

UNIWERSYTET WARMIŃSKO-MAZURSKI W OLSZTYNIE
WYDZIAŁ MATEMATYKI I INFORMATYKI

Kierunek: Informatyka

Michał Bartosz Ludwikowski

**Opracowanie automatycznego systemu
chłodzącego stacji roboczej z wykorzystaniem
Arduino**

Praca inżynierska wykonana
w katedrze Matematycznych Metod Informatyki
pod kierunkiem
dra Krzysztofa Sopyły

Olsztyn, 2016 rok

UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Field of Study: Computer Science

Michał Bartosz Ludwikowski

Developing automatic cooling system for workstation using Arduino

Engineer's Thesis is performed
in the Department of Mathematical Methods of Informatics
under supervision of
dr Krzysztof Sopyła

Olsztyn, 2016

Spis treści

Streszczenie	2
Abstract	3
Rozdział 1. Wstęp	4
1.1. Wprowadzenie	4
1.2. Cel pracy	4
1.3. Układ pracy	4
Rozdział 2. Opis technologii wykorzystanych podczas realizacji projektu	6
2.1. Arduino	6
2.1.1. Język programowania Arduino	6
2.1.2. Arduino Software (IDE)	7
2.2. Narzędzia do projektowania elektroniki	8
2.2.1. Eagle	8
2.2.2. Fritzing	8
2.3. CAD	9
2.3.1. AutoCAD	9
2.4. Narzędzia do testów wydajności	10
2.4.1. OCCT	10
Rozdział 3. Realizacja projektu systemu chłodzącego	11
3.1. Projekt techniczny	11
3.1.1. Projekt systemu chłodzącego	11
3.1.2. Wykonanie systemu chłodzącego	11
3.2. Elektroniczny układ współpracujący z Arduino	12
3.2.1. Projekt układu elektronicznego	15
3.2.2. Wykonanie układu elektronicznego	17
3.3. Implementacja sterownika układu chłodzącego na platformie Arduino	19
3.3.1. Opis algorytmu głównego	19
3.3.2. Implementacja algorytmu głównego	19
Rozdział 4. Testy wydajnościowe i analiza wyników	25
4.1. Testy wydajnościowe chłodzenia podzespołów	25
4.1.1. Opis testowanego sprzętu	25
4.1.2. Opis eksperymentu	25
4.2. Analiza wyników	25
Rozdział 5. Podsumowanie	32
Bibliografia	33
Spis rysunków	34
Lista algorytmów	35
Spis tablic	36

Streszczenie

Moc obliczeniowa komputerów znacznie wzrosła na przestrzeni ostatnich lat. Wraz ze wzrostem mocy obliczeniowej wzrosła również ilość ciepła generowana przez podzespoły komputera. Jest to oczywiście zjawisko niekorzystne i niepożądane. Jak wiadomo urządzenia elektroniczne pod wpływem ciepła przestają pracować prawidłowo, a nadmierne ich nagrzanie może doprowadzić do całkowitego ich uszkodzenia.

Komputery stacjonarne chłodzone są najczęściej powietrzem. Wykorzystuje się do tego radiatory i wentylatory. Ostatnimi czasy pojawiły się rozwiązania wykorzystujące ciecz do transportowania ciepła, które działają lepiej niż rozwiązania bazujące na chłodzeniu powietrzem, lecz te rozwiązania odprowadzają ciepło tylko z punktów, do których przyczepione są elementy układu chłodzącego, przez co nie zabezpieczają kompleksowo podzespołów komputera.

Poniższa praca stara się rozwiązać problem poprzez zaprojektowanie oraz wykonanie automatycznego systemu chłodzącego, który odprowadza ciepło z całej powierzchni elektroniki poprzez zanurzenie jej w cieczy, która nie przewodzi prądu.

Stworzone rozwiązanie wykorzystujące platformę Arduino, układ chłodzący oraz algorytm sterujący pozwala na skuteczne odprowadzanie ciepła z podzespołów komputera o czym świadczą przeprowadzone eksperymenty, co może pozwolić na zwiększenie wydajności podzespołów komputera.

Abstract

Computing power of machines has increased significantly in recent years. With the increase in computing power has also increased the amount of heat generated by the computer's components. This is obviously disadvantageous and undesirable phenomenon. As it is known, electronic devices stop working properly due to high temperatures, and excessive overheating can result in complete breakage.

Desktop PC's are usually cooled by air. They use heat sinks and fans to cool inner components. Recently solutions based on the fluid for transporting heat appeared, which works better than solutions based on air cooling which only dissipate the heat from the points where cooling system components are mounted, and cannot completely protect the system components.

The following work presents how to solve the problem by designing and creation of an automatic cooling system which removes heat from the entire surface of electronics by immersing it in a liquid that does not conduct electricity.

Solution was created using the Arduino platform, cooling system and control algorithm allows efficient heat dissipation from the computers components, as evidenced in performed experiments, which may increase efficiency of computer components

Rozdział 1

Wstęp

1.1. Wprowadzenie

Podzespoły obecnych komputerów z racji dążenia do coraz większej mocy obliczeniowej stacji roboczych generują duże ilości ciepła. Do elementów stacji roboczej, które generują to ciepło można zaliczyć: procesor, kartę graficzną, zasilacz, pamięć operacyjną oraz elementy płyty głównej takie jak sekcja zasilania procesora czy mostek północny.

Chłodzenie podzespołów komputera to bardzo istotny aspekt poprawnej pracy stacji roboczej. W dzisiejszych czasach najpopularniejszym sposobem chłodzenia są zaawansowane radiatory chłodzone powietrzem aktywnie (zastosowanie wentylatora) lub pasywnie (brak wentylatora). Coraz popularniejsze stają się bloki wodne, które zastępując radiatory umożliwiają wymianę ciepła podzespołu z cieczą. Sposób ten pozwala jedynie na przekazanie ciepła z ograniczonych powierzchni elementów, z których zbudowane są podzespoły komputera. Rozwiązaniem, które pozwoli zwiększyć te powierzchnie jest zanurzenie całego komputera w chłodziwie.

1.2. Cel pracy

Niniejsza praca ma na celu zaprojektowanie oraz wykonanie automatycznego systemu chłodzącego stację roboczą (Rysunek 1.1). W pracy poza opisami narzędzi takich jak Arduino wykorzystanych do realizacji projektu znajdziemy bogato ilustrowane opisy poszczególnych kroków dążących do stworzenia skutecznego systemu chłodzącego. Skuteczność systemu omówiona i podsumowana zostanie na wykresach temperatur podzespołów zebranych podczas wzmożonej pracy stacji roboczej.

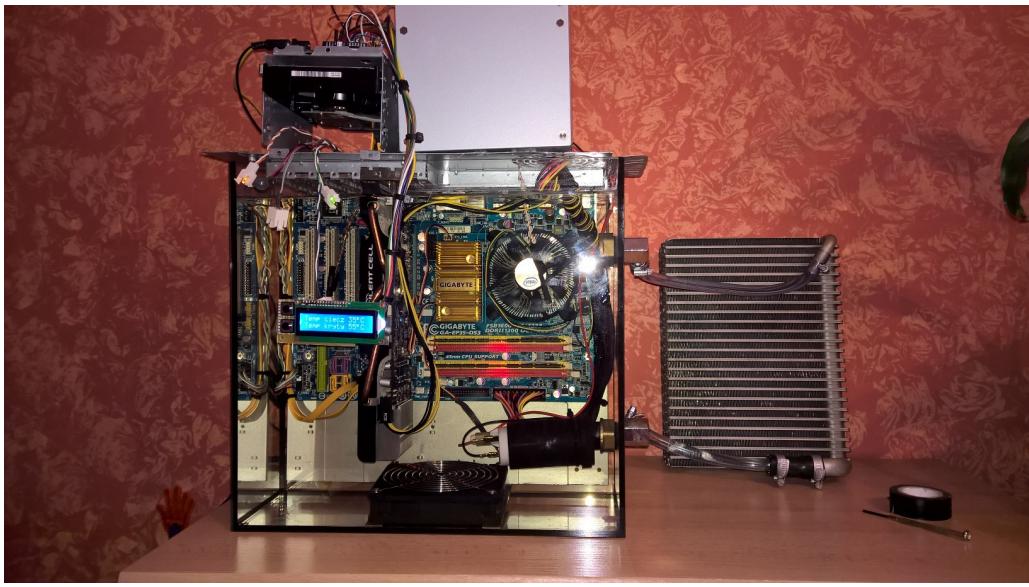
1.3. Układ pracy

Praca została podzielona na pięć rozdziałów. Kolejne rozdziały to wstęp, opis użytych technologii, sprawozdanie z wykonania projektu, testy skuteczności systemu chłodzącego oraz podsumowanie.

W drugim rozdziale opisane zostały technologie, przy pomocy których wykonano poniższy projekt. Umieszczono tam opis platformy Arduino, programów wykorzystywanych przy projektowaniu układów elektronicznych, projektowania wspomaganego komputerowo oraz programów wykorzystywanych podczas testowania skuteczności systemu chłodzącego.

Rozdział trzeci jest sprawozdaniem z procesu projektowania oraz wykonania poszczególnych elementów układu chłodzącego. Rozdział ten podzielony został na trzy sekcje dotyczące:

- układu elektronicznego, który współpracuje z Arduino,
- algorytmu sterownika,
- układu chłodzącego.



Rysunek 1.1. Gotowy projekt

Testy skuteczności zbudowanego układu zawarte zostały w rozdziale czwartym. Przedstawiono tutaj wykresy pomiaru temperatur rdzeni procesora komputera i porównano je z danymi zebranymi przed wykonaniem systemu.

Ostatni rozdział jest podsumowaniem całego projektu. Przedstawiono tu ogólną ocenę zbudowanego systemu oraz uwagi, dotyczące zastosowanych rozwiązań.

