POLITECHNIKA ŚLĄSKA W GLIWICACH
WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI
INSTYTUT AUTOMATYKI

PRACA DYPLOMOWA MAGISTERSKA

Wyświetlacz 3D

Jakub Trznadel
Promotor: dr inż. Jerzy Mościński

Za życzliwy stosunek, cenne uwagi i wskazówki,
wsparcie duchowe oraz pomoc przy opracowywaniu
niniejszej pracy składam serdeczne podziękowania

dr. inż. J. Mościńskiemu
mgr. inż. M. Filipczykowi
mgr. inż. R. Kapralskiemu

Spis treści

1 WSTĘP

Historia wykorzystywania bezwładności ludzkiego oka w celu stworzenia iluzji obrazu jest bardzo stara, sięga XVIII wieku. Wcześniej podobne urządzenie skonstruował T.A. Edison, niemniej jednak wynalazek kamery i kina przypisywany jest braciom Lumiere którzy w roku 1894 skonstruowali pierwszy kinematograf - aparat do realizacji i wyświetlania filmów.

Od tego czasu upłynął ponad wiek. Przez ten czas aparaty służące uzyskiwaniu fotorealistycznych iluzji były ciągle doskonalone. W latach 20-tych XX wieku do wizji został dodany dźwięk. W 1928 powstał pierwszy telewizor, początkowo czarno-biały. Obecne urządzenia do transmisji obrazu różnią się od pierwowzoru ilością możliwych kanałów (telewizja cyfrowa), medium transmisyjnym (satelity, internet), wygodą obsługi (pilot, interfejs OSD), sposobem kodowania informacji (algorytmy kompresji typu mpeg), rozdzielczością (HDTV). Jednak pomimo znaczącego postępu, obecne telewizory (monitory) funkcjonalnością są wciąż zbliżone do pierwowzoru - służą do przekazu dwuwymiarowego obrazu i dźwięku na odległość.

Człowiek jednak zapragnął czegoś więcej - obrazu trójwymiarowego. Można tu wymienić wiele różnych sposobów na osiągnięcie efektu trójwymiarowości:

- okulary filtrujące
  Wykorzystują normalny telewizor. Idea działania polega na tym, iż lewe oko jest przesłonięte filtrem przepuszczającym kolor czerwony, natomiast prawe oko - kolor zielony. Wskutek odpowiedniego przygotowania wyświetlanego obrazu możliwe jest osiągnięcie efektu 3D.

- okulary szybko migające
  Działają na bardzo podobnej zasadzie co poprzednie. Tu jednak soczewki na zmianę przepuszczają światło - raz lewa, raz prawa. Wskutek dużej szybkości migania (połowa częstotliwości wyświetlania obrazu na monitorze), synchronizacji z powrotem pionowym monitora oraz bardzo szybkimi zmianami obrazu na monitorze uzyskuje się żądany efekt.

- okulary 3D
  Wykorzystują dwa wyświetlacze, osobno dla lewego i prawego oka. Posiadają zwykle również czujniki położenia i pozwalają na pełne zanurzenie się w wirtualnej rzeczywistości. Efekt jest szczególnie niesamowity w przypadku grania w gry komputerowe.

- stereogramy
  Tu z kolei efekt 3D uzyskuje się poprzez skoncentrowanie wzroku nie na obserwowanym obrazie, ale poza nim. Wymaga to jednak osiągnięcia pewnej wprawy.

- wyświetlacze 3D
  Są to urządzenia wytwarzające obraz trójwymiarowy wewnątrz zamkniętej przestrzeni. Główną różnicą w porównaniu z poprzednimi rozwiązaniami jest to, iż na wyświetlany obraz można równocześnie patrzeć z różnych stron.

Celem niniejszej pracy jest stworzenie prototypowego wyświetlacza 3D. Jest to zupełnie nowe podejście do tematu obrazów trójwymiarowych - pierwsze urządzenia tego typu pojawiły się na rynku w tym roku, wprowadzone przez amerykańską firmę Actuality Systems. Docelowymi klientami firmy są głównie wojsko, porty lotnicze, firmy biotechnologiczne oraz firmy związane z przemysłem kosmicznym. Cena tego typu urządzeń jest zatem jak się łatwo domyślić - wysoka.

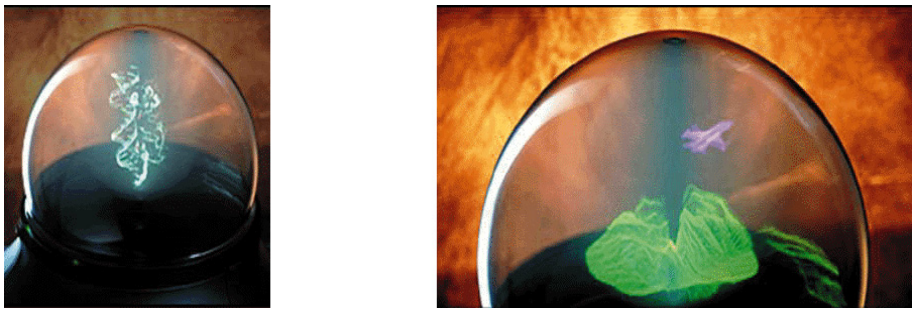
Rozdziały niniejszej pracy zostały poukładane chronologicznie, zgodnie z kolejnością tworzenia urządzenia. Pierwszy rozdział dotyczy **idei urządzenia**, obejmując źródła pomysłu oraz zakładany sposób realizacji. Następnie zamieszczono **Projekt** zawierający informacje na temat procesu projektowania urządzenia. Kolejnym rozdziałem jest **wykonanie**, w którym opisany został proces wykonywania wyświetlacza. **Opisy użytkowe** - kolejny rozdział - zawiera instrukcję obsługi wyświetlacza wraz z informacjami dla osoby pragnącej wykorzystywać wyświetlacz do swoich zastosowań. Pracę kończy **Podsumowanie**.

2 IDEA URZĄDZENIA

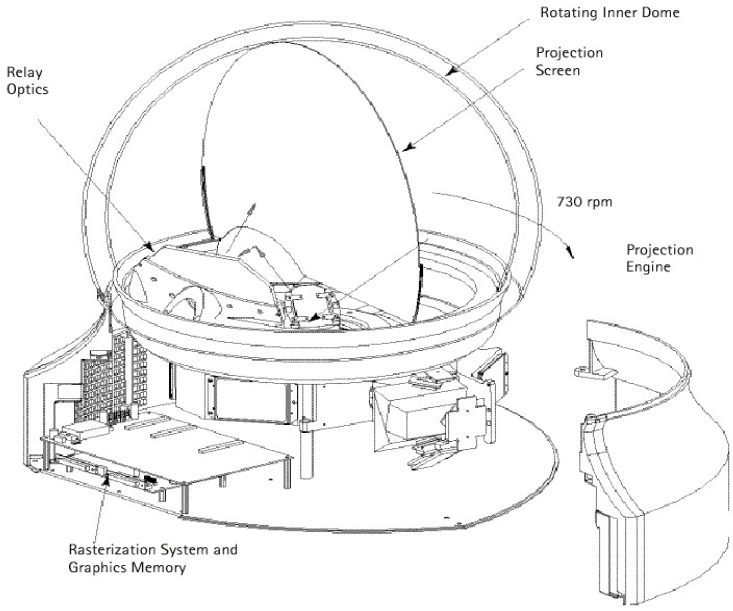
2.1 ŹRÓDŁA POMYSŁU

2.1.1 PERSPECTA 3D

W 1997 roku została założona firma Actuality Systems, której celem było stworzenie wyświetlacza 3D. W styczniu 2000 roku powstał pierwszy prototyp wyświetlacza 3D - umożliwiał on wyświetlanie obrazu 3D w rozdzielczości 64x64x64 [1] voxeli (odpowiednik piksela w przestrzeni 3D). W chwili obecnej firma ma w swojej ofercie wyświetlacz umożliwiający wyświetlanie obrazu o rozdzielczości 768x768x192 vokseli w ośmiu kolorach, przy częstotliwości odświeżania 24Hz. Jak łatwo obliczyć, obraz taki zajmować będzie 40.5MB.



Rys.1 Zdjęcia wyświetlacza firmy Actuality Systems



Rys.2 Schemat dostępny na stronie firmy Actuality Systems

[1] Współrzędne rozdzielczości podawane są w formacie: promień x wysokość x kąt

Zasada działania opiera się na wykorzystaniu szybko wirującej płaszczyzny, na którą nakierowany jest wirujący wraz z nią projektor. Przy prędkości obrotowej 24Hz człowiek przestaje widzieć płaszczyznę, zamiast niej widząc swoistą 'mgłę'. Dobrze widoczne są natomiast punkty podświetlane przez projektor.

Firma zakłada trzy główne grupy odbiorców - wojsko, przemysł biotechnologiczny, oraz lotniska, gdzie wizualizacja 3D może być użyta do kontroli bezpieczeństwa lotów. Urządzenie wręcz wydaje się stworzone do wizualizacji cząsteczek chemicznych. Na stronie producenta (http://www.actuality-systems.com/) można znaleźć więcej zdjęć urządzenia.

2.1.2 ZEGARKI DIODOWE

Są to rozmaite urządzenia tworzone przez hobbystów, które służą do wyświetlania obrazu zbliżonego do analogowego zegara. Efekt ten jest uzyskiwany poprzez umieszczenie paska diód (zwykle około sześciu) na szybko obracającej się płytce drukowanej. Gotowe zestawy można znaleźć w literaturze elektronicznej (archiwalne numery "Elektroniki dla wszystkich"). Synchronizacja położenia wykonywana jest raz na obrót z użyciem transoptora szczelinowego. Poprzez odpowiednie sterowanie zapalaniem/wyłączaniem diód podczas obrotu uzyskuje się bardzo ładny efekt przypominający zwykły zegarek.

2.2 ZAŁOŻENIA PROJEKTOWE

- **8 pasków po 8 diód LED** czerwonych, o obniżonym poborze prądu (2mA).

  Użycie większej liczby diód byłoby trudne ze względów praktycznych - już przy 64 były problemy z zaprojektowaniem płytki PCB wyświetlacza. Kolor czerwony użyty został ze względu na najlepszą sprawność źródła światła (przy zachowaniu sensownej ceny). Diody o obniżonym poborze prądu zostały użyte ze względu na pobór mocy - zapalone naraz pobierają około 64x2mA = 128mA. Normalne diody pobierałyby 64x10mA = 640mA.

- Wysokość diód regulowana za pomocą odpowiedniego przycięcia ich nóżek.

- Prędkość obrotowa około 25Hz (**1500 rpm**)

  25Hz jest często przyjmowane jako graniczna częstotliwość odświeżania obrazu przy której ludzkie oko ma wrażenie płynności (ciągłości) wyświetlania.

- Założona dokładność pomiaru położenia - **128 pozycji/obrót**

  Przy tej rozdzielczości zewnętrzne diody (leżące na promieniu ok. 45mm) pokonują drogę 2.2mm pomiędzy kolejnymi pozycjami. Jest to wartość zbliżona do rozdzielczości pionowej (3mm) i promieniowej [2] (3.6mm).

- Pamięć 1 strony obrazu = 8x8x128 b = 1kB

- Możliwość komunikacji z komputerem PC poprzez sprzęg **RS232**

  Sprzęg RS został wybrany z uwagi na prostotę jego używania przy stosunkowo dobrych parametrach transmisji. Ważną jego cechą jest też to że jest standardem.

[2] brak znanego określenia na tą rozdzielczość. Oznacza ona tutaj rozdzielczość związaną z odległością od osi wirowania.

- Odczyt położenia dokonywany jest tylko raz na obrót, z użyciem **transoptora szczelinowego**. Istniejące urządzenia typu 'zegarek diodowy' synchronizują wyświetlanie raz na obrót i jest to wystarczające.
- Zasilanie **+5/+12V**

  Dokładnie takie jakie można uzyskać z dowolnego zasilacza PC.
- Elektronika sterująca (z wyjątkiem MAX232 - konwertera stanów logicznych) wiruje razem z diodami.

  Inne rozwiązania wymagałyby doprowadzenia dużej liczby sygnałów do wirnika, co byłoby problematyczne w wykonaniu, oraz bardzo nietrwałe.
- Użycie mikrokontrolera z rodziny **PIC16F87xA** jako centralnej jednostki sterującej wyświetlaniem.

  Procesor ten został wybrany z uwagi na dużą liczbę portów I/O, obsługę sprzęgu RS. Również jego wydajność (przy taktowaniu 20MHz) wydawała się wystarczająca do sterowania wyświetlaniem.
- Użycie **pamięci SRAM** do przechowywania pamięci obrazu

  Użycie pamięci dynamicznej wymagałoby zastosowania układu odświeżającego pamięć, co dodatkowo zwiększyłoby masę wyświetlacza.
- Sprawdzenie możliwości uzyskania **stopni szarości** (czerwoności)
- **Zasilanie** (0, +5V) doprowadzone **na osi wirnika** (od dołu i od góry)

  Oś wirnika jest przerwana w środku tak że zwarcia nie ma. Z uwagi na potrzebę zachowania środka ciężkości w miarę blisko osi wirowania najcięższy układ - mikrokontroler PIC16F877A został umieszczony dokładnie w osi wirowania, i rozdziela on oś wirnika na 2 części.
- **Rx i Tx** doprowadzone do wyświetlacza **poprzez szczotki** (od dołu i od góry)

  Trudne było stwierdzenie jak będzie działać transmisja danych podczas wirowania wyświetlacza.

- Użycie **2 okrągłych płytek dwustronnych** o rozmiarach zbliżonych do standardowej płyty CD. Górna z diodami, dolna z logiką sterującą. Dodatkowa 3 płytka umieszczona poza wirnikiem, zawierająca konwerter stanów logicznych MAX232.

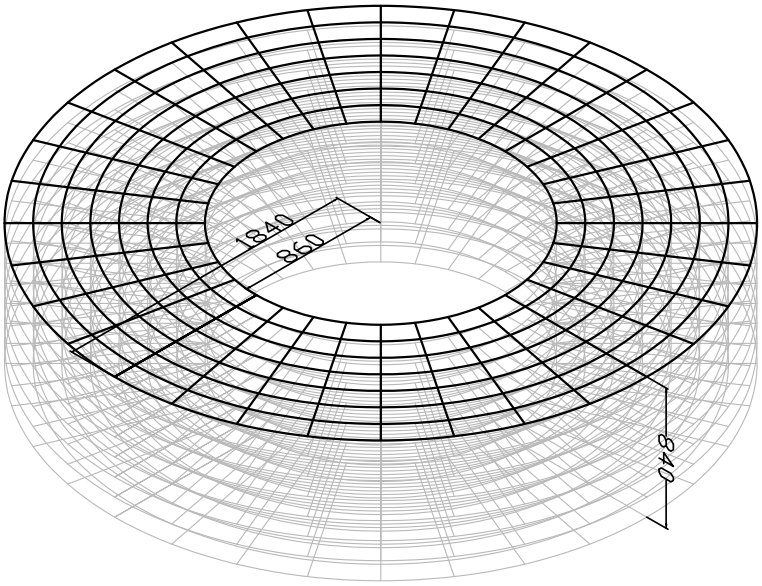
  Użycie pojedynczej wirującej płytki byłoby trudne z uwagi na problemy ze zmieszczeniem wszystkich potrzebnych układów na powierzchni wielkości płytki CD.

- Użycie **silnika magnetofonowego** jako napędu wirnika

  Po analizie dostępnych silników o prędkościach obrotowych rzędu 1500rpm (czyli 25 obrotów/sekundę) silnik magnetofonowy okazał się wystarczający. Wybrany został silnik charakteryzujący się prędkością obrotową rzędu 2200rpm (około 33 obroty/sekundę) i zasilaniu 12V. Zapas ten został wzięty dla bezpieczeństwa, a prędkość obrotowa może łatwo być zmniejszona poprzez obniżenie napięcia zasilającego np. w wyniku wpięcia szeregowo z silnikiem kilku diód (w kierunku przewodzenia).

2.3 UKŁAD WSPÓŁRZĘDNYCH

Wskutek odpowiedniego umieszczenia diód na górnej płytce wirnika, uzyskuje się (wycięty w środku) **cylindryczny układ współrzędnych**. Diody poruszają się po trajektoriach kołowych, przy czym wewnętrzne diody wirują po okręgu o promieniu 860mil [3] (ok. 21.8mm). Kolejne okręgi są umieszczone w odległości 140mil (ok. 3.6mm) od siebie. Zatem jak łatwo obliczyć (mamy 8 okręgów), zewnętrzny okrąg ma promień 1840mil (ok. 46.7mm). Pionowo diody są rozmieszczone co 3mm.



