadbano o tolerancje dla wszystkich elementów tak, że wszystko powinno pasować na swoje miejsce bez konieczności przerabiania elementów w podstawie ani instalowanych modułów.

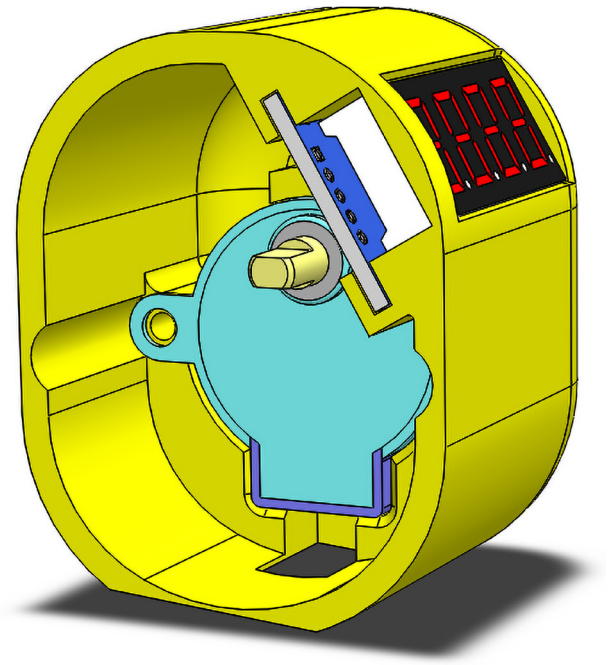
Obudowa silnika

Kluczowym elementem sterującym kontrolerem MIDI jest silnik krokowy, który będzie obracał koła z magnesami. Posiada on własną obudowę, w której, oprócz samego silnika, znajduje się również poczwórny wyświetlacz siedmiosegmentowy do prezentacji informacji na temat wybranego w danym momencie tempa rytmu.

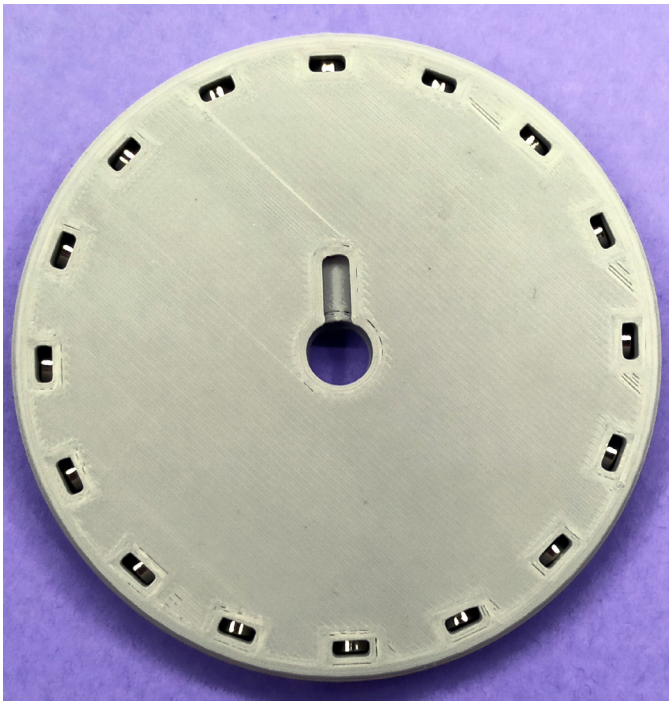
Trójwymiarowy model obudowy silnika, wraz z zainstalowanym elementami pokazano na **rysunku 3**. Do złożenia tego elementu potrzebne są:

- wydrukowana obudowa silnika,
- wydrukowana obudowa dla modułu LED,
- moduł 4-bitowego wyświetlacza siedmiosegmentowego – TM1637,
- silnik krokowy 28-BYJ48.