Rozdział 2

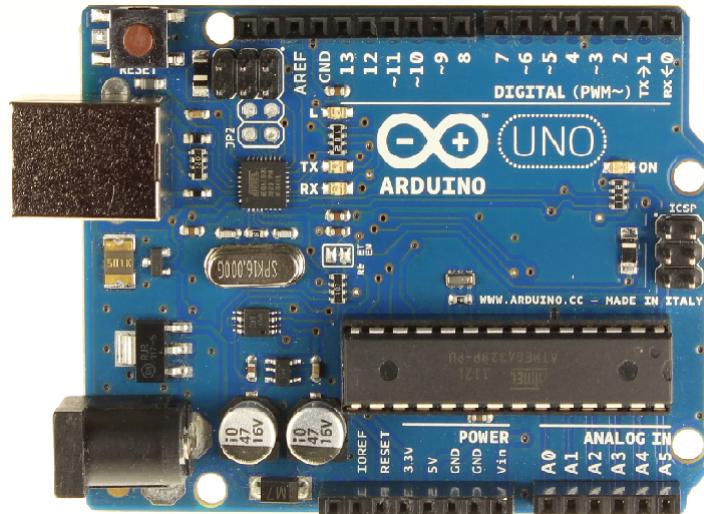
Opis technologii wykorzystanych podczas realizacji projektu

2.1. Arduino

Arduino[1] jest platformą prototypową typu open source, opartą na łatwym w użyciu hardware oraz software. Płytki Arduino (Rysunek 2.1) są w stanie odczytywać dane różnego pochodzenia - światło padające na sensor, prycięnięty przycisk, dane z sieci - i przekształcać je w żądane sygnały wyjścia - zapalenie diod LED, opublikowanie czegoś w sieci czy załączenie silnika.

Sama płytka Arduino jest mikrokontrolerem zamontowanym wraz z obsługą wejść/wyjść na pojedynczym układzie drukowanym. Na urządzeniu znajdziemy kontroler, wygodne w użyciu, cyfrowe i analogowe linie wejścia/wyjścia, interfejs UART lub USB do komunikacji z komputerem.

Celem projektu Arduino jest dostarczenie szerokiemu gronu ludzi narzędzi wykorzystywanych do tworzenia własnych interaktywnych projektów wymagających sterowania bądź przetwarzania odbieranych sygnałów.



Rysunek 2.1. Arduino UNO

Źródło: http://blog.khron.net/wp-content/uploads/2014/05/ArduinoUno_R3_Front.jpg

2.1.1. Język programowania Arduino

Platforma Arduino posiada własny język programowania Arduino, który bazuje na Wiring oraz językach C/C++.

Wiring[2] jest framework'iem na licencji typu open-source złożonym z języka programowania, zintegrowanego środowiska deweloperskiego oraz mikrokontrolera wbudowanego w jedną

płytkę (jak w przypadku Arduino). Wiring oparty został o projekt Processing, który jest językiem i zintegrowanym środowiskiem stworzonym na MIT (ang. Massachusetts Institute of Technology) Media Lab. Projekt Wiring rozpoczął się w 2003 roku w Instytucie Wzornictwa Interakcji Ivrea. Obecnie rozwijany w Szkole Architektury i Projektowania na Uniwersytecie Los Andes w Kolumbii. Wynalazcą platformy jest Hernando Barragán. Środowisko Wiring IDE zostało napisane w języku JAVA. Zaprojektowano je by wprowadzać elektroników, hobbytów i artystów do programowania elektroniki.

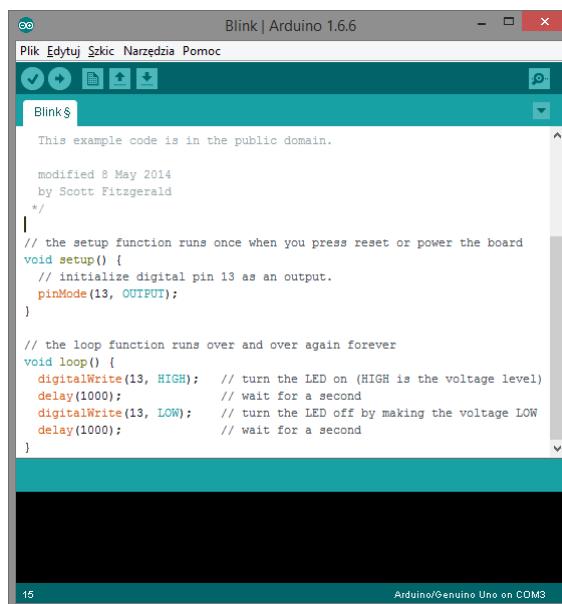
2.1.2. Arduino Software (IDE)

Wgrywanie projektów na płytę Arduino odbywa się za pośrednictwem Arduino IDE (ang. Integrated Development Environment). Jest to środowisko zawierające (Rysunek 2.2):

- edytor tekstu posiadający funkcje jak podświetlanie składni, przeszukiwanie kodu i zamianie czy automatyczne utrzymywanie wcięć,
- pole komunikatów zwracające wyniki działania Arduino Software IDE np. rezultat zapisywania czy eksportu oraz informujące o błędach, które wystąpiły,
- konsolę wyświetlającą wyjście Arduino Software oraz błędy występujące w kodzie,
- szeregowy monitor wyświetlający dane odebrane z płytki Arduino oraz umożliwiający wysyłanie danych do płytki.

Pliki będące wynikiem pracy w Arduino Software IDE nazywane są skoczami i posiadają rozszerzenie .ino. Arduino Software zbudowane jest na podstawie projektu Processing.

Processing[3] jest językiem programowania oraz zintegrowanym środowiskiem programistycznym (ang. IDE) stworzonym przez Ben'a Fry i Casey'a Reas. Powstał w MediaLab działającym w MIT (ang. Massachusetts Institute of Technology). Prace nad nim rozpoczęły się w roku 2001, na potrzeby sztuki elektronicznej oraz projektowania graficznego. Język bazuje na języku Java, jednak ma uproszczoną składnię i graficzny model programowania.



Rysunek 2.2. Arduino IDE

2.2. Narzędzia do projektowania elektroniki

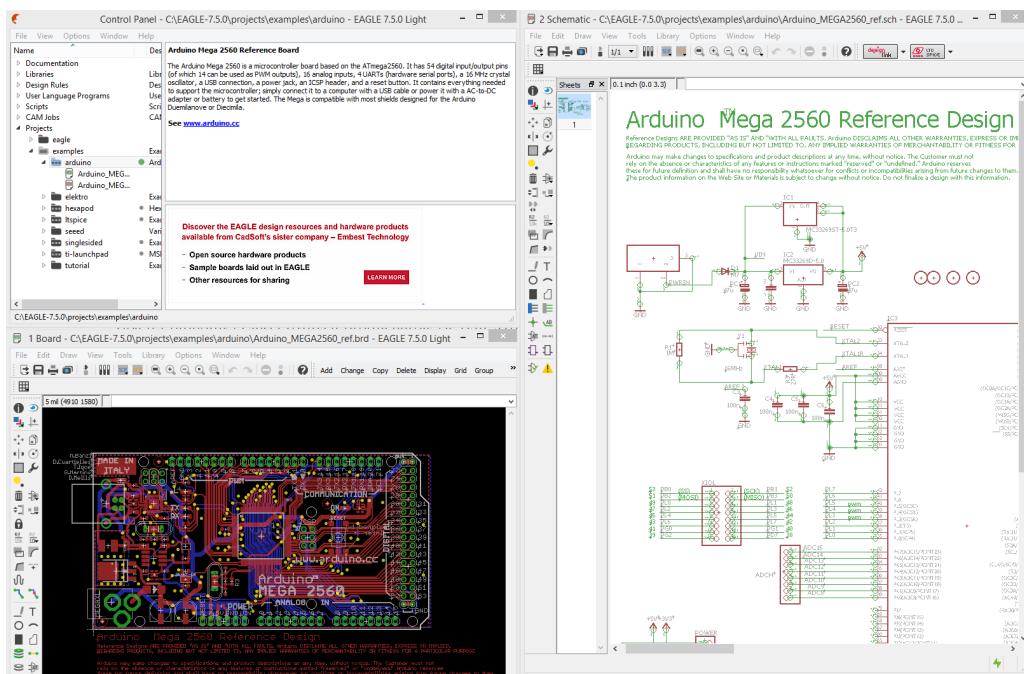
Projektowanie układów elektronicznych to złożony proces. Jednym elementem tego procesu są schematy połączeń poszczególnych elementów oraz projekty płytka PCB (z ang. Printed Circuit Board). Proces tworzenia tych schematów i projektów można ułatwić poprzez stosowanie odpowiedniego oprogramowania.

2.2.1. Eagle

EAGLE[4] (z ang. Easy Applicable Graphical Layout Editor) jest wszechstronnym oprogramowaniem do tworzenia projektów PCB. EAGLE wykorzystywany jest przez profesjonalistów pracujących dla światowych koncernów elektronicznych, hobbytów oraz studentów.

Oprogramowanie składa się z trzech modułów z identycznym interfejsem użytkownika (Rysunek 2.3):

- edytor schematów - umożliwia tworzenie symbolicznych, czytelnych schematów, które wzmacniają dokumentację projektu,
- edytor układu elektronicznego - przy pomocy tego modułu zaprojektujemy układ gotowy do wydrukowania,
- autorouter - moduł automatycznego tworzenia ścieżek elektrycznych na projektowanym układzie elektronicznym.

