
	Instytut Informatyki Politechniki Śląskiej Zespół Mikroinformatyki i Teorii Automatów Cyfrowych			
Rok akademicki:	Rodzaj studiów*: SSI/NSI/NSM	Przedmiot (Języki Asemblerowe/SMiW):	Grupa	Seksja
2020/2021	SSI	SMiW	4	2
Imię:	Mateusz	Prowadzący: OA/JP/KT/GD/BSz/GB	JP	
Nazwisko:	Kost			
<h2><i>Raport końcowy</i></h2>				
Temat projektu: <div style="text-align: center; font-size: 24px; margin-top: 40px;"> 8 kanałowy dzwonek do quizu </div>				
Data oddania: dd/mm/rrrr		13.02.2021r		

1. Temat projektu i opis założeń.

Projekt o nazwie „8 kanałowy dzwonek do quizu” posiadał poniższe założenia:

- informacja wciśniętym guziku (jego numer) będzie adekwatnie wyświetlana na wyświetlaczu
- w przypadku wciśnięcia dwóch guzików, minimalna różnica czasowa przy ich wciskaniu zostanie wychwycona przez program oraz informacja o szybciej wciśniętym guziku zostanie pokazana na wyświetlaczu
- po wychwyceniu wciśnięcia guzika, program wyśle sygnał do buzzera, który zadzwoni

Wersja finalna projektu spełnia wszystkie powyższe założenia, a do samego projektu dołożyłem jeszcze 3 diody sygnalizujące o adekwatnym stanie programu.

Funkcjonalność urządzenia opiera się na wciskaniu jednego z 8 guzików sterujących, a po wyświetleniu informacji na wyświetlaczu na temat wciśniętego guzika należy wcisnąć guzik reset, który wysła informacje do programu aby zresetował wyświetlacz i pozwolił na ponowną możliwość wciśnięcia guzika sterującego.

2. Analiza zadania i uzasadnienie wybranych elementów elektronicznych i narzędzi użytych do realizacji projektu.

a. Analiza.

Funkcjonowanie układu opiera się na wciśnięciu jednego z ośmiu guzików i adekwatnej do wciśniętego guzika informacji wyświetlanej na wyświetlaczu. W przypadku wciśnięcia więcej niż jednego guzika, minimalna różnica czasowa zostanie automatycznie wychwycona przez układ, a na wyświetlaczu pojawi się informacja o guziku, który został wciśnięty pierwszy oraz użytkownicy zostaną dodatkowo poinformowani o tym zajściu poprzez sygnał dźwiękowy.

b. dobór elementów.

- Wyświetlacz:

Jako, że w moim programie potrzebuje jedynie wyświetlanie cyfr od 1 do 8, to decyzja padła na 7 segmentowy wyświetlacz LED z wspólną anodą.

<https://kamami.pl/segmentowe/199349-wyswietlacz-led-7-segmentowy-1-cyfra-1270mm-zielony-jasny-wspolna-anoda.html>

- Buzzer:

W kolejnym punkcie musiałem dokonać wyboru spośród wielu brzęczków, podstawową różnicą jaka między nimi znalazłem jest to, że jeden posiada wbudowany generator, a drugi takowego nie ma. Jako że nie przewiduje zmiany dźwięku w moim projekcie, to decyduje się na wybranie buzzera z generatorem, dodatkowym jego atutem jest to, że nie wymaga on dodatkowego układu kontrolującego. Spośród wielu dostępnych na rynku brzęczków, wybrałem taki z większą głośnością.

<https://botland.com.pl/buzzery-generatory-dzwieku/786-buzzer-z-generatorem-5v-12mm-tht.html>

- Mikrokontroler:

Po dogłębnej analizie i wypisaniu na kartce podstawowych komponentów potrzebnych do wykonania projektu, stwierdziłem że potrzebuje około 32 wejść i/o. Na taką decyzję wpłynął wybór wyświetlacza, który wymaga ich 8 oraz jak sama nazwa wskazuje guzików dla każdej drużyny w ilości również 8, co daje nam już łącznie 16, do tego dochodzi przycisk resetujący układ jak i przycisk resetujący wynik na wyświetlaczu, do nich dochodzi buzzer, a także podstawowe elementy bez których układ by nie zadziałał takie jak np. generator kwarcowy.

Kolejnym z ważniejszych elementów w mikrokontrolerze jest pamięć, o następujących nazwach jak FLASH, EEPROM czy RAM (oprócz nich mikrokontrolery posiadają również szeregową jak i równoległą pamięć). Pierwsza z nich odpowiada za przechowywanie kodu na podstawie którego działa nasz układ. Po wstępnych oględzinach mogę stwierdzić, iż wielkość takiego programu nie jest oszałamiająca, więc 16kB czy też 32kB w zupełności wystarczy. Pamięć EEPROM, która również odpowiada za przechowywanie informacji, jest o wiele wolniejsza niż pamięć FLASH, jednakże może ona zostać wykorzystana na przechowanie informacji tymczasowych co w niektórych projektach może być przydatne, aczkolwiek w moim układzie taka rzecz nie jest niezbędna do pracy.

Pamięć RAM odpowiada za obliczenia jak i przechowywanie zmiennych, po porównaniu paru mikrokontrolerów i wstępnym opisie działania mojego projektu, stwierdziłem że takowa pamięć też nie musi być większych rozmiarów niż przykładowe 512B. Ostatnim ważnym elementem mikrokontrolera, który wziąłem pod uwagę podczas porównywania, było napięcie zasilania. Jako, że brzęczyk jak i wyświetlacz wymagają napięcia 5V, stwierdziłem że mikrokontroler mógłby wymagać podobnej wartości. Po analizie mikrokontrolerów z firm takich jak Atmel, Pic, czy też Atmega, stwierdziłem że każdy mógłby się nadać, ponieważ ich napięcie zasilania jest zbliżone (przynajmniej dla tych, które wziąłem pod lupę).