Instalacja wszystkich komponentów jest równie prosta, co w przypadku elementów elektronicznych w podstawce sterownika. W pierwszej kolejności umieszczamy na miejscu silnik, a następnie moduł LCD, który przykrywamy jego obudową. Jej zadaniem jest zasłonić



Rysunek 3. Model 3D obudowy silnika krokowego wraz z zainstalowanym silnikiem i wyświetlaczem siedmiosegmentowym



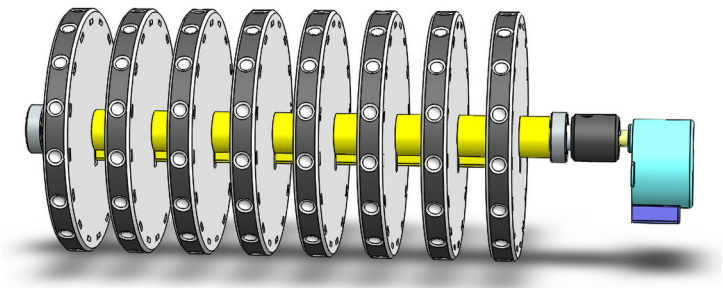
Fotografia 1. Zmontowane koło z magnesami wsuniętymi w otwory

sam wyświetlacz tak, aby możliwe było tylko przejście przez nią światła. Wszystkie elementy powinny bez problemu i konieczności przeróbek obudowy pasować na swoje miejsce. Całość montujemy bez wykorzystania kleju czy śrub.

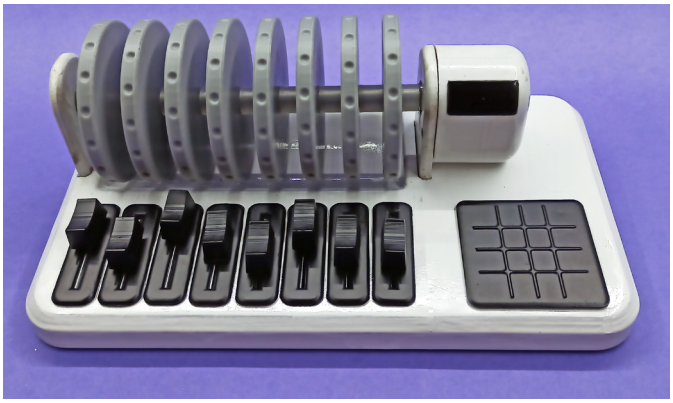
Koła z magnesami

Bardzo istotnym elementem sterownika są koła z magnesami, gdyż to one regulują rytm w urządzeniu. Sterownik posiada osiem takich kół. Po wydrukowaniu elementów, w każdym z kół osadzane jest 16 płaskich magnesów neodymowych 4×1,5 mm – służą one przytrzymywaniu magnetycznych sfer (o średnicy 5 mm) w ich gniazdach na obwodzie koła. Instalując magnesy w kole należy pamiętać o odpowiedniej ich biegunowości – wszystkie powinny mieć biegun N skierowany do środka koła, a biegun S do zewnętrza. Zmontowane koło z magnesami pokazane jest na **fotografii 1**.

Zespół ośmiu kół umieszczany jest następnie na wspólnej osi. Jako oś wykorzystano stalowy wałek o średnicy 8 mm i długości 184 mm. Taki wałek zamówić można z sklepu z akcesoriami do automatyki lub modeli RC. Pomiędzy poszczególnymi kołami umieszczane są dystanse o szerokości 10 mm, co odpowiada rastrowi, z jakim rozmieszczone są sensory Halla w podstawie urządzenia. Oś z jednej strony zakończona jest nakładką, zabezpieczającą koła przed wysunięciem się i łożyskiem 688zz. Po drugiej stronie również znajduje się dokładnie takie samo łożysko oraz uchwyt dla silnika, który pozwala na sztywne połączenie osi kół z osią silnika krokowego. Wszystkie elementy, oprócz osi i łożysk, wykonane są w technologii druku 3D. Na **rysunku 4** pokazano model 3D kół z magnesami, zainstalowanych na osi połączonej z silnikiem krokowym.



