UNIWERSYTET MORSKI W GDYNI
WYDZIAŁ ELEKTRYCZNY
Katedra Elektroniki Morskiej

Nr ewidencyjny:.....
Data złożenia pracy: 29.01.2024

PRACA DYPLOMOWA INŻYNIERSKA

Temat: Projekt i konstrukcja automatu flipper w oparciu o układ Arduino Mega 2560

Dyplomant:	Tadeusz Kazimierz Lorkowski	Numer albumu: 47530
Specjalność:	Aplikacje Internetowe i Mobilne	
Promotor:	dr hab. inż. Przemysław Ptak, prof. UMG	Ocena:
Recenzent	dr inż. Jacek Dąbrowski	Ocena:
Data egzaminu:		Wynik studiów:

Recenzent: .....Promotor: .....Dziekan: .....

Wyrażam zgodę / nie wyrażam zgody na
udostępnianie mojej pracy dyplomowej

....
datapodpis

Gdynia 2024

Tadeusz Lorkowski

Imię i Nazwisko

Gdynia, dnia 29.01.2024 r.

OŚWIADCZENIE

Świadomy/a odpowiedzialności prawnej oświadczam, że złożona praca inżynierska/magisterska* pt.: *Projekt i konstrukcja automatu flipper w oparciu o układ Arduino Mega 2560* została napisana przeze mnie samodzielnie.

Równocześnie oświadczam, że w pracy wykorzystano tylko cytowaną literaturę a więc praca nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz. U. 1994, nr 24, poz. 83) oraz dóbr osobistych chronionych prawem cywilnym.

Ponadto praca nie zawiera informacji i danych uzyskanych w sposób nielegalny i nie była wcześniej przedmiotem innych procedur urzędowych związanych z uzyskaniem dyplomów lub tytułów zawodowych uczelni wyższej.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną na płycie CD.

Na podstawie art. 75 §2 kodeksu postępowania administracyjnego wnoszę o odebranie tego oświadczenia jako dowodu prawdziwości okoliczności w nim podatnych, przy czym jestem świadomy odpowiedzialności karnej z art. 233 §1 i §6 k.k. za złożenie fałszywego oświadczenia.

……

podpis

* niepotrzebne skreślić

SPIS TREŚCI

WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW

AD KEY – *Analog-Digital Keypad*
ASIC – *Application-Specific Integrated Circuit*
CGRAM - *Character Generator Random Access Memory*
CGROM – *Character Generator Read-Only Memory*
DDRAM – *Display Data Random Access Memory*
DSP – *Digital Signal Processing*
GPIO – *General-Purpose Input and Output*
IDE – *Integrated Development Environment*
I/O – *Input/Output*
LCD – *Liquid-Crystal Display*
LED – *Light-Emitting Diode*
PWM – *Pulse Width Modulation*
SPI – *Serial Peripheral Interface*
TWI – *Two-Wire Interface*
UART – *Universal Asynchronous Receiver-Transmitter*
USART – *Universal Synchronous and Asynchronous Receiver-Transmitter*
USB – *Universal Serial Bus*

WSTĘP

Jednym z zastosowań mikroprocesorów i mikrokontrolerów są wszelkiego rodzaju automaty do gier, zarówno te, które podobnie jak komputery PC wyświetlają grę na ekranie, jak i te wykorzystujące do gry elementy elektromechaniczne. Przykładem takiego automatu są flippery, nazywane też pinballami. O ile w latach osiemdziesiątych XX wieku były one jedną z najpopularniejszych form rozrywki i lukratywnym biznesem dla właścicieli salonów gier, w XXI wieku automaty flipper są zdecydowanie mniej popularne niż w wieku ubiegłym, co potwierdza fakt, iż na rynku została już tylko jedna firma podejmująca się ich produkcji – Stern Pinball (dawniej Data East Pinball). Tego typu gry polegają na odbijaniu metalowej kulki od różnych elementów stołu za pomocą łapek (ang. *flippers*), obsługiwanych z użyciem elektromagnesów. Mikroprocesory we flipperach wykorzystuje się od prawie pięćdziesięciu lat – pierwsze tego typu automaty, produkowane od lat siedemdziesiątych XX wieku, nazywano stołami półprzewodnikowymi (ang. *solid-state*). Z ich użyciem wprowadzono rozwiązania niemożliwe we wcześniejszych, czysto elektromechanicznych stołach, takie jak cyfrowy licznik punktów, zapisywanie najlepszych wyników czy wynagradzanie gracza dodatkowymi rundami gry. Procesor jest jednym z elementów tzw. płyty automatowej (ang. *arcade system board*), do reszty należą pamięci, układy DSP (*digital system processing*) związane z odtwarzaniem dźwięku oraz układy ASIC (*application-specific integrated circuit*) łączące procesor z elementami wykorzystywanymi do gry [13].

Celem mojej pracy jest utworzenie prototypu przykładowego układu elektronicznego flippera, zarówno od strony sprzętowej, jak i programowej, oraz opisanie elementów w nim wykorzystywanych z uwzględnieniem ich topologii, zasady działania i uzasadnienia ich wykorzystania. W jej ramach mam zamiar także udowodnić wykonalność takiego projektu z użyciem popularnego i łatwego w programowaniu mikrokontrolera AVR, ATmega2560. Zawiera on nie tylko mikroprocesor i pamięć, ale także porty GPIO (*general-purpose input/output*), potencjalnie zastępujące tradycyjne układy ASIC. Projektowanie można jeszcze bardziej uprościć poprzez wykorzystanie popularnej płyty rozwojowej Arduino Mega 2560 Rev3. Poza mikrokontrolerem układ flippera powinien zawierać wskaźniki postępu gry, na przykład w formie diod LED i cyfrowego wyświetlacza LCD (*liquid-crystal display*), układ DSP służący do odtwarzania dźwięków, przykładowo DFPlayer, oraz właściwe elementy gry, oparte na solenoidach [1].

1. ANALIZA I OMÓWIENIE UKŁADÓW CYFROWYCH

W układzie automatu wykorzystano trzy gotowe komercyjne podzespoły elektroniczne – płytę rozwojową Arduino Mega 2560 oraz dwie płyty z układami cyfrowymi gotowymi do programowania z użyciem mikrokontrolera, popularnie nazywane „modułami". Pierwsza z nich, DFPlayer, jest układem DSP umożliwiającym odtwarzanie plików dźwiękowych z karty microSD przez zewnętrzny głośnik, druga natomiast to podświetlany wyświetlacz matrycowy LCD sterowany przez popularny układ HD44780. Pierwsze dwa układy, połączone ze sobą, wypełniają wszystkie funkcje płyty tzw. płyty automatowej. Wyświetlacz natomiast stanowi wskaźnik postępu gry, informując gracza o wyniku i numerze rundy.

Wykorzystanie Arduino Mega w tym projekcie motywuję dużą dostępnością i małym nakładem pracy, przy zachowaniu wystarczających walorów użytkowych wymaganych do jego realizacji. Konfiguracja połączenia płyty rozwojowej z wykorzystywanym do programowania komputerem jest bowiem o wiele łatwiejsza od łączenia samodzielnego mikrokontrolera, co zresztą dodatkowo wymaga zbudowania układu obsługującego taki mikrokontroler oraz jego programatora. Innymi słowy, wymaga to konstrukcji własnego Arduino, co ciągnie za sobą dodatkowy nakład pracy, a nie wnosi nic do procesu projektowania układu flippera. Wykorzystany mikrokontroler powinien także mieć wystarczającą ilość rejestrów i wyprowadzeń GPIO, co czyni ATmega2560 bardziej trafionym wyborem od bardziej popularnego ATmega328P, znajdującego się na tańszej płycie Arduino Uno.

Procesor dźwiękowy DFPlayer został wybrany dlatego, że w zasadzie jest on jedynym układem w swoim rodzaju. Oczywiście od wielu lat produkuje się wiele różnych procesorów DSP dedykowanych przetwarzaniu dźwięku, ten jednak jako jedyny reklamowany jest jako procesor bezpośrednio realizujący przetwarzanie sygnałów dźwiękowych z nośnika danych, a nie przykładowo z Arduino. Podobnie jak w poprzednim przypadku, głównym powodem zastosowania tego podzespołu w obwodzie automatu jest niski nakład pracy – zadaniem projektanta układu wykorzystującego DFPlayer jest wyłącznie zaprogramowanie go z użyciem UART (*Universal Asynchronous Receiver-Transmitter*). Obsługa kart microSD (do 32 GB) eliminuje także potrzebę integracji dodatkowej pamięci jako części układu automatu celem zachowania ciężkich próbek sygnałów dźwiękowych. W ten sposób

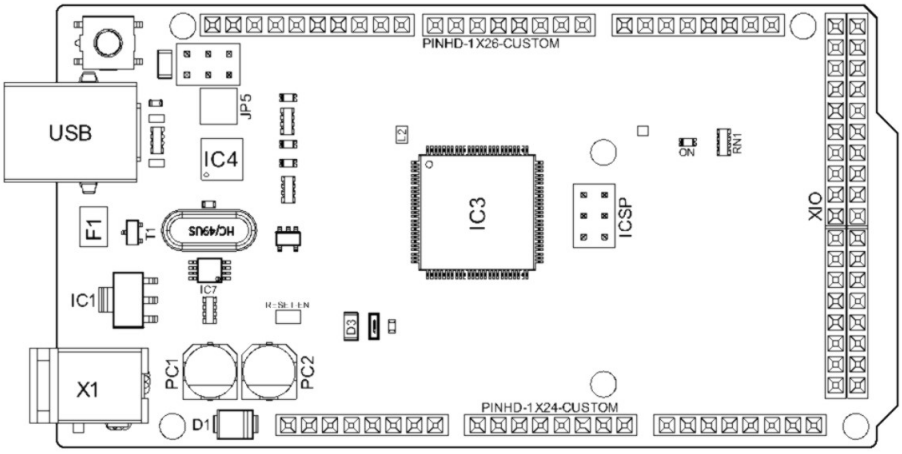
wykorzystuje się tylko dwa wyprowadzenia Arduino – dostęp do zewnętrznej pamięci flash zwiększyłby liczbę tych wyprowadzeń o przynajmniej jeden interfejs.

Motywacją do wykorzystania modułu wyświetlacza matrycowego LCD jest jego popularność i uniwersalność. Jest to najpopularniejszy model, wykorzystujący sprawdzony układ HD44780 już połączony z wyświetlaczem, co projektantowi pozostawia ponownie tylko programowanie z użyciem mikrokontrolera, tym razem za użyciem nieustandaryzowanego interfejsu. Wbudowane podświetlenie usprawnia widoczność nieemitującego światła ciekłego kryształu w gorzej oświetlonym środowisku – przykładowo w salonach gier, z którymi najczęściej kojarzy się takie automaty. Wykorzystany model wyświetla dwa wiersze po szesnaście znaków w rozdzielonych przerwami matrycach 5x7. Najnowocześniejsze flippery korzystają z większych wyświetlaczy matrycowych bez przerw na znaki, co umożliwia wyświetlanie animowanych obrazów.

1.1. Mikrokontroler ATmega2560 i płyta rozwojowa Arduino Mega 2560

Arduino Mega 2560 jest płytą rozwojową z 8-bitowym mikrokontrolerem AVR ATmega2560. W projekcie automatu flipper jest ona preferowana nad bardziej popularną płytą Arduino Uno, gdyż oddaje ona do dyspozycji 54 cyfrowe piny GPIO, to jest ponad cztery razy więcej niż wariant z ATmega328P [17]. Płyta rozwojowa ma na celu ułatwienie projektowania obwodu i programowania mikrokontrolera poprzez połączenie go z obwodem wspierającym jego działanie, złączami ułatwiającymi połączenie pinów GPIO z obwodem testowym na płytkach stykowych oraz z układem programatora. Napięcie zasilania płyt Arduino zależy od modelu. W przypadku Arduino Mega 2560 jest to od 7 do 12 Interfejs zasilacza także różni się między modelami, w wykorzystanej w projekcie płycie jest to gniazdo żeńskie DC o średnicy 5,5 mm [17].

Ułatwienie programowania mikrokontrolera Arduino przejawia się poprzez umieszczenie programatora na płycie. Tradycyjnie mikrokontrolery AVR programuje się przez interfejs SPI, a najpopularniejszym programatorem umożliwiającym przesłanie kodu z komputera przez interfejs USB jest USBasp. Konfiguracja tego połączenia jest jednak trudna, w związku z czym cały układ łączący mikrokontroler z komputerem jest częścią Arduino. Interfejs umożliwiający podłączenie płyty do komputera to żeńskie gniazdo USB typ B, powszechnie kojarzony z drukarkami [17].

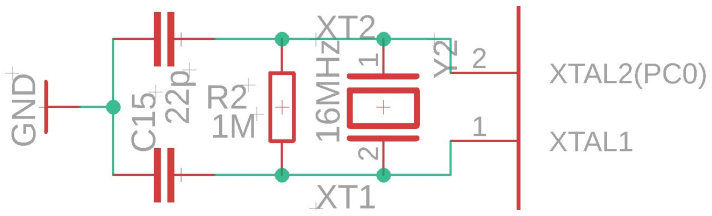


Rys. 1.1. Płyta Arduino Mega 2560

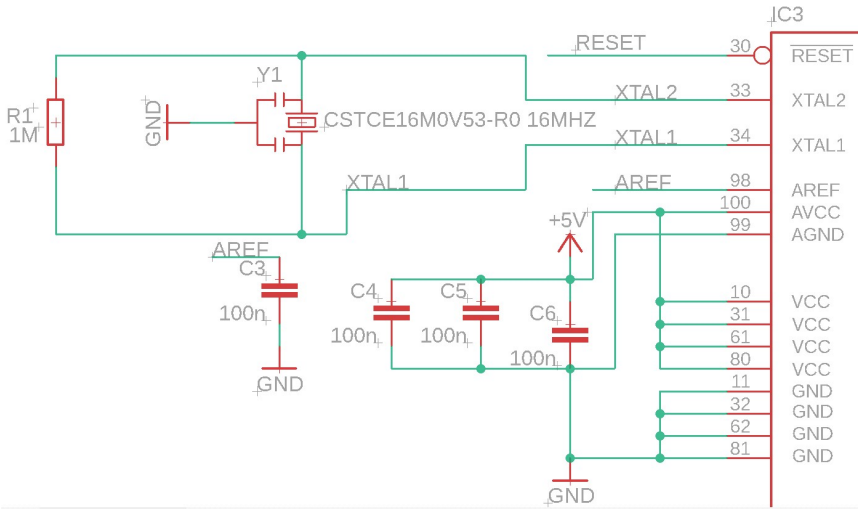
Na rysunku 1.1 przedstawiono podstawowy schemat topologii płyty, dostarczany przez producenta [18]. Do najważniejszych elementów przez niego zaznaczonych należą:

- IC3 – mikrokontroler ATmega2560 w obudowie TQFP100,
- USB – gniazdo żeńskie USB typ B, wykorzystywane do przesłania heksadecymalnego kodu aplikacji z komputera do ww. mikrokontrolera,
- IC4 – mikrokontroler ATmega8U2 pełniący rolę cyfrowego procesora sygnałowego - przetwornika sygnałów USB z komputera na sygnały SPI,
- ICSP – wyprowadzenia męskie wykorzystywane do programowania mikrokontrolera ATmega2560 przez SPI,
- JP5 – wyprowadzenia męskie wykorzystywane do programowania mikrokontrolera ATmega8U2 przez SPI,
- X1 – gniazdo żeńskie zasilające DC 5,5 mm, docelowe napięcie 7-12 V,
- IC1 – stabilizator napięcia MC33269ST-5 lub SPX1117,
- ON – dioda LED - kontrolka zasilania, świeci tak długo jak układ jest zasilany.

Do prawidłowego działania oba mikrokontrolery poza źródłem zasilania 5 V wymagają układów oscylatora, składających się z kondensatorów ceramicznych oraz rezonatorów kwarcowych [19]. Owalna obudowa rezonatora wykorzystywanego przez ATmega2560 jest widoczna na rysunku 1.1., na rysunkach 1.2. i 1.3 przedstawiono konfiguracje tych układów.