Podsumowując. Mikrokontrolery, które spełniają powyższe warunki, to:

- ATMEL AT89C51RC-24PU (FLASH 32kB, EEPROM brak, RAM 512B, napięcie: 4-5.5V, cena: 6-16zł)

<https://pl.farnell.com/microchip/at89c51rc-24pu/mcu-8bit-24mhz-dip-40/dp/3131811>

- PIC18F4550-I/P (FLASH 32kB, EEPROM 256B, RAM 2kB, napięcie: 2-5,5V, cena: ~36zł)

<https://kamami.pl/mikrokontrolery-pic/187723-pic18f4550-ip.html>

- ATMEGA 16A-PU (FLASH 16kB, EEPROM 512B, RAM 1kB, napięcie: 2,7-5,5V, cena: ~16zł)

<https://kamami.pl/mikrokontrolery-avr/207957-atmega16a-pu.html>

- ATMEGA 32A-PU (FLASH 32kB, EEPROM 1kB, RAM 2kB, napięcie: 2,7-5,5V, cena: ~18zł)

<https://kamami.pl/mikrokontrolery-avr/208962-atmega32a-pu.html>

Po głębszej analizie ceny, jak i dostępności powyższych mikrokontrolerów moją uwagę głównie skupiły mikrokontroler firmy ATMEL, jak i o 32kB pamięci FLASH mikrokontroler firmy ATMEGA. Jedną z głównych różnic między nimi, jest większy zakres napięcia wejściowego, który u ATMEGI jest dużym atutem oraz większy RAM, czy też posiadanie pamięci EEPROM, stąd wybór właśnie pada na nią.

- Zasilanie

Jednym z głównych zamysłów na działanie układu było to, by był on przenośny, co rozwiązuje zasilanie bateryjne. Po przeszukaniu internetu w celu odnalezienia zasilania, które będzie odpowiednie dla układu, wpadłem na dwa pomysły.

- Zasilanie bateryjne (bateria 9V) z stabilizatorem napięcia

https://botland.com.pl/pl/regulatory-napiecia/81-stabilizator-5v-l7805abv-tt220.html?search_query=stabilizator+napiecia&results=83 cena: 1-10zł za stabilizator + baterie

- Zasilanie poprzez baterie AA z przetwornicą step-down

<https://botland.com.pl/przetwornice-step-down/3019-d24v10f5-przetwornica-step-down-5v-1a-pololu-2831.html> cena: 30zł + baterie

Po dokładniejszym przeanalizowaniu obu możliwości, pierwsza z nich okazała się o wiele mniej opłacalna pod względem długości działania, ponieważ jej pojemność to tylko około 500mAh. Natomiast bateria AA ma takową o wiele większą i wynosi ona ponad 2000mAh (według pierwszej informacji zwykła bateria AA ma przeciętną pojemność 2500mAh ~2018r.). Zatem wybrana przeze mnie opcja, to opcja numer 2.

- Diody

Do projektu zdecydowałem dodać się 3 dodatkowe diody, które informują użytkownika o aktualnym stanie programu. Decyzja była prosta wybrałem diody, które były tanie i dosyć dobrze świecą.

<https://botland.com.pl/diody-led/11144-dioda-led-5mm-zielona-10szt.html>

<https://botland.com.pl/diody-led/13606-dioda-led-5mm-zolta-10szt.html>

<https://botland.com.pl/diody-led/11440-dioda-led-5mm-czerwona-10szt.html>

c. Narzędzia:

Do wykonania projektu potrzebowałem miernika uniwersalnego, aby badać napięcie wychodzące z przetwornicy by ubezpieczyć się przed spalaniem elementów.

<https://botland.com.pl/mierniki-uniwersalne/4608-miernik-uniwersalny-dt830b-5906881174487.html>

Do projektowania na płytce prototypowej nie potrzebowałem żadnych dodatkowych urządzeń, dopiero do samego lutowania zdecydowałem się zakupić potrzebną do lutowania lutownicę oraz kilka elementów przydatnych do tejże pracy. Takich jak cyna, czy też odsysacz do cyny.

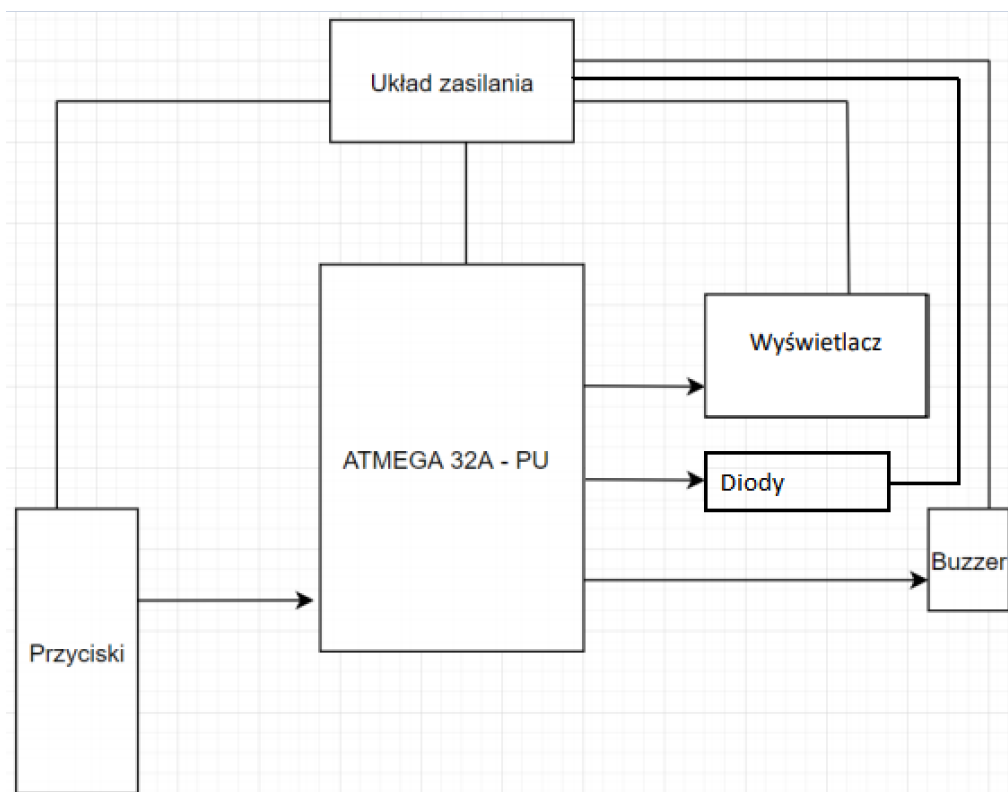
-lutownica: <https://allegro.pl/oferta/lutownica-kolbowa-precyzyjna-60w-450c-230v-groty-9819515684?snapshot=MjAyMS0wMS0yNVQwOT00MzoyMy45ODRaO2J1eWVvOzVIYmMxYmQ0YTRmOWI0NDJkMGI1ODU5NzhjZTQyODE5MThhNDAxYzAzMThiYmMwOGEzNWJkZGEwMDIzYmY0NTg%3D>