Rysunek 4. Oś z zainstalowanymi kołami z magnesami i silnikiem krokowym



Fotografia 2. Fotografia przedstawiająca w pełni zmontowany sterownik

Montaż całości urządzenia

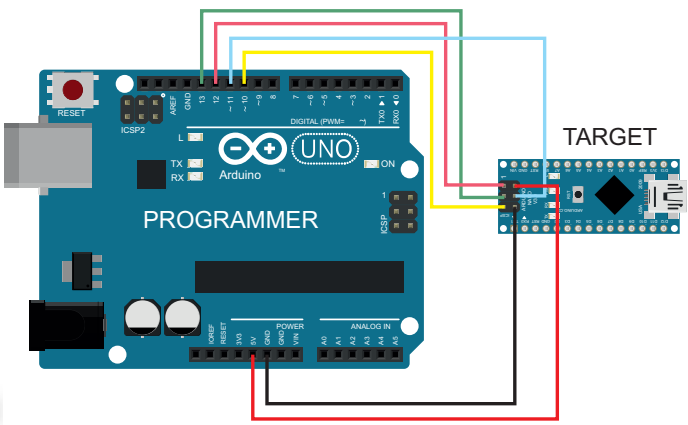
Po złożeniu wszystkich podzespołów razem, można zamontować, z wykorzystaniem wsporników, oś z kołami dla magnesów i silnikiem na podstawie urządzenia. Upewnijmy się, że oś z magnetycznymi kołami obraca się bez oporów w swoich łożyskach, a same magnesy nie ocierają o żadne elementy podstawy. Gotowe urządzenie pokazano na **fotografii 2**. Po zmontowaniu urządzenia można przystąpić do instalacji oprogramowania na poszczególnych mikrokontrolerach oraz łączenia ze sobą wszystkich elementów elektronicznych.

Firmware dla Arduino Micro

W prezentowanym sterowniku do komunikacji z komputerem wykorzystany jest interfejs MIDI, emulowany poprzez USB. Do emulacji tego interfejsu wykorzystano mikrokontroler ATmega32U4 w module Arduino Micro. Do obsługi interfejsu MIDI zastosowano oprogramowanie wbudowane mocoLUFA MIDI, skompilowane dla ATmega32U4. Korzystając z mocoLUFA, Arduino Micro będzie działać, jako natywne urządzenie MIDI, które wysyła polecenie MIDI do komputera poprzez port USB.

Ponadto Arduino z oprogramowaniem mocoLUFA posiada port szeregowy, który umożliwia Flashowanie oprogramowania wbudowanego na innych modułach przez ten kontroler, zależnie od tego, w jakim trybie się go uruchomi. Do zaprogramowania Arduino Micro potrzebny jest programator ISP, emulowany przez moduł Arduino UNO, jak pokazano na **rysunku 5**, podłączony do komputera przez USB.

Oprogramowanie dla Arduino Micro można pobrać ze strony z projektem. Po pobraniu należy je skompilować wpisując w linii komend „make all" w folderze z projektem. Następnie, po podłączeniu Arduino UNO do Arduino Micro, jak pokazano na rysunku 5, można przystąpić do programowania układu korzystając z oprogramowania AVRDUDESS. Do Arduino Micro przesyłamy pliki dualMOCO.hex oraz dualMOCO.eep. Po zaprogramowaniu Arduino Micro możemy podłączyć do układu wszystkie pozostałe elementy.