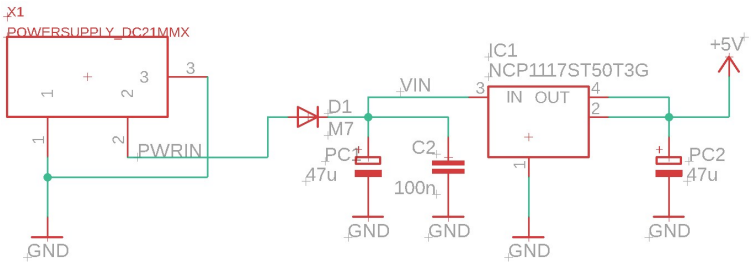


Rys. 1.2. Układ oscylatora dla ATmega8U2



Rys. 1.3. Układ oscylatora dla ATmega2560

Układ zasilający przyjmuje napięcie stałe 7-12 V, do mikrokontrolerów jednak poprowadzić należy napięcie stałe 5 V. Aby je uzyskać, w układzie Arduino zastosowano stabilizator napięcia MC33269 (w niektórych wersjach jest to SPX1117) oraz diodę prostowniczą. Do prawidłowego działania stabilizatora potrzebne są dwa kondensatory o pojemności rzędu dziesiątek mikrofaradów, w tym przypadku są to foliowe kondensatory elektrolityczne 47 µF, oraz kondensator tantalowy 100 nF [19]. Tę konfigurację przedstawiono na rysunku 1.4.



Rys. 1.4. Konfiguracja układu zasilania mikrokontrolerów

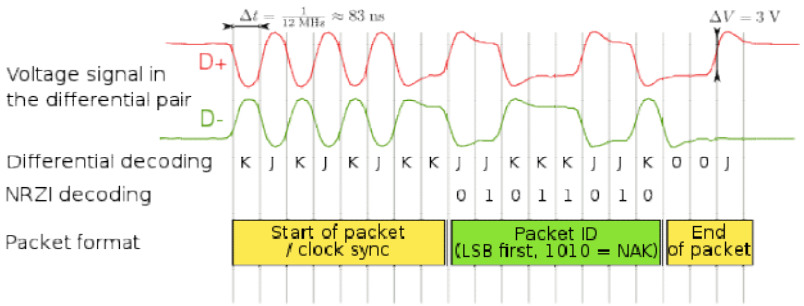
Wykorzystywany w Arduino Mega 2560 mikrokontroler – ATmega2560 – oddaje do dyspozycji osiem 8-bitowych cyfrowych i jeden 6-bitowy cyfrowy rejestr GPIO. Poza tym wyposażony jest w sześć zegarów, komparator analogowy, 10-bitowy przetwornik analogowo-cyfrowy oraz interfejsy komunikacyjne SPI (*Serial Peripheral*

Interface), TWI (*Two-Wire Interface*) i USART (*Universal Synchronous and Asynchronous Receiver-Transmitter*). Z przetwornikiem połączonych jest szesnaście linii wejścia analogowego, które wykorzystać można także jak dwa kolejne 8-bitowe rejestry cyfrowe [20].

Zastosowany mikrokontroler ATmega2560 w montowanej powierzchniowo obudowie TQFP100 ma sto wyprowadzeń [20]:

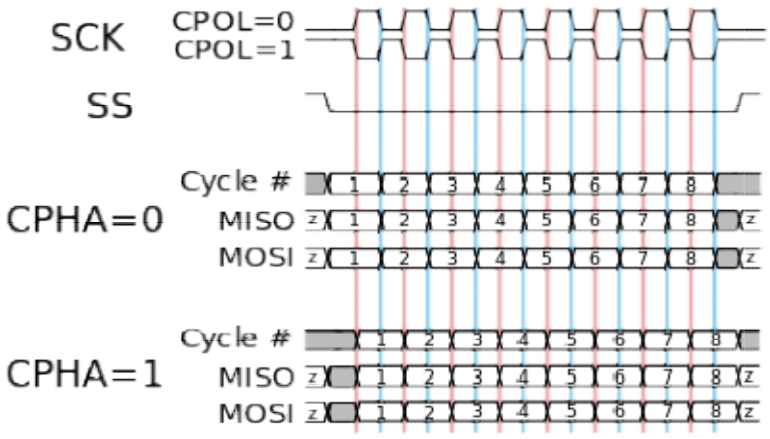
- 4 piny zasilania 5V (VCC),
- 4 piny masy (GND),
- 2 piny łączące mikrokontroler z rezonatorem (XTAL1 i XTAL2),
- 3 piny obsługujące przetwornik analogowo-cyfrowy - AVCC, AGND i AREF,
- pin RESET', będący programowo sterowaną linią interfejsu SPI,
- 16 analogowych pinów, połączonych z przetwornikiem analogowo-cyfrowym oraz cyfrowymi rejestrami portów F i K,
- 70 cyfrowych pinów GPIO, połączone z 8-bitowymi rejestrami portów A, B, C, D, E, G, H, J oraz L. PORTG jest krótszy od reszty, ma 6 bitów a zarazem wyprowadzeń.

Do programowania mikrokontrolerów AVR wykorzystuje się interfejs SPI. W ATmega2560 składają się na niego trzy piny rejestru PORTB - PB1 jako SCK, PB2 jako MOSI oraz PB3 jako MISO. Sterowany programowo pin SS' jest tożsamy z pinem RESET'. Komputery PC nie komunikują się przez SPI, ale przez USB, w związku z czym potrzebny jest procesor sygnałowy „tłumaczący" instrukcje z jednego interfejsu do drugiego. Pierwsza część tego procesu polega na przesłaniu instrukcji komputera poprzez interfejs USB do drugiego mikrokontrolera, ATmega8U2, który ma ku temu dedykowane linie D+ i D-, powiązane z liniami o tych samych nazwach w USB. Schemat komunikacji przedstawiono na rysunku 1.5.



Rys. 1.5. Schemat komunikacji interfejsu USB

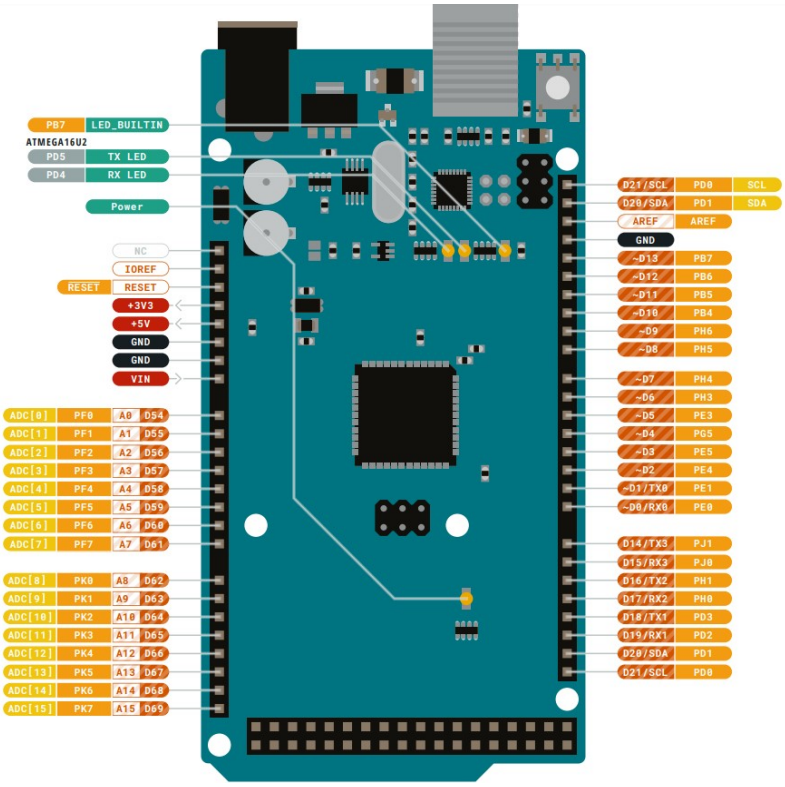
W następnej kolejności mikrokontroler ATmega8U2 dokonuje translacji odebranego sygnału USB na sygnał SPI, który wykorzystać można do programowania ATmega2560. Schemat komunikacji SPI przedstawiono na rysunku 1.6.



Rys. 1.6. Schemat komunikacji interfejsu SPI

Nie wszystkie piny mikrokontrolera są jednak możliwe do wykorzystania w Arduino. Standardowo ATmega 2560 oddaje do użytku 86 pinów GPIO, na płycie jednak dedykowane gniazda żeńskie otrzymało 70, czyli o 16 mniej. Do innych pinów nieposiadających dedykowanych gniazd żeńskich należą [18-20]:

- piny zasilania (VCC),
- piny XTAL1 i XTAL2, wykorzystywane do łączenia z układem rezonatora,
- dwa z trzech pinów obsługujących przetwornik analogowo-cyfrowy - AVCC, AGND.



Rys. 1.7. Mapa wyprowadzeń Arduino Mega 2560

Na płycie rozwojowej Arduino Mega 2560 znajdują się następujące wyprowadzenia żeńskie:

- NC - nie jest wykorzystywane,
- RESET (w odróżnieniu od „czystego” mikrokontrolera ma stan aktywny 1),
- IOREF - napięcie referencyjne logiki pinów cyfrowych, połączone z pinem 5V,
- GND (5 sztuk, z czego 2 niezaznaczone na rysunku 1.7) - masa,
- VIN – pin połączony bezpośrednio z gniazdem zasilającym, można je połączyć z innymi urządzeniami lub wykorzystać je do zasilania zamiast gniazda 5,5 mm,
- 3,3V - zapewnia logikę wyjścia 3,3 V,
- 5V (3 sztuki, z czego 2 niezaznaczone na rysunku 1.7) - zapewnia logikę wyjścia 5 V,
- D0-D21 oraz niezaznaczone na rysunku 1.8 D22-D53 - cyfrowe piny GPIO,
- A0-A15 - analogowe piny, mogą także pełnić rolę pinów cyfrowych (jako D54-D69).

Niektóre z nich mają dodatkowe funkcje [8]:

- wszystkie piny analogowe mogą korzystać z 10-bitowego przetwornika analogowo-cyfrowego,
- 14 z 54 cyfrowych pinów GPIO wspiera PWM (*Pulse Width Modulation*; zaznaczone jako ~D0-~D13),

- para D20/D21 może obsługiwać interfejs TWI, ku temu zamontowano ich dwie pary. Nie można ich wykorzystać do innych celów.
- pary D0/D1, D18/D19, D16/D17 i D14/D15 mogą obsługiwać interfejs UART.

Do programowania mikrokontrolera na oryginalnych płytach rozwojowych Arduino, ich klonów sprzętowych i niektórych innych (np. sieciowy ESP8266) zamiast asemblera albo dialektu języka C dla AVR-ów wykorzystuje się inny dialekt, używany wewnątrz przyjaznego użytkownikowi środowiska programistycznego Arduino IDE. Do głównych różnic pomiędzy standardowym C dla mikrokontrolerów AVR a dialektem wykorzystywanym w Arduino IDE zaliczyć można niestandardową strukturę kodu. Na znaną z ANSI C funkcję *main* składają się tam dwie funkcje - *setup*, wykonująca się raz przy resecie płytki (stan aktywny RESET'), oraz *loop*, wykonująca się w nieskończonej pętli. Poza tym wszelkie funkcje nie odnoszą się do rejestrów, a do pojedynczych pinów. Taki wzrost poziomu abstrakcji sprawia, że są one przyjaźniejsze użytkownikowi (nie wymagają przykładowo układania działań binarnych na całych rejestrach). Wadą tego podejścia jest mniejsza swoboda w przypadku bardziej zaawansowanych działań.

Poniżej przedstawiono przykład kodu w tym dialekcie:

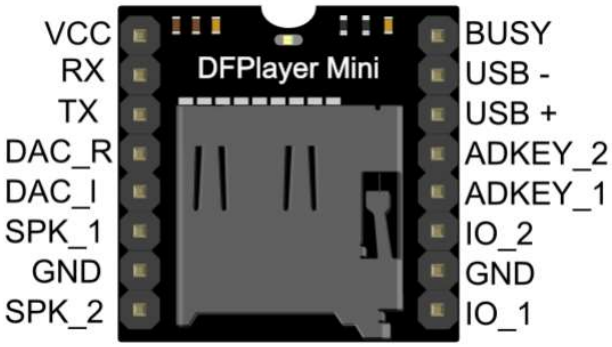
```
void setup() {
  pinMode(22, OUTPUT); // ustawia pin numer 22 jako wyjście
}

void loop() {
  digitalWrite(22, HIGH); // ustawia stan wysoki na pinie 22
  delay(1000); // czeka 1000 ms = 1 s
  digitalWrite(22, LOW); // ustawia stan niski na pinie 22
  delay(1000); // czeka 1000 ms = 1 s
}
```

Ten kod co sekundę przełącza pin D22 pomiędzy wysokim a niskim stanem. Stan wysoki oznacza ustawienie napięcia pinu IOREF, w praktyce jest to napięcie 5 V, natomiast niski – 0 V. Włączenie pomiędzy D22 i GND diody LED z odpowiednim rezystorem sprawi, że dioda będzie migać.

1.2. Układ DSP DFPlayer

DFPlayer Mini jest modułem do odtwarzania plików dźwiękowych w formacie .mp3, zapisanych na karcie microSD o maksymalnej pojemności 32 GB. Producentem oryginalnego układu jest DFRobot. Do działania wykorzystuje gniazdo na tę kartę oraz układ YX5200-24SS, będący procesorem DSP przystosowanym do konwersji plików

.mp3 systemu FAT16 i FAT32 na sygnały audio, które przed wysłaniem na zewnętrzny głośnik wzmacniane są przez układ 8002A [7].



Rys. 1.8. Wyprowadzenia układu DFPlayer Mini

Przedstawiony na rysunku 1.8 moduł DFPlayer zawiera szesnaście wyprowadzeń, wykorzystywanych do komunikacji i odtwarzania dźwięku [21]:

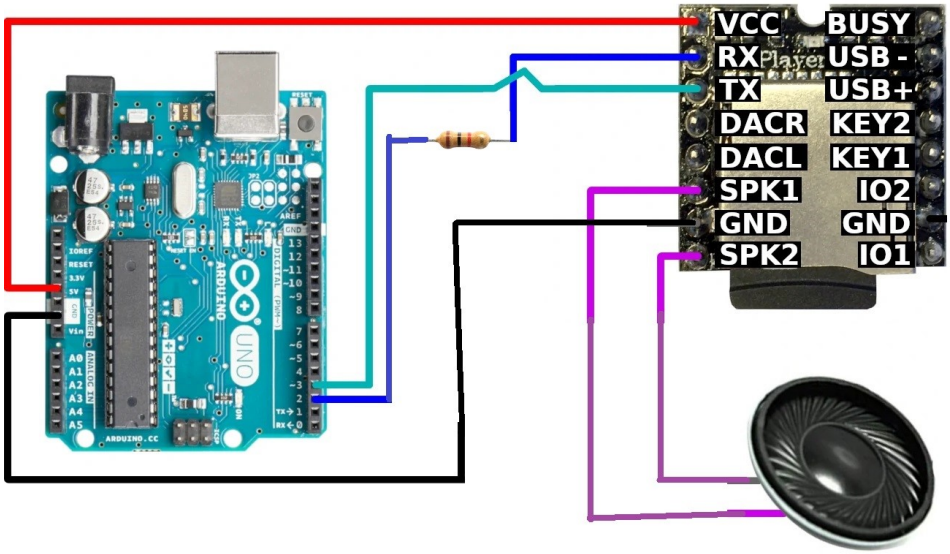
- Vcc - zasilanie 5 V,
- GND (dwa razy) - masa,
- RX/TX - piny obsługiwane przez interfejs UART,
- USB-/USB+ - piny D- i D+ interfejsu USB,
- ADKEY_1/ADKEY_2 - piny obsługiwane przez specjalizowaną klawiaturę producenta, wykorzystywaną w trybie komunikacji AD KEY,
- IO1/IO2 - sterowanie głośnością i numerem utworu w trybie komunikacji I/O,
- DAC_L/DAC_R - wyjście dla wzmacniaczy i głośników o mocy powyżej 3 W,
- SPK_1/SPK_2 - wyjście dla głośników do 3 W,
- BUSY - wyjście informujące czy dźwięk jest odtwarzany, o stanie aktywnym 0.