-cyna: <https://botland.com.pl/cyny-i-pasty-lutownicze/834-cyna-cynel-lc60-100g-056mm-5902884600060.html>

-odsysacz: <https://allegro.pl/oferta/odsysacz-lutowniczy-do-nadmiaru-cyny-zd-190-9833880982?snapshot=MjAyMS0wMS0yNVQwOT00MzoyNC4wMTBaO2J1eWVvOzY3YjE1MTFlMGU1NWMyMWE1ZDIiNjMyZTVmODU0ZTBhNDBjNTBINzlkZDVjMGVhZmI3NjFhZmJkOWMzMjIhY2U%3D>

3. Specyfikacja wewnętrzna urządzenia.

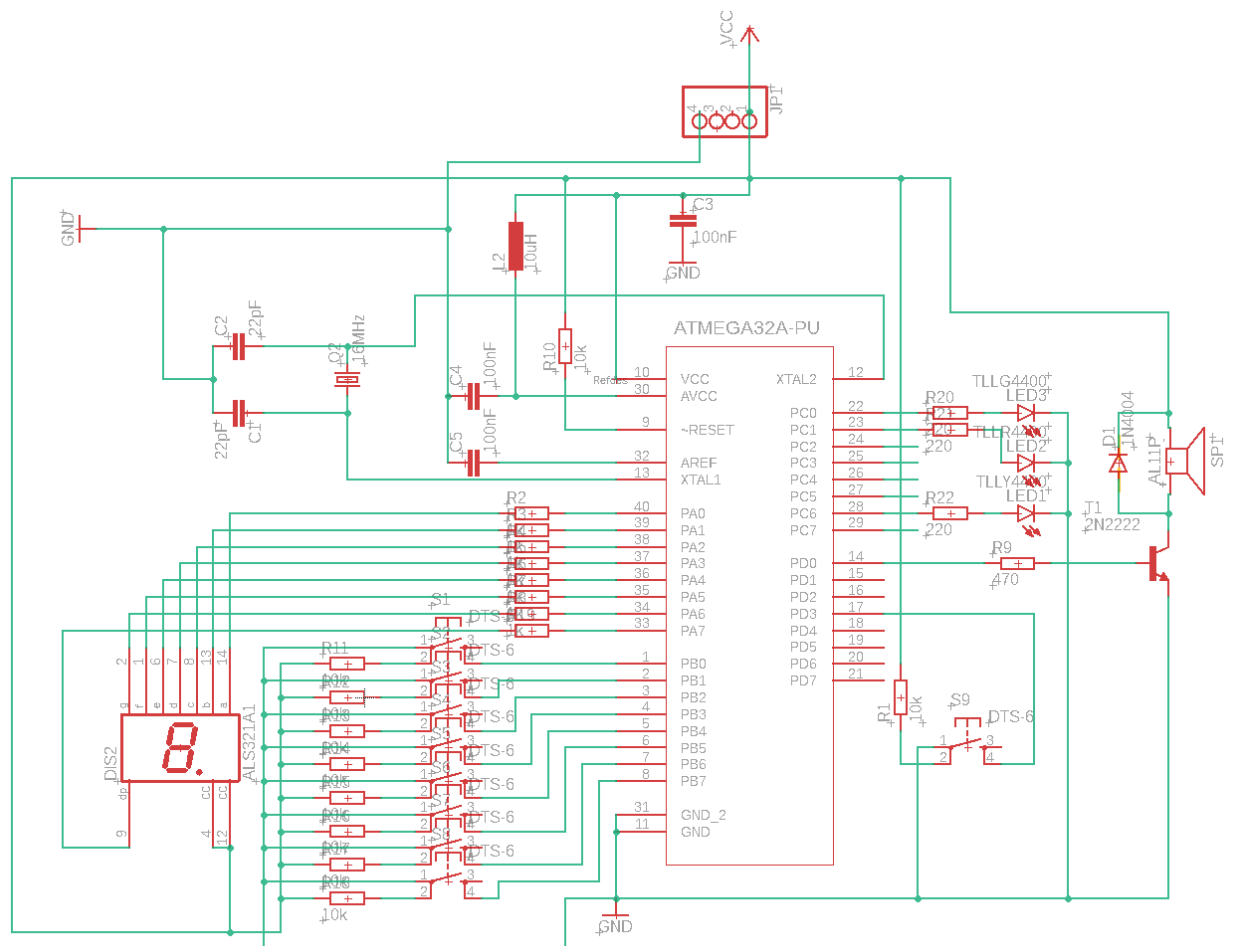
a. schemat blokowy i ideowy:



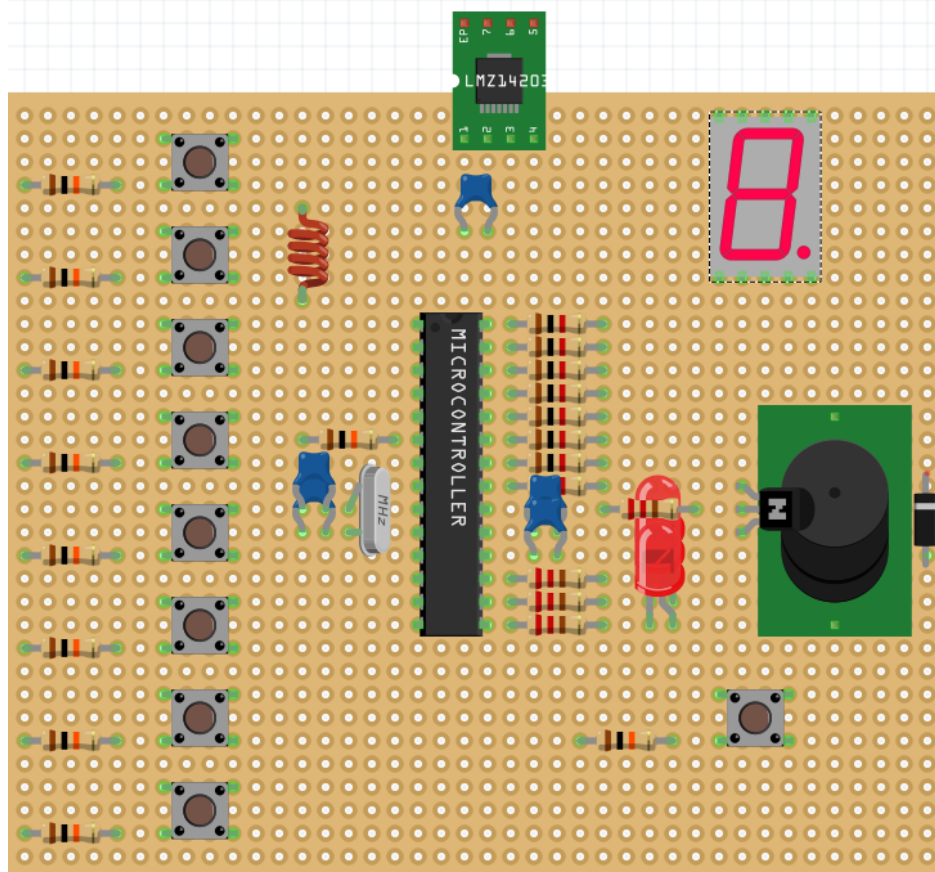
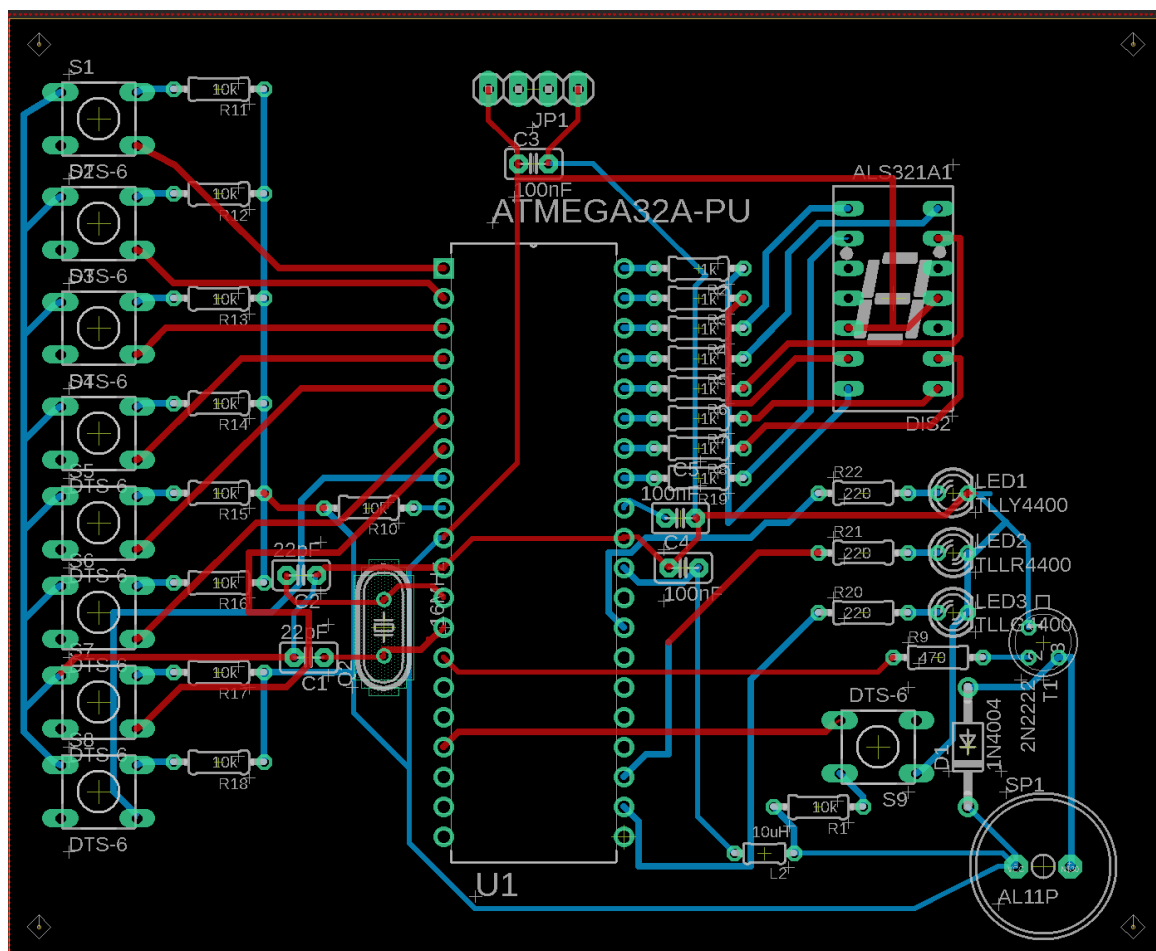
Na schemacie widocznych jest 6 bloków, które pełnią poniższe funkcje:

- Układ zasilania – jest to przetwornica która zmienia napięcie wejściowe na 5V, a następnie doprowadza je do kolejnych bloków.
- Atmega 32A-PU – mikrokontroler, jest to układ sterujący urządzeniem, odciera sygnał dochodzący z guzików, wyświetla informacje na wyświetlaczu, włącza diody w zależności od etapu działania programu, jak i operuje buzzerem.
- Wyświetlacz – jest to 7 segmentowy wyświetlacz LED, na który pokazywana jest informacja adekwatna do wciśniętego guzika.
- Przyciski – jest to blok, w którym znajdują się przyciski odpowiedzialne za pracę układu, są one głównym źródłem naszej gry, po wciśnięciu jednego z nich, wysyłana jest informacja do mikrokontrolera, który ją przetwarza i działa dalej, wyświetlając wynik, zaświecając diodę i włączając buzzer.
- Buzzer – jest to dzwonek, który zostaje włączany po wykryciu wciśnięcia guzika.

Schemat Ideowy



b. Schemat montażowy



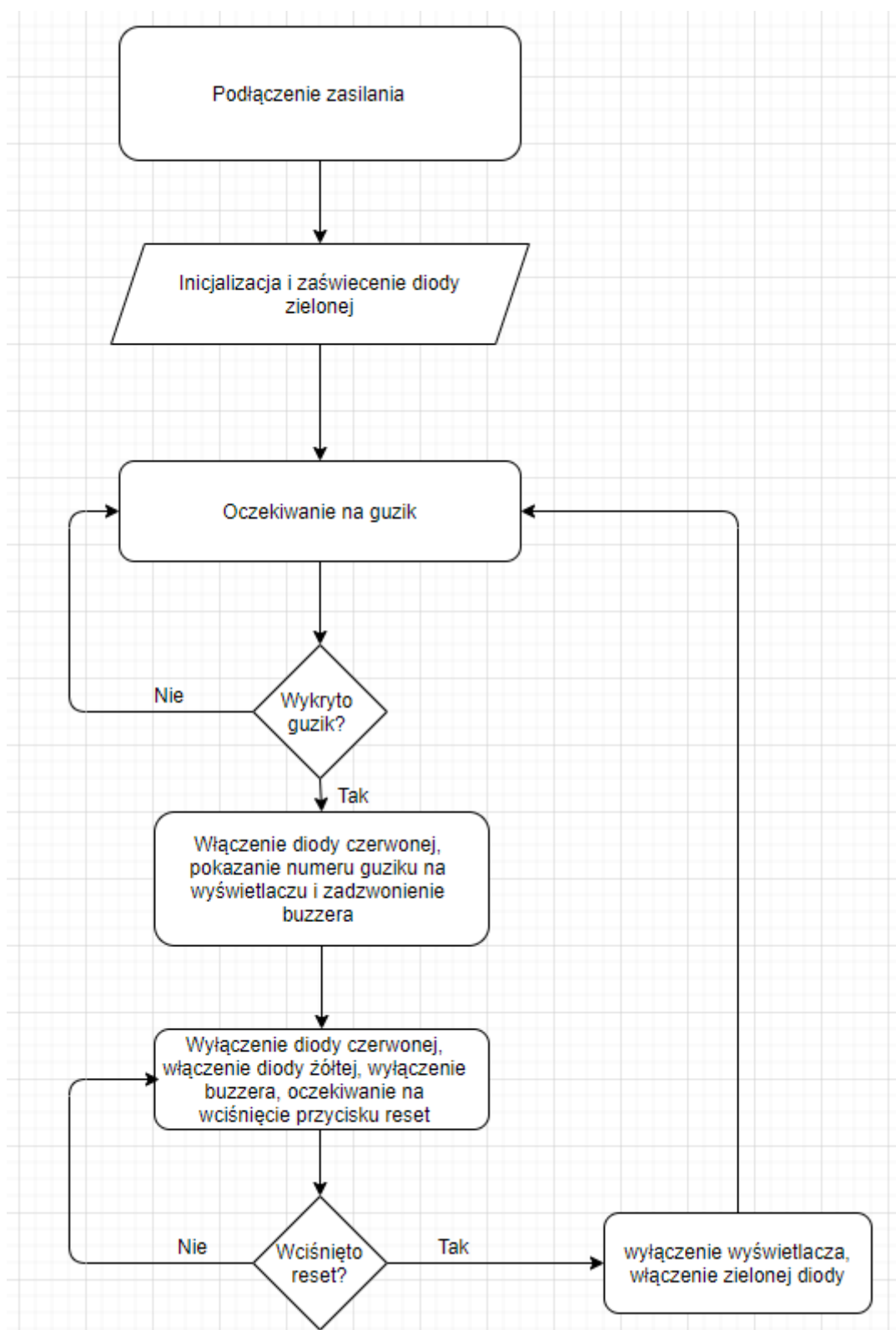
rozmieszczenie na płytce uniwersalnej

c. Lista elementów:

- Mikrokontroler Atmega 32A-PU
<https://kamami.pl/mikrokontrolery-avr/208962-atmega32a-pu.html>
- Wyświetlacz 7-segmentowy LED (zielony)
<https://kamami.pl/segmentowe/199349-wyswietlacz-led-7-segmentowy-1-cyfra-1270mm-zielony-jasny-wspolna-anoda.html>
- Buzzer
<https://botland.com.pl/buzzery-generatory-dzwieku/786-buzzer-z-generatorem-5v-12mm-tht.html>
- Przetwornica step down
<https://botland.com.pl/przetwornice-step-down/3019-d24v10f5-przetwornica-step-down-5v-1a-pololu-2831.html>
- Diody LED koloru zielonego, czerwonego i żółtego
- Rezonator kwarcowy 16Mhz
- Tranzystor bipolarny NPN 2N3904
- Dioda prostownicza 1N4004
- Rezystor 220 x3
- Rezystor 1k x 8
- Rezystor 10k x 10
- Rezystor 470
- Kondensator ceramiczny 100nF x 3
- Kondensator ceramiczny 22pF x 2
- Dławik 10uH
- Podstawka DIP-40 do mikrokontrolera
- tact switch x 9