Rys.3 Układ współrzędnych

Rozdzielczość kątowa zakładana była początkowo na bezpiecznym poziomie 128 pikseli/kąt pełny (ok 2.81 stopnia). Niemniej jednak doświadczenia wykazały iż można ją śmiało zwiększyć do 256, a nawet 512 pikseli/kąt pełny. Przy tej ostatniej jednak diody LED nie nadążają z wyświetlaniem i pojedyncze voksele są słabo widoczne - pozwala to na próbowanie osiągnięcia stopni szarości (czerwoności) na wyświetlaczu. Częstotliwość pracy diód przy rozdzielczości kątowej 512 pozycji i zmierzonej prędkości wirnika około 33Hz wynosi 33*512Hz, co daje 17kHz.

Wyświetlacz ma możliwość zmiany rozdzielczości kątowej w czasie pracy - do wyboru są rozdzielczości: 512, 256, 128, 64, 32 pikseli/obrót.

[3] mil - jednostka długości w systemie opartym o cal. Bardzo popularna w elektronice.

2.4 MOŻLIWE EFEKTY ANIMACJI

Przy zakładanej rozdzielczości kątowej 128 pikseli/kąt obraz zajmuje 128x8x8b, czyli 1kB. Wskutek założenia pamięci obrazu urządzenia na poziomie 128kB możliwe jest przechowywanie w pamięci **128 klatek** animacji. Przy w miarę płynnej animacji na poziomie 8 klatek na sekundę pozwala to na odtworzenie 16 sekund animacji bezpośrednio z pamięci urządzenia.

Niemożliwe jest uzyskiwanie płynnej animacji na bieżąco - zakładana prędkość transmisji danych po sprzęgu RS na poziomie 19200 bodów nie pozwala na płynną animację. Maksymalna prędkość takiej animacji z uwzględnieniem strat w transmisji wynosi 1 klatkę na sekundę.

Innym możliwym efektem jest powolne **obracanie wyświetlanego obrazu** dookoła osi wirowania - efekt taki można uzyskać poprzez powolne przesuwanie początkowego adresu wyświetlanego obrazu w górę pamięci obrazu. Jeżeli 2 sąsiadujące ze sobą obrazy są takie same uzyskamy efekt powolnego obracania się wyświetlanej bryły.

W przypadku gdy 2 kolejne obszary nie zawierają tego samego obrazu, uzyskujemy efekt płynnego przejścia pierwszego obrazu w drugi - wynika to z ogranizacji pamięci graficznej wyświetlacza.

3 PROJEKT

Część projektowa była bardzo multidyscyplinarna i składała się z następujących części:

- mechanicznej
  Dobór silnika oraz mocowania wyświetlacza. Konieczne było tu zwracanie uwagi też na część elektroniczną.

- elektronicznej
  Dobór elementów, projektowanie schematu oraz wykonywanie projektów PCB. Konieczne było tu też pamiętanie o wymaganiach mechanicznych oraz zapewnienie odpowiedniej szybkości działania układów pod kątem ich późniejszego używania w części informatycznej.

- informatycznej
  Ogólne obliczenia sprawdzające czy zadany procesor będzie w stanie obsłużyć zadanie sterowania wyświetlaniem oraz komunikacją z komputerem. Konieczne było tu też odpowiednie zaprojektowanie struktury systemu w celu zapewnienia działania oprogramowania na poziomie Soft RealTime by zaspokoić wymagania fizyczne (bardzo dokładny pomiar położenia kątowego). Rozważany był też model Hard Realtime.

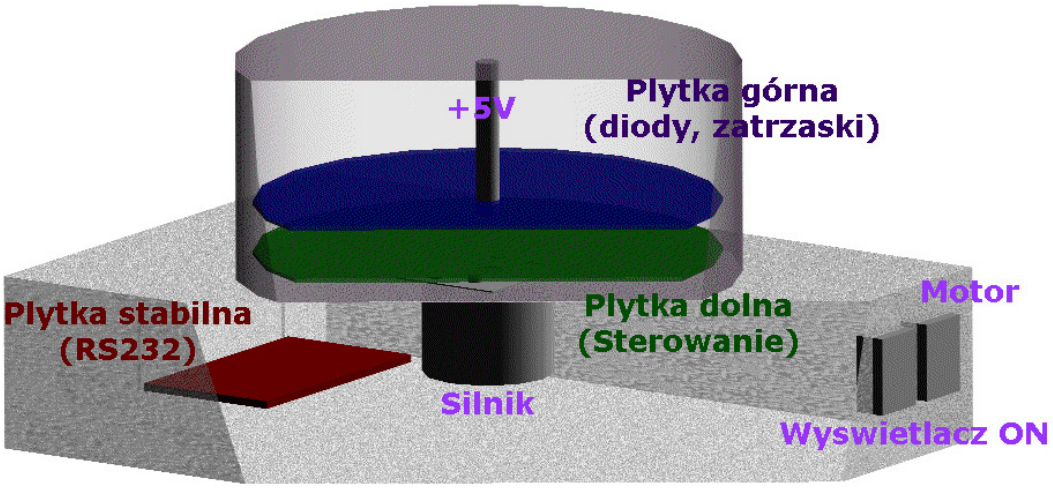
- oprogramowania PC
  Ustalenie głównych funkcji oprogramowania po stronie PC. Konieczne tu było odpowiednie dobranie algorytmów kontroli przesyłu danych do urządzenia z uwagi na spodziewane spore straty w transmisji danych przez sprzęg RS232.

Trudne jest rozdzielenie tych elementów od siebie, jednak w celu zachowania porządku podrozdziały są poukładane w wyżej wymienionej kolejności.

3.1 MECHANIKA

3.1.1 OBUDOWA

Podstawowym problemem było **dobranie wymiarów urządzenia**. Ponieważ pierwszym pomysłem było wykorzystanie mechaniki z napędu CD-ROM, więc założono dla płytek wirnika rozmiar płytek CD. Mechanika napędu CD okazała się za słaba, mimo to np. obudowa urządzenia została wzięta właśnie z napędu CD.



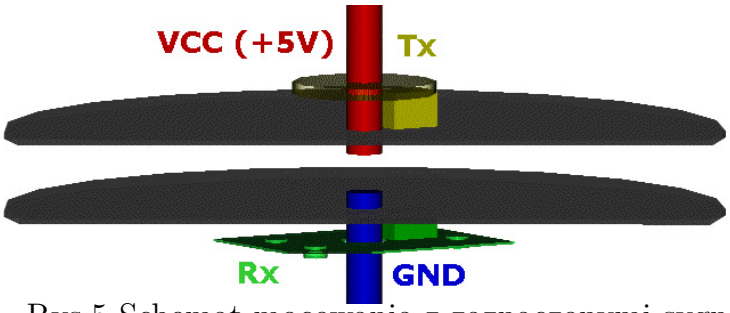
Rys.4 Poglądowy przekrój przez wyświetlacz

3.1.2 MOCOWANIE

Z uwagi na to, iż do płytek wirnika musi być dostarczone zasilanie oraz sygnały RX i TX (czyli 4 sygnały), pojawił się problem odpowiedniego zaprojektowania mocowania tak by spełnić ten wymóg. Zostało użyte rozwiązanie, doprowadzające sygnały:

- GND
  poprzez oś silnika.
- VCC
  poprzez górną oś mocująca wirnik
- Rx
  poprzez szczotkę umieszczoną na spodniej stronie dolnej płytki wirnika.

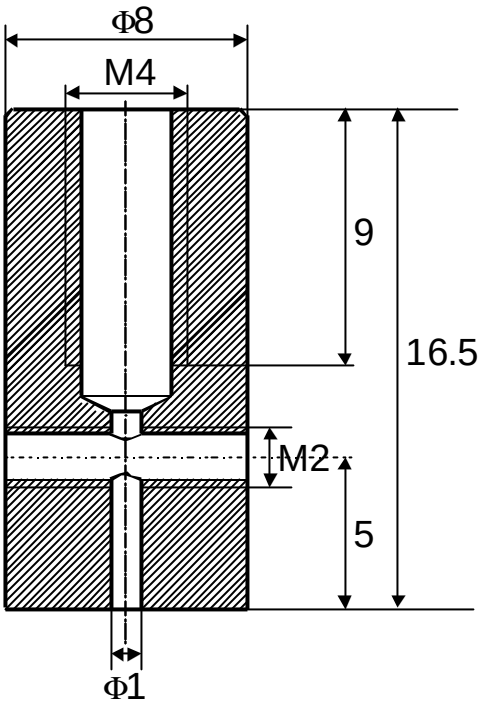
- Tx
  poprzez szczotkę umieszczoną na górnej stronie górnej płytki wirnika.



Rys.5 Schemat mocowania z zaznaczonymi sygnałami

W celu sztywnego połączenia osi silnika z dolną płytką wirnika został zaprojektowany specjalny element wykonany z mosiądzu (rys. 6). Podobny element został użyty do połączenia górnej płytki wirnika z górnym wałem.

Istotnym problemem było też umieszczenie **środka ciężkości** wirnika możliwe najbliżej osi wirowania. Zostało to osiągnięte poprzez dobór odpowiedniego rozmieszczenia elementów na wirniku, dobór kształtu płytki, oraz dobór odpowiednio lekkich elementów. Więcej na ten temat znajduje się w części poświęconej projektowaniu PCB.



Rys.6 Element łączący wał silnika z wirnikiem

3.2 ELEKTRONIKA

Część elektroniczna została zaprojektowana tak, aby spełniała następujące funkcje:

- Sterowanie diodami LED
  występują tu dwie trudności - duża liczba diód (64) oraz szybkość ich przełączania. Przy założonych 25Hz prędkości obrotowej oraz rozdzielczości kątowej 128 pikseli/kąt pełny szybkość zmian stanu diód wynosi 128*25Hz = 3.2kHz.

- pomiar położenia kątowego
  Znajomość pozycji kątowej wirnika jest konieczna aby wiedzieć w którym momencie należy zmienić stan diód.

- odpowiednio duża pamięć
  Pamięć powinna umożliwiać oglądanie 10-sekundowej w miarę płynnej animacji.

- komunikacja z PC
  Urządzenie powinno mieć możliwość komunikacji z komputerem PC w trakcie pracy.

3.2.1 MIKROKONTROLER PIC16F877A

Wybrany został procesor **PIC16F877A**. Jest on taktowany zegarem **20MHz**, jednak wewnętrznie instrukcje wykonują się z częstotliwością 5MHz (jedna instrukcja zajmuje 4 cykle). Procesor ten posiada następujące ważne dla nas właściwości:

- 40 nóżek
  7 jest wykorzystanych na obsługę rezonatora kwarcowego, doprowadzenie zasilania i masy, oraz do programowania w układzie.

- 33 piny I/O ogólnego przeznaczenia

- układ USART
  Wykorzystywany do obsługi RS232.

- zewnętrzne przerwania
  Przerwanie jest potrzebne do obsługi synchronizacji obrotów.
- 3 wbudowane liczniki (timery)
  Układy potrzebne do pomiaru czasu obrotu, oraz wyznaczania położenia kątowego.
- 8kx14b pamięci programu typu Flash
  Nietypowa organizacja (po 14 bitów) wynika z architektury procesora. Pamięć ta też może być wykorzystana do przechowywania danych, łącznie z jej zapisywaniem w trakcie działania (acz nie jest to zalecane). Pamięć ta jest podzielona na 4 banki.
- 368B pamięci RAM
  Część zajmują rejestry specjalnego przeznaczenia. Nie jest to pamięć RAM o typowym sposobie obsługi - całość to rejestry. Jest ona podzielona na 4 banki.
- 256B pamięci EEPROM danych
- Architektura typu RISC
  Tylko 35 instrukcji.
- Możliwość programowania w układzie

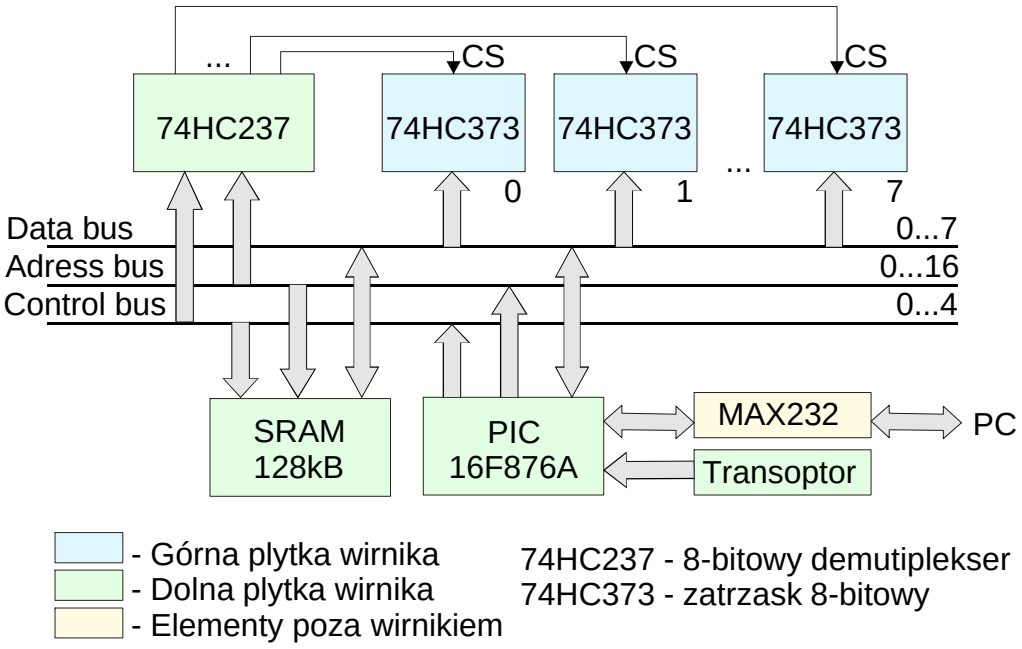
3.2.2 SCHEMAT LOGICZNY

Do dyspozycji użytkownika mikrokontroler PIC16F877A udostępnia 33 nóżki I/O. Dwie z nich od razu trzeba przeznaczyć na obsługę RS232. jedną na synchronizację położenia (sygnał z transoptora). W związku z tym zostaje 30 nóżek na obsługę pamięci SRAM 128kB oraz sterowanie stanem 64 diód. Żeby sterować wszystkim bezpośrednio potrzebne byłoby 92 sygnały, z czego 64 na sterowanie stanami diod oraz 28 na obsługę pamięci SRAM. Sterowanie bezpośrednie jest zatem niemożliwe. Na liczbę 28 sygnałów potrzebnych dla prawidłowej pracy pamięci SRAM składa się 17 bitowa szyna adresowa, 8-bitowa szyna danych oraz 3 sygnały sterujące.

W związku z tym iż sygnałów sterujących dla diód nie da się bezpośrednio wyprowadzić z procesora, użytych zostało 8 układów 74HC373 [4]. Wyjścia zatrzasków sterują stanami diody. Wejścia zatrzasków są wspólne, oddzielone są natomiast sygnały CS. To rozwiązanie ogranicza liczbę potrzebnych sygnałów sterujących diodami do 16. Sygnały te znajdują się na czterech złączkach pomiędzy górną i dolną płytka wirnika.

Niemniej jednak ciągle potrzebne są 44 sygnały (16+ 17+8+3). Dlatego wejścia zatrzasków są współdzielone również z szyną danych pamięci, co ogranicza liczbę potrzebnych sygnałów do 36 (8+17+8+3). Jest to ciągle wartość większa od liczby dostępnych nóżek mikrokontrolera (30).

Aby rozwiązać ten problem, sygnały CS układów 74HC373 nie są wyprowadzane bezpośrednio z procesora. Użyty tu został układ 74HC237 [5]. Jest on sterowany bezpośrednio z procesora 2 sygnałami sterującymi. Natomiast wejścia adresowe demultipleksera są współdzielone z 3-ma najmłodszymi liniami adresowymi pamięci SRAM. Co ogranicza liczbę potrzebnych sygnałów do 30 (2+17+8+3). A tyle akurat jest dostępnych.



Rys.7 Poglądowy schemat logiczny wyświetlacza

[4] nienegujących trójstanowych zatrzasków 8 bitowych typu D
[5] nienegujący demultiplekser 3 na 8 z zatrzaskiem adresu

Możliwe było ograniczenie liczby potrzebnych sygnałów do 28, gdyż liczbę sygnałów sterujących do pamięci da się ograniczyć do 2, a dla demultipleksera do 1. Niemniej jednak nie było to konieczne, ani nie było też dodatkowych zastosowań dla wolnych nóżek.

W celu synchronizacji położenia użyta została para **dioda - fototranzystor** (czyli transoptor). Raz na obrót pomiędzy te elementy wlatuje **przesłona**, co powoduje ustawienie fototranzystora w stan zaporowy. Sygnał wyjściowy z układu fototranzystora jest wprowadzany do procesora jako złącze zewnętrznego **przerwania** wyzwalanego zboczem.

Schamty wszystkich omówionych tu układów znajdują się w załączniku na końcu niniejszej pracy.