Układ obsługuje trzy tryby pracy [21]:

- Tryb I/O (*input/output*; wejścia/wyjścia) jest najprostszy i do sterowania wykorzystuje piny IO_1, którego krótkie zwarcie z zasilaniem zaczyna odtwarzanie następnego utworu, a długie - zmniejszenie mocy sygnału, oraz IO_2, którego działanie jest odwrotne - odpowiada za zwiększenie mocy sygnału lub wybór poprzedniego utworu. Moc sygnału przekłada się na głośność dźwięku.
- Tryb AD KEY (*Analog-Digital KEYpad*) wykorzystuje specjalizowaną klawiaturę producenta, zawierającą przyciski z rezystorami o różnych wartościach rezystancji. W zależności od wybranego przycisku utworzony zostaje odpowiedni dzielnik napięcia, którego napięcie wyjściowe jest zamieniane na odpowiednią komendę przez przetwornik analogowo-cyfrowy.

- Tryb Serial (szeregowy) jest przystosowany do działania z układem mikroprocesorowym, wykorzystując interfejs UART lub USB.

Komunikacja z Arduino odbywa się w trybie Serial, za pomocą UART. Aby ją zrealizować, należy podłączyć dwa piny Arduino (na przykład D10 jako RX i D11 jako TX) do odtwarzacza. Na linii RX powinien się znaleźć rezystor 10 kΩ, chroniący przed uszkodzeniem obu układów w razie zwarcia. Do działania należy także podłączyć zasilanie 5V (do pinów Vcc i GND) oraz zewnętrzny głośnik 3 W pomiędzy piny SPK_1 i SPK_2. Pozostałe piny nie są używane w tej konfiguracji. Przedstawiono ją poniżej, na rysunku 1.9.



Rys. 1.9. Wyprowadzenia układu DFPlayer Mini

Aby zainicjować interfejs UART w Arduino IDE, należy załączyć wbudowaną bibliotekę SoftwareSerial oraz utworzyć strukturę reprezentującą interfejs, podając kolejno numery pinów RX i TX:

```
#include "SoftwareSerial.h"

SoftwareSerial Ss(10, 11);
```

Komunikacja z DFPlayer odbywa się poprzez wysyłanie 10-bajtowych instrukcji, będących sekwencją: *$S, VER, Len, CMD, Feedback, para1, para2, checksum, $O*, gdzie [21]:

- *$S* – bajt startowy, zawsze równy 0x7E,
- *VER* – wersja DFPlayera, zawsze równa 0xFF,
- *Len* – długość komendy, liczba bajtów po Len - w praktyce zawsze równa 0x06,

- *CMD* – komenda do wykonania przez system,
- *Feedback* – określa, czy ma zostać odesłane potwierdzenie, 0x00 albo 0x01,
- *para1* – 8 starszych bitów parametru komendy,
- *para2* – 8 młodszych bitów parametru komendy,
- *checksum* – 16-bitowa suma kontrolna, będąca sumą wszystkich pozostałych parametrów za wyjątkiem *$S* i *$O*, ze znakiem minus,
- *$O* – bajt końcowy, zawsze równy 0xEF.

Aby wydać odtwarzaczowi polecenie, należy utworzyć 10-bajtowy bufor. Suma kontrolna jest 16-bitowym słowem (*word*) i wymaga rozbicia na starszy i młodszy bajt.

```
byte komenda[10] = { 0x7E, 0xFF, 0x06, CMD, 0x00, para1, para2,
highByte(checksum), lowByte(checksum), 0xEF};
```

oraz wysłać je poprzez utworzoną wcześniej strukturę interfejsu:

```
for (byte i=0; i<10; i++)
 Ss.write(komenda[i]);
```

Powyższe polecenia połączyć można w jedną funkcję, której parametrami są polecenie i parametry tego polecenia:

```
void wykonaj(byte CMD, byte para1, byte para2) {
 word checksum = -(VER + Len + CMD + Feedback + para1 + para2);
 byte komenda[10] = { 0x7E, 0xFF, 0x06, CMD, 0x00, para1, para2,
 highByte(checksum), lowByte(checksum), 0xEF};
      for (byte i=0; i<10; i++)
       Ss.write(komenda[i]);
}
```

W tabeli 1.1 przedstawiono przykładowe polecenia DFPlayera. Nie są to wszystkie dostępne opcje, a jedynie te, które znajdują zastosowanie w projekcie flippera. Przykładowo aby odtworzyć utwór 0001.mp3, należy zapisać i uruchomić poniższy kod:

```
Ss.begin(9600); // rozpocznij działanie interfejsu UART z
prędkością 9600 Bd
delay(1000); // czekaj 1 s
wykonaj(0x3F, 0, 0); // uruchom moduł
delay(500); // czekaj 0,5 s
wykonaj(0x06,0,30); // nastaw głośność na wartość maksymalną
delay(2500); // czekaj 2,5 s
wykonaj(0x03,0,1); // odtwórz utwór 0001.mp3
delay(500); // czekaj 0,5 s
```

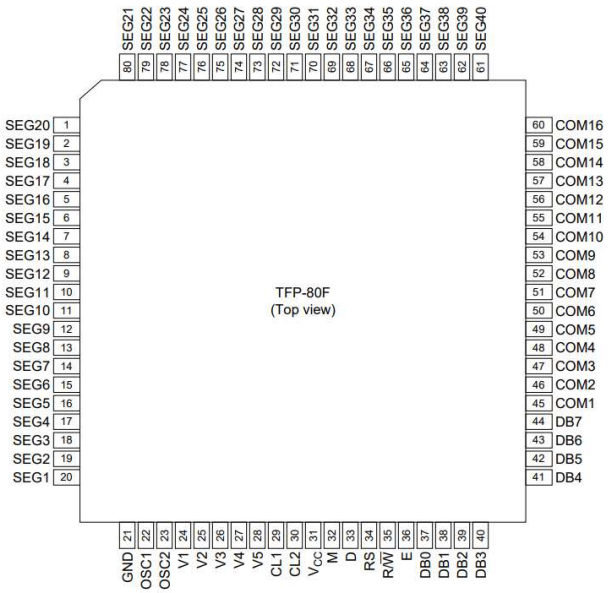
Tabela 1.1 Wybrane polecenia DFPlayera

polecenie	wartość *CMD*	wartość *par1*	wartość *par2*
uruchomienie modułu	0x3F	0x00	numer układu Slave, dla jednego DFPlayer jest to zawsze 0x00
odtworzenie utworu raz	0x03	starsze 8 bitów numeru utworu od 1 do 2999	młodsze 8 bitów numeru utworu od 1 do 2999
odtworzenie utworu w pętli	0x08	starsze 8 bitów numeru utworu od 1 do 2999	młodsze 8 bitów numeru utworu od 1 do 2999
pauza	0x0E	0x00	0x00
zmiana poziomu głośności	0x06	0x00	głośność od 0x00 do 0x1D

Aby odtwarzać pliki .mp3, podobnie jak karta microSD muszą być one zgodne z systemem partycjonowania FAT16 lub FAT32 oraz przyjmować odpowiednie nazewnictwo - XXXX - Y.mp3, gdzie XXXX jest liczbą od 0001 do 2999, a Y to dowolny tekst. Utwory domyślnie wyszukiwane są w katalogu głównym karty pamięci, chociaż istnieje możliwość organizacji ich w katalogach o nazwach od 1 do 10 oraz wybierania ich za pomocą innych poleceń [21].

1.3. Wyświetlacz LCD ze sterownikiem HD44780

Oryginalnie zaprojektowany przez Hitachi sterownik HD44780 jest układem ASIC służącym do sterowania monochromatycznym wyświetlaczem LCD, zawierającym 16 matryc po 40 komórek ciekłego kryształu, ułożonych w prostokąty 5x8. Sterownik rozszerzenia zwiększa liczbę tych matryc do 32 [22]. Wyprowadzenia tego układu przedstawiono na rysunku 1.10.



Rys. 1.10. Wyprowadzenia układu HD44780

Niezależnie od obudowy układ zawiera 80 wyprowadzeń, z czego 69 dotyczy zasilania układu i prezentacji danych na wyświetlaczu:

- SEG1-SEG40 - sygnały wyjściowe segmentów,
- COM1-COM16 - sygnały wyjściowe wspólne,
- CL1, CL2, M, D - sygnały wyjściowe do sterownika rozszerzenia, podwaja liczbę dostępnych pól,
- Vcc - zasilanie układu 2,7-5,5 V,
- GND - masa,
- V1-V5 - zasilanie wyświetlacza LCD, maksimum 11 V,
- OSC1, OSC2 - połączenie z zewnętrznym zegarem lub rezonatorem.

Pozostałe 11 pinów natomiast dotyczy komunikacji z mikrokontrolerem:

- RS - wybór rejestru danych DR (na 1) lub rejestru instrukcji IR (na 0),
- R/W’ - wybór odczytu (na 1) lub zapisu (na 0),
- E - wraz z wykryciem zbocza opadającego wykonuje podaną instrukcję,

- DB0-DB7 - w trybie odczytu na te piny wyprowadza bity do odczytania, w trybie zapisu natomiast odczytuje bity zadane przez mikrokontroler.

Sterownik korzysta z następujących rejestrów:

- IR - 8-bitowy rejestr instrukcji - zawiera kody instrukcji i informacje o adresie w pamięci,
- DR - 8-bitowy rejestr danych - zawiera dane do zapisania do pamięci lub do odczytu z pamięci,
- Busy Flag - ustawiana automatycznie w czasie wykonywania instrukcji i czyszczona w momencie ich zakończenia, wtedy inne instrukcje nie są wykonywane,
- AC - 7-bitowy licznik adresu - przydziela adres do pamięci.

oraz z trzech pamięci:

- DDRAM (*Display Data Random Access Memory*) - pamięć danych wyświetlacza - zawiera kody znaków do wyświetlania. Na 80 B może pomieścić maksymalnie 80 znaków.
- CGROM (*Character Generator Read-Only Memory*) - zawiera wzory predefiniowanych znaków,
- CGRAM (*Character Generator Random Access Memory*) - zawiera wzory maksymalnie ośmiu niestandardowych znaków.

Aby wyświetlacz wykonał instrukcję w momencie nadania zbocza opadającego na E, należy wybrać, czy dane mają być zapisywane czy odczytywane (R/W'), oraz czy podane mają zostać dane, czy instrukcja (RS). W tabeli 1.2 przedstawiono wybraną operację dla wszystkich czterech kombinacji tych bitów. Wybrana instrukcja zapisu (dla wyczyszczonych RS i RW') natomiast zależy od liczby kolejnych zer na pinach DB7-DB1, do pierwszej jedynki. Bity znajdujące się po tej jedynce mogą być parametrami instrukcji. W tabeli 1.3 przedstawiono instrukcje zapisu, które można przesłać do wyświetlacza [22].

Tabela 1.2 Konfiguracje bitów RS i R/W'

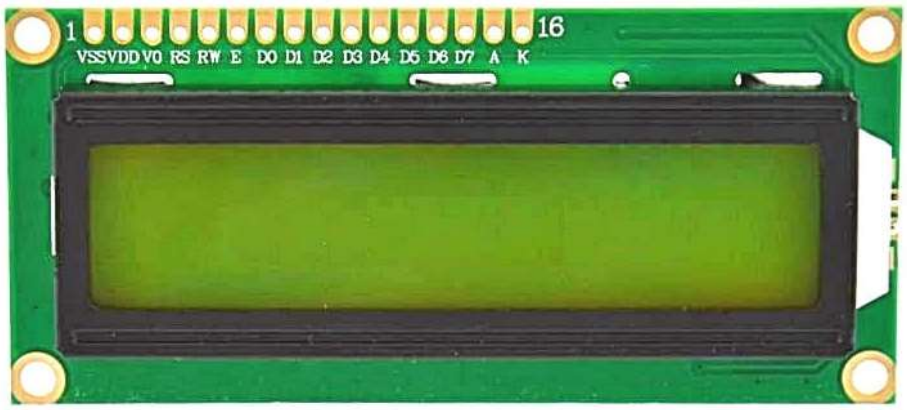
RS	R/W'	operacja
0	0	zapis i wykonanie instrukcji
0	1	odczyt Busy Flag na DB7 oraz AC na DB0-DB6
1	0	zapis danych do DDRAM (wyświetlanie) lub CGRAM (tworzenie własnych znaków)
1	1	odczyt danych z DDRAM lub CGRAM

Tabela 1.3 Lista poleceń HD44780

liczba kolejnych zer przed jedynką	parametry	operacja
7	-	czyszczenie całego ekranu i powrót na pozycję 0 w DDRAM
6	-	powrót na pozycję 0 w DDRAM i cofnięcie przesunięcia wyświetlacza
5	DB0 - jeżeli „1”, wyświetlacz też się przesuwa DB1 – „0” oznacza dekrementację, „1” inkrementację	przesuwanie kursora i wskaźnika DDRAM
4	DB0 - przełącza miganie kursora DB1 - przełącza widoczność kursora DB2 - przełącza wyświetlacz	przełączanie kursora i wyświetlacza
3	DB2 – „0” przesuwa w lewo, „1” w prawo DB3 - jeżeli „1”, przesunięciu ulega wyświetlacz, jeżeli „0”, kursor	przesuwanie wyświetlacza lub kursora
2	DB2 – „1” ustawia wysokość znaku na 10 punktów, „0” na 8 punktów DB3 – „1” ustawia liczbę linijek na 2, „0” na 1 linijkę DB4 – „1” ustawia długość interfejsu na 8 bitów, „0” na 4 bity	zmiana sprzętowych ustawień, należy dostosować do wyświetlacza i źródła instrukcji
1	DB0-DB5 - adres CGRAM	wybór adresu CGRAM
0	DB0-DB6 - adres DDRAM	wybór adresu DDRAM

Aby zapisać znak na danej pozycji, przed nadaniem sygnału (zbocze opadające na E) należy wpisać kod znaku na liniach DB0-DB7 lub DB4-DB7 (zależnie od długości interfejsu), ustawić linię RS oraz wyczyścić R/W’. Należy pamiętać, że po zapisaniu znaku AC się inkrementuje, co oznacza, że następny zapis odbędzie się na pozycji o jeden większej. Jedna linia zawiera 40 znaków, z czego tylko 16 jest widocznych bez przesunięcia wyświetlacza. Oznacza to, że adres DDRAM pierwszej pozycji drugiej linijki to nie 16, ale 40 [22]. Instrukcje odczytu DDRAM, AC i Busy Flag są rzadko używane, w tym projekcie nie mają żadnego zastosowania.

Na najpopularniejszy komercyjny moduł wyświetlacza, przedstawiony na rysunku 1.11, składa się sterownik HD44780 połączony z wyświetlaczem LCD z podświetleniem, a także układ umożliwiający podłączenie go do źródła zasilania i instrukcji, oraz regulację jasności.