Atmega 32A-PU	17,34
Wyświetlacz 7-segm Anoda	1,75
Buzzer	0,9
Przetwornica step-down	29,9
Dioda LED czerwona	1,5
Dioda LED zielona	1,5
Dioda LED żółta	1,5
Rezonator kwarcowy	0,95
tranzystor bipolarny npn	0,99
Dioda prostownica 1M4004	1,2
Rezystory	16,9
Kondensatory	6,21
Dławik	2
Podstawka dip 40	2,5
Płyta uniwersalna	13,42
Programator AVR	25
Płytki stykowe	29,2
Przewody połączeniowe	17,4
Lutownica + cyna	59,9
SUMA	230,06

d. schemat blokowy algorytmu



d*. Algorytm oprogramowania.

```
#include <xc.h>
#include <avr/io.h>
#include <util/delay.h>

void dzwon()
{
    PORTD=0b00000001; // ustaw stan wysoki
    _delay_ms(1000); // czekanie 1 sekundy
}

void reset()
{
    PORTC=0b00000010;

    PORTD=0b00000000; // ustaw stan niski
    while(bit_is_set(PIND, 3));
    PORTA=0xff;
}

int main(void)
{
    DDRB=0b00000000; // ustawienia pinow w porcie B jako wejsc dla guzikow
    DDRA=0b11111111; //piny wyjsc dla wyswietlacza
    DDRD=0b00000001; // ustawia port D0 jako wyjscie dla buzera
    PORTA=0xff; // ustawianie stanów wysokich na wyswietlaczu zeby nie bylo nic widac,
    wyswietlacz dziala na stanach niskich
    DDRC=0b01000011;

    while(1)
    {
        PORTC=0b00000001;

        while(PINB == 0b11111111);

        PORTC=0b01000000;

        if(bit_is_clear(PINB, 0))
        {
            PORTA=0b01111001;
            dzwon();
            reset();
        }

        if(bit_is_clear(PINB, 1))
        {
            PORTA=0b00100100;
            dzwon();
            reset();
        }

        if(bit_is_clear(PINB, 2))
        {
            PORTA=0b00110000;
            dzwon();
            reset();
        }

        if(bit_is_clear(PINB, 3))
        {
            PORTA=0b00011001;
            dzwon();
        }
    }
}
```

```

        reset();
    }

    if(bit_is_clear(PINB, 4))
    {
        PORTA=0b00010010;
        dzwon();
        reset();
    }

    if(bit_is_clear(PINB, 5))
    {
        PORTA=0b00000010;
        dzwon();
        reset();
    }

    if(bit_is_clear(PINB, 6))
    {
        PORTA=0b01111000;
        dzwon();
        reset();
    }

    if(bit_is_clear(PINB, 7))
    {
        PORTA=0b00000000;
        dzwon();
        reset();
    }
}
return 0;
}

```

e. Opis funkcji:

- void dzwon() – funkcja odpowiedzialna za włączenie buzzera na dokładnie sekundę oraz włącza diodę czerwoną sygnalizującą działanie buzzera.
- void reset() – funkcja, która wyłącza diodę czerwoną, włącza żółtą i oczekuje na wciśnięcie guzika reset, gdzie po wciśnięciu wyświetlacz zostaje wyłączony.
- int main(void) – główna funkcja programu, która na początku ustawia wejścia i wyjścia atmegi, zaświeca diodę zieloną sygnalizującą możliwość wciśnięcia guzików odpowiedzialnych za dalszą pracę programu. Następnie posiadamy pętlę while, w której po wejściu program oczekuje na wciśnięcie jednego z guzików sterujących. Jeżeli jeden z nich zostanie wykryty, program przechodzi do dalszej części, w której odczytuje jaki guzik został wciśnięty, po wykryciu wyświetla na wyświetlaczu adekwatny do guzika numer i wywołuje funkcję dzwoń, a następnie reset. Po wykryciu guzika resetującego wyświetlacz, program wraca na początek pętli, gdzie wyłącza diodę żółtą, włącza zieloną i czeka na ponowne wciśnięcie guzika.

Funkcje bibliotek:

- bit_is_clear() – funkcja która sprawdza czy wartość wejściowa określonego pinu jest równa 0 (stan niski), jako argumenty przyjmuje, które to jest wejście, jak i jego numer
- bit_is_set() - analogicznie jak poprzednia funkcja, jedyna różnica polega na tym, że ta czeka aż wartość wejściowa będzie równa 1 (stan wysoki).
- _delay_ms() - funkcja która zatrzymuje program na określonej długości czasu wyrażoną w milisekundach, po czym wywołuje kolejne instrukcje.

4. Specyfikacja zewnętrzna urządzenia.

a. Funkcje elementów sterujących urządzeniem.

Jedynym elementem sterującym istniejącym w moim projekcie jest Atmega 32A-PU, która jest odpowiedzialna za odbieranie sygnałów z guzików oraz za wysyłanie odpowiednich informacji do wyświetlacza. Zapala ona również diody, w zależności od momentu wykonywania programu.

b. Funkcje elementów wykonawczych.