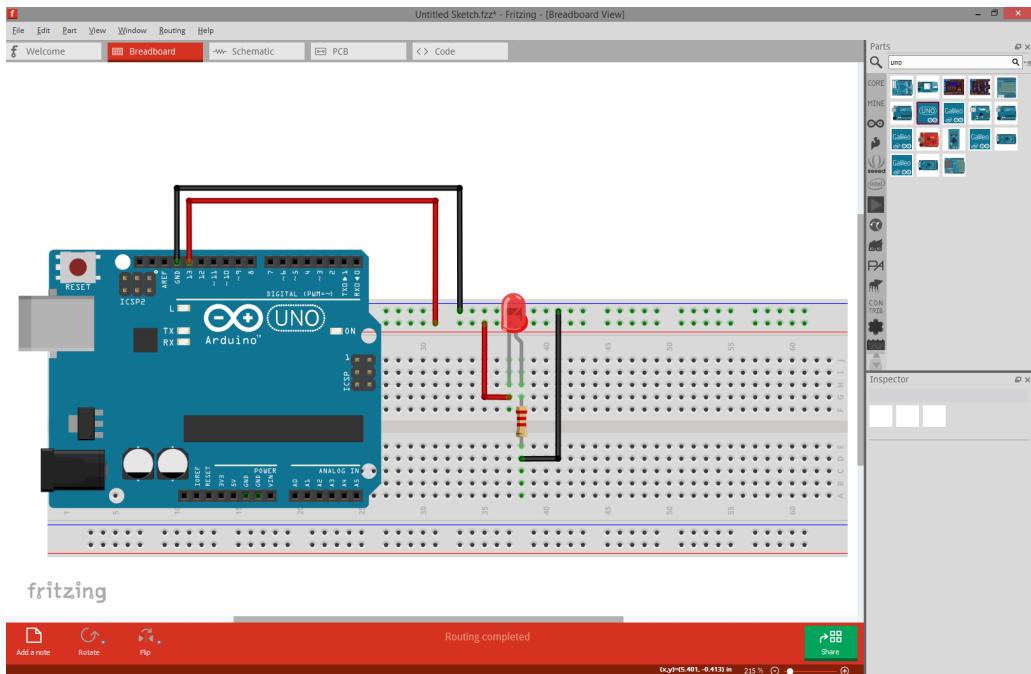


Rysunek 2.3. Program EAGLE - wszystkie okna

2.2.2. Fritzing

Fritzing to narzędzie służące do tworzenia schematów oraz rysunków poglądowych projektów elektronicznych. Fritzing jest narzędziem open source. Narzędzie to pozwala na tworzenie czytelnych profesjonalnych schematów, jednak to co wyróżnia to narzędzie to schematy poglądowe (Rysunek 2.4), które pokazują nam faktyczny wygląd skomponowanego układu.

Bogata biblioteka części różnych producentów (w tym Arduino), ciągle rośnie ze względu na możliwość dodawania własnych części i dzieleniem się nimi z innymi użytkownikami Fritzing.



Rysunek 2.4. Program Fritzing - edytor schematów poglądowych

2.3. CAD

Komputery i oprogramowanie od wielu lat znajdują zastosowanie w różnych dziedzinach naszego życia. CAD (ang. computer aided design) jest wykorzystaniem systemów komputerowych do wspomagania tworzenia, modyfikacji, analizy lub optymalizacji projektu. Znajduje zastosowanie w takich dziedzinach jak: inżynieria mechaniczna, inżynieria elektryczna czy inżynieria budowlana.

2.3.1. AutoCAD

Program[5] umożliwiający projektowanie wspomagane komputerowo. Program stworzony został przez firmę Autodesk, a jego pierwsza wersja zaprezentowana została na targach COMDEX w Las Vegas w listopadzie 1982 roku, sprzedaż produktu ruszyła miesiąc później. Program jest nadal rozwijany i jego najnowsza wersja nosi nazwę AutoCAD 2016. AutoCAD umożliwia wspomaganie projektowania 2D, 3D oraz dzięki specjalnym dodatkom 2.5D. Pierwotnie wykorzystywany był tylko przez mechaników, jednak dzięki rozszerzeniu programu przez wiele nakładek np. AutoCAD Electrical, AtuoCAD Mechanical, Architectural Desktop, Civil Design itp. stał się platformą wykorzystywaną przez architektów i projektantów różnych dziedzin inżynierii.

2.4. Narzędzia do testów wydajności

Podzespoły komputera nagrzewają się podczas intensywnej pracy. Aby sprawdzić skuteczność zbudowanego systemu chłodzącego konieczne będzie obciążenie stacji roboczej. W celu przeprowadzenia testów wykorzystamy oprogramowanie, które umożliwia obciążenie komputera oraz monitorowanie temperatury.

2.4.1. OCCT

OCCT[6] (ang. OverClock Checking Tool) jest programem służącym do sprawdzania stabilności pracy stacji roboczej. Poza stabilnością sprawdzimy także temperatury osiągane przez poszczególne podzespoły czy napięcie występujące na tych podzespołach. Poza wymienionymi danymi program dostarcza również informacje o posiadanym procesorze czy karcie graficznej. Program podczas działania bardzo mocno obciąża komputer (wykorzystanie procesora ~98 - 100%), jednocześnie dając możliwość stałego kontrolowania wyżej wymienionych danych. W programie dostępne są cztery testy: autorski, test na bazie biblioteki LINPACK stworzonej przez Intel, test sprawdzający kartę graficzną pod względem renderowania obiektów 3D oraz test zasilacza. Wyniki przeprowadzonych testów obrazowane są na wykresach zapisanych na dysku w postaci plików .png.

Rozdział 3

Realizacja projektu systemu chłodzącego

3.1. Projekt techniczny

System chłodzący jest połączeniem sterownika na bazie Arduino z podzespołami wykonawczymi oraz zamknięcie tego w szczelnej obudowie.

3.1.1. Projekt systemu chłodzącego

Głównym celem projektu jest zanurzenie podzespołów w cieczy, jednak nie każda ciecz może być użyta w tym projekcie. Główną cechą zastosowanej cieczy musi być jej przewodnictwo elektryczne, a właściwie jego zupełny brak. Woda destylowana nie przewodzi prądu, jednak jest bardzo uniwersalnym rozpuszczalnikiem co prowadzi do zmiany tej cechy. Cieczą zastosowaną w projekcie jest parafina ciekła C10 - C13. Nie przewodzi ona prądu oraz ma ciepło właściwe mniejsze o połowę od ciepła właściwego wody. Skutkuje to mniejszą ilością energii potrzebnej do ogrzewania zadanej ilości masy danego ciała o zadaną ilość stopni. Oznacza to, że parafina szybciej odbierze temperaturę z radiatora.

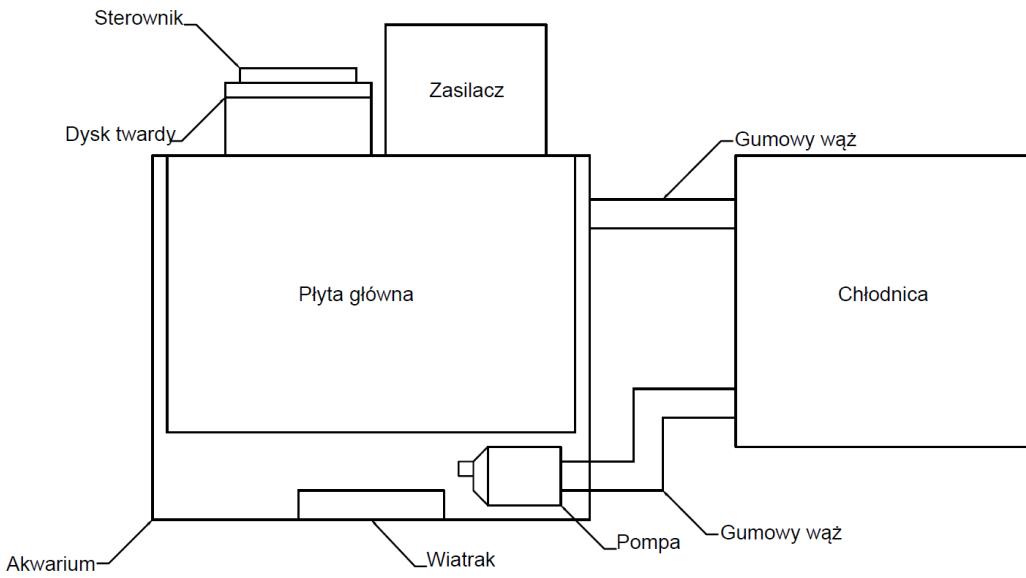
Projekt zakłada umieszczenie podzespołów stacji roboczej w cieczy, jednak nie wszystkie zostaną umieszczone wewnętrz pojemnika. Dyski twarde HDD, pomimo hermetycznego zamknięcia talerzy, posiadają zawory wyrównujące ciśnienie wewnętrz obudowy dysku. Zawory te mogłyby nie wytrzymać kontaktu z cieczą, co doprowadziło by do ich zniszczenia. Zasilacz, ze względu na swoje rozmiary również został umieszczony poza pojemnikiem. Ciecz chłodząca podzespoły komputera oddawać ciepło będzie w chłodnicy umieszczonej obok pojemnika.

Plan rozmieszczenia poszczególnych elementów, oraz planowanych połączeń przedstawiono na rysunku poglądowym (Rysunek 3.1).

3.1.2. Wykonanie systemu chłodzącego

Jednym z ważniejszych elementów tego projektu jest pojemnik, w którym zanurzony zostanie komputer. Z materiałów dostępnych na rynku wybrano szkło. W porównaniu do innych materiałów, np. szkła akrylowego (popularnie nazywanego „plexi”), szkło jest tańsze oraz nie rysuje się. Wykonanie specjalnego pojemnika zlecono zewnętrznej firmie. Pojemnik jest prostopadłosciennym naczyniem z otworami w prawej ścianie. Rama z zamontowanymi podzespołami opierać się będzie na krawędziach naczynia w związku z czym wybrano szkło o grubości 5mm. Projekt wraz z wymiarami przedstawiono na rzucie izometrycznym (Rysunek 3.2). Projekt został przesłany do firmy wykonującej akwarium. Wymiary wyrażone są w milimetrach i są to wymiary wewnętrzne akwarium.