3.2.3 PROJEKTOWANIE PCB

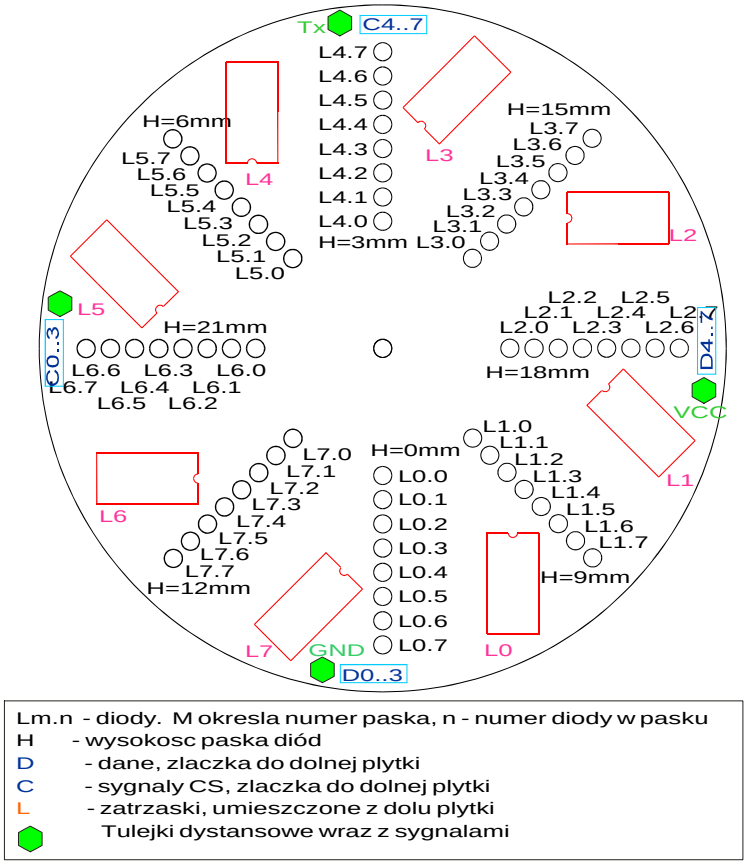
W celu zaprojektowania płytek zostało użyty pakiet **Protel**. Programowanie to zostało wybrane z następujących powodów:

- cena
  jest darmowy dla zastosowań niekomercyjnych - wersja testowa pozwala na używanie programu przez miesiąc.
- standard
  jest standardem przemysłowym, chętnie używanym w biurach projektowych.
- funkcjonalność
  pozwala na zaprojektowanie kompletnego schematu PCB bez potrzeby sięgania do dodatkowego oprogramowania.
- znajomo obsugi
  Protel nie jest programem łatwym w obsłudze - wprowadza własne skróty klawiszowe, niekompatybilne z obowiązującymi standardami. Dlatego dużym ułatwieniem była znajomość Protela wyniesiona z wcześniejszych projektów.

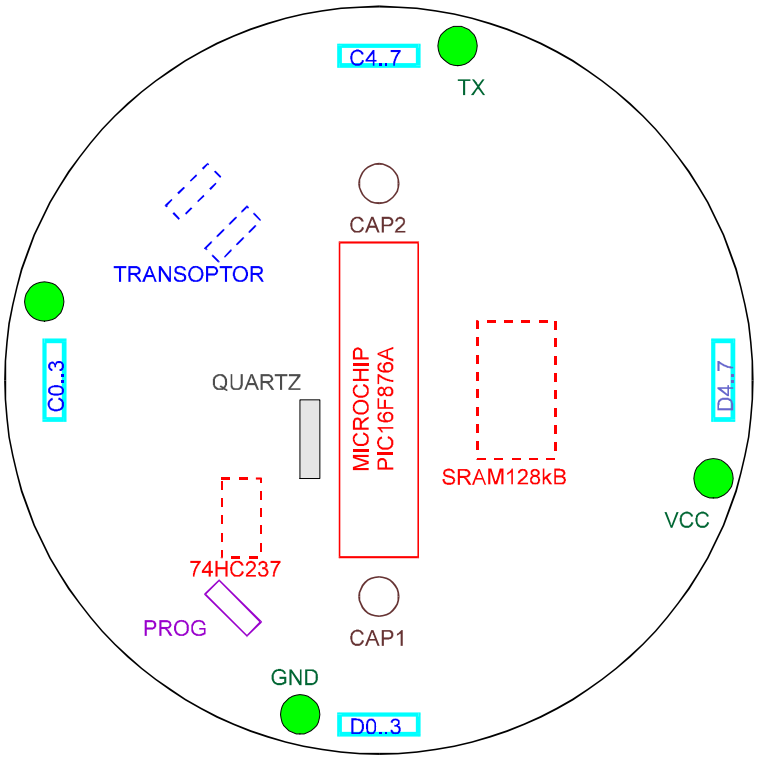
Głównym problemem było takie zaprojektowanie płytek drukowanych wyświetlacza, aby ich środek ciężkości leżał możliwie blisko osi wirowania. W tym

celu został użyty bardzo nietypowy kształt płytek - są one okrągłe. Również elementy elektroniczne zostały umieszczone na płytkach tak, aby możliwie jak najbardziej zbliżyć środek ciężkości do osi wirowania - najlepiej widać to na górnej płytce wyświetlacza, w której elementy (diody, zatrzaski, złączki, rezystory) zostały umieszczone tak, by dla każdego elementu po drugiej stronie wirnika znajdował się element o takiej samej masie (symetria osiowa). Było to możliwe z uwagi na parzystą liczbę zatrzasków (8), diód i rezystorów (po 64), złączek (4) oraz tulejek dystansowych (4).

Również wysokość nóżek diód wyświetlacza była dobierana tak, by na przeciwko siebie znajdowały się diody o zbliżonej długości nóżek.



Rys.8 Rozłożenie elementów na górnej płytce wirnika



Rys.9 Rozłożenie elementów na dolnej płytce wirnika

Dolna płytka nie mogła mieć już tak symetrycznie rozmieszczonych elementów, jakkolwiek tu też zostały poczynione starania by środek ciężkości umieścić w osi wirowania - najcięższy element tej płytki, mikrokontroler PIC16F877A został umieszczony w samym środku (na podstawce uniwersalnej). Wadą tego rozwiązania jest to iż brak jest mocnego mechanicznego połączenia górnej i dolnej osi wirnika, co wpływa na zwiększone drgania całego urządzenia.

Na dolnej płytce znajdują się też 2 kondensatory 470uF, umieszczone symetrycznie względem osi wirowania. Inne elementy dolnej płytki zostały rozmieszczone tak blisko osi wirowania jak było to możliwe.

W celu zmniejszenia wagi całości wirnika zostały użyte elementy przeznaczone do montażu powierzchniowego, z uwagi na ich mniejszą wagę od elementów w obudowach przewlekanych. I tak:

- rezystory

  Wszystkie 64 rezystory górnej płytki, jak i 6 rezystorów dolnej płytki to elementy SMD.

- kondensatory
  Na płytkach wirnika znajdują się 4 kondensatory - 2x470uF (do stabilizacji zasilania) i 2x22pF (do rezonatora kwarcowego). Kondensatory 22pF są typu SMD.

- układy scalone serii 74HCxxx
  Wszystkie układy scalone rodziny 74HCxxx (8x74HC373, 1x74HC237) są przeznaczone do montażu powierzchniowego (obudowy TQFP)

- pamięć SRAM 128kB
  Została użyta kość D431000AGW-70LL, produkcji NEC, również w obudowie TQFP.

Mikrokontroler jest jedynym elementem scalonym na płytce w obudowie przewlekanej (DIP40). Została ona użyta z powodu trudności w zakupie pojedynczego egzemplarza układu w obudowie do montażu powierzchniowego (SMD, TQFP44) na terenie Polski.

3.3 OPROGRAMOWANIE

Oprogramowanie mikroprocesora PIC16F877A zostało napisane z wykorzystaniem środowiska programistycznego **MpLAB**. Jest to środowisko firmowe, stworzone przez producenta układów serii PIC. Jest ono całkowicie darmowe, najnowszą wersje można zawsze pobrać ze strony www.microchip.com.

Jako język programowania został użyty asembler mikroprocesora PIC. Kompilatory języków wyższego poziomu (C, Basic) są dostępne odpłatnie i nie były wykorzystywane. Istnieje też port GCC[6] dla mikroprocesorów ośmio-bitowych (stdcc) zawierający wsparcie dla mikrokontrolerów PIC, niemniej jednak po krótkich testach został on odrzucony z uwagi na błędy w kompilacji.

Zaletą środowiska MPLAB jest też obsługa programatora **PICSTART plus**. Tani programator kompatybilny z nim można nabyć w Gliwicach. Więcej informacji można znaleźć na stronie producenta - http://ajpic.zonk.pl/

Głównymi zadaniami mikrokontrolera są:

- obsługa peryferiów
  Obsługa elektroniki zawartej na wirniku - pamięci SRAM, odczyt sygnału synchronizacji z transoptora, obsługa 64 diód LED poprzez wspomagającą elektronikę.

- komunikacja
  Komunikacja z komputerem PC poprzez sprzęg RS. Interpretacja przychodzących komend i potwierdzanie ich.

- wyświetlanie
  Ustalanie pozycji kątowej wirnika na podstawie sygnału synchronizacji z transoptora, zmienianie stanu diód w zależności od pozycji kątowej.

Aby zapewnić działanie tych funkcji, zostały użyte następujące przerwania (wymienione od najwyższego do najniższego priorytetu obsługi):

- synchronizacji
  Przerwanie zewnętrzne podpięte do wyjścia z transoptora. Zeruje licznik

[6] GNU C Compiler. Darmowy kompilator języka C oparty o zasady licencji GNU

nr 1. Ustawia częstotliwość przerwania od licznika 2. Zeruje pozycje kątową.

- odczytu RS
  Przerwanie danych przychodzących UART. Dane są niezwłocznie kopiowane do bufora programowego.

- Licznika 1
  Jest to 16-bitowy licznik zliczający czas potrzebny na wykonanie pełnego obrotu wirnika. Jest on zerowany przez przerwanie synchronizacji. W przypadku gdy nastąpi przepełnienie tego licznika stwierdzane jest iż wirnik się nie obraca. wyłączane jest z tego powodu przerwanie licznika 2.

- Licznika 2
  Główne przerwanie wyświetlania. Jego częstotliwość jest ustalana w przerwaniu synchronizacji na postawie odczytu z Licznika 1. Inkrementuje zmienną zawierającą współrzędną kątową. Dokonuje odczytu danych z pamięci SRAM (jeżeli odczyt ten nie jest zabezpieczony semaforem) i odpowiednio do odczytanych danych ustawia stan 64 diód LED.

Przy odświeżaniu 25Hz i zakładanej rozdzielczości kątowej 128 wokseli, częstotliwość odświeżania stanu diód wynosi 128*25 = 3,2kHz. Przy tej częstotliwości procesor ma 5MHz/3.2kHz = 1565 instrukcji, podczas których musi wysterować odpowiednio diody. Wydaje się to bezpieczną wartością.[7]

[7] W praktyce okazało się iż potrzebne jest około 160 instrukcji by tego dokonać. Kątowa rozdzielczość maksymalna została zwiększona do 512 wokseli/kąt pełny. Możliwa była też jeszcze większa rozdzielczość 1024 wokseli/kąt pełny, ale nie została wprowadzona - efekt przy 512 wokselach/kąt pełny jest i tak nadto zadawalający.

3.4 OPROGRAMOWANIE PC

Jako środowisko developerskie został wybrany pakiet Microsoft Visual C++. Został on wybrany z następujących powodów:

- standardowość
  Jest to najbardziej standardowe środowisko do tworzenia oprogramowania dla systemu operacyjnego Windows, stworzone przez firmę będącą również twórcą systemu operacyjnego. Jako takie jest to najpewniejsze środowisko programistyczne.
- obsługa
  Autor pracy miał wcześniejszą styczność z tym środowiskiem, dlatego nauka środowiska nie była potrzebna.
- przyjazność
  Środowisko zapewnia duży zestaw narzędzi ułatwiających i przyśpieszających tworzenie kodu.

Głównymi zadaniami oprogramowania na PC jest:

- komunikacja
  Komunikacja z wyświetlaczem poprzez sprzęg RS232. Musi ona być asynchroniczna, oraz uwzględniać systuację typu timeout. Możliwa ma być komunikacja również podczas pracy wyświetlacza.
- obsługa danych
  Możliwość obsługi danych w 2 formatach - graficznej (jako pojedynczy obraz), oraz jako duplikatu (mirror) zawartości pamięci wyświetlacza.
- opcja symulacji
  Wstępne oglądanie obrazu przed wczytaniem go do urządzenia.
- ustalanie trybu pracy
  Możliwość ustalenia trybu pracy wyświetlacza.
- program graficzny
  Praca w trybie programu graficznego.

4 WYKONANIE

4.1 MECHANIKA

Projektowanie części mechanicznej prawie nie miało miejsca. Etap wykonywania był natomiast ciekawszym procesem. Część mechaniczna urządzenia w celu ograniczenia kosztów została bowiem wykonana w większości z materiałów pozostałych z uszkodzonych urządzeń, opakowań - słowem, tego co akurat było dostępne. W takiej sytuacji jakiekolwiek projektowanie nie wchodziło w grę. Dla przykładu:

- obudowa
  została wykonana z obudowy po uszkodzonym napędzie CD-ROM, zaślepce od napędu z komputera HP Kayak XA, pustego pudełka po płytach CD, kawałka pleksi. Całość obudowy została pomalowana na czarno.
- mocowanie wirnika
  zostały użyte m.in.: fragment starej dętki rowerowej (do amortyzacji silnika), fragment laminatu miedzianego (po którym porusza się szczotka), szczotki z myszki komputerowej (pewne stare modele posiadały takie szczotki zamiast pary dioda-fototranzystor), zatyczka długopisu (do mocowania powierzchni po której porusza się górna szczotki), fragment uszkodzonego frezu (służy jako górna oś obrotów wirnika).

4.1.1 TRUDNOŚCI

Głównym problemem wykonawczym na jaki natrafiono podczas prac było wytłumienie drgań podczas pracy urządzenia. Trudności sprawiało takie zamocowanie wirnika, by jego środek ciężkości znajdował się możliwie blisko osi wirowania. Problem ten był blisko powiązany z inną trudnością - realizowacją mocowania pomiędzy płytkami wirnika a osią silnika. Oś silnika ma średnicę około milimetra. Potrzebny był element przejściowy pomiędzy osią a płytkami. Jest to jedyny fragment pracy, w którym autor musiał skorzystać z usług zewnętrznej firmy - zakładu ślusarskiego, który takie mocowanie wykonał.

Płytki są mocowane do elementu łączącego za pomocą śrubki M4. Z powodu małej twardości laminatu trudno jest przykręcić wirnik w spobób powtarzalny.

Tak więc pozycjonowanie go by wytłumić drgania jest konieczne przy każdej operacji składania wirnika. Było to szczególnie uciążliwe na wstępnym etapie prac, kiedy zachodziła konieczność częstego rozkładania wirnika.

4.2 ELEKTRONIKA

Płytki drukowane zostały wykonane przez autora samodzielnie. Proces drukowania negatywu, naświetlania, wytrawiania, wiercenia dziur, robienia przelotek, nadawania kształtu oraz lutowanie był wykonany w domowych warunkach. Cały proces został przeprowadzony w następujący sposób: [8]

- drukowanie negatywu
  Użyte narzędzia - drukarka atramentowa HP.
- naświetlanie i przygotowanie laminatu
  Użyte narzędzia i odczynniki - gilotyna, amatorska naświetlarka z 6-ścio watową świetlówką UV, szybka, piekarnik elektryczny, spray Positiv, denaturat, płyn do czyszczenia WC, laminat.

  Laminat został wstępnie przycięty do rozmiarów o 1-2 cm (w każdej osi) większych niż potrzebne. Zostawiony zapas 1-2cm ułatwia nanoszenie emulsji "Positiv", która ma tendencję do zostawiania grubszej warstwy przy brzegach płytki. Następnie z użyciem płynu do czyszczenia WC (jest to 9% roztwór kwasu fosforowego) warstwa miedzi na laminacie została oczyszczona z tlenku miedzi. Potem laminat został przemyty denaturatem by pozbyć się resztek płynu do czyszczenia WC. Następnie pierwsza warstwa laminatu została spryskana emulsją "Positiv", po czym płytka została na godzinę zamknięta w ciemnym miejscu. Kolejnym etapem było włożenie jej na 30 minut do piekarnika, w którym temperatura nie przekraczała 70 stopni (wygrzewanie nie jest konieczne, ale bez niego płytka powinna schnąć około doby). Po wyjęciu z piekarnika emulsja "Positiv" została naniesiona na drugą stronę płytki (po jej uprzednim przemyciu denaturatem). Laminat ponownie powędrował na godzinę do ciemni, a następnie został wygrzany w piekarniku.

[8] zamieszczony opis metody wykonywania płytek w warunkach domowych powstał w wyniku długotrwałego procesu eksperymentowania z tą technologią przez autora. Zostaje on zamieszczony z uwagi na to iż może być przydatny dla czytelnika chcącego poznać tą technologię.