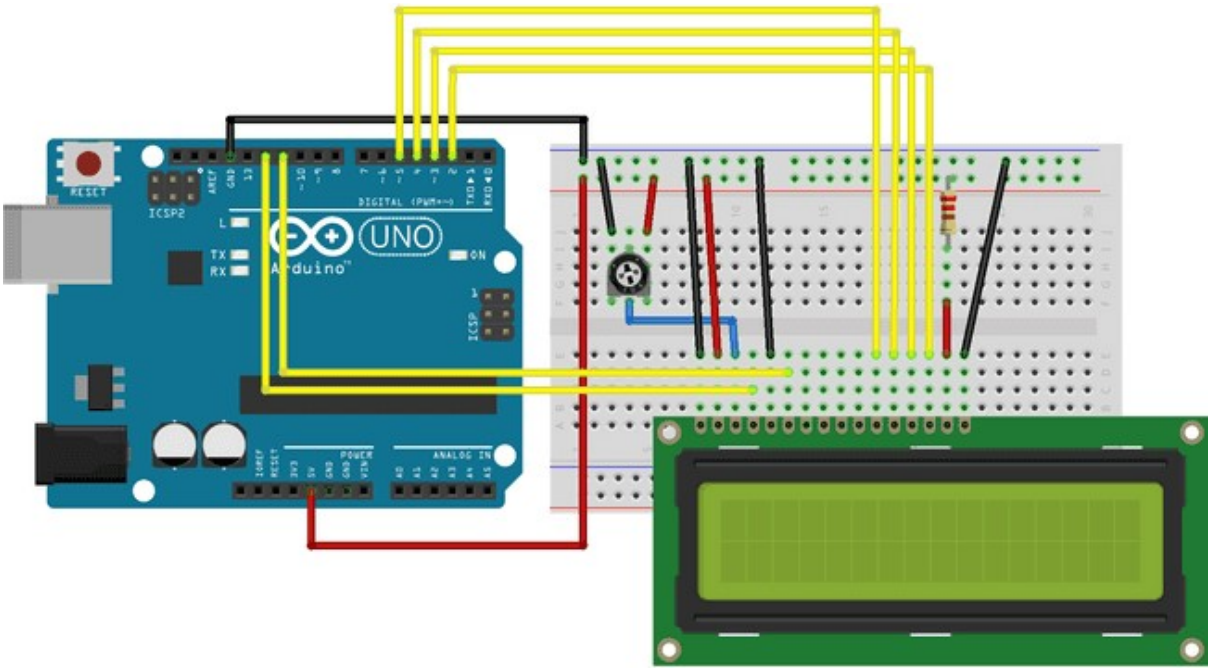


Rys. 1.11. Moduł wyświetlacza sterowanego przez HD44780

Moduł ma 16 kolejnych pinów o następujących zastosowaniach [23]:

- VSS (czasami jako GND) - masa,
- VDD - zasilanie 5 V,
- V0 - pin podłączany do wyjścia dzielnika napięcia, wykorzystywany do regulacji jasności,
- RS - wybór rejestru, bezpośrednio połączony z RS w HD44780,
- RW - wybór pomiędzy odczytem i zapisem, bezpośrednio połączony z R/W' w HD44780,
- E - zbocze opadające na tej linii wykonuje instrukcję, bezpośrednio połączony z E w HD44780,
- D0-D7 - linie danych, bezpośrednio połączone z DB0-DB7 w HD44780,
- A (czasami jako BLA) - anoda podświetlającej ekran diody LED,
- K (czasami jako BLK) - katoda tej diody.

Aby sterować takim modułem za pomocą dowolnego modułu Arduino, należy połączyć piny D0-D7 lub D4-D7 (zależnie od długości interfejsu), RS oraz E z pinami cyfrowymi płytki, VDD i A (BLA) do zasilania (5V), VSS (GND) i K (BLK) do masy (GND). Regulacja jasności dodatkowo wymaga utworzenia dzielnika napięcia, którego wyjście zostanie wyprowadzone na V0 - do jego realizacji i eksperymentalnego wyznaczenia stosuje się potencjometr. Opcjonalnie można sterować pinem RW, w projekcie jednak wykorzystywany jest wyłącznie tryb zapisu, a zatem RW jest zwarty do masy. Połączenie podświetlającej diody LED wymaga dodatkowo rezystora 220 Ω na linii A (BLA) lub K (BLK) [11]. Schemat tych połączeń dla 4-bitowego interfejsu przedstawiono na rysunku 1.12.



Rys. 1.12. Połączenie Arduino z wyświetlaczem LCD.

Do sterowania wyświetlaczem za pomocą Arduino wykorzystać można wbudowaną bibliotekę LiquidCrystal, która dostarcza funkcji na wysokim poziomie abstrakcji. Każda z tych funkcji pozyskuje zrozumiałe dla człowieka parametry (np. ciąg wejściowy do wyświetlenia) i tłumaczy je na instrukcje dla sterownika. W związku z tym nie trzeba pisać kodu odwołującego się do danych pinów, poza zainicjowaniem struktury powiązanej z tymi pinami:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(5, 4, 3, 2, 1, 0);
```

Konstruktor struktury LiquidCrystal przyjmuje kolejno numery pinów Arduino połączone z: RS, E, D4, D5, D6, D7 [9]. Następnym krokiem wymaganym do uruchomienia wyświetlacza jest wywołanie funkcji początkowej w funkcji *setup*, jako parametry podając liczbę znaków w jednej linijce oraz ilość linijek, w przypadku tego modułu 16x2:

```
lcd.begin(16, 2); // wyczyść podłączony moduł wyświetlacza 16x2
```

Następnie celem zapisania ciągu znaków należy podać pozycję, od której ma zacząć wypisywać (setCursor), oraz nadać instrukcję napisania tego ciągu print, podając ten ciąg:

```
lcd.setCursor(1, 0); // zacznij pisać od drugiej pozycji górnej
linii

lcd.print("BALL"); // zapisz tam wyraz "BALL"
```

Instrukcje następują bezpośrednio po sobie i wykonują się prawidłowo. Większość instrukcji trwa zaledwie 37 μs, w związku z czym, a zatem nie jest potrzebne ręczne sprawdzanie wartości Busy Flag [10].

2. OMÓWIENIE ELEMENTÓW I UKŁADÓW MAJĄCYCH ZASTOSOWANIE W GRZE

Gra na automacie flipper polega na utrzymywaniu w grze kulki poprzez nadawanie jej ruchu. W rzeczywistym stole przez cały czas stacza się ona z planszy nachylonej pod kątem 6,5° [5], a kiedy ją opuści, runda się kończy. Aby temu zapobiec, odrzuca się ją w górę planszy z użyciem dwóch łapek (ang. *flippers*). Ich mechanizm opiera się na elektronicznym siłowniku z dwóch powodów – trzycentymetrowa, stalowa kulka jest ciężka i ręczny mechanizm może wymagać od gracza przyłożenia dużej siły, poza tym w nowszych maszynach z przytrzymaniem łapki wiąże się działanie programu. Kiedy przycisk z boku automatu jest przytrzymany, solenoid przewodzi i przytrzymuje łapkę, wówczas jest ona zwrócona „do góry". Łapka przy nieprzewodzącym solenoidzie jest opuszczona w dół. Kształt łapki i kąt jej nachylenia sprawiają, że kulka w kontakcie z opuszczoną łapką dalej stacza się w dół planszy, z przewodzącą łapką natomiast zatrzymuje się w miejscu. Przyłożenie napięcia do solenoidu, podczas gdy kulka jest na opuszczonej łapce sprawia, że zostaje ona „wystrzelona" w górę pochylonej planszy.

Liczba punktów zdobytych w grze nie wynika jednak bezpośrednio z czasu trwania rundy, a z trafiania różnych urządzeń, które mają zostać w pewien sposób „aktywowane" przez dotknięcie kulką. Najbardziej znane przykłady także opierają swoje działanie na solenoidach. Obejmują one przede wszystkim elementy odpychające kulkę, czyli tzw. „grzybki" (ang. *bumper*) w kształcie grzybów, „proce" (ang. *slingshot*) w kształcie trójkątów rozwartokątnych, czy wreszcie zwykłych siłowników. Te ostatnie mają zastosowanie przy podawaniu kulki między rundami w miejsce, z którego „wystrzeliwuje" się kulkę), zwykle ulokowane po prawej stronie stołu. Urządzenie to nazywa się „korytem na kulki" (ang. *ball trough*). Wyrzutnie dla gracza także mogą być obsługiwane przez solenoid, w większości przypadków jest to jednak ręczna wyrzutnia ze sprężyną (ang. *loaded spring*). Nie wszystkie punktowane elementy nadają kulce ruch – poza nimi istnieją „tarcze opuszczane" (ang. *drop target*), które po trafieniu „chowają się" pod planszą, także z użyciem solenoidu, do czasu odebrania kolejnego sygnału. Tarcze niewykorzystujące elektromagnesów nazywa się „tarczami punktowymi" (ang. *spot target*) i zwykle mają one inny kształt od „tarcz opuszczanych". Poza tym istnieją dziesiątki innych elementów dających punkty, wymyślanych w zależności od tematyki automatu. Najprostszym w działaniu elementem, który można aktywować przez obecność kulki jest przycisk, który zwierany

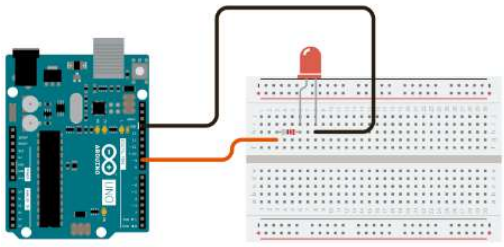
jest przez przetoczenie się po nim kulki – stąd nazwa „przetoka” (ang. *rollover*). Najczęściej jest to przycisk, do którego przytwierdzono od góry kawałek odpowiednio zgiętego drutu, po którym kulka może się przetoczyć bez straty prędkości [4].

Aby poinformować gracza o jego postępach w grze, wykorzystuje się elementy optoelektroniczne. Wspomniany wcześniej wyświetlacz ze sterownikiem przedstawia przynajmniej punkty i numer rundy. Tradycyjnie na planszy umieszcza się także wskaźniki związane z innymi zasadami danego automatu, na przykład możliwości zwiększenia liczby rund czy wielokrotności otrzymywanych punktów. Wskaźniki te realizuje się z użyciem diod elektroluminescencyjnych (LED) umieszczonych pod planszą tak, aby były widoczne, a zarazem nie blokowały kulki ani innych elementów. Aby zapewnić ich widoczność, umieszcza się je pod transparentnymi kształtami z plastiku [5].

2.1. Elementy optoelektroniczne – wyświetlacz LCD i diody LED

W projekcie wykorzystano dwa rodzaje elementów optoelektronicznych - emitujące światło diody LED oraz wyświetlacz ciekłokrystaliczny (LCD), który jedynie pełni rolę modulatora światła (sam z siebie nie emituje światła). Wykorzystany model zawiera jednak podświetlającą go diodę LED.

Diody elektroluminescencyjne (LED) są szczególnym przypadkiem diody półprzewodnikowej, w której zachodzi zjawisko elektroluminescencji, czyli emisji promieniowania pod wpływem przepływu prądu elektrycznego. Warunkiem zajścia tego zdarzenia jest rekombinacja promienista, czyli emisja fotonów. W czystym krzemie zjawisko to nie występuje. Szerokość promieniowania, a zarazem barwa świecenia zależy od szerokości przerwy energetycznej w danym materiale półprzewodnikowym [14]. Aby sterować diodą LED za pomocą Arduino, należy włączyć obwód z diodą i odpowiednio dobranym rezystorem pomiędzy wybrany pin i masę (GND), tak jak na rysunku 2.1.



Rys. 2.1. Połączenie Arduino z diodą LED

Pisząc program sterujący diodą należy ustawić połączony z diodą pin Arduino jako wyjściowy:

```
pinMode(22, OUTPUT);
```

Zapalić diodę można, zadając stan wysoki (5 V) za pomocą funkcji *digitalWrite* z parametrem *HIGH*, podając także numer tego pinu:

```
digitalWrite(22, HIGH);
```

Gaszenie diody odbywa się poprzez zadanie stanu niskiego (0 V), czyli *LOW*:

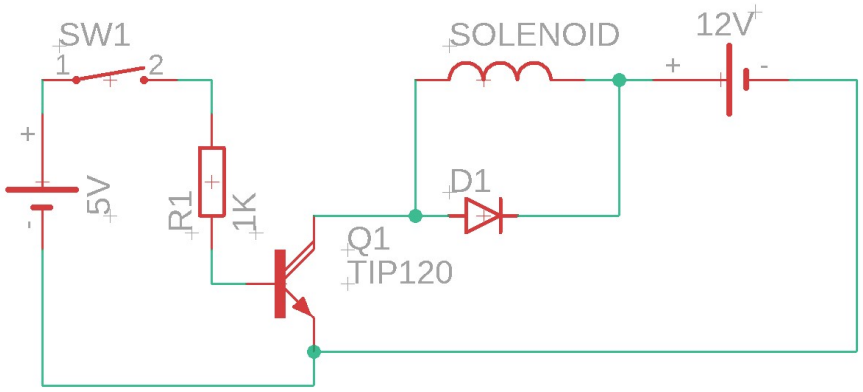
```
digitalWrite(22, LOW);
```

2.2. Elektromagnesy oraz ich sprzętowo-programowe sterowniki

Aby wprawiać w ruch elementy automatów do gier z wykorzystaniem układów elektronicznych, stosuje się elektromagnesy. Składają się na nie tzw. solenoidy, czyli cewki o wielu ciasno zwiniętych zwojach cienkiego drutu, oraz ferromagnetyczne rdzenie. Do swojego działania wykorzystują zjawisko pola elektromagnetycznego, będące sprzężeniem pola elektrycznego i magnetycznego.

W projekcie wykorzystano pięć elektromagnesów przyciągających wbudowany bolec na sprężynie o sile 4,5 kG, dwa trzymające o sile 20 kG oraz jeden pchający bolec o sile 2 kG (1 kG w przybliżeniu odpowiada 10 N, czyli sile grawitacji działającej na ciało o masie 1 kg na Ziemi). Wielkości odpowiadają masie przedmiotów ferromagnetycznych, na które mogą zadziałać te elektromagnesy. Siła przyciągania jest wprost proporcjonalna do natężenia pola elektrycznego, w związku z czym powinno ono być jak najwyższe. Aby osiągnąć pożądany efekt, w obwodzie nie powinno być żadnych elementów rezystywnych poza samą cewką. Wymienione wartości siły przyciągania osiągane są dla napięcia 12 V oraz rezystancji samego solenoidu.

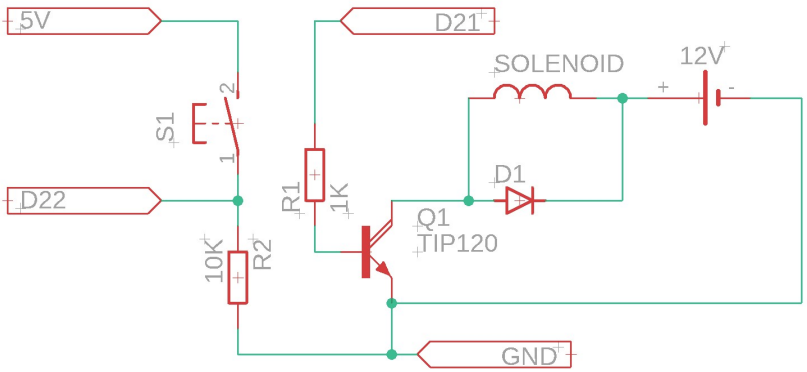
Tak wysokie wartości natężenia prądu elektrycznego nie mogą zostać zastosowane w bezpośrednim połączeniu z Arduino, ponieważ już dla 30 mA układ może się przegrzać i nieodwracalnie uszkodzić. W związku z tym układ sterowania za pomocą mikrokontrolera musi być oddzielony od obwodu zasilania, na przykład z użyciem przekaźników [6]. W tym projekcie zaproponowano prostszy układ, zaprezentowany na rysunku 2.2.



Rys. 2.2. Sprzętowy sterownik elektromagnesu

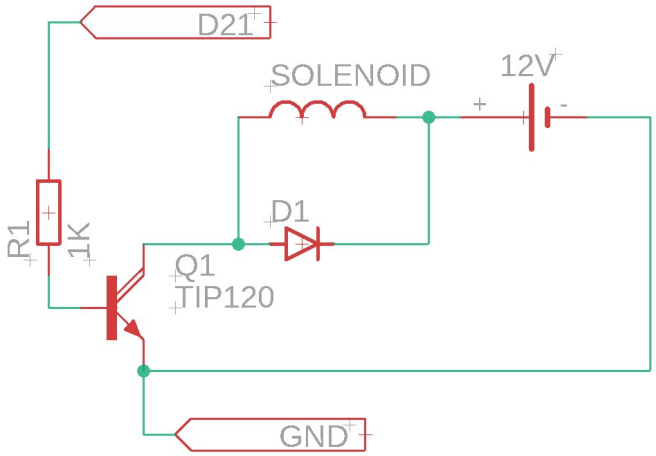
W celu zwiększenia wydajności prądowej wyjść modułu Arduino zastosowano tranzystor TIP120, pracujący w układzie Darlingtona [3]. W tym zastosowaniu występuje on jako przełącznik, przechodząc w zakres nasycenia po zwarciu klucza i zadaniu napięcia 5 V na bazę, w zakres odcięcia natomiast po rozwarciu klucza. Rezystor 1 kΩ ogranicza prąd bazy, który może wynieść najwyżej I_B = 120 mA [24], a także zapobiega przegrzaniu Arduino, nie wpływa jednak na prąd zadany na solenoid. Przez układ sterowania płynie wówczas prąd rzędu pojedynczych miliamperów, przez układ elektromagnesu natomiast prąd 1,5 lub 2 A, zależnie od cewki. Równolegle do elektromagnesu podłącza się także krzemową diodę prostowniczą w kierunku zaporowym względem źródła napięcia, celem wygładzenia ewentualnych przepięć. Przykładem niepożądanego zachowania wynikającego z takiego przepięcia jest całkowite wyłączenie niezwiązanego z tymi układami wyświetlacza LCD.

Zamieniając źródło napięciowe 5 V i szeregowy przełącznik na Arduino z przyciskiem sprzętowo-programowo uodpornionym na drżenie styków otrzymuje się układ zaprezentowany na rysunku 2.3.



Rys. 2.3. Sprzętowo-programowy sterownik elektromagnesu z przyciskiem

W tym obwodzie D21, D22, 5V i GND to wyprowadzenia Arduino. Zwierając przycisk S1 pin cyfrowy D22 przyjmuje stan wysoki. Wówczas mikrokontroler powinien propagować stan wysoki na D21 na krótki czas, otwierając tranzystor, a następnie go ponownie zamknąć aż do odczytania następnego zbocza narastającego na D22. W ten sposób sterować można elektromagnesami elementów przełączanych przez kulkę, na przykład „grzybków” i „proc”. Sterowany czysto programowo elektromagnes nie wymaga przycisków i ich obwody sprowadzają się wyłącznie do otwarcia tranzystora przez program w odpowiednim momencie. Można to osiągnąć z użyciem układu na rysunku 2.4. W ten sposób obsługiwać można „koryto”.



Rys. 2.4. Sprzętowo-programowy sterownik elektromagnesu bez przycisku

Aby programowo przełączać elektromagnes, należy zmieniać stan na pinie wyjściowym, połączonym z bazą tranzystora. Poniższy kod ma zastosowanie w cewkach sterowanych programowo, gdzie do bazy włączony jest pin D21:

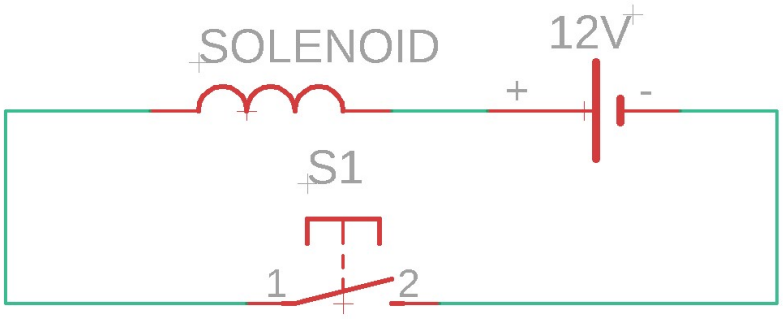
```
void aktywujGrzybek() {
  digitalWrite(21,HIGH);
  delay(100);
  digitalWrite(21,LOW);
}
```

W tym przypadku elektromagnes jest aktywowany na czas 100 ms, a następnie ponownie dezaktywowany do następnego przełączenia. Przekłada się to na szybkie pociągnięcie lub popchnięcie „bolca”. Funkcja wywoływana jest w momencie wykrycia stanu wysokiego na pinie wejściowym, w tym przypadku D22.