Podzespoły komputera potrzebują sztywnej ramy, do której zostaną przyjmocowane. W tym projekcie postanowiono wykorzystać elementy dotychczasowej obudowy komputera. Tak jak w zwykłej obudowie głównym elementem ramy utrzymującej będzie tacka z nawierconymi otworami pod kołki dystansowe. Tył obudowy wraz z panelem maskującym wykorzystany został do stworzenia pokrywy akwarium oraz wykorzystany został jako element utrzymujący



Rysunek 3.1. Schemat poglądowy układu chłodzenia wykonany w AutoCAD 2016

podzespoły w pojemniku. Poniżej przedstawiono elementy wykorzystane przy tworzeniu ramy podzespołów (Rysunek 3.3)

Tacka, na której zamontowana jest płyta główna została przycięta do rozmiarów pasujących do pojemnika, w którym zostaną umieszczone podzespoły komputera. Wycięta tylna część obudowy wraz z maskownicą płyty głównej obudowy została wydłużona i wyposażona w uchwyty zapobiegające się przemieszczaniu całej ramy po akwarium. Po złączeniu elementów ramy i przykręceniu do nich podzespołów komputera zajęto się poprowadzeniem przewodów. Przewody zostały umieszczone tak by nie zaklinować któregoś z wiatraków znajdujących się w pojemniku oraz by całość wyglądała estetycznie (Rysunek 3.4).

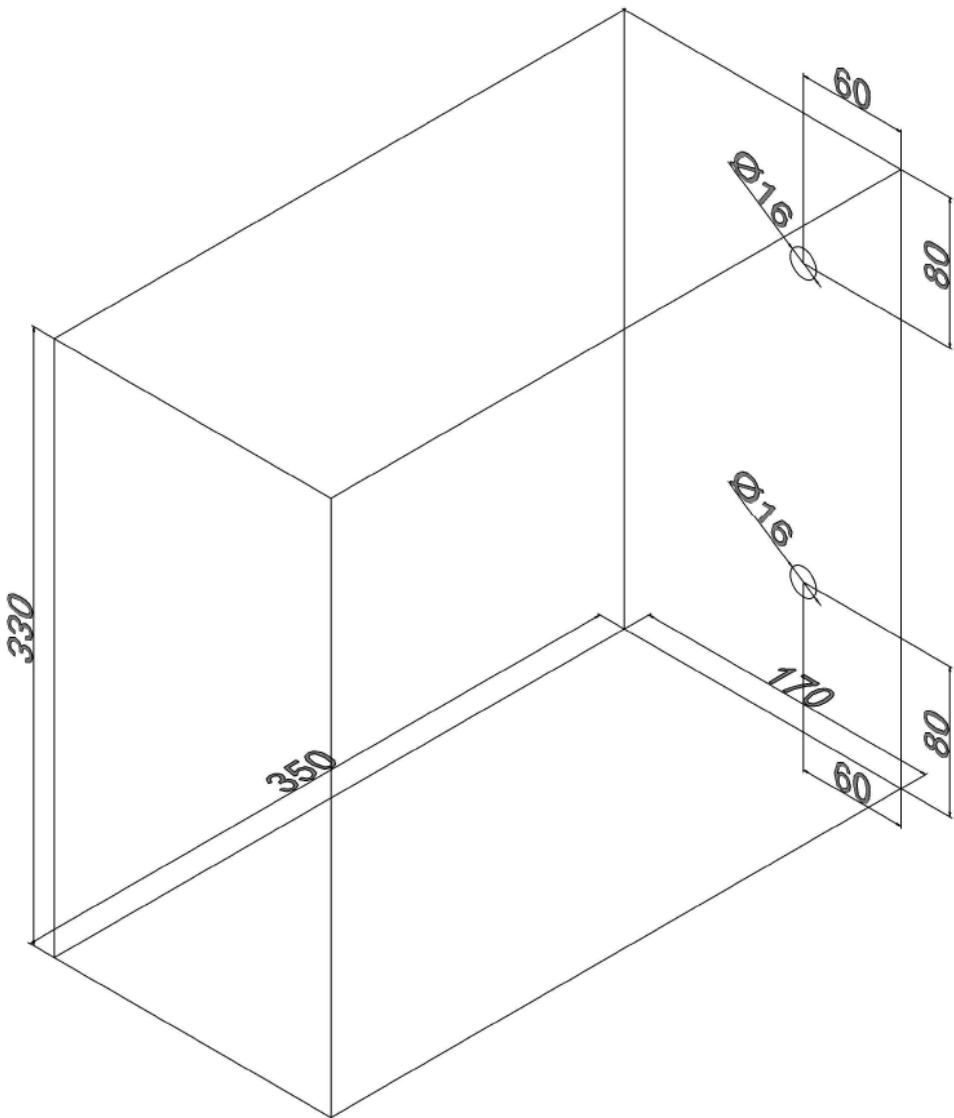
Następnym etapem było przygotowanie akwarium wraz z pompą i chłodnicą. W otworach w ściance akwarium umieszczono reduktory mosiężne z uszczelniającymi o ringami, na gwinty reduktorów nakręcono mosiężne kolanka z wyprowadzeniem na podłączenie węza. Gwinty uszczelniono silikonem silnikowym. Gumowe węże przymocowano do kolanek, a następnie przymocowano do chłodnicy. Do projektu wykorzystano chłodnicę układu klimatyzacji z Volkswagena Passat B5. W reduktorze umieszczonym w dolnym otworze zamocowano pompę. Zastosowana pompa to pompa paliwa z Fiata Punto (Rysunek 3.5).

Kolejnym etapem było umieszczenie ramy z podzespołami komputera w wypełnionym pojemniku, odpowietrzenie chłodnicy oraz zaprojektowanie wykonanie układu elektronicznego odpowiedzialnego za sterowanie tworzonym projektem.

3.2. Elektroniczny układ współpracujący z Arduino

Platforma Arduino została stworzona w celu popularyzacji tworzenia interaktywnych projektów elektronicznych oraz układów automatyzacji. Płytki Arduino zapewnia wyprowadzenie pinów komunikacyjnych mikrokontrolera. Zadaniem osoby budującej projekt na jego podstawie jest zaprojektowanie i wykonanie całego zestawu urządzeń periferyjnych.

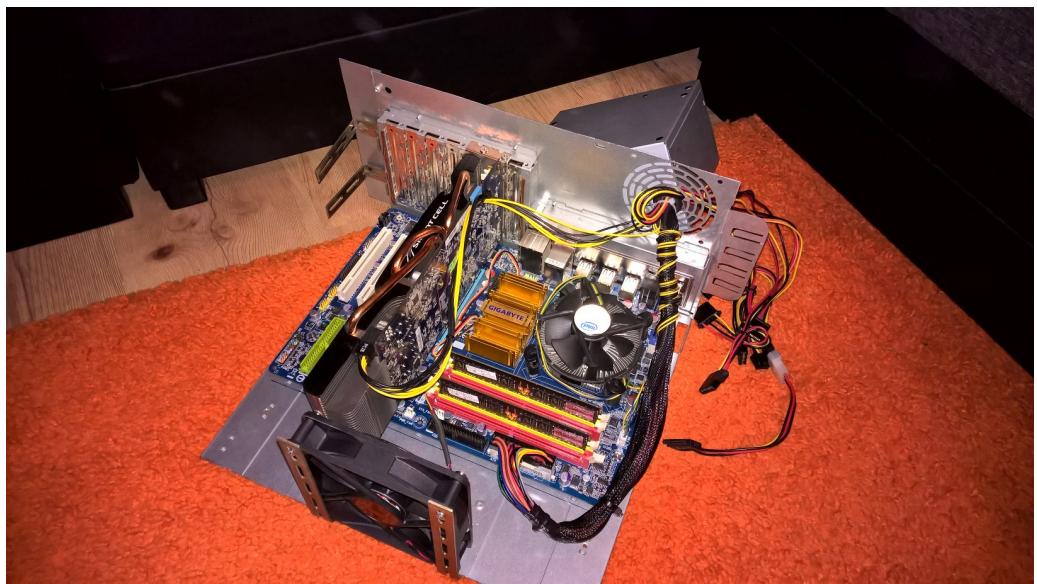
W przypadku tego projektu niezbędne będą urządzenia do komunikacji z użytkownikiem, ponieważ to on decyduje o momencie załączenia pompy oraz będzie miał możliwość śledzenia temperatury cieczy. Cały układ poza odebraniem danych od użytkownika musi być w stanie



Rysunek 3.2. Rzut izometryczny bryły pojemnika wraz z wymiarami wykonany w AutoCAD 2016



Rysunek 3.3. Elementy obudowy wykorzystane w projekcie



Rysunek 3.4. Rama z zamontowanymi podzespołami komputera



Rysunek 3.5. Zmontowane akwarium z chłodnicą i pompą

zmierzyć temperaturę cieczy[7] wewnątrz pojemnika z komputerem. Jak wiadomo płyta główna jest sporym podzespołem więc rozmiary pojemnika, w którym zanurzony będzie komputer będą znaczące dla odczytu temperatury. Zastosowano więc dwa czujniki temperatury, które odczytywać będą temperaturę z dwóch różnych miejsc w pojemniku. Ciecz znajdująca się wewnątrz pojemnika musi mieć gdzie oddać ciepło przejęte z podzespołów. W tym celu zastosowano pompę, która przetłoczy ciecz przez zewnętrzną chłodnicę. Pracując przy komputerze użytkownicy często skupiają się tylko na swoim zadaniu. By system był efektywny zastosowano właśnie automatyczne sterowanie załączeniem pompy, wykorzystujące Arduino.

3.2.1. Projekt układu elektronicznego

Komunikacja z mikrokontrolerem jest istotnym elementem tego projektu. Arduino ma generować dane: aktualną średnią temperaturę cieczy, oraz temperaturę po przekroczeniu której załączona zostanie pompa, które będą wyświetlane. Temperatura załączenia pompy ustawiana jest przez użytkownika. Tak więc do komunikacji niezbędny będzie wyświetlacz. Do tego typu danych wybrany został wyświetlacz LCD posiadający dwa rzędy znaków po 16 znaków w każdym. Do ustawiania temperatury wykorzystane zostały dwa przyciski monostabilne, których zwarcie monitorowane będzie przez mikrokontroler.