Po przygotowaniu laminatu zostało w nim wywierconych kilka otworów w celu umożliwienia synchronizacji położenia między górną i dolną warstwą - bez tego trudno by było przyłożyć negatyw z obu stron z zachowaniem należytej dokładności.

Następnie na laminat zostały położone folie z negatywem, które zostały dociśnięte do laminatu z użyciem szybki. Całość została poddana naświetlaniu promieniami UV przez czas 6 minut 30 sekund (dobrany eksperymentalnie przez autora dla świetlówki o mocy 6W, jest to parametr krytyczny procesu). Potem naświetlona została druga strona płytki.

- wytrawianie
  Użyte narzędzia i odczynniki - piekarnik, kuweta, rękawice gumowe, pipetki, gąbka, preparat SENNO 4007, wodny roztwór chlorku żelaza, denaturat.

  Naświetlony laminat został włożony do kuwety z przygotowanym preparatem SENNO 4007 (można też używać zasady sodowej, w tym np. popularnego płynu do przetykania rur "Kret"), służącym do wywoływania naświetlonego laminatu. Na tym etapie przydatna była gąbka (z rączką), do przecierania powierzchni laminatu (co pozwala dokładniej sterować działaniem preparatu SENNO4007 i kompensować różnice w grubości naniesionej warstwy emulsji Positiv). Dobrym pomysłem jest wykorzystanie gumowych rękawic w celu ochrony skóry rąk. Po stwierdzeniu że płytka jest wywołana została ona wypłukana pod bieżącą wodą w celu usunięcia reszty wywoływacza.

  Następnie laminat został włożony do wodnego roztworu chlorku żelaza i umieszczony w piekarniku w temperaturze około 70 stopni. Wygrzewanie nie jest konieczne, jednak pozwala na przyśpieszenie trawienia. Po stwierdzeniu iż płytka została należycie wytrawiona, została ona ponownie wypłukana w wodzie. Następnie została przetarta denaturatem w celu usunięcia emulsji Positiv.

- wiercenie otworów
  Użyte narzędzia - amatorska wiertarka do robót precyzyjnych.

- nadawanie kształtu
  Użyte narzędzia - gilotyna, szlifierka z zestawu do wiertarki, papier ścierny, pilniki.

  Podczas projektowania PCB na projekcie został narysowany pierścień o grubości 1cm, wyznaczający granicę płytki. Wewnętrzny promień pierścienia wyznaczał brzeg płytki o promieniu 60mm. Płytki wirnika zostały wstępnie przycięte do kształtu jak najbardziej zbliżonego do koła (tak by nie uciąć nic leżącego wewnątrz pierścienia), a następnie z użyciem narzędzi szlifierskich została zeszlifowana reszta pierścienia. W ten sposób uzyskany został okrągły kształt płytek wirnika.

- robienie przelotek i lutowanie
  Użyte narzędzia i substancje - stacja lutownicza Elwik RTC-24, pincety, lupa, kalafonia, płyn do lutowania elementów SMD, cyna, odsysacz do cyny, taśma do rozlutowywania.

4.2.1 TRUDNOŚCI

Podczas wykonywania płytek natknięto się na następujące problemy:

- drukowanie negatywu
  Uzyskiwane krycie na folii jest częstokroć niewystarczające. By uniknąć problemów, zostały użyte dwie folie z negatywem i naświetlanie odbywało się poprzez 2 warstwy folii.

- naświetlanie
  Podczas naświetlania płytek dwustronnych występuje problem synchronizacji położenia warstw. Osiągnięty rezultat nie do końca jest zadawalający, niemniej jednak uzyskane płytki były możliwe do wykorzystania.

- wytrawianie
  Spodnia część płytki ulega trawieniu się znacznie szybciej niż jej górna część. Spowodowane jest to prawdopodobnie tym, iż wskutek działania grawitacji cząsteczki chlorku miedzi jako cięższe od roztwóru chlorku żelaza opadają na dno, co przyśpiesza reakcję. Efekt ten był zaskakujący dla autora niniejszej pracy.

- wiercenie
  Wskutek braku stojaka do wiertarki otwory nie są wywiercone dokładnie pod kątem prostym do powierzchni. Powoduje to dodatkowe rozjeżdżanie się otworów pomiędzy dolną i górną warstwą płytki.

- nadawanie kształtu
  Mimo nietypowego kształtu płytek nie sprawiło to trudności.

- robienie przelotek
  Najbardziej nieprzyjemny i pracochłonny etap. Tylko na jednej z płytek była konieczność wykonania 100 przelotek, a następnie sprawdzenie poprawności lutów i poszukiwanie zwarć. Cały proces zajął nieco ponad jeden dzień roboczy. Same przelotki zostały wykonane z nóżek rezystorów.

- lutowanie
  Przy wlutowywaniu elementów scalonych w obudowach SMD uwagę trzeba było poświęcić testowaniu połączeń i szukaniu zimnych lutów. Nie było to jednak bardzo uciążliwe.

Nakład pracy na wykonanie płytek o tym stopniu skomplikowania w warunkach domowych jest duży. Potrzebne jest na to kilka dni pracy. Zdecydowanie lepszym rozwiązaniem w takim przypadku jest zlecenie wykonania specjalistycznej firmie.

4.3 OPROGRAMOWANIE

Oprogramowanie mikrokontrolera zostało napisane w asemblerze, z użyciem środowiska programistycznego **MpLAB**. Celem zachowania porządku tekst źródłowy programu został podzielony na następujące pliki:

- my.asm
  Główny program. Znajdują się tu procedury inicjalizujące peryferia mikrokontrolera (initialization), deklaracje zmiennych globalnych oraz procedury interpretujące dane otrzymane za pośrednictwem sprzęgu RS.

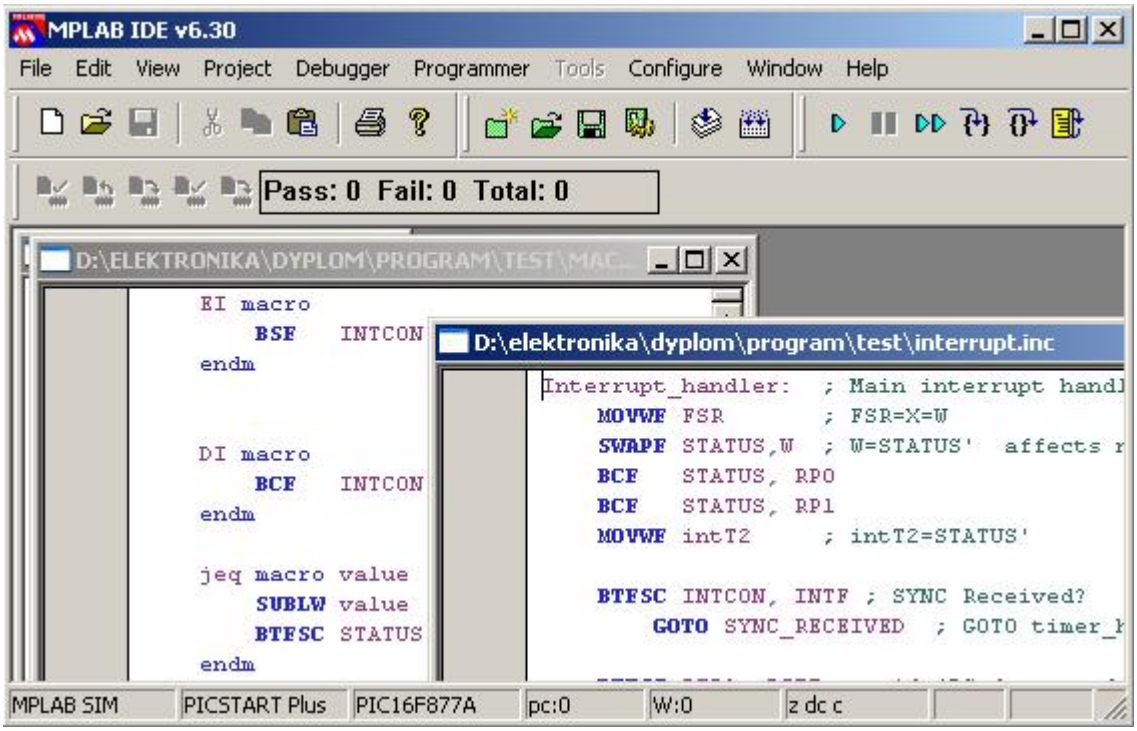
- interrupt.inc
  Część odpowiedzialna za prawidłową obsługę przerwań.

- tables.inc
  Z uwagi na architekturę mikrokontrolera (brak pamięci jako takiej, w zamian duża liczba rejestrów) występuje problem realizacji tablic statycznych. Użyte rozwiązanie zawiera dane jednobajtowe bezpośrednio w kodzie, co sprawia że jest szybkie. Wadą są ograniczenia w alokacji kodu tablic[9]. W celu zapanowania nad alokacją wszystkie tablice znajdują się w osobnym pliku, który jest dołączany (linkowany) w taki sposób by znajdował się na samym początku pamięci programu. W pliku tym znajdują się również inne funkcje czułe na umiejscowienie w pamięci.

- tools.inc
  Plik ten zawiera rozmaite funkcje pomocnicze m.in. do obsługi pamięci SRAM, realizacji opóźnień itp.

- macros.inc
  Zbiór makr ułatwiających pisanie kodu w asemblerze mikrokontrolera, jak np. implementacja znanych z architektury x86 mnemoników EI oraz DI.

[9]Dzieje się tak ponieważ do obsługi tablic używa się operacji arytmetycznych (dodawania) na mniej znaczącym bajcie rejestrze PC. W przypadku przepełnienia górna połowa nie jest zmieniana, co prowadzi do błędów.

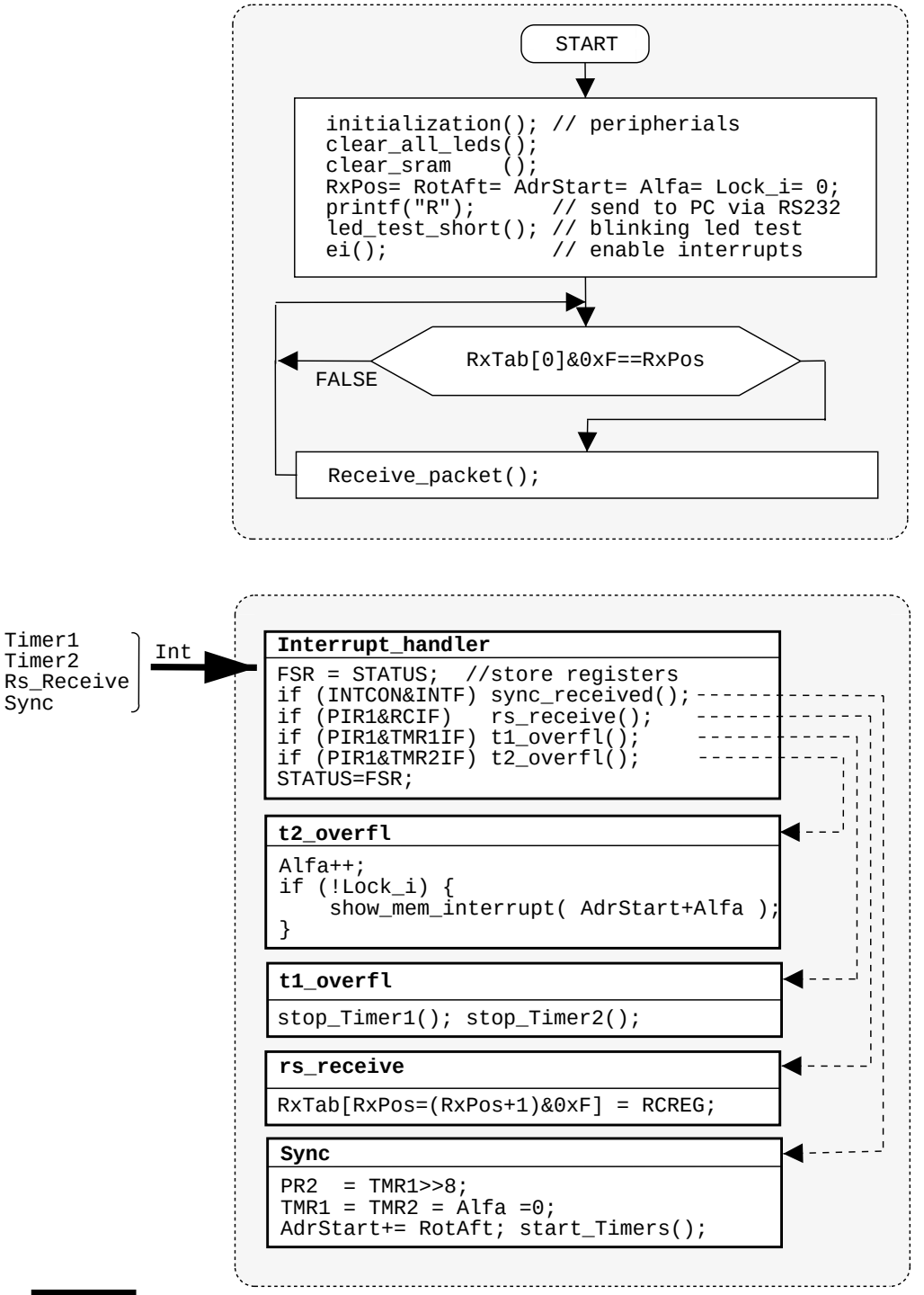


Rys.10 Środowisko MpLAB

Działanie systemu opiera się w głównej mierze na przerwaniach. Przerwania w całości obsługują zarówno **wyświetlanie** obrazu, jak i **odbiór danych** z sprzęgu RS. Jedynym 'zadaniem' wykonywanym poza przerwaniami jest interpretacja danych odebranych ze sprzęgu RS i wykonywanie odebranych komend. Dzięki temu nie występują zakłócenia w wyświetlaniu podczas wykonywania komend (za wyjątkiem komend operujących na pamięci SRAM).

Ponieważ występuje konflikt w dostępie do pamięci SRAM pomiędzy przerwaniem wyświetlania (na przerwaniu licznika 2) a operacjami zapisu/odczytu pamięci zlecanymi zdalnie przez sprzęg RS, został zaimplementowany programowy semafor (**Lock_i**). Pozwala on procedurze obsługi sprzęgu RS232 zablokować dostęp do pamięci przerwaniu wyświetlającemu. Jest to konieczne, gdyż adresowanie pamięci wymaga wystawienia odpowiednich sygnałów na porty: A, B, C, D, E. W przypadku gdyby nie było semafora i przyszło przerwanie, wystawiony na portach adres byłby niezgodny z pożądanym.

Wyznaczanie czasu pojedynczego obrotu wirnika jest oparte na liczniku 2, działającym na podstawie czasowej równej 625kHz. Ponieważ użyty licznik



Rys.11 Uproszczony schemat algorytmu dla mikrokontrolera PIC

jest 2-bajtowy, pozwala on na zliczenie czasu równego niecałe 105ms. W przypadku gdy czas ten jest dłuższy następuje przepełnienie wyłączające główne przerwanie wyświetlające oparte na liczniku 1. Do właściwej pracy wyświetlacz musi mieć zatem częstotliwość odświeżania większą niż 10Hz.

Licznik drugi działa na podstawie czasowej równej 1250kHz. Wystawia on przerwanie gdy zliczona wartość jest równa zawartości rejestru **PR2** (1-bajtowego). Maksymalny czas wynosi zatem nieco ponad 0.2 ms (niecałe 5 kHz). W praktyce, przy wysokiej rozdzielczości wyświetlacza (512 pozycji/obrót) i prędkości obrotowej równej 32 Hz potrzebne jest przerwanie generowane z częstotliwością około 16.4 kHz, co idealnie spełnia przerwanie licznika 2. Istnieje też możliwość wystawiania co n-tego przerwania przy przepełnieniu licznika, dla n=1...16. Jest to wykorzystywane przy ustawianiu rozdzielczości pracy wyświetlacza.

Samo przerwanie licznika drugiego inkrementuje jedynie licznik pozycji obrotowej **Alfa**, oraz ustawia na diodach stan pobrany z pamięci SRAM z adresu Alfa+AdrStart [10] (w jednostce 8-bajtowej). **AdrStart** jest początkiem pamięci obrazu wyświetlanego. Ustawianie odbywa się pod warunkiem że semafor **Lock_i** jest wyzerowany.

Ostatnie przerwanie związane z wyświetlaniem obsługuje synchronizację kątową. Jest ono wyzwalane gdy przesłona przechodzi przez transoptor szczelinowy umocowany z dołu dolnej płytki wirnika. Dokonywany jest wtedy odczyt zawartości licznika pierwszego i na jego podstawie dokonywane jest obliczenie i ustawienie rejestru **PR2**, używanego przez licznik drugi. Następuje wyzerowanie wartości liczników pierwszego i drugiego (rejestry: **TMR1H**, **TMR1L**, **TMR2**), jak również licznika pozycji kątowej **Alfa**.