```
void loop() {
```

```
  int odczyt = digitalRead(22);

   if (odczyt == HIGH) {
     aktywujGrzybek();
   }
}
```

W układzie automatu znajdują się także dwa elektromagnesy całkowicie niezależne od Arduino, które nie wymagają klucza tranzystorowego ani diody, a przełącza się je jedynie z użyciem przycisku, jak pokazano na rysunku 2.5.

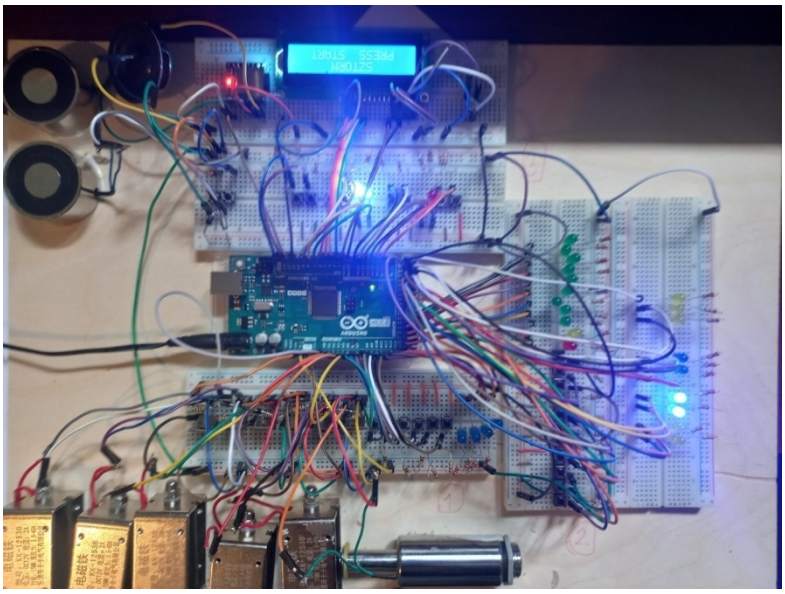


Rys. 2.5. Sterownik bez tranzystora

Te elektromagnesy są odpowiedzialne za ruch łapek (flipperów) i są obsługiwane przez gracza poprzez przytrzymanie palcem przycisku S1. Elektromagnes trzyma łapkę tak długo, jak przycisk jest wciśnięty. Powodem zastosowania elektromagnesu trzymającego jest niższy pobór prądu – ten jest długo trzymany, w związku z czym zużywa dużą część mocy zasilacza. Wykorzystane modele elektromagnesów ciągnących pobierają dwa razy więcej prądu od trzymających.

3. KONSTRUKCJA PROTOTYPU UKŁADU I APLIKACJI STERUJĄCEJ

Znając elektroniczne podstawy układów opisanych w poprzednich dwóch rozdziałach można skonstruować prototyp obwodu, który umieścić można w automacie flipper. Z wykorzystaniem tych wzorców wykonano jeden egzemplarz obwodu, który zawiera wszystkie opisywane wcześniej elementy i podzespoły. Prototyp został wykonany na płytkach stykowych.



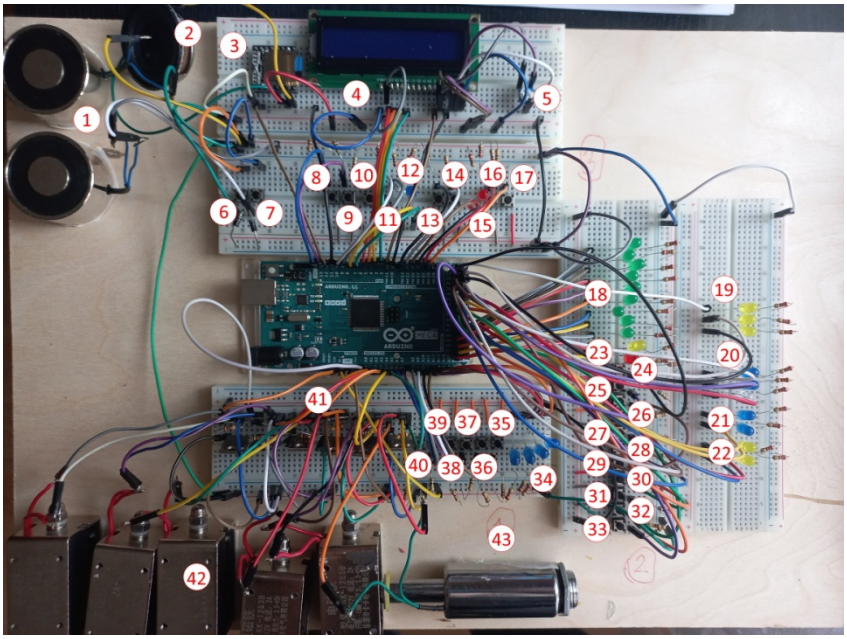
Rys. 3.1. Uruchomiony prototyp układu flippera

Poszczególne połączenia pomiędzy elementami w sposób bardziej czytelny opisuje schemat elektroniczny w Załączniku 1. Układ poza Arduino składa się z 29 sterowanych za jego pomocą diod LED, 27 przycisków TACT połączonych z płytą, ośmiu elektromagnesów, z czego sześcioma także steruje się z użyciem mikrokontrolera, oraz cyfrowych podzespołów – DFPlayera i wyświetlacza LCD. Wszystkim tym elementom towarzyszą rezystory o odpowiednio dobranych wartościach rezystancji. Elektromagnesy wymagają napięcia 12 V do działania, a zatem zadziałają one kiedy układ jest zasilany z sieci, ale nie podczas programowania z użyciem USB.

Tak zaprojektowany układ nie jest jednak bez wad. Ze względu na dużą ilość solenoidów i duży pobór mocy elementy często tracą na sprawności względem siebie. Do zasilania układu wykorzystano powszechny zasilacz o napięciu wyjściowym 12 V i prądzie 1,5 A, co sprawia, że zasilanie nawet jednego solenoidu negatywnie wpływa na działanie innego elementu.

3.1. Układ elektroniczny i propozycja przeniesienia go na automat

Przetwornik analogowo-cyfrowy nie jest wykorzystywany w tym projekcie, a wszelkie sygnały analogowe działają niezależnie od Arduino, w związku z czym można potraktować wszystkie szesnaście pinów analogowych jako cyfrowe. Oznacza to, że w praktyce na Arduino znajduje się siedemdziesiąt pinów, które można wykorzystać w projekcie flippera, to jest obsługa modułów, przełączników i elementów optoelektronicznych wchodzących w skład automatu. Skonstruowany prototyp wykorzystuje jednak o dwa piny mniej – nie wykorzystano pary 20 i 21, gdyż jest ona zgodna wyłącznie z interfejsem TWI, który nie znalazł w tym projekcie żadnego zastosowania. Czytelny schemat całego układu oraz zestawienie numerów pinów i urządzeń do nich podłączonych zamieszczono w Załączniku 1. Napisana aplikacja sterująca będzie działać wyłącznie dla tej konfiguracji. Na podstawie ilości poszczególnych elementów zdefiniować można projekt i zasady rzeczywistego automatu.



Rys. 3.2. Prototyp układu flippera z oznaczeniami elementów

Elementy tego samego prototypu, oznaczone na rysunku 3.2 liczbami od 1 do 43, mogą mieć następujące zastosowania w rzeczywistym flipperze:

- 1 - elektromagnesy łapek; w rzeczywistym stole powinny być to solenoidy ciągnące, a nie trzymające. Aby przetestować ich działanie, należy przyłożyć ferromagnetyk.
- 2 - głośnik, z którego odtwarzać można muzykę i efekty dźwiękowe.

- 3 - moduł MP3 DFPlayer, który odtwarza pewne utwory z karty microSD, w zależności od tego, jak zostanie zaprogramowany z użyciem mikrokontrolera. W prototypie zamiast rzeczywistych dźwięków jest nagranie, co automat robi w danym momencie (“gra trwa”, “trafiono cel” etc.)
- 4 - moduł wyświetlacza, wyświetla wynik, numer rundy, a przed rozpoczęciem gry wyświetla nazwę potencjalnego automatu - “SZTORM”.
- 5 - potencjometr połączony z modułem wyświetlacza,
- 6,7 - przyciski służące do sterowania łapkami, tak długo jak są wciśnięte, łapki w grze są skierowane do góry (na chwilę obecną zamiast tego aktywuje elektromagnesy trzymające),
- 8 - przycisk mający zastosowanie w jednej z „przetok” (ang. *rollover*, przycisku zwieranego przez przetoczenie przez niego kulki). Wciśnięcie przycisku powoduje odtworzenie dźwięku i dodane 15000 punktów jeżeli świeci dioda LED połączona z pinem D41. W przeciwnym razie dodane zostanie tylko 5000 punktów.
- 9 - drugi przycisk o podobnej funkcjonalności, 15000 punktów zostanie dodane jeżeli świeci dioda LED połączona z pinem D39, w przeciwnym razie 5000 punktów.
- 10 - przycisk mający zastosowanie w „tarczy” (ang. *target*, elementu aktywowanego przez uderzenie w niego kulką). Wciśnięcie spowoduje odtworzenie dźwięku, dodanie 1000 punktów i zapalenie diody LED połączonej z pinem D41, oraz zgaszenie diody LED połączonej z pinem D39.
- 11 - j.w., ale załączana i gaszona dioda LED zamienione są ze sobą.
- 12 - dwie diody LED zawsze załączone, kontrolka zasilania Arduino,
- 13 - przycisk mający zastosowanie w jednej z „przetok”, odtwarza dźwięk oraz dodaje 1500 punktów gdy dioda LED na pinie D39 świeci, w przeciwnym razie dodaje 500 punktów,
- 14 - przycisk mający zastosowanie w jednej z „przetok”, odtwarza dźwięk oraz dodaje 1500 punktów gdy dioda LED na pinie D41 świeci, w przeciwnym razie dodaje 500 punktów,
- 15 - dwie diody LED związane z dodatkową rundą. Jeżeli dioda LED połączona z pinem D16 jest załączona, oznacza to, że będzie miała miejsce dodatkowa runda. Jeżeli dioda na pinie D17 jest załączona, zwarcie przycisku połączonego z pinem D32 zgasi tę diodę, jednocześnie załączając tę na pinie D16.
- 16,17,23,27,28,29,30,31,32 - przyciski połączone z dziewięcioma innymi „tarczami”, na rzeczywistym stole zostaną pogrupowane w trzy zestawy celów obok siebie, nazywane “bankami”. Naciśnięcie przycisku powoduje odtworzenie dźwięku, dodanie

1000 punktów, zwiększenie licznika bonusu o 1 oraz zwiększenie licznika trafionych celów o 1. Jeżeli licznik trafionych „tarcz" jest podzielny przez 3 albo 5, bonus zwiększa się jeszcze o 1, jeżeli jest równy 15, powstanie możliwość otrzymania dodatkowej rundy przez wciśnięcie przycisku połączonego z D32 (na zdjęciu numer 28). Licznik trafionych celów zeruje się z końcem każdej rundy.

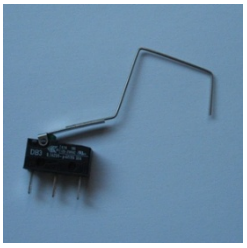
- 18 - diody LED informujące o podstawowej wysokości bonusu. Zielone diody symbolizują liczbę jedności licznika bonusu (od 1 do 9), żółta dodaje 10, czerwona natomiast 20. Maksymalna wartość licznika wynosi 39. Bonus otrzymywany jest na końcu rundy, jako wartość licznika pomnożona przez 1000 oraz dodatkowo przez mnożnik, który może wynieść 1,2,3,4 albo 5. Licznik bonusu zeruje się z końcem każdej rundy.
- 19 - diody LED przy trzech przetokach w górnej części stołu, jeżeli dioda przy danej przetoce jest załączona, oznacza to, że dana przetoka została już trafiona.
- 20 - diody LED do zamontowania wewnątrz „grzybków". Jeżeli przyciski połączone z grzybkami (A6,A7 i A8) trafi się wystarczająco dużo razy (kolejno 30, 50 i 80 razy), diody zostaną załączone.
- 21 - diody LED do zamontowania wewnątrz „proc". Są zawsze załączone.
- 22 - diody LED połączone z pinami D39 i D41; informują, czy dodatek punktowy za trafienie przycisków połączonych z pinami D12, D13, D14 i D15 zostanie potrojony.
- 24,25,26 – przyciski do połączenia z trzema „przetokami" w górnej części stołu. Jeżeli dioda LED koło danej „przetoki" jeszcze nie świeci, dodane zostanie 2000 punktów, w przeciwnym razie tylko 500. Jeżeli wszystkie trzy przyciski zostaną wciśnięte, mnożnik bonusu zwiększy się o 1, a diody zgasną. Za każdym razem dodatkowo zostanie odtworzony dźwięk.
- 33 - przycisk startu lub restartu gry; zeruje postęp gry oraz aktywuje wyrzutnię, a także powoduje odtwarzanie w tle muzyki (w prototypie jest to zapętlony głos "gra trwa").
- 34 - diody LED symbolizujące mnożnik bonusu; wynosi on 1 + liczba załączonych diod.
- 35 - przycisk montowany w „przetoce" w najniższej części stołu; jego zwarcie oznacza koniec rundy. W tej sytuacji na wyświetlaczu pojawia się wysokość otrzymanego bonusu, uruchamiana jest wyrzutnia oraz zaczyna się kolejna runda.
- 36,37,38,39,40 - przyciski symbolizujące zwarcie "spódnicy" (ang. *skirt*, przełącznika „procy" lub „grzybka" o charakterystycznym kształcie). Zarówno w przypadku „proc" jak i „grzybków" oznacza to wykonanie szybkiego (0,1 s) ruchu elektromagnesem

ciągnącym. Trafienie „procy” zwiększa wynik o 200 punktów, niebieskiego „grzybka” o 50, białego natomiast o 100. W przypadku „grzybków” z załączoną diodą premia jest mnożona razy 4.

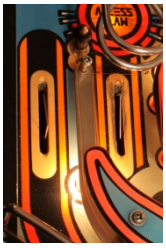
- 41 - układy sterujące elektromagnesami wyrzutni, „proc” i „grzybków”.
- 42 - elektromagnesy ciągnące (solenoidy) „proc” i „grzybków”, w rzeczywistym stole zostaną wykorzystane te same modele, dwa z bolcem o długości 1 cm, trzy z bolcem o długości 2 cm.
- 43 - elektromagnes pchający wyrzutni kulek.

Przenosząc te elementy do rzeczywistego automatu należy wykonać pewne niezależne od struktury obwodu modyfikacje związane z ich kształtem. Przykładowo diody LED należy umieszczać pod powierzchnią planszy, wewnątrz transparentnych elementów z plastiku, na które często nakłada się napisy, np. numer celu. Ich kolor powinien być taki sam co kolor światła diody. Elementy te znajdują zastosowanie także na pionowej tablicy wyników (ang. *backglass*), montowanej za planszą na wysokości głowy gracza, celem jej podświetlenia w często przyciemnionym pomieszczeniu salonu gier. To właśnie tam swoje miejsce znajduje wyświetlacz LCD. Arduino oraz drugi podzespół cyfrowy, DFPlayer, nie powinny być widoczne dla gracza. Do tego drugiego podłącza się głośnik, który też powinien być schowany za ścianą automatu, należy jednak przewidzieć otwory w tej ścianie umożliwiające wyjście fali dźwiękowej poza nią.

Realizacja elementów bezpośrednio związanych z punktacją wymaga bardziej zaawansowanych modyfikacji. Z użyciem standardowych przycisków TACT zrealizować można co najwyżej przyciski po bokach automatu, służące do sterowania łapkami, oraz cele do trafiania przez kulkę. Do konstrukcji „przetok” wykorzystuje się płytsze przyciski ze zgiętym drutem, którego część wystaje powyżej powierzchni planszy [4].

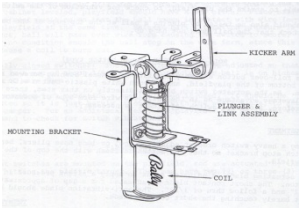


Rys. 3.3. Tradycyjny przycisk używany w „przetoce”



Rys. 3.4. „Przetoka”

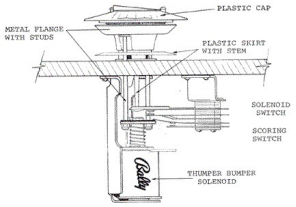
Najtrudniejsze w realizacji są elementy wykorzystujące solenoidy, czyli przykładowo „proce”, „grzybki” i łapki. Te pierwsze do działania wykorzystują obrotowe ramię, które jest przyciągane przez przewodzący solenoid. Dzieję się tak, kiedy kulka ma styczność z najdłuższym bokiem trójkątnej „procy”. W celu pokrycia całego tego boku pomiędzy trzy tworzące trójkąt wkręty naciąga się gumową opaskę, która po przyciągnięciu ramienia odpycha kulkę [15].



Rys. 3.5. Mechanizm działania „procy”

Rys. 3.6. „Proca”

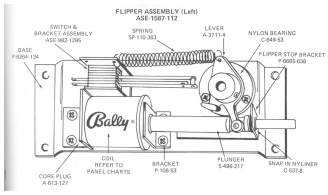
Celem realizacji „grzybków” na koniec bolca rdzenia solenoidu nakłada się okrągły element w kształcie stożka ściętego, który w momencie styczności z kulką zostaje na niej „zatrzaśnięty”, odpychając ją od siebie. Do aktywacji wykorzystuje się plastikowy przełącznik, nazywany „spódnicą” (ang. *skirt*), który po poruszeniu przez kulkę wytrącany jest z równowagi i zwiera obwód [12].



Rys. 3.7. Mechanizm działania „grzybka”

Rys. 3.8. „Grzybek”

Najważniejszy elektromechaniczny element flippera, czyli łapki, w działaniu przypominają „proce”. Tutaj też wykorzystuje się ruch obrotowy wynikający z przyciągania przez solenoid obrotowego ramienia z osią łapki [15]. Mechanizm łapki jest chiralny.



Rys. 3.9. Mechanizm działania lewej łapki

Rys. 3.10. Łapka flippera

Przed przełożeniem układu do prawdziwego flippera należy dobrać odpowiednie modele elektromagnesów (na przykłada zamiast trzymających zastosować ciągnące) oraz zasilacz o wyższej mocy. Tradycyjnie układy flipperów z odpowiednimi solenoidami zasila się napięciem 48 V (dawniej 70 V) o natężeniu prądu wynoszącym 5 A [2]. Należy pamiętać, że Arduino jest zasilane z 12 V, w związku z czym w układzie wówczas powinna znaleźć się przetwornica typu *buck*.

3.2. Program sterujący mikrokontrolerem

Kod źródłowy aplikacji sterującej mikrokontrolerem został napisany w środowisku Arduino IDE i wykorzystuje zaledwie 3% pamięci flash i 10% pamięci zmiennych globalnych. Na początku kodu zdefiniowano piny wejściowe i wyjściowe, a także informacje o postępie gry – na przykład wynik, numer rundy (nazywany też czasami numerem kulki), albo czy graczowi przysługuje dodatkowa runda. W momencie uruchomienia automatu dodatkowo inicjuje się wyświetlacz i DFPlayer – gra nie zaczyna się zaraz po uruchomieniu, zamiast tego na ekranie wyświetla się tradycyjny napis „*PRESS START*”. Z głośnika natomiast słychać zapętlony głos mówiący, że automat jest w trybie *attract mode*, czyli w trybie zachęcania do gry. Po skończonej rozgrywce automat wraca do tego trybu.

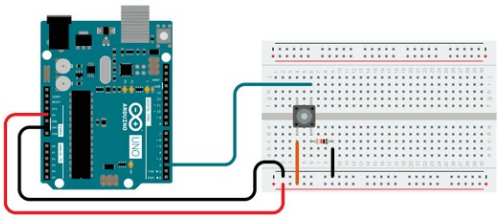
Przełącznik połączony z pinem A11 docelowo ma znaleźć się w dolnej części stołu, poniżej łapek. Wówczas runda się kończy i licznik kulek powinien zostać zwiększony o jeden, czemu powinna towarzyszyć odpowiednia informacja. Podobnie po zakończeniu wszystkich trzech rund w to miejsce powinna trafić stosowna informacja.