Układ elektroniczny systemu chłodzącego musi również być w stanie mierzyć temperaturę cieczy. Do tego zadania wybrane zostały dwa czujniki LM35, które cechują się szerokim zakresem pomiaru (od -50°C do $+150^{\circ}\text{C}$), w którym mieści się zakres odczytywanych temperatur. Ciecz minimalnie będzie osiągała temperaturę pokojową, a maksymalnie nie przekroczy 100°C .

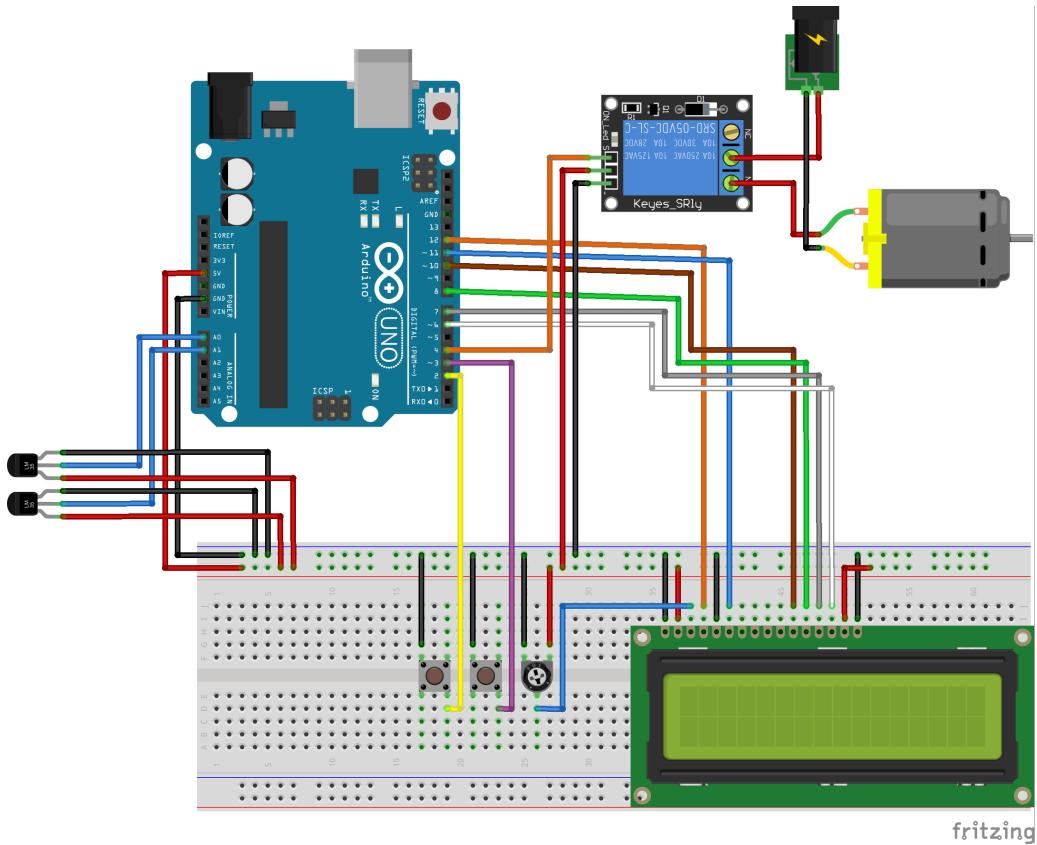
Nagrzaną przez podzespoły cieczy należy wychłodzić poza zbiornikiem z komputerem. Aby przetłoczyć ciecz poza pojemnik wykorzystano pompkę elektryczną, której wymagane zasilanie znacznie przewyższa możliwości Arduino. Wykorzystano więc przekaźnik, który wysterowany przez Arduino załączy pompkę w odpowiednim momencie.

Podsumowując, nasz układ elektroniczny składać się będzie z:

- płytki Arduino UNO rev. 3,
- wyświetlacza LCD 2 razy po 16 znaków JHD162A-B-W,
- dwóch przycisków monostabilnych,

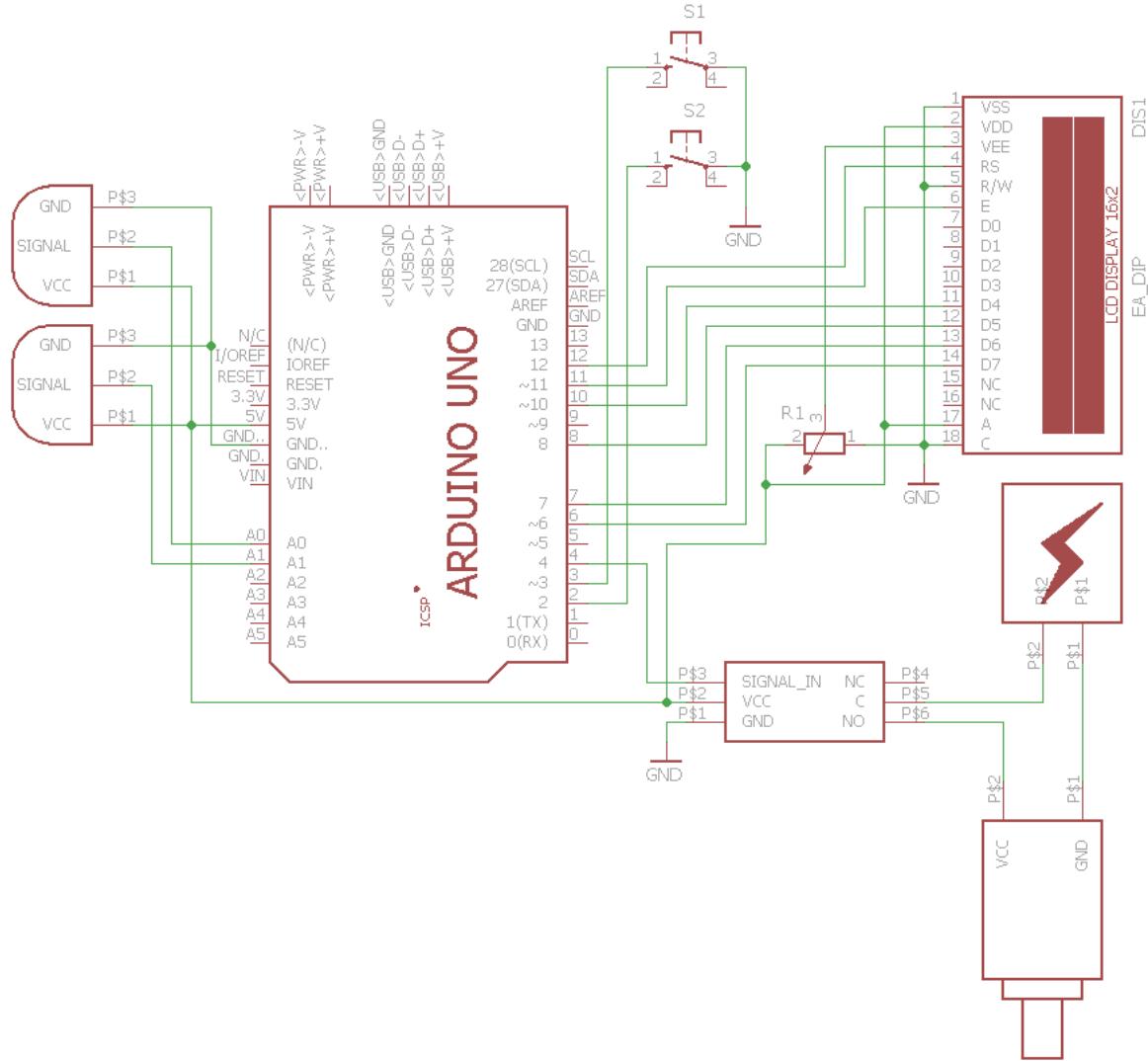
- dwóch czujników temperatury LM35,
- przekaźnika,
- pompki,
- zewnętrznego źródła zasilania.

Połączenia między wszystkimi podzespołami zostały zaprojektowane zgodnie z notami katalogowymi poszczególnych podzespołów. Połączenia między poszczególnymi elementami przedstawione zostały na schemacie poglądowym (Rysunek 3.6). Tworzenie takich schematów ma na celu łatwe odczytanie projektu[8] przez innych twórców. Na tym schemacie urządzenia przedstawione zostały za pomocą grafik przybliżający faktyczny wygląd poszczególnych elementów (nie zostały zachowane proporcje ze względu na zróżnicowany rozmiary elementów układu), oraz przedstawione wszystkie niezbędne fizyczne połączenia.