- Wyświetlacz, jest on odpowiedzialny za pokazanie, który z guzików został wciśnięty najszybciej.
- Czerwona dioda LED - informuje nas o tym, że guzik został wciśnięty.
- Żółta dioda LED - informuje nas, że program czeka na wciśnięcie guzika reset, który wyłącza wyświetlacz.
- Zielona dioda LED - informuje nas o stanie gotowości programu, który oczekuje wciśnięcia jednego z guzików odpowiedzialnego za sterowanie.
- Buzzer – sygnałem głosowym informuje nas o wciśnięciu guzika.

c. Reakcje oprogramowania na zdarzenia zewnętrzne.

Zadaniem oprogramowania jest wychwycenie wciśnięcia jednego z ośmiu guzików sterujących, po jego wykryciu na wyświetlaczu pojawia się adekwatna do guzika cyfra (np. guzik nr 1 został wciśnięty, na wyświetlaczu pojawia się cyfra '1'), dzwoni buzzer i zaświeca się dioda czerwona pokazująca, że program działa.

d. Instrukcja obsługi.

- Położyć urządzenie na stabilnym miejscu.
- Podłączyć do źródła baterie, o napięciu wyższym niż 5V, dioda zielona powinna się załączyć, sygnalizuje ona oczekiwanie mikrokontrolera na wciśnięcie guzika.
- Wciśnij jeden, bądź więcej z ośmiu guzików sterujących, program wychwyci najszybciej wciśnięty z nich. Na wyświetlaczu pojawi się adekwatna cyfra do wciśniętego guzika, zadzwoni buzzer, zielona dioda zgaśnie i zaświeci się czerwona, która sygnalizuje pracę programu.
- Po zmienienu się zaświeconej diody z czerwonej na żółtą i wyłączeniu buzzera wciśnij guzik reset, spowoduje on wyłączenie wyświetlacza i wyłączenie diody żółtej i oświecenie diody zielonej, która pozwala Ci na ponowne skorzystanie z urządzenia.
- W celu zresetowania urządzenia odłącz i podłącz ponownie zasilanie.
- W celu wyłączenia urządzenia odłącz i podłącz ponownie zasilanie.

e. Opis złączy.

Urządzenie posiada jeden zestaw zewnętrznych złączy i są to złącza odpowiednie za programowanie Atmegi. Jest to dokładnie 6 koniecznych złączy, bez których nie zaprogramujemy naszego układu. Poszczególne z nich to MISO, MOSI, SCK, RST, a GND i VCC są podłączone do ścieżek zasilania.

Oprócz wyżej wymienionych złączy układ używa pinów PB0-PB7 odpowiedzialnych za wejścia guzików, PA0-PA7 które sterują wyświetlaczem, PC0, PC1, PC6 – diody, PD0 – buzzer, PD3 – guzik reset

f. Opis montażu układu.

Układ w całości został polutowany w technologii THT, wszystkie jego elementy zostały połączone na spodzie dwuwarstwowej płytki uniwersalnej, a ich rozłożenie można zobaczyć w rozdziale 3.b.

Polutowany układ został umieszczony na krawędziach pudełka, tak aby kable łączące ścieżki nie zostały naruszone.

g. Opis programowania układu.

Do zaprogramowania urządzenia został wykorzystany programator USBasp oraz oprogramowanie Microchip Studio wraz z AVRDUDE. Po zainstalowaniu potrzebnych sterowników do pracy na mikrokontrolerze można było przejść do samego programowania.

Układ został przełączony na pracę z 16MHz rezonatorem kwarcowym. Początkowo nie obyło się bez problemów, microchip studio nie chciał odczytać mojej Atmegi, lecz po znalezieniu pomocy, jak i dodatkowych informacji, problem ten został zażegnany.

W pierwszej fazie programu zostały przetestowane każde części programu z osobna, aby sprawdzić czy każda z nich jest w stu procentowo sprawna, dopiero po tym można było się zabrać za programowanie wersji finalnej.

h. Uruchamianie i testowanie.

Początkowo urządzenie było uruchamiane na płytce stykowej. Na pierwszy ogień poszło testowanie działania buzzera, początkowo dzwonek ten dzwonił strasznie cicho, problemem okazała się odwrócenie wpięta dioda prostownicza, gdy wpiąłem ją na odwrót, buzzer zagrał poprawnie. Kolejnym z elementów do testowania był wyświetlacz segmentowy, początkowo podpięty do pinów PC0-PC7, jednakże działały tylko 4 z 8 pinów co sprawiło, że zmieniłem je na piny PA0-PA7, gdzie już wszystko ładnie działało. Po przetestowaniu działania wyświetlacza jak i buzzera, zdecydowałem się na podłączenie przycisków. Na szczęście każdy z nich działał i mogłem zacząć programować.

Pierwszym z napotkanych problemów było wyświetlanie konkretnej liczby na wyświetlaczu, dopiero po paru próbach zorientowałem się, że wystarczy wysłać sygnał stanu niskiego do wyświetlacza aby działać na konkretnym pinie, po tym już było z górki.

Do samej obsługi urządzenia dodałem 3 diody sygnalizujące o aktualnym stanie układu, aby pomóc użytkownikowi w trakcie jego korzystania.

Testowanie po lutowaniu miało już charakter sprawdzenia, czy wszystko zostało przeze mnie dobrze polutowane, na początku nie dzwonił buzzer, problemem okazał się nie przylutowanie kabel dający mu zasilanie. Po sprawdzeniu każdej części, przystąpiłem do testowania wciskając kilka guzików naraz, program odczytywał poprawnie szybciej wciśnięty guzik i informował użytkownika o jego numerze.

5. Wnioski i uwagi z przebiegu pracy.

a. Problemy podczas montażu

Początkowo prace związane z układem nie skończyły się sukcesem. Pierwszym napotkanym przeze mnie problemem była cicha praca buzzera, jak się okazało wynikała ona z błędnie podpiętej diody prostowniczej, przypadkowo wpiąłem ją na odwrót. Kilka godzin zajęło mi dojście do tego błędu, sprawdziłem pozostałe buzzery i każdy z nich działał identycznie, dopiero po przybliżeniu oczu do układu zauważyłem swój błąd, po zmianie wpięcia buzzer działał poprawnie.