Osobne przerwanie zajmuje się obsługą danych przychodzących na sprzęgu RS, pracującego z przepustowością 19 200 bodów (1 bit stopu, bez kontroli parzystości). Dana przychodząca zostaje zapisana do 16 bajtowej tablicy **RxTab**, pod indeks **RxPos**. Przerwanie nie zajmuje się interpretacją danych przychodzących, zostawiając to procedurze obsługi poza przerwaniem.

[10] przez to uzyskuje się dość egzotyczną organizację pamięci obrazu

Tablica RxTab **nie jest** implementacją bufora cyklicznego. Nie jest to konieczne - protokół transmisji nie zezwala na wysłanie dwóch komend jedna po drugiej bez czekania na potwierdzenie poprzedniej. Procedura interpretująca dane przychodzące zeruje zawartość **RxPos** po skończeniu wykonania komendy, po czym wysyła potwierdzenie wykonania zezwalające jednocześnie na kontynuację transmisji. Bufor cykliczny (klasyczne rozwiązanie) nie został zaimplementowany z uwagi na trudności w obsłudze tablic sprawianą przez architekturę procesora.

Przerwanie danych wychodzących sprzęgu RS **nie zostało** zaimplementowane. Nie ma takiej potrzeby dzięki zachowaniu w protokole transmisji zasady nie wysyłania dwóch komend po sobie bez czekania na potwierdzenie, oraz - dzięki temu iż procesor poza przerwaniami nie zajmuje się niczym innym niż obsługa danych ze sprzęgu RS.

4.3.1 TRUDNOŚCI

Głównymi problemami na jakie się natknięto podczas realizacji oprogramowania kontrolującego praca mikrokontrolera były:

- wyścigi

  Pomiędzy przerwaniami dochodzi do wyścigów. Problem ten wystąpił pomiędzy przerwaniem synchronizacji a wyświetlania, co były świetnie obserwowalne na wyświetlaczu w postaci chwilowych drgań obrazu. Wystąpił też konflikt pomiędzy procedurami obsługi pamięci z poziomu przerwania wyświetlającego i procedury obsługi danych sprzęgu RS.

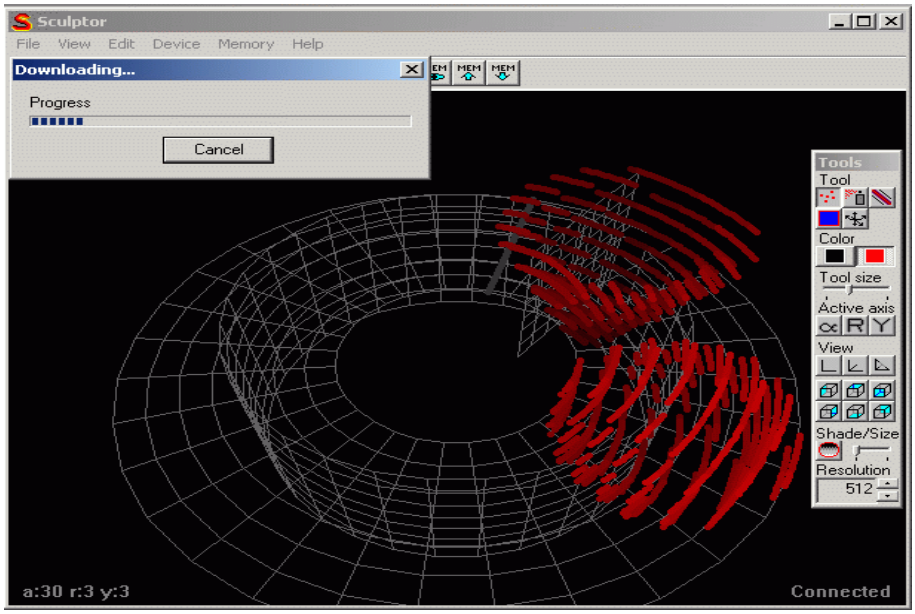
- asembler

  Używanie asemblera mikrokontrolera PIC nie należy do przyjemności. Hasło reklamowe firmy - "tylko 35 rozkazów do nauczenia" nieco mija się z prawdą. Bo o ile rozkazów jest rzeczywiście 35, to zrealizowanie adresowania pośredniego pamięci (de facto rejestrów) wymaga korzystania z dodatkowych rejestrów specjalnego przeznaczenia. Podobnie mimo iż do dyspozycji jest 8kB pamięci programu, to jest ona podzielona na 4 strony. I aby kontekst programu przenosił się między stronami trzeba używać dodatkowych rejestrów specjalnego przeznaczenia. Przykłady takie moż-

na mnożyć. Programowanie w asemblerze przez to bardziej przypomina walkę z architekturą niż pisanie kodu.

- programowanie
  Programowanie mikrokontrolera wymagało za każdym razem zatrzymania wirnika, częściowego demontażu osłony, podpięcia kabelka do programatora, zaprogramowania, po czym ponownego poskładania w celu przetestowania. Były to czynności bardzo żmudne.

4.4 OPROGRAMOWANIE PC



Rys.12 Program Sculptor podczas pracy

Oprogramowanie zostało napisane w Windows API, z użyciem środowiska programistycznego Microsoft Visual C++ 4.0. Program kontrolujący pracę wyświetlacza został nazwany **Sculptor** (ang. rzeźbiarz). Tekst źródłowy programu zostały podzielony na następujące pliki:

- config.cpp
  Moduł zawierający kilka globalnych definicji.

- connection.cpp
  Implementacja klasy connection, odpowiedzialnej za komunikację z wyświetlaczem oraz obsługę plików z danymi.

- display.cpp
  Implementacja klasy display, odpowiedzialnej za renderowanie obrazu. Klasa ta posiada również funkcje pomocnicze dla programu graficznego.

- dyplom2.cpp
  Moduł zawierający obsługę GUI.

Dokładny opis procesu tworzenia programu nie jest istotny dla treści pracy, dlatego zostaje pominięty. Opis możliwości programu został zawarty w rozdziale 5 (instrukcja obsługi).

4.4.1 TRUDNOŚCI

Głównymi trudnościami podczas tworzenia programu były:

- komunikacja asynchroniczna
  Konieczne było zaimplementowanie pełnej komunikacji asynchronicznej, wraz z obsługą zdarzenia typu timeout. Dokumentacja na ten temat jest niezbyt przyjazna dla programisty, sposób implementacji - zagmatwany. Implementacja tego samego zagadnienia w systemach typu Unix (np. Linux) jest zdecydowanie bardziej przejrzysta.

- tworzenie GUI
  Implementacja 12 okienek dialogowych wraz z procedurami (handlerami) obsługi była procesem bardzo pracochłonnym.

- renderowanie 3D
  Konieczne było stworzenie klasy odpowiedzialnej za obsługę renderowania wyświetlanego obrazu. Klasa ta umożliwia m.in. obracanie obrazu, oglądanie w 3 rzutach (płaski, ukośny oraz perspektywiczny), dobór wielkości wyrenderowanych voxeli (automatyczny bądź manualny), cieniowanie w zależności od odległości voxela od patrzącego. Nie została do tego użyta żadna biblioteka 3D.

5 OPISY UŻYTKOWE

5.1 PROGRAM SCULPTOR

Środowisko spełnia 2 główne funkcje, operujące na różnych typach obiektów:

- programu graficznego
  Program operuje tu na obrazie.
- komunikacji z wyświetlaczem
  Program operuje tu na lokalnej kopii pamięci SRAM wyświetlacza.

Możliwe są konwersje pomiędzy tymi obiektami - program nie wprowadza tu żadnych ograniczeń. Należy mieć na uwadze, iż w pamięci SRAM może się zmieścić wiele obrazów. Możliwe jest wklejanie obrazu w dowolne miejsce pamięci, jak również ściągnięcie (download) obrazu z dowolnego miejsca.

5.1.1 EDYTOR 3D

Do podglądu oraz edycji obrazu służą menu **View**, **Edit**, oraz paleta narzędzi. Operują one na lokalnym obrazie, który jest renderowany w czasie rzeczywistym na ekranie monitora. Obracanie wyrenderowanego obrazu jest możliwe z użyciem myszki - przytrzymywanie naciśniętego prawego klawisza myszki podczas poruszania nią powoduje obracanie się obrazu.

Na rysunku 13 przedstawione jest menu **View**. Pozwala ono na wybór sposobu renderowania obrazu oraz wybór pozycji patrzenia. I tak:

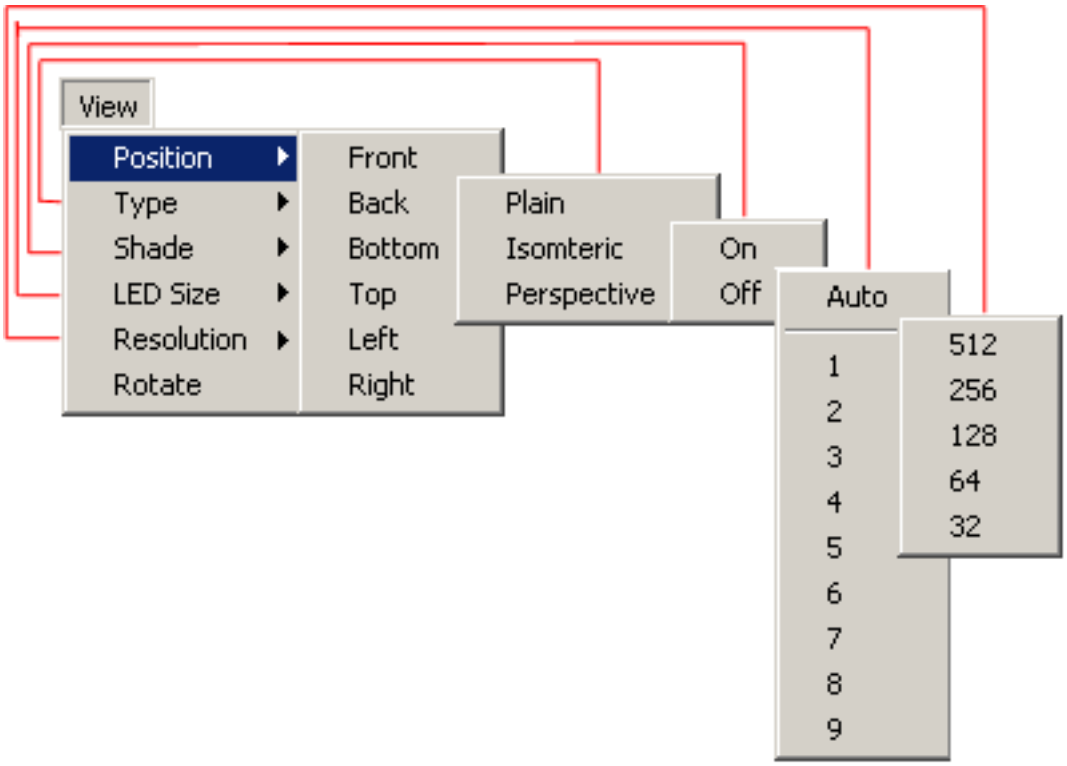
- podmenu **Position**
  Wybór pozycji patrzenia - od przodu, od tyłu itd.
- podmenu **Type**
  Wybór sposobu rzutowania - rzut prosty, rzut ukośny, rzut perspektywiczny.
- podmenu **Shade**
  Włączenie/wyłączenie cieniowania diód w zależności od odległości od patrzącego. Efekt ten nie występuje w rzeczywistym wyświetlaczu, ale poprawia odczucie przestrzenności obrazu po renderingu.

- podmenu **LED Size**
  Ustawianie wielkości pojedynczej diody w pikselach. W przypadku wyboru trybu **AUTO** wielkość ta jest zależna od kąta patrzenia na wyświetlacz - efekt ten występuje w rzeczywistości. Jest on związany z charakterystyką jasności diody w zależności od kąta patrzenia.

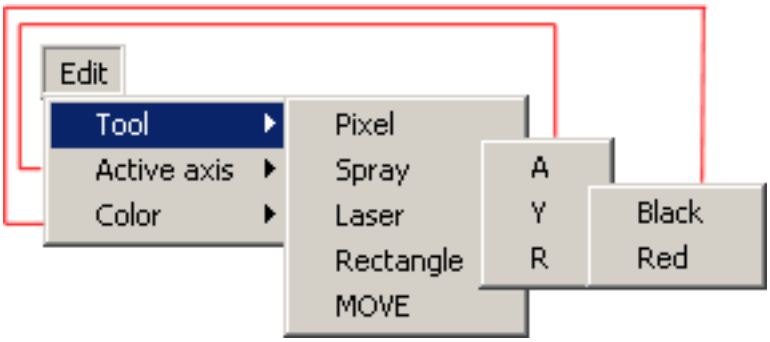
- podmenu **Resolution**
  Ustawienie rozdzielczości kątowej obserwowanego obrazu (ale nie wyświetlacza).

- opcja **Rotate**
  Włączenie/wyłączenie samoczynnego obracania się obrazu.

Wszystkie opcje dostępne w podmenu **View** (z wyjątkiem Rotate) są dostępne również przez paletę narzędzi (rys. 15).



Rys.13 Menu view



Rys.14 Menu Edit

Menu **Edit** (rys. 14) pozwala na wybór narzędzi i ich parametrów, służących do modyfikacji obrazu:

- Tool
  Wybór konkretnego narzędzia.
- Active axis
  Wybór aktywnej osi.
- Color
  Wybór koloru.

Wszystkie opcje dostępne w menu **Edit** dostępne są również z użyciem palety narzędzi (rys. 15). Dodatkową opcją, niedostępną z menu edit, jest zmiana wielkości obszaru działania konkretnego narzędzia. Opcja ta jest dostępna jedynie z **palety narzędzi**. Narzędzi używa się naciskając lewy klawisz myszki na obrazie.

Na obrazie zaznaczona jest siatka układu współrzędnych. Pozwala ona na ograniczanie zasięgu działania narzędzia do pojedynczej płaszczyzny bądź prostej (w szczególności punktu). Dokonuje się tego poprzez wybór aktywnych osi - jeżeli oś jest zaznaczona jako nieaktywna to narzędzia działają tylko w pojedynczej, wyszczególnionej współrzędnej danej osi. Gdy np. aktywne osie to R i Y, wtenczas możliwe jest modyfikowanie obrazu tylko w jednej współrzędnej kątowej. [11]

[11] Opis ten jest niezbyt czytelny bez sprawdzenia działania w praktyce.

Dostępnych jest 5 narzędzi:

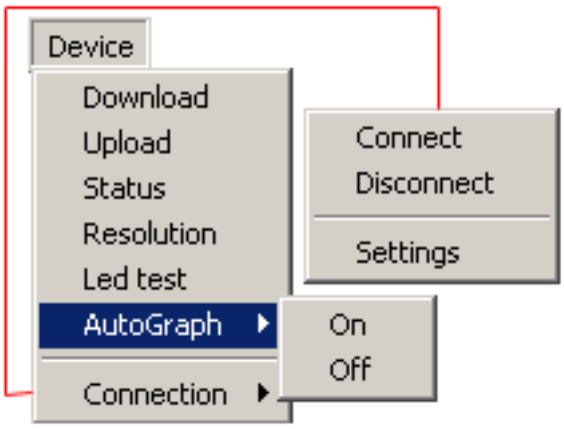
- Pixel
  Modyfikuje pojedynczy piksel.
- Spray
  Działa podobnie jak Laser (patrz niżej), ale modyfikuje jedynie losowo wybrane punkty z obszaru.
- Laser
  Modyfikuje okrągły obszar, o promieniu zależnym od ustawienia Tool size (opcja dostępna z palety narzędzi).
- Rectangle
  Modyfikuje obszar prostokątny, który zaznacza się przeciągając myszkę z wciśniętym lewym klawiszem.
- MOVE
  Pozwala na przesuwanie siatki układu współrzędnych.

Dostępne narzędzia pozwalają na podstawową obróbkę obrazu. Możliwe jest wgrywanie obrazu 'w locie' do wyświetlacza, z użyciem opcji **AutoGraf** z menu Device. Efekt jest bardzo ciekawy dla obserwatora.

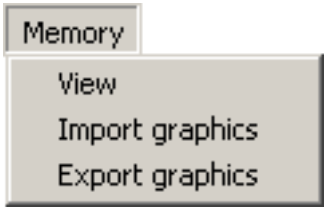
Rys.15 Paleta narzędzi

5.1.2 KOMUNIKACJA Z WYŚWIETLACZEM

Za komunikację z wyświetlaczem odpowiadają polecenia z menu **Device** oraz część paska narzędzi. Dodatkowo w menu **memory** dostępne są opcje pozwalające dokonywać operacji na lokalnej kopii pamięci SRAM wyświetlacza.