Rysunek 3.6. Schemat poglądowy wykonany we Fritzing

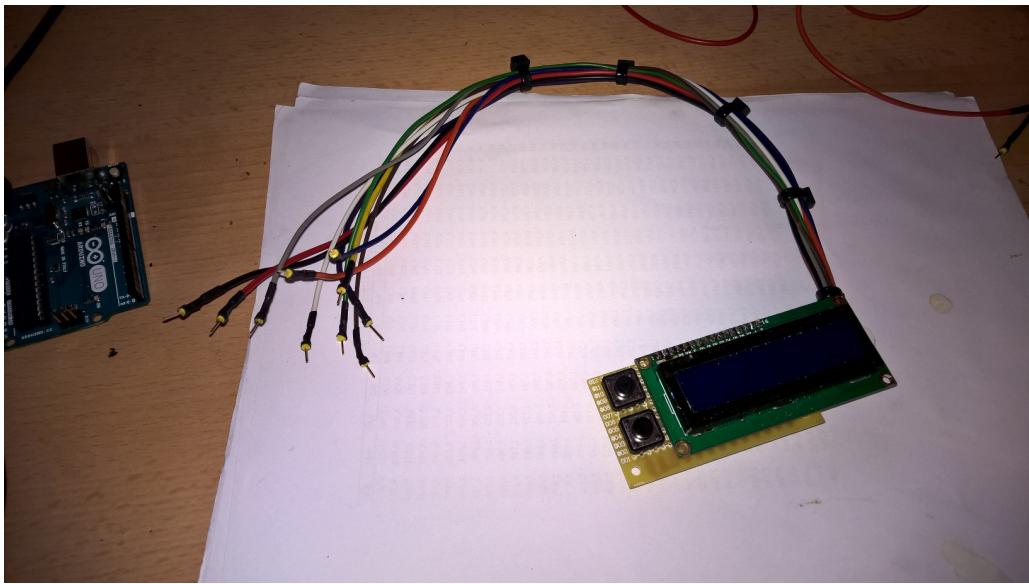
W celu stworzenia odpowiedniej dokumentacji projektu, układ elektryczny został zaprezentowany na funkcjonalnym schemacie elektrycznym[9] (Rysunek 3.7).



Rysunek 3.7. Schemat elektryczny układu współpracującego z Arduino wykonany w EAGLE

3.2.2. Wykonanie układu elektronicznego

Wykonanie układu elektronicznego rozpoczęto od wykonania modułu odpowiedzialnego za komunikację z użytkownikiem. Elementy zamontowane na płytce stykowej projektu (Rysunek 3.6) zostały zlutowane zgodnie z zasadami lutowania miękkiego[9] na płytce prototypowej. Montaż części odpowiedzialnych za komunikację na jednej płytce umożliwił wyprowadzenie tego modułu na długich przewodach i umieszczenie ich w widocznym miejscu obudowy (na przedniej ścianie akwarium). Przewody zakończone zostały goldpinami w celu umożliwienia podpięcia ich do płytki Arduino (Rysunek 3.8).

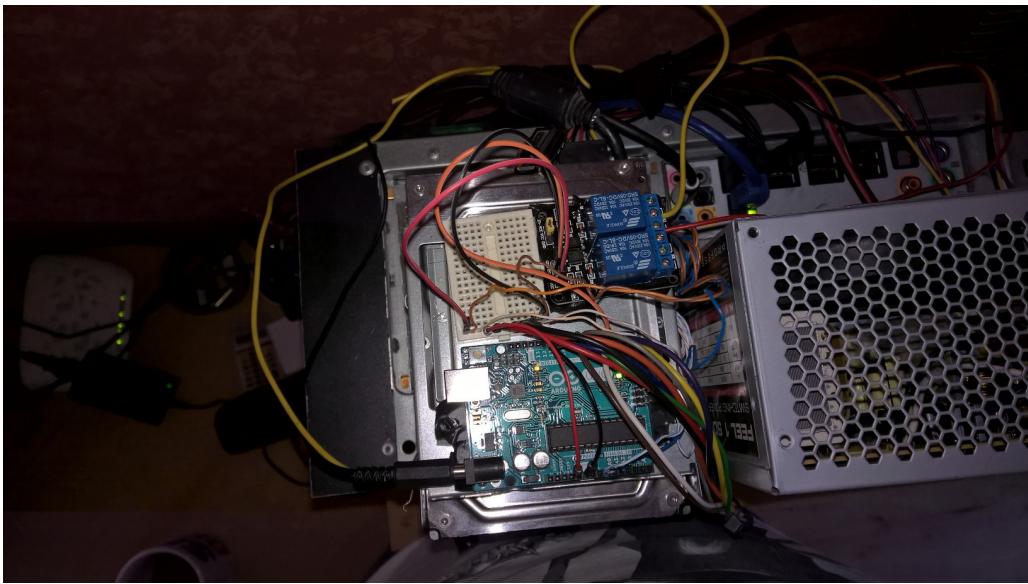


Rysunek 3.8. Zmontowany moduł komunikacji

Czujniki temperatury również zostały przylutowane do długich przewodów by można było umieścić je w odpowiednich miejscach w obudowie. Jeden tuż przy radiatorze procesora, drugi w pobliżu radiatora karty graficznej. Zasilacz, pompa i przekaźnik są elementami gotowymi. Wystarczyło tylko wykonać odpowiednie połączenia do płytki Arduino, zgodnie z połączonymi przedstawionymi na schemacie (Rysunek 3.7). Kolejno podłączono:

- dwa czujniki temperatury LM35: piny GND czujników wpięto do wspólnej masy całego układu, piny VCC czujników wpięto do zasilania 5V wyprowadzonego z płytki Arduino, piny SIGNAL czujników wpięto do analogowych wejść płytki Arduino A0 i A1,
- wyświetlacz JHD162A-B-W: piny VSS, R/W, C wyświetlacza podpięto do wspólnej masy całego układu, piny VDD i A wyświetlacza wpięto do zasilania 5V wyprowadzonego z płytki Arduino, piny RS, E, D4, D5, D6, D7 zostały podpięte do kolejnych cyfrowych wejść/wyjść płytki Arduino 12, 11, 10, 8, 7, 6,
- przyciski monostabilne: pin 1 przycisków monostabilnych zostały wpięte do cyfrowych wejść/wyjść płytki Arduino 3 i 2, pin 3 obu przycisków podpięto do wspólnej masy układu,
- przekaźnik: pin VCC przekaźnika wpięto do zasilania 5V wyprowadzonego z płytki Arduino, pin GND przekaźnika podpięto do wspólnej masy układu, pin SIGNAL_IN podłączono do 4 cyfrowego wejścia/wyjścia płytki Arduino, linia zasilania pompy podpięta została podłączona do zacisku NO (ang. normally open) przekaźnika, linia uziemienia pompy połączona została z uziemieniem źródła zasilania pompy natomiast napięcie źródła zasilania pompy wpięto do zacisku C przekaźnika.

Po wykananiu odpowiednich połączeń (Rysunek 3.9) należało zaimplementować pracę podłączonych elementów układu w oprogramowaniu Arduino.



Rysunek 3.9. Podłączony układ elektryczny współpracujący z Arduino

3.3. Implementacja sterownika układu chłodzącego na platformie Arduino

3.3.1. Opis algorytmu głównego

Algorytm główny sterownika ma kilka zadań. Jednym z nich jest stałe monitorowanie odczytu czujników temperatury i na podstawie tych odczytów, obliczanie i wyświetlanie temperatury cieczy na ekranie. Pomiar temperatury chłodziwa znajdującego się w obudowie jest również wyznacznikiem momentu załączenia pompy. Gdy temperatura parafiny w obudowie przekroczy ustawioną przez użytkownika temperaturę krytyczną, Arduino ma za zadanie załączenie pompy, która przetłoczy ciecz przez chłodnicę. Gdy temperatura chłodziwa spadnie poniżej poziomu krytycznego, Arduino wyłączy pompę. Poziom krytyczny, o którym mowa ustawiany jest przez użytkownika przy pomocy przycisków monostabilnych. Nasłuchiwanie wspomnianych przycisków to również jedno z zadań sterownika. Praca sterownika przedstawiona została na diagramie przepływu (Rysunek 3.10).

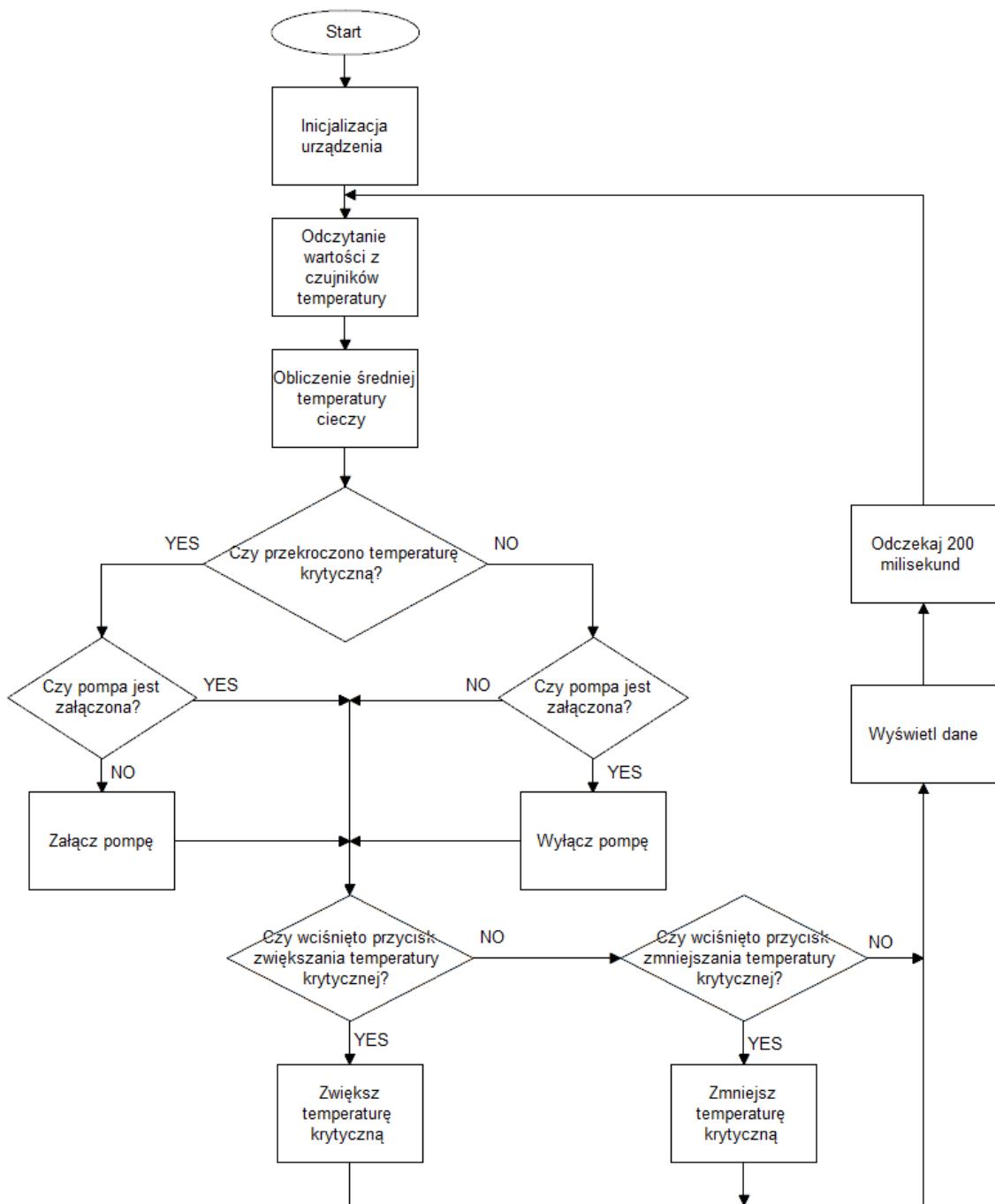
3.3.2. Implementacja algorytmu głównego

Programy wykonywane przez Arduino pisane są w specjalnym języku Arduino, który bazuje na językach C/C++ oraz środowisku Wiring. Kod programu Arduino składa się głównie z 3 bloków[8]:

- część odpowiedzialna za inicjalizację urządzenia - deklaracja zmiennych globalnych, stałych, definicji funkcji, dodanie bibliotek,
- funkcji setup - funkcja wykonywana raz po uruchomieniu urządzenia,
- funkcji loop - funkcja wykonywana nieprzerwanie przez urządzenie.

Poniżej przedstawiono kod sterownika podzielony na wyżej wymienione bloki.

Przedstawiony listing Algorytm 1 odpowiada za inicjalizację urządzenia. Jest on wykonywany tylko raz, zaraz po włączeniu urządzenia. Linie 1 i 2 odpowiadają za dodanie biblioteki obsługującej wyświetlacz LCD oraz jego inicjalizację na wyjściach płytki Arduino o numerach



Rysunek 3.10. Diagram przepływu pracy sterownika

Algorytm 1 Inicjalizacja urządzenia Arduino

```
1 #include <LiquidCrystal.h>
2 LiquidCrystal lcd(12, 11, 10, 8, 7, 6);
3
4 #define relay 4
5 #define buttonUp 3
6 #define buttonDown 2
7 #define lm35_1_pin 0
8 #define lm35_2_pin 1
9
10 int tempCritical = 55;
11 int tempAvg;
12 float tempLm35_1;
13 float tempLm35_2;
14 int lm35_1;
15 int lm35_2;
16 bool pump = false;
17
18 bool increaseTemp() {
19     int zwiększenie = !digitalRead(buttonUp);
20     return zwiększenie;
21 }
22
23 bool decreaseTemp() {
24     int zmniejszenie = !digitalRead(buttonDown);
25     return zmniejszenie;
26 }
```

12, 11, 10, 8, 7, 6. Linie od 4 do 8 są definicją pinów urządzeń podpiętych do płytki. Kolejno są to:

- przerzutnik na pinie 4,
- przycisk monostabilny odpowiedzialny za zwiększenie temperatury krytycznej na pinie 3,
- przycisk monostabilny odpowiedzialny za zmniejszenie temperatury krytycznej na pinie 2,
- dwa czujniki temperatury LM35 na pinach 0 i 1.

Następnie zdefiniowano zmienne, w liniach od 10 do 16:

- temperatura krytyczna,
- temperatura średnia,
- obliczona temperatura dla obu czujników,
- odczyt wartości z obu czujników,
- stan załączenia pompy.

Linie od 18 do 26 zawierają bliźniacze funkcje. Funkcje te odpowiadają za wykrywanie przyciśnięcia przycisków odpowiadających za ustawianie temperatury krytycznej. Wykorzystano w nich funkcję dostępną w języku Arduino, digitalRead(). Funkcja ta zwraca stany HIGH lub LOW w zależności od tego jaki stan znajduje się na pinie, którego numer podajemy jako parametr. Stany HIGH i LOW równoważne są z wartościami zmiennych logicznych true i false oraz 1 i 0.

Algorytm 2 Funkcja setup()

```
1 void setup() {  
2     lcd.begin(16, 2);  
3     pinMode(buttonUp, INPUT_PULLUP);  
4     pinMode(buttonDown, INPUT_PULLUP);  
5     pinMode(relay, OUTPUT);  
6     digitalWrite(relay, HIGH);  
7 }
```

Zaprezentowany listing Algorytm 2 to funkcja setup(). Jest to funkcja, którą urządzenie wykona tylko raz zaraz po uruchomieniu. W jej wnętrzu można ustawić wartości zmiennych, ustawić rolę poszczególnych pinów płytki Arduino czy wywoływać funkcje dostępne w bibliotekach, które wcześniej dodano do sketch'u.

W linii 2, wywołana została funkcja z biblioteki obsługującej wyświetlacz ciekłokrystaliczny. W jej parametrach podano kolejno ilość znaków w wierszu, w tym przypadku 16 oraz ilość wierszy czyli 2. Linie 3 i 4 odpowiadają za ustawienie funkcji pinów, do których podłączone zostały przyciski monostabilne za pomocą, których ustawiana jest temperatura krytyczna. Jako parametry funkcji ustawiającej piny pinMode() podano kolejno numer pinu, w tym przypadku wykorzystano stałą, która została zadeklarowana w tym celu i tryb w jakim pracować ma zadanego pin, w tym przypadku INPUT_PULLUP, który wykorzystuje wbudowane w mikrokontroler rezystory podciągające i działa jako wejście odporne na skoki napięcia, co zapobiega występowaniu zakłóceń związanych z pracą pinu wejściowego. Linia 5 to ustawienie pinu, do którego podłączony jest przekaźnik jako pin wyjścia, dzięki czemu w części głównej algorytmu można ustawić żądany stan na zadanym pinie. Linia 6 jest ustawieniem pinu, do którego podłączony jest przekaźnik na stan wysoki, co skutkuje wyłączeniem pompy.

Funkcja loop przedstawiona na listingu Algorytm 3, wykonywana jest przez cały czas załączenia urządzenia, w niej realizowane są zadania, które wykonać ma mikrokontroler. Linie 2 i 3 odpowiadają za odczytanie wartości przetwornika analogowo-cyfrowego na pinach, do

Algorytm 3 Funkcja loop()

```
1 void loop() {
2     lm35_1 = analogRead(lm35_1_pin);
3     lm35_2 = analogRead(lm35_2_pin);
4
5     tempLm35_1 = lm35_1 * 0.48828125;
6     tempLm35_2 = lm35_2 * 0.48828125;
7
8     tempAvg = (tempLm35_1 + tempLm35_2)/2;
9
10    if (tempCritical < tempAvg && !pump) {
11        digitalWrite(relay, LOW);
12        pump = true;
13    }
14
15    if (tempCritical > tempAvg && pump) {
16        digitalWrite(relay, HIGH);
17        pump = false;
18    }
19
20    if (decreaseTemp())
21        tempCritical--;
22    if (increaseTemp())
23        tempCritical++;
24
25    lcd.clear();
26
27    lcd.setCursor(0, 0);
28    lcd.print("Temp\u0142ciecz\u0142");
29    lcd.print(tempAvg);
30    lcd.print((char)223);
31    lcd.print("C");
32
33    lcd.setCursor(0, 1);
34    lcd.print("Temp\u0142kryty\u0142");
35    lcd.print(tempCritical);
36    lcd.print((char)223);
37    lcd.print("C");
38
39    delay(200);
40 }
```

których podpięte są czujniki temperatury. Do tego celu wykorzystano funkcje analogRead(), której parametrami są numery pinów, z których odczyt ma być wykonany. Mikrokontroler wbudowany w Arduino posiada 10 bitowy przetwornik analogowo-cyfrowy, czyli odczyt będzie z zakresu od 0 do 1023. Linie 5 i 6 mają za zadanie przetworzenie odczytanej wartości na liczbę reprezentującą ilość stopni Celsjusza. Liczba 0,48828125 została wyliczona ze wzoru:

$$T = \frac{ADC * \frac{AREF}{1024}}{10mV} \quad (3.1)$$

gdzie:

T	liczba reprezentująca temperaturę w stopniach Celsjusza,
ADC	odczyt z przetwornika analogowo-cyfrowego,
AREF	napięcie odniesienia używane przez przetwornik analogowo-cyfrowy, w tym przypadku 5V = 5000mV,
1024	zakres odczytu przetwornika analogowo-cyfrowego,
10mV	ilość mV, która równoważna jest 1°C.

Linia 8 to wyliczenie średniej temperatury cieczy znajdującej się w pojemniku z komputerem. W liniach od 10 do 13 zamieszczono instrukcję warunkową odpowiedzialną za załączenie pompy. Warunek w tej instrukcji jest prawdziwy, gdy średnia temperatura cieczy jest większa od ustawionej temperatury krytycznej i gdy pompa nie jest załączona. Spełnienie warunku powoduje ustawienie stanu niskiego na pinie, do którego podpięty jest przekaźnik i ustawienie zmiennej przetrzymującej stan załączenia na prawdę. Linie od 15 do 18 zawierają analogiczną instrukcję warunkową, z tym że tutaj wykrywany jest moment wyłączenia. Warunek jest prawdziwy gdy średnia temperatura cieczy jest niższa niż temperatura krytyczna i pompa jest załączona. Spełnienie warunku powoduje ustawienie stanu wysokiego na pinie, do którego podpięty jest przekaźnik i ustawienie zmiennej przetrzymującej stan załączenia pompy na fałsz.