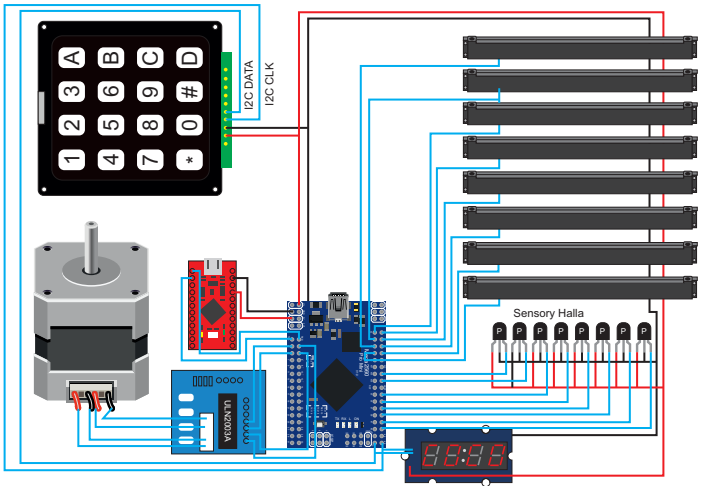


Rysunek 5. Schemat podłączenia Arduino UNO, jako programatora do Arduino Micro

Listing 1. Fragment listingu programu dla Arduino Mega

```
uint16_t BPM_INTERVAL = 120;
const uint16_t stepsPerRevolution = 4096;

MIDI_CREATE_INSTANCE(HardwareSerial, MIDI, midiOut);
TM1637 bpm_display(tm1637Clock,tm1637Data);
TTP229 touch_pad(ttp229Clock, ttp229Data);
Unistep2 stepper_motor(motorPin1, motorPin2, motorPin3, motorPin4, stepsPerRevolution, 1000000/((BPM_INTERVAL/60)*(stepsPerRevolution/16)));

void updateDisplay(uint16_t data);
void readActuatorsCallback();
void controlStepperCallback();

Task readActuators(50, TASK_FOREVER, &readActuatorsCallback);
Task controlStepper(TASK_IMMEDIATE, TASK_FOREVER, &controlStepperCallback);

Scheduler runner;

void setup() {
    for (size_t i = 0; i < 8; i++) {
        pinMode(hallSensors[i], INPUT);
        pinMode(sliderPots[i], INPUT);
        previous_hallSensorsState[i] = digitalRead(hallSensors[i]);
        previous_sliderPotsState[i] = (map(analogRead(sliderPots[i]), 0, 1023, 0, 127)) / 10;
    }
    #ifdef DEBUG
        DEBUG.begin(115200);
        DEBUG.println(„MesoTune Started”);
        DEBUG.print(„\n”);
        DEBUG.print(„Hall Sensor   =>\t”);
        for (size_t i = 0; i < 8; i++) {
            DEBUG.print(previous_hallSensorsState[i]);
            i != 7 ? DEBUG.print(„\t”) : DEBUG.print(„\n”);
        }
        DEBUG.print(„Potentiometer =>\t”);
        for (size_t i = 0; i < 8; i++) {
            DEBUG.print(previous_sliderPotsState[i]);
            i != 7 ? DEBUG.print(„\t”) : DEBUG.print(„\n”);
        }
        DEBUG.print(„\n”);
    #endif
    // Inicjalizacja kanałów MIDI
    midiOut.begin(MIDI_CHANNEL_OMNI);
    // Rozpocznij odczyt z sensorów
    runner.addTask(readActuators);
    readActuators.enable();
    // Konfiguracja wyświetlacza tempa
    bpm_display.init();
    //Różne poziomy jasności: BRIGHT_TYPICAL = 2,
/   //BRIGHT_DARKEST = 0, BRIGHTEST = 7;
    bpm_display.set(BRIGHT_TYPICAL);
    // Ustaw aktualne tempo na wyświetlaczu
    updateDisplay(BPM_INTERVAL);
    // Uruchom silnik krokowy
    runner.addTask(controlStepper);
    controlStepper.enable();
}

void loop() {
    runner.execute();
    stepper_motor.run();
}

void readActuatorsCallback() {
    // Odczyt wartości sensorów Halla
    for (size_t i = 0; i < 8; i++) {
        current_hallSensorsState[i] = digitalRead(hallSensors[i]);
        if (current_hallSensorsState[i]  != previous_hallSensorsState[i] ) {
            #ifdef DEBUG
                DEBUG.print(„Hall Sensor change at „);
                DEBUG.print(i);
                DEBUG.print(„ from „);
                DEBUG.print(previous_hallSensorsState[i]);
                DEBUG.print(„ to „);
                DEBUG.println(current_hallSensorsState[i]);
            #endif
            current_hallSensorsState[i] ? midiOut.sendNoteOn(48, 127, i) : midiOut.sendNoteOff(48, 0, i);
            previous_hallSensorsState[i] = current_hallSensorsState[i];
        }
    }
    // Odczyt wartości potencjometrów
    for (size_t i = 0; i < 8; i++) {
        current_sliderPotsState[i] = (map(analogRead(sliderPots[i]), 0, 1023, 0, 127)) / 10;
        if (current_sliderPotsState[i]  != previous_sliderPotsState[i] ) {
            #ifdef DEBUG
                DEBUG.print(„Potentiometer change at „);
                DEBUG.print(i);
                DEBUG.print(„ from „);
                DEBUG.print(previous_sliderPotsState[i]);
                DEBUG.print(„ to „);
                DEBUG.println(current_sliderPotsState[i]);
            #endif
            midiOut.sendControlChange(i, map(analogRead(sliderPots[i]), 0, 1023, 0, 127), i);
            previous_sliderPotsState[i] = current_sliderPotsState[i];
        }
    }
    uint8_t key = ttp229.GetKey16();
    // Odczyt z klawiatury dotykowej
    switch (key) {
    case 1:
        if (BPM_INTERVAL > 50) {
            BPM_INTERVAL--;
            updateDisplay(BPM_INTERVAL);
            stepper_motor.updateTime(1000000/((BPM_INTERVAL/60)*(stepsPerRevolution/16)));
        }
        break;
    case 4:
        if (BPM_INTERVAL < 250) {
            BPM_INTERVAL++;
            updateDisplay(BPM_INTERVAL);
            stepper_motor.updateTime(1000000/((BPM_INTERVAL/60)*(stepsPerRevolution/16)));
        }
```

```
Listing 1. cd.
            break;
        default:
            break;
        }
}

void updateDisplay(uint16_t data) {
    size_t i = 3;
    while (data > 0) {
        bpm_display.display(i, data % 10);
        data /= 10;
        i--;
    }
}

void controlStepperCallback() {
    stepper_motor.move(stepsPerRevolution);
}
```