Rys.16 Menu device



Rys.17 Menu memory

Rys.18 Pasek narzędzi

Menu **device** udostępnia następujące opcje (ikony z boku oznaczają odpowiadające pozycje **paska narzędzi**) :

- Download 
  Wczytanie z wyświetlacza fragmentu pamięci SRAM do lokalnej kopii.
- Upload 
  Wgranie do wyświetlacza fragmentu pamięci SRAM z lokalnej kopii.
- Status 
  Odczytanie bieżących parametrów pracy wyświetlacza.
- Resolution 
  Ustawienie rozdzielczości pracy wyświetlacza, adresu początku pamięci obrazu, prędkości obracania się obiektu w wyświetlaczu.

- Led test 

Włączenie specjalnego trybu Led Test, sprawdzającego pracę diód za pomocą krótkiej animacji.

- AutoGraph 

Włączenie/wyłączenie specjalnego trybu pracy, w którym na bieżąco modyfikowana jest zawartość pamięci graficznej wyświetlacza tak by wyświetlany obraz był taki sam jak obraz renderowany przez program (część graficzną Sculptora). Wymagane jest tu ustawienie początku pamięci obrazu na zero i ustawienie prędkości obracania się obrazu na zero (ustawienia początkowe są właśnie takie po włączeniu wyświetlacza).

- Connection - settings 

Możliwość wyboru portu szeregowego do którego podłączony jest wyświetlacz (COM1, COM2, COM3, COM4)

- Connection - connect 
Inicjacja komunikacji z wyświetlaczem. Nie powinno się inicjować komunikacji przed włączeniem wyświetlacza.

- Connection - disconnect 
Odłączenie od wyświetlacza.

Menu **memory** udostępnia następujące opcje:

- View 

Podgląd lokalnej kopii pamięci SRAM w trybie heksadecymalnym.

- Import graphics 

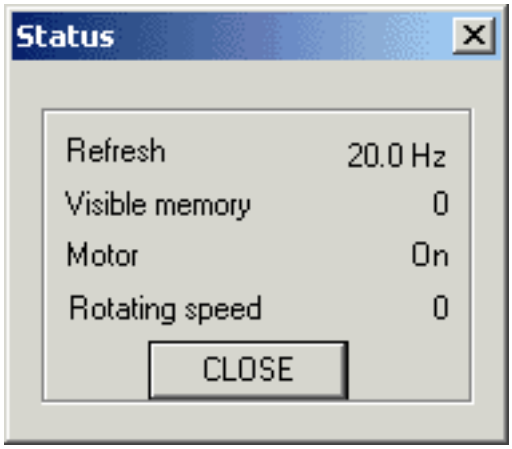
Przekopiowanie bieżąco renderowanego obrazu do określonej lokalizacji w pamięci SRAM.

- Export graphics 

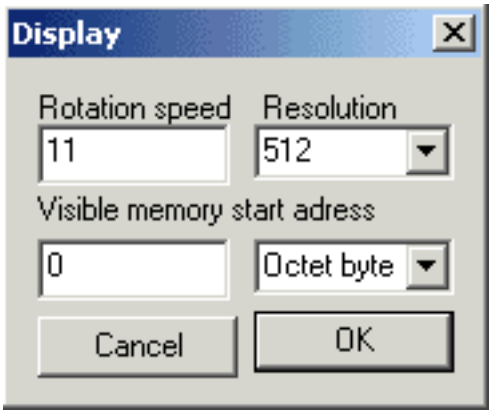
Potraktowanie fragmentu lokalnego obrazu pamięci (o określonej lokalizacji) SRAM jako obrazu i przekopiowanie go do części odpowiedzialnej za renderowanie.

Możliwa jest komunikacja z wyświetlaczem podczas pracy urządzenia, jednak przy transferze dużych ilości danych zalecane jest wyłączenie silnika wyświetlacza - transmisja danych podczas wirowania jest dużo wolniejsza.

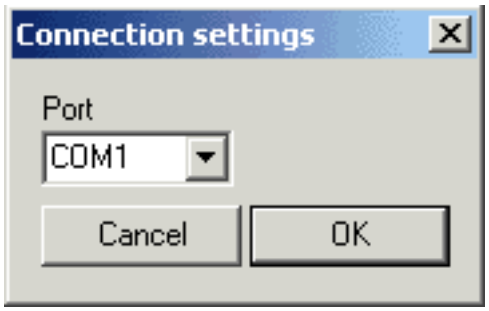
Przy włączaniu programu tworzony jest plik **packet.log** (w bieżącym katalogu). Zawiera on log obrazujący komunikację między środowiskiem Sculptor a urządzeniem. Jest to plik tekstowy.



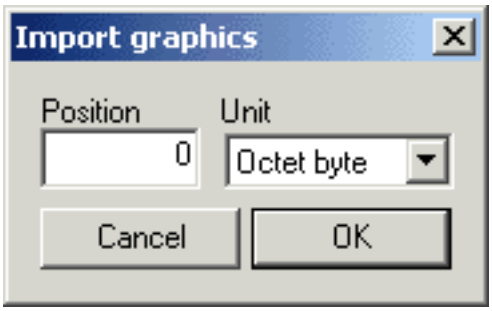
a) Device - status



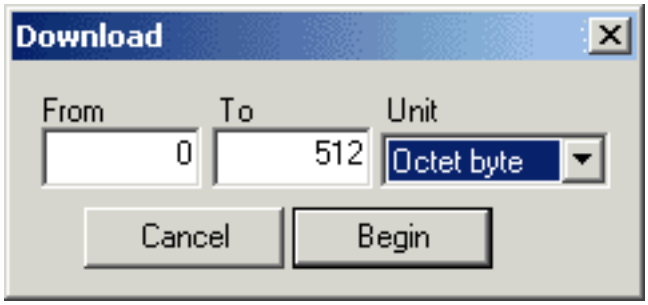
b) Device - display



c) Device - connection - settings



d) Memory - import (export)



e) Device - download (upload)

Rys.19 Okna dialogowe

5.2 FORMATY PLIKÓW

Samo środowisko ma ograniczone możliwości tworzenia obrazów - nie da się np. podać funkcji do wyświetlenia. Aby była możliwość tworzenia własnych obrazów poza środowiskiem Sculptor, konieczna jest znajomość formatów plików jakimi się on posługuje. Występują 2 typy plików:

- *.3d
  pliki z rozszerzeniem **3d** zawierają pojedynczy obraz.
- *.mem
  pliki z rozszerzeniem **mem** zawierają obraz całej pamięci urządzenia, tj. 128kB.

Pliki **3d** składają się z 512 linii, z których każda zawiera 16 cyfr heksadecymalnych (czyli 8 bajtów danych) zakończonych kombinacją 0Dh 0Ah (znak końca linii). Cyfry heksadecymalne to: {0-9, A-F}. Muszą być używane duże litery. Numer linii odpowiada współrzędnej kątowej wyświetlacza (0..512). Każdy z 8-miu bajtów zawiera dane dla innej współrzędnej pionowej wyświetlacza (0..7). Każdy bit takiego bajtu oznacza innej rozdzielczość promieniową - najbardziej znaczący bit to najbardziej wewnętrzny okrąg.

Pliki **mem** składają się z 16 384 lini, z których każda zawiera 8 par po dwie cyfry heksadecymalne poprzedzielane znakiem spacji (20h). Każda linia kończy się kombinacją 0Dh 0Ah (znak końca linii). Do zapisu cyfr heksadecymalnych muszą być używane duże litery (i - oczywiście - cyfry). Pierwsza linia pliku odpowiada początkowi pamięci urządzenia, ostatnia - końcowi.

Do własnych zastosowań celowe wydaje się używanie formatu **3d**. Tworzenie plików **mem** możliwe jest bezpośrednio w środowisku Sculptor na podstawie plików **3d**.

W celu ułatwienia tworzenia plików **3d** poniżej zamieszczony zostaje przykładowy program w języku C tworzący prosty rysunek, a następnie zapisujący go do pliku **3d**.

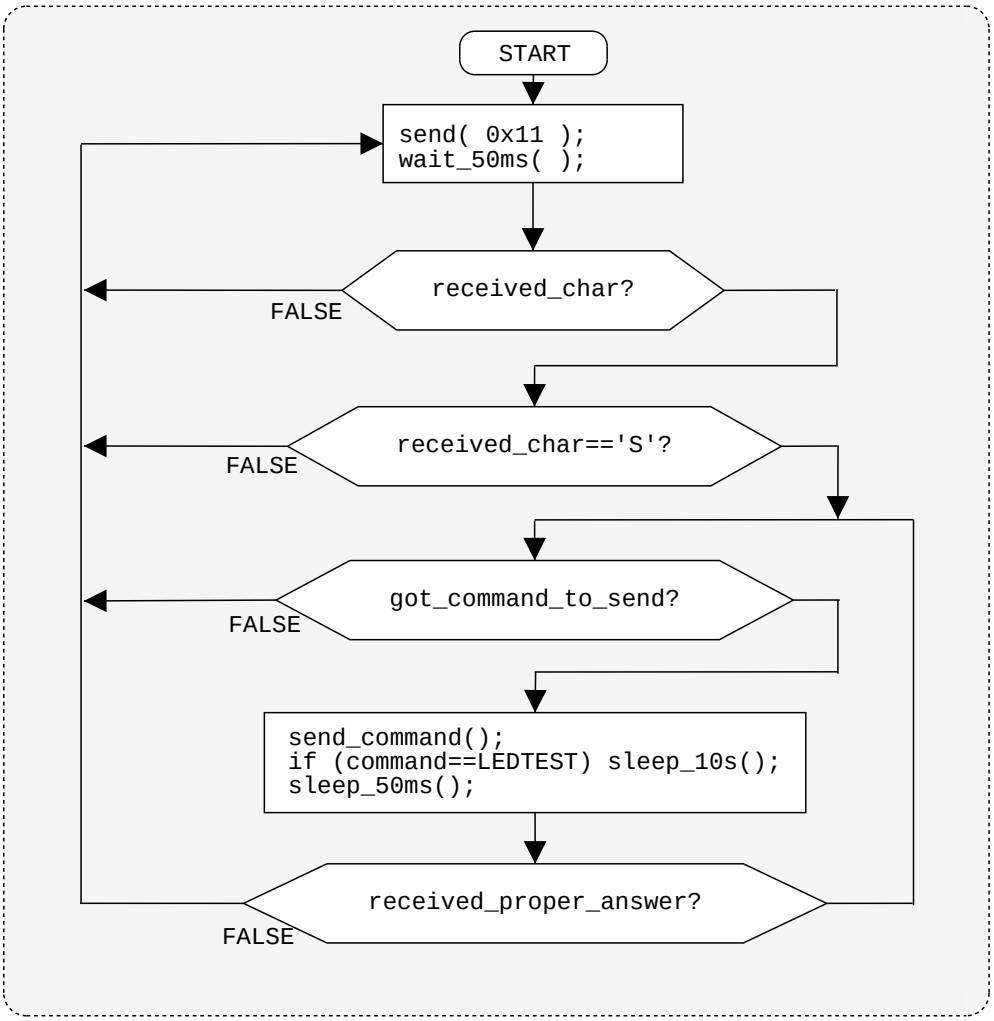
```
#include <stdio.h>
#include <math.h>

unsigned char T[512*8]; /* Tablica zawierająca obraz */

void putpixel(unsigned int a, unsigned int r, unsigned int y)
{
     T[ ( (a%511) *8 + (y%8)]|=1<< (r%8);
}

void main()
{
      FILE *f;
      int i,r,z;
      f=fopen("sin8.3d","w+");
      for( i=0; i<512; i+=2)
           for(r=0; r<8; r++) {
                z = (i+r)/16;
                putpixel(i,r,z);
           }
      for( i=0; i<512; i++) {
           for(r=0; r<8; r++)
                fprintf(f,"%2.2X",T[i*8+r]);
           fprintf(f,"\r\n");
      }
      fclose(f);
}
```

5.3 OPIS PROTOKOŁU KOMUNIKACJI



Rys.20 Schemat algorytmu komunikacji od strony PC

Wyświetlacz pracuje w trybie SLAVE - odpowiada na komunikaty pojawiające się na złączu RS232, samemu nie inicjując komunikacji. Protokół komunikacyjny jest uproszczony do granic możliwości. Zakłada on dzielenie komunikacji na pakiety. Pakiet składa się z 2-bajtowego nagłówka, po którym następują dane. Istnieje też specjalny pakiet synchronizacyjny 1-bajtowy.

Bajt	0		1	...
Bit	0...3	4..7	0..7	...
Znaczenie	KOD	DŁUGOŚĆ	SUMA KONTROLNA	DANE

Rys.21 Struktura pakietu

Pierwsze 4 bity pakietu (z wyjątkiem jednobajtowego pakietu synchronizacji) zajmuje **KOD**. Pozwala on na wybór typu pakietu:

- Pakiet **WRITE** | KOD 1h
  Pozwala na pisanie do pamięci obrazu.
- Pakiet **SYNC** | KOD 1*h
  Jest to pakiet wyjątkowy. Jest on jednobajtowy, i jego kod jest taki sam jak pakietu WRITE - rozróżnienie następuje na podstawie pola DŁUGOŚĆ. Cały pakiet składa się zatem z jednego bajtu - 11h. Pakiet ten służy do synchronizacji komunikacji w przypadku zgubienia danych.
- Pakiet **READ** | KOD 4h
  Pozwala na czytanie z pamięci obrazu.
- Pakiet **ANIM** | KOD 2h
  Pozwala na ustalenie trybu pracy wyświetlacza (rozdzielczość, początek pamięci obrazu, prędkość obracania obrazu).
- Pakiet **STATUS** | KOD 3h
  Pozwala na pobranie z urządzenia adresu początku pamięci obrazu, prędkości obrotowej wirnika itd.
- Pakiet **CLEAR** | KOD 7h
  Pozwala na wyczyszczenie zawartości pamięci RAM.
- Pakiet **LEDTEST** | KOD 5h
  Włącza urządzenie w specjalny tryb diagnostyczny LED TEST, który pozwala na łatwe testowanie poprawności działania diód. Jest to zarazem bardzo wygodny komunikat do testowania komunikacji w kierunku do-wyświetlacza.

Jak widać, tylko 6 z 16 możliwych kodów jest w użyciu. Nadmierne przeładowywanie protokołu komunikatami o dyskusyjnej użyteczności zostało przez autora uznane za niecelowe.

Pole **DŁUGOŚĆ** zawiera długość pakietu w bajtach.

Pole **SUMA KONTROLNA** jest wykorzystywane do sprawdzania poprawności komunikacji. Wartość tego pola jest dobrana tak, by po przeprowadzeniu operacji XOR wszystkich bajtów paczki wynik był równy zero. W przypadku, gdy suma kontrolna jest niepoprawna wysyłany jest komunikat o błędzie transmisji.

Dokładny opis pakietów jest następujący:

- Pakiet **WRITE** | KOD 1h

Bajt	0	1	2	3	4..11
Zawartosc	1Ch		AdrHI	AdrLO	Data0...Data7

  Pakiet pozwala na zapis do pamięci obrazu. Adres jest podawany poprzez kombinację AdrHI (górny bajt adresu) oraz AdrLO (dolny bajt adresu). Tu uwaga - pamięć nie jest adresowana pojedynczymi bajtami, ale ósemkami bajtów (tj. zmiana najmłodszego bitu AdrLO powoduje przesunięcie się w pamięci rzeczywistej o 8 bajtów). Po adresie następuje ciąg ośmiu bajtów, które zostaną zapisane do pamięci. Po pomyślnym dokonaniu zapisu zwracany jest znak 'W' (57h).

- Pakiet **SYNC** | KOD 1h

Bajt	0
Zawartosc	11h

  Pakiet ten pozwala na synchronizację komunikacji po wystąpieniu błędów. Po odebraniu tego pakietu zwracany jest znak 'S' (53h). W przypadku wystąpienia jakiegoś błędu pakiet ten powinien być wysyłany co 50 ms tak długo, aż uzyskana zostanie prawidłowa odpowiedź.

- Pakiet **READ** | KOD 4h

Bajt	0	1	2	3
Zawartosc	44h		AdrHI	AdrLO

  Pakiet pozwala na odczyt z pamięci obrazu. Adres jest podawany poprzez

kombinację AdrHI (górny bajt adresu) oraz AdrLO (dolny bajt adresu). Tu uwaga - pamięć nie jest adresowana pojedynczymi bajtami, ale ósemkami bajtów (tj. zmiana najmłodszego bitu AdrLO powoduje przesunięcie się w pamięci rzeczywistej o 8 bajtów). Zwracany jest pakiet następujący:

Bajt	0	1...8	9
Zawartosc	'R' (52h)	Data0...Data7	Suma kontrolna

- Pakiet **ANIM** | KOD 2h

Bajt	0	1	2	3	4,5
Zawartosc	26h		Res	RotAft	AdrHi,Lo

Pakiet ten pozwala na ustawienie następujących parametrów:

Res - bity 6,5,4,3 służą do ustalenia rozdzielczości pracy wyświetlacza, pozostałe są ignorowane. Rozdzielczość wynikowa dana jest wzorem:

Res=512/(1+x), gdzie x to liczba zawarta w bitach.