```
    void kulka_stracona() {
      kulka++; // zwiększ numer kulki = numer rundy

      if(kulka<3) {
        lcd.setCursor(0, 1);
        lcd.print("     BALL      "); // 5 spacji przed i 7 po
wyrazie "BALL"
        lcd.setCursor(10, 1);
        lcd.print(kulka+1); // zapisz jednocyfrowy numer kulki na
pozycji 11
      }
      else {
        koniec_gry = true;
        lcd.setCursor(0, 1);
        lcd.print("   GAME  OVER   "); // 3 spacje przed wyrazem
"GAME" i po wyrazie "OVER", 2 pomiędzy nimi
      }
    }
```

Gra zaczyna się po wciśnięciu przycisku połączonego z pinem D42 Arduino. Wówczas w pętli po kolei sprawdza się wszystkie przyciski, czy aby nie zostały wciśnięte i czy nie należy nagrodzić gracza dodatkowymi punktami, zwiększeniem bonusu czy nawet dodatkową rundą. Po otrzymaniu punktów należy na bieżąco aktualizować stosowną informację na wyświetlaczu LCD.

```
void dodaj_punkty(int punkty) {
  wynik += punkty; // dodaj punkty do wyniku
  lcd.setCursor(0, 0); // nadpisuj znaki od początku pierwszej
linijki
  lcd.print(wynik); // zapisz wynik
}
```

Przedstawione funkcje są podstawą do realizacji programowej części automatu i w ostatecznej wersji zostały dostosowane do reszty kodu źródłowego. Funkcję *dodaj_punkty* powinno się wołać w momencie aktywacji pewnego elementu zwiększającego wynik, np. trafienie grzybka albo przetoczenie się kulki przez przycisk, *kulka_stracona* natomiast po jej utracie. Elementy te powinny zwierać obwody ustawione jako wejściowe, a także implementować tzw. debouncing, czyli mechanizm eliminujący drżenie styków. Wśród metod debouncingu można wyróżnić mechaniczne, elektroniczne i programowe [16]. Przyciski poza połączeniem z płytką muszą także być zwarty do masy rezystorem pull-up o odpowiednio dużej rezystancji, aby nie tworzyć zbędnego dzielnika napięcia. W tym przypadku wynosi ona 10 kΩ.



Rys. 3.11. Połączenie przycisku z debouncingiem

Następnie dla każdego z tych pinów wejściowych należy zapamiętać ich numer, ostatni stan oraz czas od wciśnięcia:

```
const int piny_wejsciowe[2] = {6, 7};
int odczyt[2] = {LOW, LOW};
int ostatni_odczyt[2] = {LOW, LOW};
unsigned long czas_od_wcisniecia[2] = {0,0};
```

W funkcji setup należy ustawić ich tryb na wejście:

```
for(int i=0; i<2; i++)
    pinMode(piny_wejsciowe[i],INPUT);
```

W funkcji loop natomiast należy mierzyć czas od ostatniego zbocza narastającego i wołać funkcje tylko jeżeli pomiędzy naciśnięciami upłynęło przynajmniej 50 ms, nie blokując jednocześnie działania reszty programu. Zarówno obwód, jak i algorytm są oficjalnym rozwiązaniem zaproponowanym przez producenta płyt Arduino [25].