Rysunek 6. Szczegóły połączeń elektrycznych całego systemu

Połączenie wszystkich komponentów

Po zaprogramowaniu modułu Arduino Micro można podłączyć go do Arduino Mega, które pełni w sterowniku kluczową rolę. Jak pokazano na **rysunku 6**, wszystkie elementy systemu sterowane są właśnie przez ten moduł. Szczegóły połączeń elektrycznych w systemie zawarto na **rysunku 6** jak i w **tabeli 1**. W tabeli pominięto dla uproszczenia połączenia linii zasilania i masy.

Oprogramowanie dla Arduino Mega

Po podłączeniu wszystkich elementów zgodnie z listą zamieszczoną w tabeli 1 można przystąpić do ostatniego kroku budowy sterownika – programowania modułu Arduino Mega, będącego sercem urządzenia. Program został napisany w środowisku Arduino, najistotniejszy

Tabela 1. Lista połączeń Arduino Mega z innymi układami

Arduino Mega	Peryferia
RXD	TXD (Arduino Micro)
TXD	RXD (Arduino Micro)
2	IN1 (ULN2003)
4	IN2 (ULN2003)
5	IN3 (ULN2003)
3	IN4 (ULN2003)
A0...A7	Środkowy odczep potencjometrów, włączonych pomiędzy linię zasilania a masę (od 1 do 8)
D32...D39	Wyjścia sensorów Halla (od 1 do 8)
46	CLK (TTP229)
47	SDO (TTP229)
46	CLK (TM1637)
47	SDO (TM1637)

fragment pokazany jest na **listingu 1**. Wykorzystuje on następujące zewnętrzne biblioteki, które trzeba dodatkowo zainstalować w naszym środowisku Arduino IDE:

- UniStep2,
- Grove_4Digital_Display,
- Arduino MIDI,
- TaskScheduler.

Po przekopiowaniu źródła do Arduino IDE z zainstalowanymi bibliotekami można już zaprogramować Arduino Mega. Programatorem jest tutaj Arduino Serial, który realizowany jest przez Arduino Micro. Aby wprowadzić je w stan programowania należy ściągnąć pin 16 tego modułu do masy przed podłączeniem do niego kabla USB. Dzięki temu moduł uruchomi się, jako wirtualny port szeregowy (tak powinien wykryć go komputer).

Po załadowaniu programu do Arduino Mega można rozłączyć połączenie 16 pinu Arduino Micro z masą i zresetować ten moduł przykładając na chwilę do pinu resetu stan niski. Teraz moduł załaduje się, jako sterownik MIDI i jest już gotowy do działania. Może sterować dowolnym programem, który przyjmuje sygnał z urządzenia MIDI do sterowania np. wielu cyfrowych syntezatorów i programów typu DAW (*Digital Audio Workstation* – cyfrowa stacja audio).

Nikodem Czechowski, EP

Źródło: ***http://bit.ly/3bUvHoI***