Rozdzielczości nie-całkowite są dozwolone, jakkolwiek nie są zalecane.

RotAft - prędkość obracania obrazu w wyświetlaczu (w pikselach kątowych/obrót)

AdrHi,Lo - początek pamięci wyświetlanej obrazu (w jednostce 8-bajtowej)

Po wykonaniu tej operacji zwracany jest znak 'A' (41h).

- Pakiet **STATUS** | KOD 3h

Bajt	0	1
Zawartosc	32h	32h

Pakiet ten pozwala na pobranie stanu wyświetlacza. Zwracany jest następujący pakiet:

Bajt	0	1	2	3,4	5,6	7	8
Zaw.	'S'(53h)	T2CON	PR2	AdrLO,HI	AlfaLo,Hi	RotAft	Ctrl

T2CON - zawartość rejestru T2CON. Pozwala na stwierdzenie rozdzielczości kątowej wyświetlacza oraz czy jest włączony silnik.

Bit	7	6..3	2	1,0
Zawartosc	0	Postscale	Timer On	01

Pole **Postscale** pozwala na odczytanie rozdzielczości kątowej wyświetlacza wg. wzoru Res=512/(1+Postscale). Pole **Timer On** jest wyzerowane jeśli wirnik się nie obraca, lub obraca zbyt wolno.

PR2 - zawartość rejestru PR2, pozwalającego na ocenę szybkości obracania się wirnika wg wzoru freq = 5000/(2048*PR2) kHz;

AdrLo,Hi - początek pamięci wyświetlanej obrazu (w jednostce 8-bajtowej);

AlfaLo,Hi - aktualna pozycja kątowa wyświetlacza (przy najwyższej rozdzielczości w przedziale 0..511);

RotAft - prędkość obracania obrazu w wyświetlaczu (w pikselach kątowych/obrót);

Ctrl - suma kontrolna. Operacja XOR na bajtach 1,2,...,8 musi dawać zero;

- Pakiet **CLEAR** | KOD 7h

Bajt	0	1
Zawartosc	72h	72h

Pakiet ten powoduje wyzerowanie zawartości pamięci graficznej (SRAM) wyświetlacza. Po wykonaniu tej operacji zwracany jest znak 'C' (43h).

- Pakiet **LEDTEST** | KOD 5h

Bajt	0	1
Zawartosc	52h	52h

Pakiet ten powoduje uruchomienie procedury testowej, wyświetlającej prostą animację na diodach. Podczas działania tej procedury wyłączone są

przerwania. Po odebraniu tego rozkazu (ale przed wywołaniem procedury) zwracany jest znak 'L' (4Ch). Animacja trwa pewien czas (kilka sekund), przez który komunikacja z wyświetlaczem powinna być zaniechana.

5.4 ORGANIZACJA PAMIĘCI WYŚWIETLACZA

Z uwagi na maksymalne uproszczenie algorytmu odpowiedzialnego za realizację wyświetlania, organizacja pamięci graficznej staje się nieco złożona. Otóż każda pozycja kątowa wirnika ma przyporządkowane jej 8 bajtów danych (odpowiadającym 64 diodom). Pozycja kątowa wirnika nie jest jednak tożsama z pozycją w cylindrycznym układzie współrzędnych - tożsamość zachodzi jedynie dla najniżej położonego paska diód. Wynika to z faktu, że kolejne paski diód są rozłożone co 45 stopni - i o wielokrotności tych 45 stopni współrzędne cylindryczne odbiegają od pozycji kątowej wirnika.

Co gorsza, sprawę komplikuje fakt iż wysokość pasków diód była dobierana w sposób mający minimalizować odchylenia wirnika od środka ciężkości. Na rysunku 22 przedstawiona jest zależność wysokości, offsetu w pamięci (w bajtach) oraz przesunięcia kątowego w stopniach. Jest ona podana dla pierwszych 8 bajtów pamięci, gdzie pole **offset** wyznacza pozycję w tych 8 bajtach. Jak łatwo zauważyć, pole **kąt** jest w liniowej zależności od pola **offset**. Kolejne ósemki bajtów mają pozycję kątową zwiększoną o kąt zależny od rozdzielczości wyświetlacza (wg. zależności kąt = 360/rozdzielczość). Na następnej stronie zamieszczony jest przykładowy kod w C ilustrujący sposób adresowania pamięci.

h	offset	kąt
0	0	0
1	4	180
2	5	225
3	1	45
4	7	315
5	3	135
6	2	90
7	6	270

Rys.22 tabela pomocnicza do adresowania pamięci

```
#define MEMORY_SIZE 128*1024

int Res=512; \* Rozdzielczość kątowa (512, 256, 128, 64, 32) *\
byte memory[MEMORY_SIZE];

\* Zapala punkt o współrzędnych (a,y,r) *\
void putvoxel(int a, int y, int r)
{
    static int hTable[]={ 0, 4, 5, 1, 7, 3, 2, 6};
    int offset = ( (hTable[y]+1)*Res + 8*a ) % (Res*8);
    memory[ VisibleMemoryStart + offset ] |= 1<<r;
}
```

5.5 ZAWARTOŚĆ ZAŁĄCZONEJ PŁYTKI CD

Do pracy została dołączona płytka CD, zawierająca:

- fragmenty stron do których istnieją odniesienia w treści pracy w katalogu **HTML**;
- zewnętrzną dokumentację do elementów użytych w pracy w katalogu **PDF**;
- źródła programu sterującego pracą wyświetlacza w katalogu **PICASM**;
- projekt części elektronicznej w katalogu **PROTEL**;
- środowisko Sculptor (wraz z źródłami) w katalogu **SCULPTOR**;
- źródło pracy dyplomowej w katalogu **TEX** ;
- przykładowe pliki z obrazami w katalogu **3D**.

6 PODSUMOWANIE

Stworzony system wyświetlający spełnił pokładane w nim oczekiwania - udało się uzyskać na nim obraz trójwymiarowy oraz proste efekty animacji. Wszystkie założenia projektowe udało się w pełni zrealizować, a w niektórych przypadkach nawet przekroczyć zakładane parametry.

Przyjęte ryzykowne założenie, iż komunikację szeregową da się przeprowadzić poprzez szczotki zweryfikowała praktyka. Jest to możliwe, jednak z uwagi na dużą liczbę przekłamań prędkość transmisji maleje około dziesięciokrotnie. Jest to niestety uciążliwe. Zastosowanie szczotek było wariantem najtańszym. W zastosowaniach profesjonalnych konieczna byłaby komunikacja np. z użyciem diód IR.

Udało się uzyskać odświeżanie o częstotliwości około 30Hz (przy zakładanym 25Hz), co dobrze eliminuje efekt 'migotania' obrazu. Brak stabilizacji prędkości obrotowej nie przeszkadza w wyświetlaniu. Inne założone parametry pracy zostały również osiągnięte, a nawet przekroczone. Udało się uzyskać rozdzielczość kątową 512 voxeli/kąt pełny. Zakładana rozdzielczość maksymalna wynosiła 128 voxeli/kąt pełny. Początkowe obawy czy wirnik wytrzyma prędkość obrotową na poziomie 1800rpm okazały się płonne.

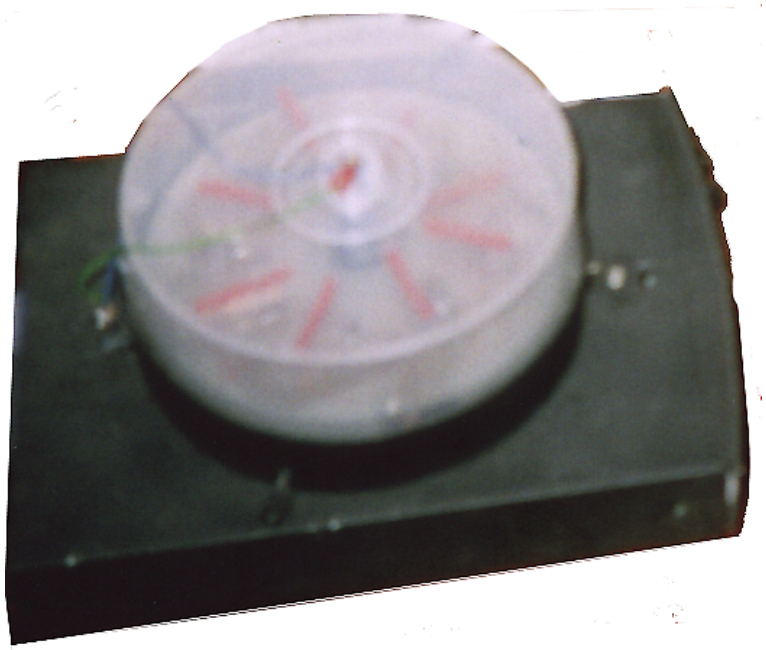
System niezbyt efektownie prezentuje się podczas pracy w świetle dziennym. Swiatło emitowane przez diody jest wówczas słabo widoczne. Natomiast podczas pracy w ciemnym pomieszczeniu uzyskuje się bardzo efektowny wygląd. Profesjonalne systemy również mają tę wadę.

Charakterystyka świecenia diód LED niestety wpływa destrukcyjnie na jakość obrazu. Diody z boku widoczne są jako 'krótka czerwona linia'. z góry natomiast jak 'czerwony okrąg'. Efekt ten nie był zaskoczeniem, niemniej jednak uniknięcie go wymagałoby użycia diód o innej charakterystyce świecenia, niestety droższych. Wybrane zostało rozwiązanie tańsze - użycie typowych diód LED.

System nie nadaje się do zastosowań profesjonalnych z uwagi na zbyt niską rozdzielczość wyświetlacza (8x8x512, 2 kolory). Jak łatwo obliczyć, pamięć obrazu w takiej rozdzielczości wynosi 4 kB. Dla porównania - Zx Spectrum

miał 6.75 kB. Jest to zdecydowanie za mało by zademonstrować główną zaletę wyświetlacza trójwymiarowego, którą jest szybkość analizy przez człowieka danych w ten sposób prezentowanych.

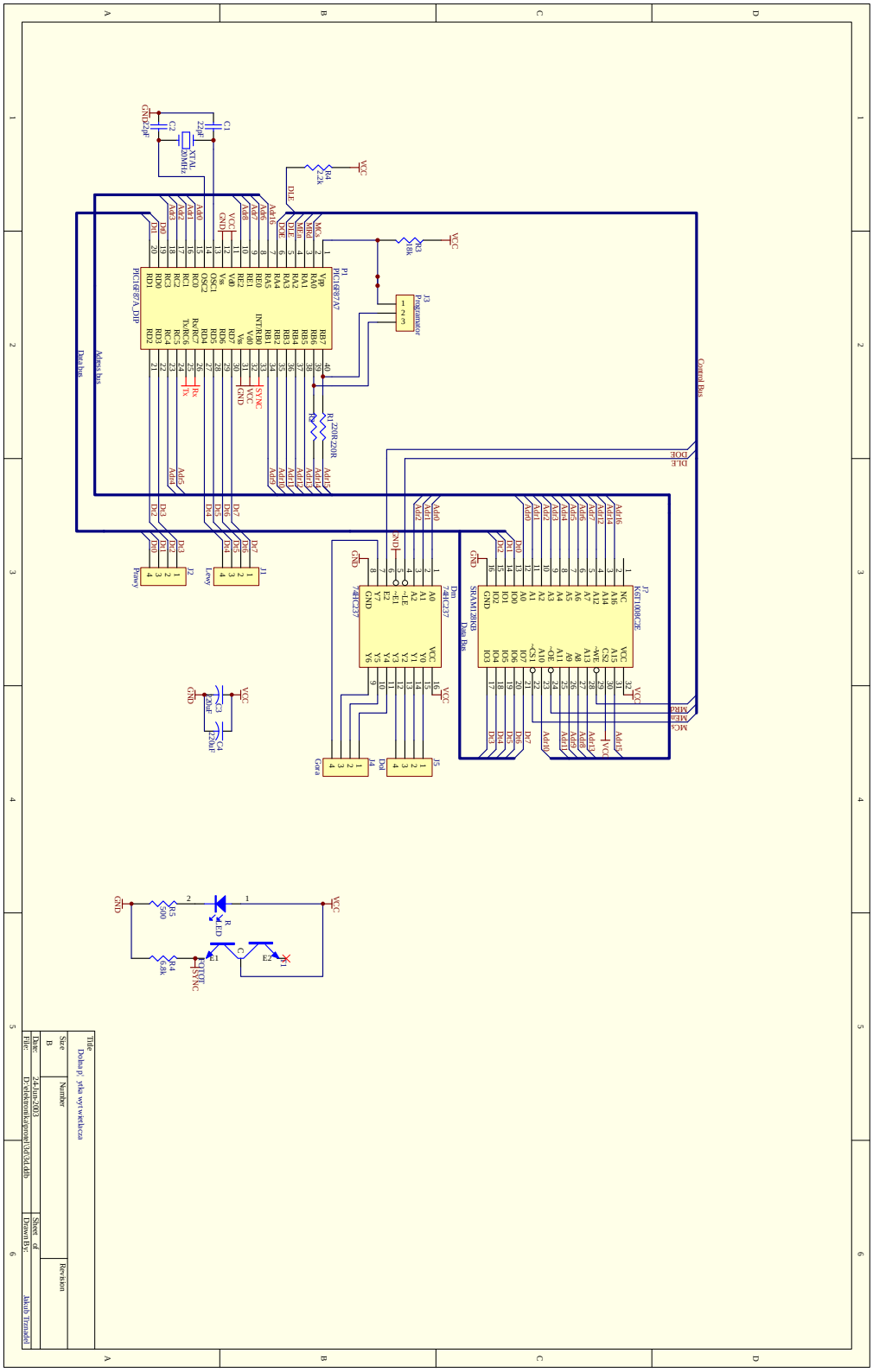
Wyświetlacz spisuje się natomiast świetnie jako demonstracja możliwości techniki - tego co można jeszcze osiągnąć. Pokazuje że nawet za pomocą łatwo dostępnych elementów można stworzyć coś nowego. Cały system został skonstruowany kosztem około 300zł, gdyż jego duża część została stworzona z elementów pochodzących 'z odzysku' - z innych, uszkodzonych urządzeń. Wątek ekonomiczny był obecny w całym procesie tworzenia wyświetlacza.