```
    for(int i=0; i<2; i++) {
        int nowy_odczyt = digitalRead(piny_wejsciowe[i]);
        // licz tylko jeżeli zmienił się stan (zbocze)
        if (nowy_odczyt != ostatni_odczyt[i]) {
          czas_od_wcisniecia[i] = millis();
        }
        // sprawdź, czy minęło przynajmniej 50 ms
        if (millis() - czas_od_wcisniecia[i] > 50) {
          if (nowy_odczyt != odczyt[i]) {
            odczyt[i] = nowy_odczyt;
        // jeżeli minęło 50 ms i zbocze jest narastające, wołaj
funkcję
            if (odczyt[i] == HIGH) {
              if(piny_wejsciowe[i] == 7)
                kulka_stracona();
              else
                dodaj_punkty(200);
            }
          }
        }
        ostatni_odczyt[i] = nowy_odczyt;
      }
```

Pełny kod aplikacji sterującej automatem, uwzględniający wszystkie zdefiniowane wcześniej zasady gry, znajduje się w Załączniku 2.

PODSUMOWANIE I WNIOSKI KOŃCOWE

Badania rozpocząłem od znalezienia powszechnie dostępnych cyfrowych podzespołów umożliwiających konstrukcję takiego układu. W ich trakcie brałem pod uwagę dostępność, interfejs komunikacyjny i poziom trudności ich programowania. Po dobraniu odpowiedniego wyświetlacza i procesora sygnałowego plików dźwiękowych wyselekcjonowałem ich funkcje, które znajdują zastosowanie w projekcie automatu, a następnie przedstawiłem sprzętowo-programowe propozycje wykorzystania tych funkcji w układzie. W ten sposób udowodniłem, iż mikrokontroler ATmega2560 wykorzystać można w zakresie odtwarzania dźwięków i informowania o postępach w grze.

Po pomyślnym teście układów cyfrowych, zarówno od strony sprzętowej jak i programowej, zbadałem sposoby na sterowanie diodami LED oraz elektromagnesami z użyciem mikrokontrolera. W przypadku elektromagnesów konieczna okazała się konstrukcja sterowników wzmacniających wydajność prądową wyjść Arduino. Poprzez pomyślne przetestowanie tych układów udowodniłem wykonalność wykorzystania wspomnianego mikrokontrolera jako sterownika urządzeń biorących bezpośredni udział w grze w pinball.

Produktem końcowym badań, projektowania, prac konstrukcyjnych i testów związanych z tą pracą jest prototyp obwodu elektronicznego do umieszczenia w automacie flipper, oparty o płytę ewaluacyjną Arduino Mega 2560, oraz kod aplikacji sterującej układem, o zachowaniu adekwatnym do charakterystyki elementów i zasad gry na takim automacie. Prototypowi towarzyszy jego schemat, na podstawie którego można, w bardziej profesjonalny sposób, zreplikować układ do produkcji rzeczywistego automatu. Poprzez tę pracę udowodniłem wykonalność wykorzystania popularnego mikrokontrolera ATmega2560 w takim układzie. Podczas analizy tematu także samemu nauczyłem się nowych rzeczy – przykładowo jak po części programowo realizować debouncing, oszczędzając kondensator, a także zasady działania mniej znanego układu DFPlayer.

W rozdziale trzecim opisałem także sprawdzone, istniejące od lat trzydziestych XX wieku metody realizacji elementów gry, które mają zastosowanie w przemysłowo produkowanych stołach do dzisiaj, a od strony elektronicznej wykorzystują elementy przeze mnie opisane. W przemysłowej produkcji zamiast Arduino należy jednak zastosować autorską konfigurację mikrokontrolera ATmega2560.

BIBLIOGRAFIA

Bibliografia w kolejności alfabetycznej nazwisk autorów:

1. Chad J.: Pinball: A Graphic History of the Silver Ball, First Second Books, Nowy Jork 2022.
2. De Jager H.: Pinball Machine Maintenance, H.P. De Jager, Amsterdam 1991.
3. Hemalatha C., Nagarajan R., Suresh P., Ganesh Shankar P.: Brushless DC Motor Controlled by using Internet of Things, JSTE - International Journal of Science Technology & Engineering, Volume 3, Issue 09, 2017.
4. Kamoroff B.B.: Pinball Machine Care and Maintenance, 2nd Edition, Bell Springs Publishing, Willits 2011.
5. Kamoroff B.B.: Your Pinball Machine: How to Purchase, Adjust, Maintain, and Repair Your Own Machine, Schiffer Publishing, Limited, Atglen 2021.
6. Kang S., Yeo H., Yoon J.: Applying Chemistry Knowledge to Code, Construct, and Demonstrate an Arduino−Carbon Dioxide Fountain, Journal of Chemical Educaction 2019, 96, 313−316.
7. Kartika L., Hapsari G.I., Mutiara G.A.: Smart-Cane for The Blind with Wind Direction Position based-on Arduino, The 4th Annual South East Asian International Seminar (ASAIS), Dżakarta 2015.
8. Kurnia Y., Li Sie J.: Prototype of Warehouse Automation System Using Arduino Mega 2560 Microcontroller Based on Internet of Things, Journal Binary Digital – Technology, Vol.1, No.3, s. 124-130.
9. Monk S.: Arduino dla początkujących. Podstawy i szkice, Wydawnictwo Helion, Gliwice 2014.
10. Mui E.NC.: FPGA Interfacing of HD44780 Based LCD Using Delayed Finite State Machine (FSM), Texco Enterprise Ptd. Ltd, Apex 2007.
11. Parihar V.R., Tonge A.Y., Ganorkar P.D.: Heartbeat and Temperature Monitoring System for Remote Patients using Arduino, International Journal of Advanced Engineering Research and Science (IJAERS), Vol-4, Issue-5, s. 57.
12. Pring S.R., Budd C.J.: The Dynamics of a Simplified Pin-Ball Machine, IMA journal of applied mathematics, Volume: 76 Issue: 1, Oxford 2011, s. 2-3.
13. Rossignoli M.: The Complete Pinball Book: Collecting the Game & Its History, Schiffer Publishing, Limited, Atglen 2002.
14. Stepowicz W.J.: Elementy półprzewodnikowe, Wydawnictwo Akademii Morskiej w Gdyni, Gdynia 2009.
15. Tolbert C.F., Tolbert J.A.: Tilt, The Pinball Book: Home Maintenance, History, Hot Tips, Creative Arts Book Company, Berkeley 1978.
16. Yershov R.D., Voytenko V.P., Bychko V.A.: Software-based Contact Debouncing Algorithm with Programmable Auto-Repeat Profile Feature, 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kijów 2019.

Źródła internetowe w kolejności występowania w tekście:

17. https://docs.arduino.cc/hardware/mega-2560, Mega 2560 Rev3 - Arduino Documentation, (data dostępu 29.01.2024 r.).
18. https://docs.arduino.cc/resources/datasheets/A000067-datasheet.pdf, Nota katalogowa płyty Arduino Mega 2560 Rev3, (data dostępu 29.01.2024 r.).
19. https://community.element14.com/products/arduino/w/documents/2969/arduino-mega-2560-rev3-pinout-atmega2560-pin-mapping-eagle-files-schematics-and-more, Schematy elektroniczne płyty rozwojowej Arduino Mega 2560 Rev3, (data dostępu 29.01.2024 r.).

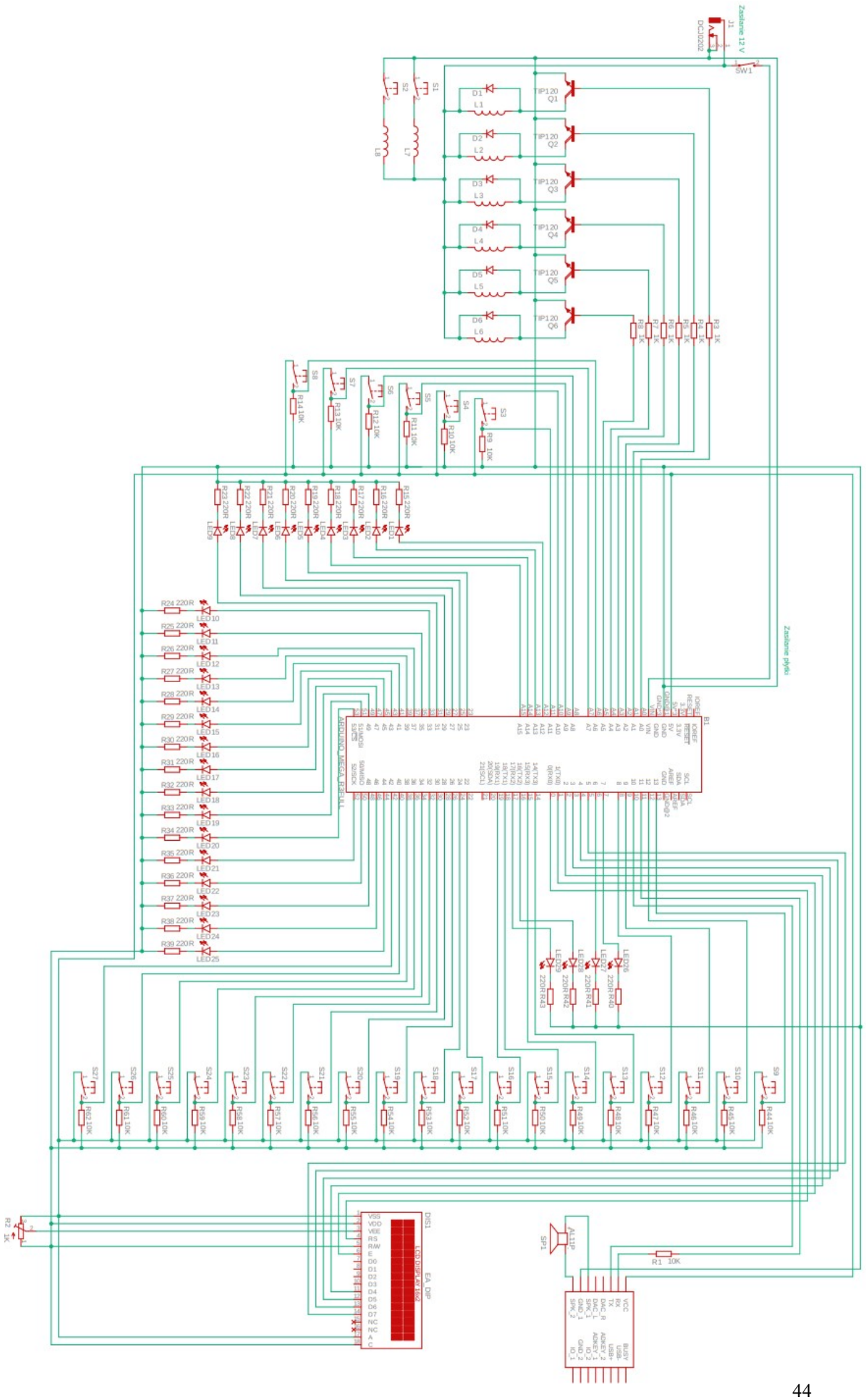
20. https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf, Nota katalogowa mikrokontrolera ATmega2560, (data dostępu 29.01.2024 r.).
21. https://docs.rs-online.com/58c2/A700000006944490.pdf, Nota katalogowa układu DFPlayer, (data dostępu 29.01.2024 r.).
22. https://www.sparkfun.com/datasheets/LCD/HD44780.pdf, Nota katalogowa układu HD44780, (data dostępu 29.01.2024 r.).
23. http://krzysztof.halawa.staff.iiar.pwr.wroc.pl/wyswietlacz_LCD_HD44780.pdf, Wyświetlacz LCD z układem HD44780 – Politechnika Wrocławska, (data dostępu 29.01.2024 r.).
24. https://www.onsemi.com/pdf/datasheet/tip120-d.pdf, Nota katalogowa układu Darlingtona mocy TIP120, (data dostępu 29.01.2024 r.).
25. https://docs.arduino.cc/built-in-examples/digital/Debounce, Debounce on a Pushbutton, (data dostępu 29.01.2024 r.).

ZAŁĄCZNIK 1: SCHEMAT ELEKTRONICZNY PROTOTYPU I SPIS URZĄDZEŃ POŁĄCZONYCH Z ARDUINO

Poniższa tabela zawiera urządzenia podłączone do poszczególnych pinów Arduino i ich funkcje w potencjalnym automacie flipper korzystającym z zaprojektowanego układu.

Pin Arduino	Podłączony element	Funkcja w potencjalnym automacie	Pin Arduino	Podłączony element	Funkcja w potencjalnym automacie
D0	Wyświetlacz - RS	Sterowanie wyświetlaczem	D35	LED11 - niebieska	LED w lewej procy
D1	Wyświetlacz - E		D36	S24	Bank 3, cel 1
D2	Wyświetlacz - D4		D37	LED12 - niebieska	LED w prawej procy
D3	Wyświetlacz - D5		D38	S25	Bank 3, cel 2
D4	Wyświetlacz - D6		D39	LED13 - żółta	Lampa lewej przetoki
D5	Wyświetlacz - D7		D40	S26	Bank 3, cel 3
D6	LED27 - niebieska	Dioda LED na szkle tylnym	D41	LED14 - żółta	Lampa prawej przetoki
D7	LED26 - biała	Dioda LED na szkle tylnym	D42	S27	(Re)start gry
D8	S12	Lewy cel na środku stołu	D43	LED15 - zielona	Lampa bonusu 1
D9	S11	Prawy cel na środku stołu	D44	LED25 - zielona	Lampa bonusu 2
D10	DFPlayer - RX	Sterowanie modułem DFPlayer	D45	LED16 - zielona	Lampa bonusu 3
D11	DFPlayer - TX		D46	LED24 - zielona	Lampa bonusu 4
D12	S10	Przetoka przy lewym pasie końcowym	D47	LED17 - zielona	Lampa bonusu 5
D13	S9	Przetoka przy prawym pasie końcowym	D48	LED23 - zielona	Lampa bonusu 6
D14	S13	Przetoka przy lewym pasie flippera	D49	LED18 - zielona	Lampa bonusu 7

D15	S14	Przetoka przy prawym pasie flippera	D50	LED22 - zielona	Lampa bonusu 8
D16	LED28 - czerwona	Lampa “strzelaj ponownie”	D51	LED19 - zielona	Lampa bonusu 9
D17	LED29 - czerwona	Lampa dodatkowej kulki	D52	LED21 - żółta	Lampa bonusu 10
D18	S15	Bank 1, cel 1	D53	LED20 - czerwona	Lampa bonusu 20
D19	S16	Bank 1, cel 2	A0	L1	„Grzybek” 1
D20	nie są używane		A1	L2	„Grzybek” 2
D21			A2	L3	„Grzybek” 3
D22	S17	Bank 1, cel 3	A3	L4	Lewa „proca”
D23	LED5 - żółta	Lampa przetoki górnej 1	A4	L5	Prawa „proca”
D24	S18	Przetoka górna 1	A5	L6	Wyrzutnia kulek
D25	LED6 - żółta	Lampa przetoki górnej 2	A6	S8	„Spódnica grzybka” 1
D26	S19	Przetoka górna 2	A7	S7	„Spódnica grzybka” 2
D27	LED7 - żółta	Lampa przetoki górnej 3	A8	S6	„Spódnica grzybka” 3
D28	S20	Przetoka górna 3	A9	S5	„Spódnica” lewej „procy”
D29	LED8 - niebieska	LED w grzybku 1	A10	S4	„Spódnica” prawej „procy”
D30	S21	Bank 2, cel 1	A11	S3	Przełącznik końca rundy
D31	LED9 - niebieska	LED w grzybku 2	A12	LED1 - niebieska	Lampa mnożnika x2
D32	S22	Bank 2, cel 2	A13	LED2 - niebieska	Lampa mnożnika x3
D33	LED10 - biała	LED w grzybku 3	A14	LED3 - niebieska	Lampa mnożnika x4
D34	S23	Bank 2, cel 3	A15	LED4 - niebieska	Lampa mnożnika x5


Zasilanie 12 V
DCJ0202
SW1
TIP120
Zasilanie płytki
ARDUINO_MEGA_R3-ULL
LCD DISPLAY 16x2
EA_DIP
SPK1

ZAŁĄCZNIK 2: KOD ŹRÓDŁOWY PROGRAMU STERUJĄCEGO W JĘZYKU ARDUINO C

```
#include <LiquidCrystal.h>
#include "SoftwareSerial.h"

LiquidCrystal lcd(0,1,2,3,4,5);
SoftwareSerial Ss(10, 11);

unsigned long wynik = 0;
int kulka = 0;
bool koniec_gry = false;
bool gra_rozpoczeta = false;
bool dodatkowa_kulka = false;
int bonus = 0;
int wspolczynnik_bonus = 1;

bool aktywna_lewa = false;
bool dodatkowa_kulka_gotowa = false;
bool trzy_grzybki[3] = {false, false, false};
bool trzy_przetoki[3] = {false, false, false};
int ukonczone_cele = 0;
int uderzenia_w_grzybki = 0;
unsigned long czas_konca_gry = 0;

const int piny_wejsciowe[25] = {8, 9, 12, 13, 14, 15, 18,
19, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, A6, A7,
A8, A9, A10, A11};
int odczyt[25] = {LOW, LOW, LOW, LOW, LOW, LOW, LOW, LOW,
LOW, LOW, LOW, LOW, LOW, LOW, LOW, LOW, LOW, LOW, LOW,
LOW, LOW, LOW, LOW, LOW, LOW};
int ostatni_odczyt[25] = {LOW, LOW, LOW, LOW, LOW, LOW,
LOW, LOW, LOW, LOW, LOW, LOW, LOW, LOW, LOW, LOW, LOW,
LOW, LOW, LOW, LOW, LOW, LOW, LOW, LOW};
unsigned long czas_od_wcisniecia[25] =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

const int piny_wyjsciowe[35] = {6, 7, 16, 17, 23, 25, 27,
29, 31, 33, 35, 37, 39, 41, 43, 44, 45, 46, 47, 48, 49,
50, 51, 52, 53, A0, A1, A2, A3, A4, A5, A12, A13, A14,
A15};
const int ledy_przetok[3] = {23,25,27};
const int ledy_wsp[4] = {A12, A13, A14, A15};

int dzwiek = 1;
bool czy_petla = false;
const unsigned long dlugosci_dzwiekow[10] =
{3000,2000,2000,4000,3500,4000,5000,3000,4000,3000};
unsigned long moment_odtworzenia = 0;

void setup() {
  Ss.begin(9600);

  delay(500);
  modul_mp3(0x3F, 0, 0);
  delay(500);
  modul_mp3(0x06,0,15);
  delay(500);
  modul_mp3(0x08,0,1);
  delay(500);

  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("     SZTORM     ");
  lcd.setCursor(0, 1);
  lcd.print("  PRESS  START  ");


  for(int i=0; i<25; i++)
    pinMode(piny_wejsciowe[i], INPUT);
  for(int i=0; i<35; i++)
    pinMode(piny_wyjsciowe[i], OUTPUT);

  zapal_leda(6);
  zapal_leda(7);
  zapal_leda(35);
  zapal_leda(37);
}

void loop() {
for(int i=0; i<25; i++) {
  int nowy_odczyt = digitalRead(piny_wejsciowe[i]);

  if (nowy_odczyt != ostatni_odczyt[i]) {
    czas_od_wcisniecia[i] = millis();
  }

  if (millis() - czas_od_wcisniecia[i] > 50) {
    if (nowy_odczyt != odczyt[i]) {
      odczyt[i] = nowy_odczyt;
      if (odczyt[i] == HIGH) {
        switch(piny_wejsciowe[i]) {
          case 8: //Lewy cel na środku stołu
            odtworz_dzwiek(4,false);
            ustaw_aktywna_strone(true);
            dodaj_punkty(1000);
            break;
          case 9: //Prawy cel na środku stołu
            odtworz_dzwiek(4,false);
            ustaw_aktywna_strone(false);
            dodaj_punkty(1000);
            break;
          case 12: //Przetoka przy lewym pasie końcowym
            odtworz_dzwiek(6,false);
            if(aktywna_lewa)
              dodaj_punkty(15000);
            else
              dodaj_punkty(5000);
            break;
          case 13: //Przetoka przy prawym pasie końcowym
            odtworz_dzwiek(6,false);
            if(!aktywna_lewa)
              dodaj_punkty(15000);
            else
              dodaj_punkty(5000);
            break;
          case 14: //Przetoka przy lewym pasie flippera
            odtworz_dzwiek(6,false);
            if(aktywna_lewa)
              dodaj_punkty(1500);
            else
              dodaj_punkty(500);
            break;
          case 15: //Przetoka przy prawym pasie flippera
            odtworz_dzwiek(6,false);
            if(!aktywna_lewa)
              dodaj_punkty(1500);
            else
              dodaj_punkty(500);
            break;
          case 18: //Bank 1, cel 1
            odtworz_dzwiek(5,false);
            sprawdz_cele(false);
            break;
          case 19: //Bank 1, cel 2
            odtworz_dzwiek(5,false);
            sprawdz_cele(false);
            break;
          case 22: //Bank 1, cel 3
            odtworz_dzwiek(5,false);
            sprawdz_cele(false);
            break;
          case 24: //Przetoka górna 1
            sprawdz_trzy_przetoki(0);
            break;
          case 26: //Przetoka górna 2
            sprawdz_trzy_przetoki(1);
            break;
          case 28: //Przetoka górna 3
            sprawdz_trzy_przetoki(2);
            break;
          case 30: //Bank 2, cel 1
            odtworz_dzwiek(5,false);
            sprawdz_cele(false);
            break;
          case 32: //Bank 2, cel 2
```