Kolejnym napotkanym przeze mnie problemem okazało się wyświetlanie na wyświetlaczu, początkowo podpiąłem go do PC0-PC7, lecz tylko 4 z nich działały. Początkowo myślałem, że to wina wyświetlacza, dopiero po przeniesieniu się z portu C na port A, zauważyłem że był to problem portu, może wina fabryczna Atmegi ? Nie jestem specjalistą, więc się nie dowiedziałem, a jako że wyświetlacz zaczął funkcjonować poprawnie, to się tym już nie przejmowałem.

Kolejnym z większych problemów było podpięcie guzików, na schemacie umieściłem początkowo 2 pinowe guziki, a sam zamówiłem 4, co lekko mnie zmyliło. Nie przywiązałem do tego większej uwagi i próbowałem działać tak samo jak na 2 pinowych, lecz nie zaowocowało to poprawnym działaniem, dopiero po zmianieniu podpięcia guzików zaczęły one działać poprawnie.

Jednym z większych problemów podczas działania na płytce prototypowej były kable łączące ze sobą elementy, jedno małe ruszenie czy nie dopięcie którejś z jego końcówek powodowało słabe działanie programu, bądź jego nie działanie, co wymusiło na mnie wielką ostrożność.

Mając już sprawny i przetestowany układ na płytce prototypowej zabrałem się za lutowanie, jako że było to moje pierwsze spotkanie z lutowaniem, nie szło mi to jakoś wybitnie. Prowadzenie ścieżek, czy też samo montowanie każdej z części nie była taka prosta, lecz po zamontowaniu większości już mi to jakoś szło. Końcowo cały etap lutowania zajął mi około 5 godzin. Ku mojemu zdziwieniu, cały zalutowany układ działał prawie poprawnie, jedyny błąd jaki zrobiłem, a raczej pomyłkę, to to że przypadkowo nie przylutowałem kabla z zasilania do buzzera, co powodowało brak dźwięku, po szybkiej naprawie, mogłem śmiało przetestować cały zalutowany układ, który działał bez żadnych zarzutów.

b. Testy poprawności.

W celu ułatwienia poprawności działania mojego układu, domontowałem 3 diody mówiące mi o aktualnym momencie działania programu, ponieważ patrząc na kolor włączonej diody od razu byłem pewien w czym jest problem.

Jednym z błędów jakie napotkałem było zapętlenie się na wciśnięciu guzika reset, a raczej jego nie wychwycenie, mogłem się o tym przekonać dzięki ciągle włączonej diodzie żółtej, która sygnalizowała użytkownikowi, aby wcisnął guzik reset. Błędem okazało się nie dociśnięcie kabla na płytce prototypowej, po dociśnięciu układ działał poprawnie.

Dzięki diodom, uprzedziłem się też przed zbyt częstym wciskaniem przycisków odpowiadających za sterowanie układem. Program wychwytywał je tylko, gdy zielona dioda się świeciła w przeciwnym przypadku nie odczytywał z nich informacji.

6. Wnioski końcowe.

Jedną z większych rzeczy jakie wyniosłem z realizacji tego projektu, była nauka lutowania, która na pewno przyda mi się w życiu, a nie mogę powiedzieć że mi się to nie podobało, sama taka praca fizyczna dzięki której można szybko zauważyć efekty była bardzo satysfakcjonująca.

Co do samego układu, dopiero przy praktycznej części wykonywania tego projektu przekonałem się jak ważny jest dobór odpowiednich rezystorów czy też kondensatorów do poprawnego działania układu. Przekonałem się, że przy błędnie dobranym rezystorze szybko można spalić choćby diodę, czy też sam wyświetlacz .

Projekt przede wszystkim pozwolił mi na praktyczne wykorzystanie wiedzy, którą zyskałem w trakcie ćwiczeń czy też laboratorium. Do samego programowania bardzo przydała mi się wiedza z laboratorium z AVRów, gdzie dowiedziałem się o środowisku Microchip Studio oraz jak w nim działać, bardzo ułatwiło mi to pracę podczas programu.

Jedną z rzeczy której nie byłem pewien był projekt pcb mojego układu, jako że pierwszy raz pracowałem nad nim bardziej, nie byłem przekonany czy na pewno dobrze go wykonałem oraz czy nie dodam niczego do swojego układu, dlatego też zdecydowałem się na zalutowanie swojego układu na płytce uniwersalnej. Końcowo był to trafny wybór, ponieważ w trakcie praktycznej części programu wiele zmieniłem pod względem schematu ideowego.

Podsumowując realizację swojego projektu, mogę śmiało powiedzieć że jestem z niej naprawdę zadowolony. Programowanie układu nie było takie trudne jak mogło by się wydawać, jednakże moja znikoma wiedza na temat pracy przy takich urządzeniach mi to utrudniła, aczkolwiek szybko znalazłem potrzebne informacje i udało mi się zrealizować program. Jestem zdania, że nabyłem też wiedzę na temat montowania urządzeń elektronicznych i pracy z nimi, która może mi się przydać w przyszłości. Końcowo stwierdzam, że mój 8 kanałowy dzwonek do quizu działa tak jak zakładałem, a nawet lepiej chociażby ze względu na dodane diody ułatwiające pracę z urządzeniem.

7. Literatura.

Do realizacji projektu wykorzystałem poniższe materiały:

https://dl.btc.pl/kamami_wa/atmega32a.pdf

<https://majsterkowo.pl/5-prostych-projektow-avr-w-c-czesc-1/>

<https://atnel.pl/programator-atb-usbasp.html>

Exercise 7 AVR Microcontrollers – part I IDE & MEMORY & GPIO - mgr inż. Jarosław Paduch,

Exercise 9 AVR Microcontrollers – part II Timer/Counters & Interrupts - mgr inż. Jarosław Paduch

<http://edutute.blogspot.com/2013/03/7-segment-display-interfacing-with-8051.html>