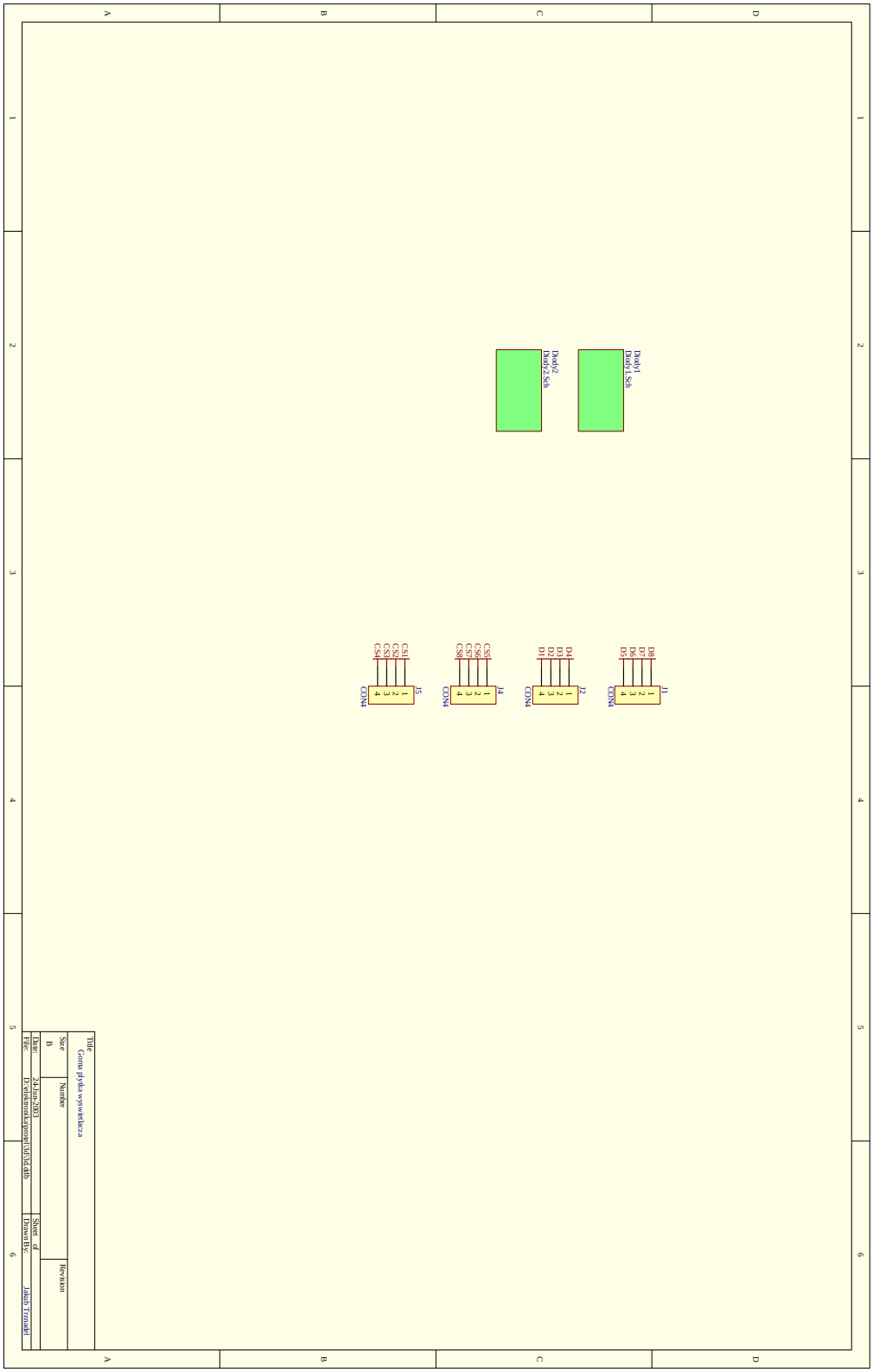


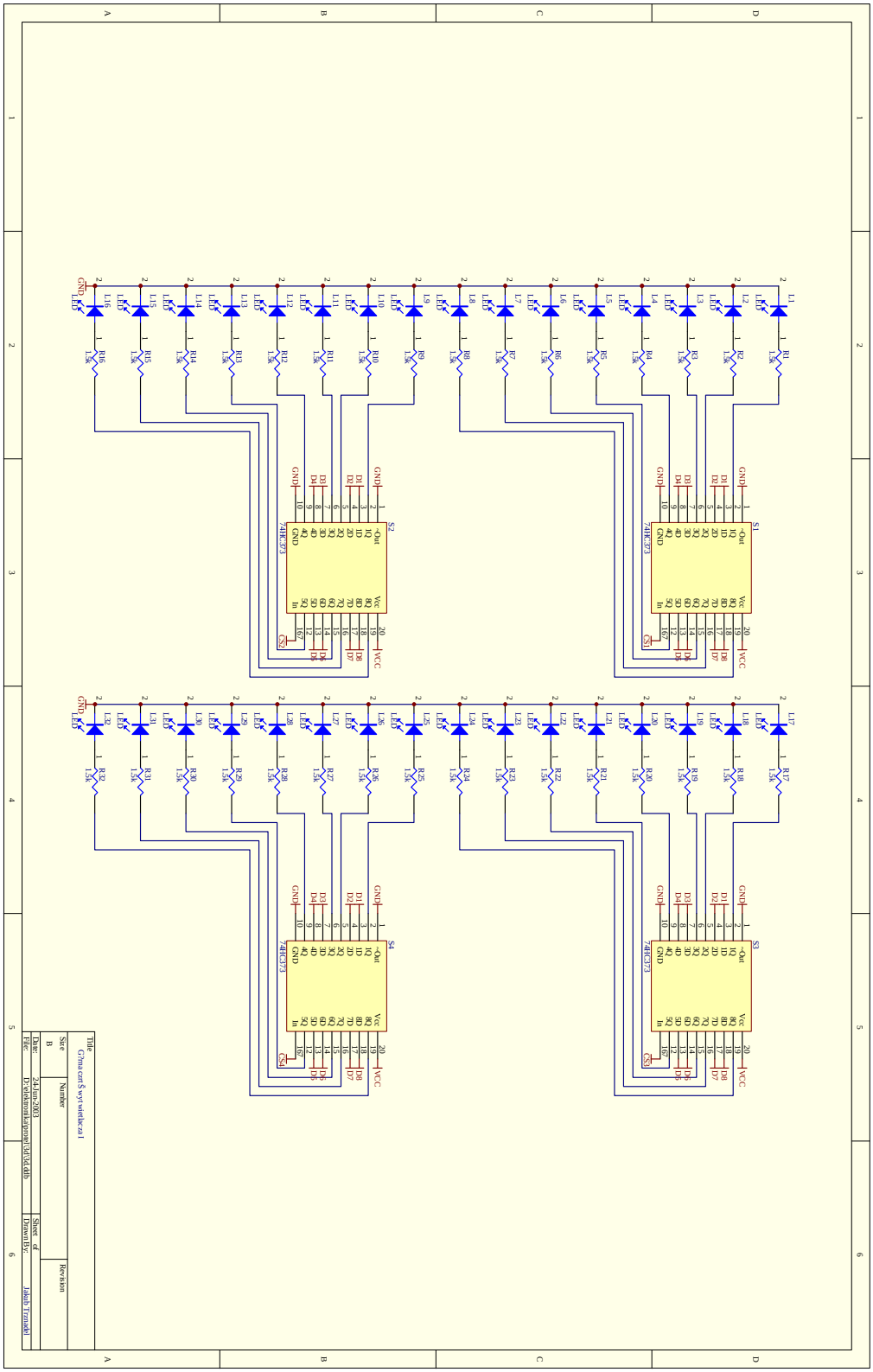
Rys.23 Zdjęcie wykonanego wyświetlacza w stanie spoczynku

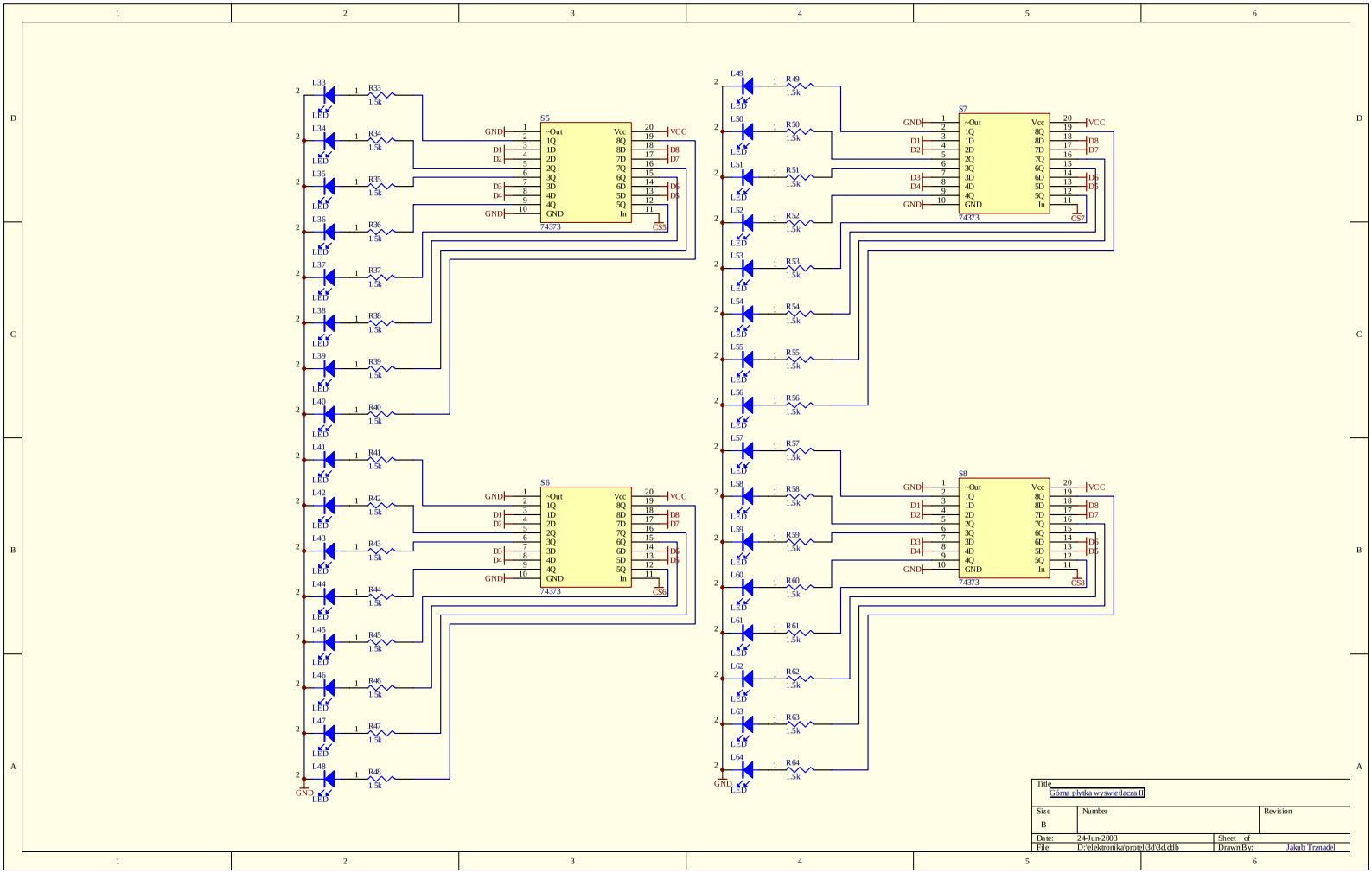
Załącznik A

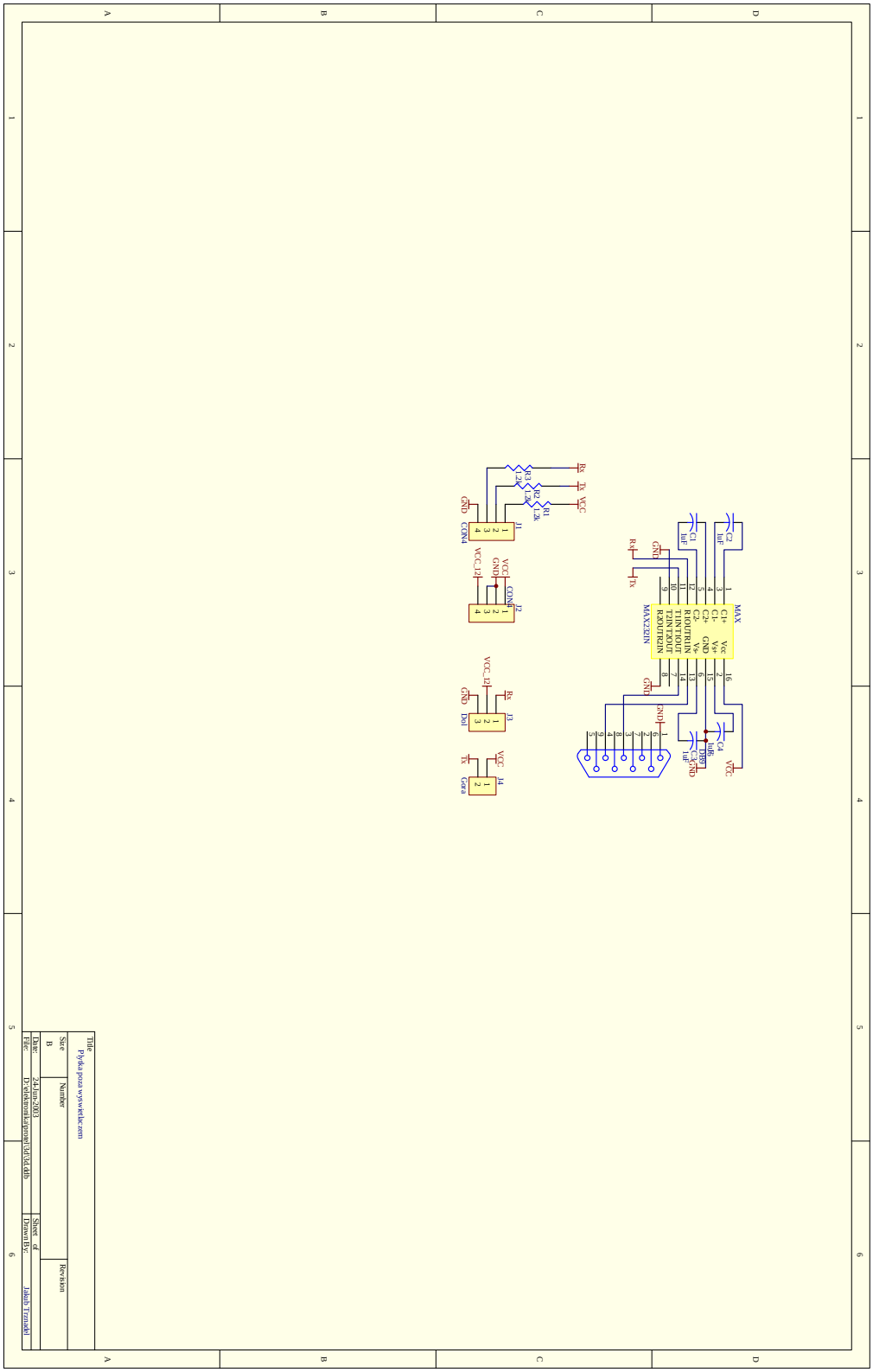
SCHEMATY CZĘŚCI ELEKTRONICZNEJ


Title
Deska pł. syda wyświetlacza
Size
B
Number
Revision
Date:
24-Jun-2003
File:
D:\elektronika\projekt\3d\3d.ddb
Sheet of
Drawn By:
Jakub Tiznadel
P1
PIC16F877A/CP
PIC16F87A_DIP
J3
Programmer
R1
220R/220R
R3
10k
R4
2.2k
C1
22pF
C2
22pF
XTAL
4.000MHz
C3
120nF
C4
120nF
R5
500
R4
6.8k
LED
Q1
Q2
J7
AS1100MC2E
SRAM128Kb
Data Bus
Address bus
Control Bus
74HC237
Prawy
Lewy
Gora
Dol
VCC
GND
DLE
DOE
MCS
MWE
MRD


Diody1
Diody1.Sch
Diody2
Diody2.Sch
J1
CON4
J2
CON4
J4
CON4
J5
CON4
Title
Gorna plyska wyswietlacza
Size
B
Number
Revision
Date:
22-Lip-2003
Sheet of
File:
D:\elektronika\projekt\3d\3d.ddb
Drawn By:
Jakub Trzuaskl


Tytuł
Gra na czas Świetlica 1
Rozmiar
B
Numer
Wersja
Data: 24-lip-2003
Plik: D:\elektronika\projekt\3d\3d.ddb
Arkusz z
Rysował:
Jakub Trzuszki


L33
R33
1.5k
LED
L34
R34
L35
R35
L36
R36
L37
R37
L38
R38
L39
R39
L40
R40
L41
R41
L42
R42
L43
R43
L44
R44
L45
R45
L46
R46
L47
R47
L48
R48
GND
S5
~Out
1Q
1D
2D
2Q
3Q
3D
4D
4Q
Vcc
8Q
8D
7D
7Q
6Q
6D
5D
5Q
In
74373
VCC
D1
D2
D3
D4
D8
D7
D6
D5
CS5
S6
CS6
L49
R49
L50
R50
L51
R51
L52
R52
L53
R53
L54
R54
L55
R55
L56
R56
L57
R57
L58
R58
L59
R59
L60
R60
L61
R61
L62
R62
L63
R63
L64
R64
S7
CS7
S8
CS8
Title
Górna płytka wyswietlacza II
Size
B
Number
Revision
Date:
24-Jun-2003
Sheet of
File:
D:\elektronika\protel\3d\3d.ddb
Drawn By:
Jakub Trznadel


Title
Płytka pozaWyswietlacz.sch
Size
B
Number
Revision
Date:
22-Jun-2003
Sheet of
File:
D:\elektronika\pomoc\3d\3d.ddb
Drawn By:
Jakub Trzuaskl
MAX
MAX232IN
R2OUT
R2IN
T1IN
T1OUT
T2IN
T2OUT
R2OUTB/N
Vs+
Vs-
GND
Vcc
C1+
C1-
C2+
C2-
R1OUT
R1IN
C1
1uF
C2
1uF
C3
1uF
C4
100nF
VCC
GND
Rx
Tx
J1
CON4
J2
CON2
J3
Dio1
J4
Gora
VCC_12
R1
1.2k
R2
1.2k
R3
1.2k

Załącznik B

LITERATURA

[1] "PIC16F87XA Data Sheet", [12]
Microchip Technology Inc., 2003;

[2] "MM74HC373 3-STATE OCTAL D-Type Latch", [12]
Fairchild Semiconductor, 1999;

[3] "CD74HC137, CD74HCT137, CD54HC237, CD74HC237, CD54HCT237", [12]
Texas Instruments, 2003;

[4] "128Kx8 bit Low Power CMOS Static RAM", [12]
Samsung Electronics, 2000;

[5] "Embeded Control Handbook, Volume 1",
Microchip Technology Inc., 1997;

[6] "Programming Windows", Charles Petzold,
1998, Microsoft Press;

[7] "Spatial 3D: The End of Flat-Screen Thinking", [12]
Actuality Systems, Inc., 2003;

[8] "Programator JuPic", [12]
http://ajpic.zonk.pl/;

[11] Dokument znajduje się na załączonej do pracy płytce CD