```
              odtworz_dzwiek(5,false);
              sprawdz_cele(true);
              break;
            case 34: //Bank 2, cel 3
              odtworz_dzwiek(5,false);
              sprawdz_cele(false);
              break;
            case 36: //Bank 3, cel 1
              odtworz_dzwiek(5,false);
              sprawdz_cele(false);
              break;
            case 38: //Bank 3, cel 2
              odtworz_dzwiek(5,false);
              sprawdz_cele(false);
              break;
            case 40: //Bank 3, cel 3
              odtworz_dzwiek(5,false);
              sprawdz_cele(false);
              break;
            case 42: //(Re)start gry
              restart_gry();
              break;
            case A11: //Przełącznik wejścia kulki do
fartucha
              kulka_stracona();
              break;
            case A6: //Spódnica grzybka 1
              if(uderzenia_w_grzybki>=30)
                dodaj_punkty(200);
              else
                dodaj_punkty(50);
              aktywuj_grzybek(A0);
              break;
            case A7: //Spódnica grzybka 2
              if(uderzenia_w_grzybki>=50)
                dodaj_punkty(200);
              else
                dodaj_punkty(50);
              aktywuj_grzybek(A1);
              break;
            case A8: //Spódnica grzybka 3
              if(uderzenia_w_grzybki>=80)
                dodaj_punkty(400);
              else
                dodaj_punkty(100);
              aktywuj_grzybek(A2);
              break;
            case A9: //Spódnica lewej procy
              zamien_aktywna_strone();
              aktywuj_proce(A3);
              break;
            case A10: //Spódnica prawej procy
              zamien_aktywna_strone();
              aktywuj_proce(A4);
              break;
          }
        }
      }
    }

    ostatni_odczyt[i] = nowy_odczyt;
  }

  if(millis() - moment_odtworzenia >
dlugosci_dzwiekow[dzwiek] && gra_rozpoczeta && !czy_petla)
    odtworz_dzwiek(2,true);

}

void dodaj_punkty(int punkty) {
  wynik += punkty;
  lcd.setCursor(0, 0);
  if(gra_rozpoczeta)
    lcd.print(wynik);
}

void zapal_leda(int pinLeda) {
  digitalWrite(pinLeda,HIGH);
}

void zgas_leda(int pinLeda) {
  digitalWrite(pinLeda,LOW);
}

void restart_gry() {
  wynik = 0;
  kulka = 0;
  dodatkowa_kulka = false;
  bonus = 0;
  wspolczynnik_bonus = 1;

  dodatkowa_kulka_gotowa = false;
  for(int i=0; i<3; i++) {
    trzy_grzybki[i] = false;
    trzy_przetoki[i] = false;
  }
  ukonczone_cele = 0;
  uderzenia_w_grzybki = 0;

  koniec_gry = false;
  gra_rozpoczeta = true;
  odtworz_dzwiek(2,true);
  delay(500);


  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.print(wynik);
  dodaj_punkty(0);
  lcd.setCursor(0, 1);
  lcd.print("     BALL      ");
  lcd.setCursor(10, 1);
  lcd.print(kulka+1);

  for(int i=0; i<35; i++)
    zgas_leda(piny_wyjsciowe[i]);
  zapal_leda(6);
  zapal_leda(7);
  zapal_leda(35);
  zapal_leda(37);
  ustaw_aktywna_strone(false);

  aktywuj_wyrzutnie();
}

void kulka_stracona() {
  odtworz_dzwiek(10,false);
  int obliczony_bonus = 1000*bonus*wspolczynnik_bonus;
  dodaj_punkty(obliczony_bonus);

  lcd.setCursor(0, 0);
  lcd.print("  BONUS    X    ");
  lcd.setCursor(13, 0);
  lcd.print(wspolczynnik_bonus);
  lcd.setCursor(0, 1);
  lcd.print("+                ");
  lcd.setCursor(1, 1);
  lcd.print(obliczony_bonus);
  delay(2000);

  if(dodatkowa_kulka) {
    lcd.setCursor(0, 0);
    lcd.print("  SHOOT  AGAIN  ");
    lcd.setCursor(0, 1);
    lcd.print("                ");
    delay(1000);
    dodatkowa_kulka = false;
  }
  else
    kulka++;


  bonus = 0;
  wspolczynnik_bonus = 1;
  dodatkowa_kulka_gotowa = false;
  ukonczone_cele = 0;
  for(int i=0; i<3; i++) {
    trzy_przetoki[i] = false;
    zgas_leda(ledy_przetok[i]);
  }
```

```
  zgas_leda(16);
  zgas_leda(17);
  ledy_bonus();

  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 0);
  lcd.print(wynik);

  if(kulka<3) {
    lcd.setCursor(0, 1);
    lcd.print("     BALL      ");
    lcd.setCursor(10, 1);
    lcd.print(kulka+1);
    aktywuj_wyrzutnie();
    odtworz_dzwiek(2,true);
  }
  else {
    koniec_gry = true;
    for(int i=0; i<3; i++)
    trzy_grzybki[i] = false;
    zgas_leda(29);
    zgas_leda(31);
    zgas_leda(33);
    lcd.setCursor(0, 0);
    lcd.print("   GAME  OVER   ");
    lcd.setCursor(0, 1);
    lcd.print("                ");
    lcd.setCursor(0, 1);
    lcd.print("Score:          ");
    lcd.setCursor(7, 1);
    lcd.print(wynik);
    odtworz_dzwiek(3,false);
    czas_konca_gry = millis();
    gra_rozpoczeta = false;

    delay(5000);

    lcd.setCursor(0, 0);
    lcd.print("     SZTORM     ");
    lcd.setCursor(0, 1);
    lcd.print("  PRESS  START  ");
    odtworz_dzwiek(1,true);
  }
}

void zwieksz_wspolczynnik() {
  if(wspolczynnik_bonus<5)
    wspolczynnik_bonus++;
  else
    dodaj_punkty(10000);

  ledy_bonus();
}

void zwieksz_bonus(int ile) {
  if(bonus<39)
    bonus += ile;

  ledy_bonus();
}

void ledy_bonus() {
  for(int i=1; i<10; i++) {
    if(bonus%10>=i)
      zapal_leda(42+i);
    else
      zgas_leda(42+i);
  }

  if(bonus>=30 || (bonus>=10 && bonus<20))
    zapal_leda(52);
  else
    zgas_leda(52);

  if(bonus>=20)
    zapal_leda(53);
  else
    zgas_leda(53);

  for(int i=0; i<4; i++) {
    if(wspolczynnik_bonus>=i+2)
      zapal_leda(ledy_wsp[i]);
    else
      zgas_leda(ledy_wsp[i]);
  }
}

void ustaw_aktywna_strone(bool lewa) {
  aktywna_lewa = lewa;

  if(aktywna_lewa) {
    zapal_leda(39);
    zgas_leda(41);
  }
  else {
    zapal_leda(41);
    zgas_leda(39);
  }
}

void modul_mp3(byte CMD, byte para1, byte para2) {
  word checksum = -(0xFF + 0x06 + CMD + 0x00 + para1 +
para2);
  byte komenda[10] = { 0x7E, 0xFF, 0x06, CMD, 0x00, para1,
para2, highByte(checksum), lowByte(checksum), 0xEF};
  for (byte i=0; i<10; i++) {
    Ss.write(komenda[i]);
  }
}

// 1 - attract mode, 2 - gra, 3 - koniec gry, 4 - cel 1, 5
- cel 2, 6 - przetoka 1, 7 - przetoka 2,
// 8 - przetoka 3, 9 - dodatkowa kulka, 10 - koniec rundy
void odtworz_dzwiek(int dzw, bool petla) {
  dzwiek = dzw;
  czy_petla = petla;
  moment_odtworzenia = millis();
  if(petla)
    modul_mp3(0x08,0,dzw);
  else
    modul_mp3(0x03,0,dzw);
  delay(10);
}

void aktywuj_grzybek(int pinGrzybka) {
  uderzenia_w_grzybki++;
  if(uderzenia_w_grzybki>=30)
    zapal_leda(29);
  if(uderzenia_w_grzybki>=50)
    zapal_leda(31);
  if(uderzenia_w_grzybki>=80)
    zapal_leda(33);

  digitalWrite(pinGrzybka,HIGH);
  delay(100);
  digitalWrite(pinGrzybka,LOW);
}

void aktywuj_proce(int pinProcy) {
  dodaj_punkty(200);

  digitalWrite(pinProcy,HIGH);
  delay(100);
  digitalWrite(pinProcy,LOW);
}

void aktywuj_wyrzutnie() {
  digitalWrite(A5,HIGH);
  delay(1000);
  digitalWrite(A5,LOW);
}

void sprawdz_trzy_przetoki(int przetoka) {
  if(trzy_przetoki[przetoka])
    dodaj_punkty(500);
  else
    dodaj_punkty(2000);

  trzy_przetoki[przetoka] = true;
  zapal_leda(ledy_przetok[przetoka]);
```

```
  bool ukonczone = trzy_przetoki[0] && trzy_przetoki[1] &&
trzy_przetoki[2];
  if(ukonczone) {
    odtworz_dzwiek(8,false);
    zwieksz_wspolczynnik();
    for(int i=0; i<3; i++) {
      trzy_przetoki[i] = false;
      zgas_leda(ledy_przetok[i]);
    }
  }
  else
    odtworz_dzwiek(7,false);
}

void sprawdz_cele(bool dod_kulka) {
  dodaj_punkty(1000);
  ukonczone_cele++;

  if(dod_kulka && dodatkowa_kulka_gotowa) {
    odtworz_dzwiek(9,false);
    dodatkowa_kulka = true;
    dodatkowa_kulka_gotowa = false;
    zapal_leda(16);
    zgas_leda(17);
  }
  if(ukonczone_cele==15) {
    dodatkowa_kulka_gotowa = true;
    zapal_leda(17);
  }
  zwieksz_bonus(1);
  if(ukonczone_cele%5==0)
    zwieksz_bonus(1);
  if(ukonczone_cele%3==0)
    zwieksz_bonus(1);
}

void zamien_aktywna_strone() {
  aktywna_lewa = !aktywna_lewa;
  Serial.println(aktywna_lewa);

  if(aktywna_lewa) {
    zapal_leda(39);
    zgas_leda(41);
  }
  else {
    zapal_leda(41);
    zgas_leda(39);
  }
}
```

STRESZCZENIE

Autor: Tadeusz Lorkowski

Promotor: dr hab. inż. Przemysław Ptak, prof. UMG

Tytuł pracy: Projekt i konstrukcja automatu flipper w oparciu o układ Arduino Mega 2560

Praca pt. „Projekt i konstrukcja automatu flipper w oparciu o układ Arduino Mega 2560” opisuje problem wykonalności zastosowania popularnego mikrokontrolera ATmega2560 w zręcznościowym, nieopartym na grafice, automacie do gier. Poruszane są w niej zagadnienia związane z elektroniką, praktyczną techniką mikroprocesorową i programowaniem. W pierwszym rodziale pracy opisuję zasadę działania wykorzystanych cyfrowych podzespołów – płyty rozwojowej Arduino Mega 2560 ze wspomnianym mikrokontrolerem, układu DSP umożliwiającego odtwarzanie plików MP3 z popularnego nośnika – karty microSD o nazwie DFPlayer, wreszcie sprawdzonego sterownika wyświetlacza LCD – HD44780, wraz z jego zastosowaniem układowym w najpopularniejszym podświetlanym wyświetlaczu. Dla każdego z tych trzech układów opisano ich topologię, zasady komunikacji i zastosowania poszczególnych wyprowadzeń, a dla DFPlayera i wyświetlacza dodatkowo metodę łączenia ich z Arduino, zarówno od strony sprzętowej, jak i programowej. Drugi rodział pracy dotyczy innych elementów, niebędących bezpośrednimi podmiotami programowania. Zaliczają się do nich urządzenia bezpośrednio kształtujące zasady i doświadczenia z gry, czyli elementy optoelektroniczne będące wskaźnikami postępów gracza – wyświetlacz LCD i diody LED, a także elektromagnesy, nadające kulce ruch. W obu przypadkach uwzględniono analizę fizycznych zasad działania, zastosowania układowe oraz metodę ich programowania z pomocą Arduino. W zastosowaniach układowych uwzględnione zostały obwody diod LED z doborem odpowiedniego napięcia zasilania i rezystora oraz sterowniki elektromagnesów, oparte o tranzystory Darlingtona mocy, projektowane w oparciu o wartości maksymalnego prądu celem ochrony Arduino przed uszkodzeniem. Trzeci rodział natomiast opisuje działający prototyp układu, w którym zastosowano elementy opisane w pierwszych dwóch rodziałach pracy. W ten sposób udowadniam wykonalność automatu flipper z użyciem mikrokontrolera ATmega 2560. W tej części zaproponowano zestaw zasad gry na takim automacie oraz przykładową jego realizację. Opisane zostały także sprawdzone mechanizmy urządzeń wykorzystywanych w rzeczywistych automatach, które wykorzystują solenoidy i mogą współpracować z opisanymi w drugim rozdziale sterownikami. Propozycja zawiera także podstawowe wzorce do wykorzystania w oprogramowaniu dla automatów flipper. Produktem końcowym pracy nad tym tematem jest działający prototyp układu ze schematem umożliwiającym jego replikację oraz kod aplikacji nim sterującej, napisany w obowiązującym w Arduino IDE dialekcie języka C.

Słowa kluczowe:

Arduino, ATmega2560, AVR, DFPlayer, HD44780, driver, język programowania C, flipper, pinball, automat do gier, gra elektroniczna

ABSTRACT

Author: Tadeusz Lorkowski

Supervisor: DSc, PhD Eng. Przemysław Ptak, Prof. GMU

Thesis topic: Project and develop automatic flipper game by Arduino Mega 2560

The thesis "Project and develop automatic flipper game by Arduino Mega 2560" describes the problem of the feasibility of using the popular ATmega2560 microcontroller in a non-video arcade game machine. In the thesis, I discuss issues related to electronics, practical microprocessor technology and programming. In the first chapter of the work, I describe the principles of operation of the used digital components - the Arduino Mega 2560 development board with the aforementioned microcontroller, a digital signal procesor DFPlayer used to play MP3 files from a popular storage medium – the microSD card, and finally a proven LCD display driver – HD44780, along with its system application in the most popular backlit display. For each of these three systems, I describe their topology, ways of communication and the applications of their individual pins, and for DFPlayer and the display, I also describe the methods of communicating them with Arduino, both from the hardware and software side. The second chapter of my thesis concerns other elements that are not subject to direct programming. These include devices that directly shape the rules and experience of the game, i.e. optoelectronic elements that indicate the player's progress - an LCD display and LED diodes, as well as electromagnets that give motion to the ball. In both cases, I include an analysis of the physical principles of operation, their system applications and the method of their programming using Arduino. These system applications include LED circuits with the selection of the appropriate supply voltage and resistor, as well as electromagnet drivers based on power Darlington transistors, designed based on the maximum current values to protect the Arduino against damage. The third chapter describes a working prototype of the circuit in which the elements described in the first two chapters of the work were used. In this way, I prove the feasibility of a pinball machine using the ATmega 2560 microcontroller. In this part, I propose a set of rules for playing such a machine and an example of its implementation. Verified mechanisms of devices used in real automatic machines that use solenoids and can cooperate with the controllers described in the second chapter are also described. The proposal also includes several basic patterns to be used in software for pinball machines. The final product of work on this topic is a working prototype of the system with a circuit diagram enabling its replication and the code of the application controlling it, written in the Arduino IDE’s dialect of the C language.

Keywords:

Arduino, ATmega2560, AVR, DFPlayer, HD44780, driver, C programming language, flipper, pinball, arcade game machine, electronic game