Linie od 20 do 23 odpowiadają za wykrycie zwarcia przycisków monostabilnych. Wykorzystano tu funkcje decreaseTemp() oraz increaseTemp(), które opisano przy opisie listingu Algorytm 1. Wykrycie zwarcia odpowiedniego przycisku skutkuje zwiększeniem lub zmniejszeniem temperatury krytycznej.

Linie od 25 do 37 odpowiadają za wyświetlanie informacji na wyświetlaczu. W tej części kodu wykorzystano kilka funkcji dostępnych w bibliotece LiquidCrystal.h. Funkcję clear(), która czyści zawartość wyświetlacza. Funkcję setCursor(), która przyjmuje w parametrach pozycję kurSORA i ustawia go w niej. Pierwsza liczba to numer znaku w wierszu, druga jest numerem wiersza. Funkcję print(), która jako parametr przyjmuje tablicę znaków, które mają być wyświetlone. Tablica może być wyrażeniem zawartym w cudzysłów, zmienną, lub kodem znaku odczytanym z dokumentacji technicznej przekonwertowanym na znak.

Ostatnia linia kodu to instrukcja delay(), która za parametr przyjmuje ilość milisekund przez które ma wstrzymać wykonywanie kodu. W tym przypadku podano 200. Oznacza to iż cały kod wykona się 5 razy w ciągu sekundy. Zapewnia to odpowiednio częsty pomiar temperatury by zauważać jej zmiany, jednocześnie nie zakłócając obrazu wyświetlanego na wyświetlaczu.

Rozdział 4

Testy wydajnościowe i analiza wyników

4.1. Testy wydajnościowe chłodzenia podzespołów

Określenie skuteczności zbudowanego systemu chłodzącego wymagało przeprowadzenia serii eksperymentów. Zostały one przedstawione w poniższym rozdziale. Zaprezentowano również konfigurację testowanego komputera oraz opisano jego standardowy system chłodzenia.

4.1.1. Opis testowanego sprzętu

Komputer, którego podzespoły zostały użyte w projekcie (Tablica 4.1) chłodzony był w sposób standardowy. Procesor chłodzony był przez wentylowany, aluminiowy radiator w kształcie walca średnicy około 10 cm i wysokości około 1 cm, dostarczony wraz z procesorem przez jego producenta. Karta graficzna znajdująca się w komputerze, posiadała chłodzenie pasywne. Masywny radiator wykonany z aluminium oraz wyposażony w miedziane rurki cieplne (ang. heat pipe) odprowadzał ciepło bez użycia żadnych wentylatorów. Całość zamknięta była w metalowej obudowie wentylowanej jednym wentylatorem.

4.1.2. Opis eksperymentu

Podczas przeprowadzania eksperymentu sprawdzającego wydajność zbudowanego układu wykorzystano oprogramowanie OCCT. Wykonano szereg testów z wykorzystaniem biblioteki LINPACK zmodernizowanej przez Intel®. Testy benchmark oparte na tej bibliotece mają za zadanie rozwiązać gęsty układ równań liniowych.

Układ chłodzący zrealizowany w tym projekcie ma możliwość ustawienia temperatury załączenia pompy. W związku z tym przeprowadzono po trzy próby testów dla kolejnych ustawień temperatury krytycznej: 45°C, 50°C, 55°C. Każda z przeprowadzonych prób trwała 30 minut oraz rozpoczynała się gdy temperatura cieczy znajdującej się w akwariu nie przekraczała 30°C.

Temperatura poszczególnych rdzeni procesora rejestrowana była co 30 sekund. Na podstawie zebranych danych wykonano wykresy przedstawiające wyniki eksperymentu.

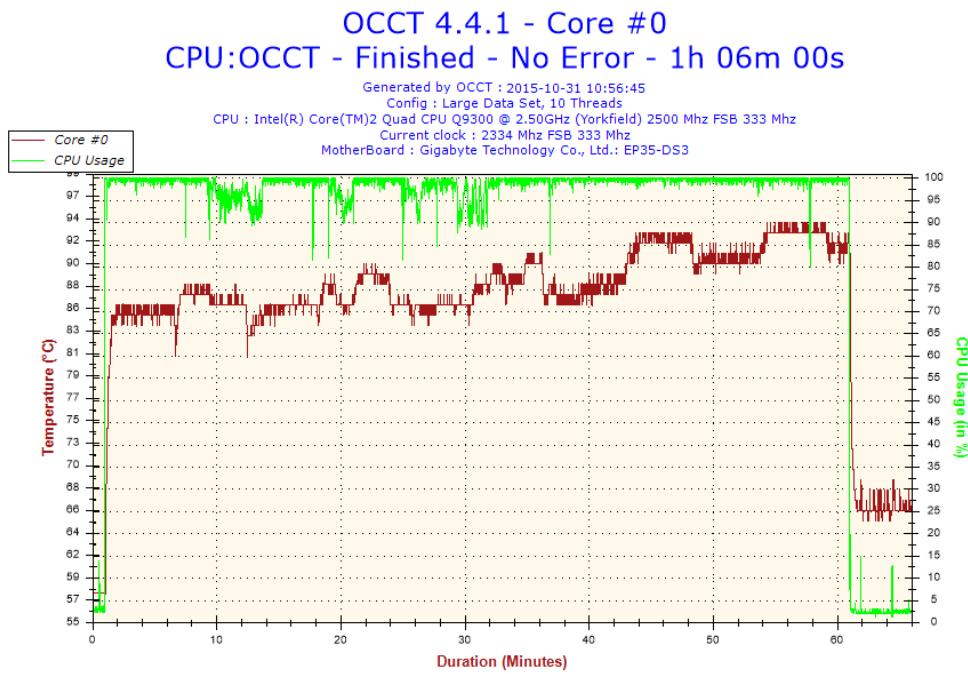
4.2. Analiza wyników

Punktem odniesienia dla przeprowadzanego eksperymentu była wydajność chłodzenia standardowego. W tym celu przeprowadzono test przy użyciu programu OCCT. Załączono test trwający godzinę, który maksymalnie obciążał procesor. Wyniki testu przedstawione zostały na wykresach (Rysunek 4.1, Rysunek 4.2, Rysunek 4.3, Rysunek 4.4).

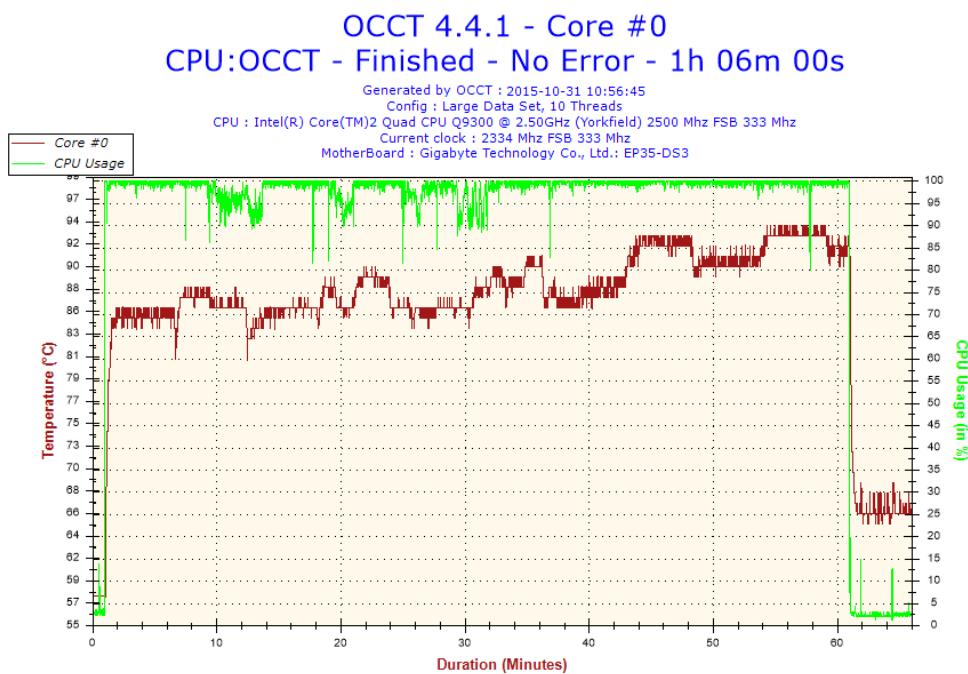
Podczas testu standardowego chłodzenia zaraz po obciążeniu procesora temperatura na wszystkich czterech rdzeniach momentalnie wzrasta i oscyluje w okolicach poziomu 88°C. Widoczne na wykresie kilkustopniowe chwilowe wzrosty temperatury spowodowane są sterowaniem

Podzespół	Zainstalowany sprzęt
Płyta główna	Gigabyte® GA-EP35-DS3 Mostek północny: Chipset Intel® P35 Express Mostek południowy: Intel® ICH9 FSB: 1600 (O.C.)/1333/1066/800 MHz Złącza PCI: 1 x PCI Express x16, 3 x PCI Express x1 RAM: DDR2 1200(O.C.)/1066/800/667 MHz
Procesor	Intel® Core™ 2 Quad Q9300 Rdzenie: 4 x 2.50 GHz FSB: 1333 MHz Zestaw instrukcji: 64 Litografia 45 nm
Karta graficzna	Gigabyte® GV-N98TSL-1GI Chipset: GeForce 9800 GT Pamięć: GDDR3 1 GB Magistrala pamięci: 256 bitów Magistrala karty graficznej: PCI-E 2.0 Wsparcie DirectX: 10 Wsparcie OpenGL: 2.1
Pamięć RAM	GEIL PC2-6400 800Mhz Rodzaj pamięci: DDR2 Pojemność pamięci: 4 GB (2x2GB)
Dysk twardy	Western Digital WD5000AZRX Pojemność: 500 GB Pamięć cache: 64 MB Prędkość obrotowa: 7200 obr./min. Interfejs: Serial ATA
Zasilacz	MODECOM FEEL 1 500ATX Moc: 500 W Standard: ATX 2.2

Tablica 4.1. Wykaz podzespołów testowanego komputera



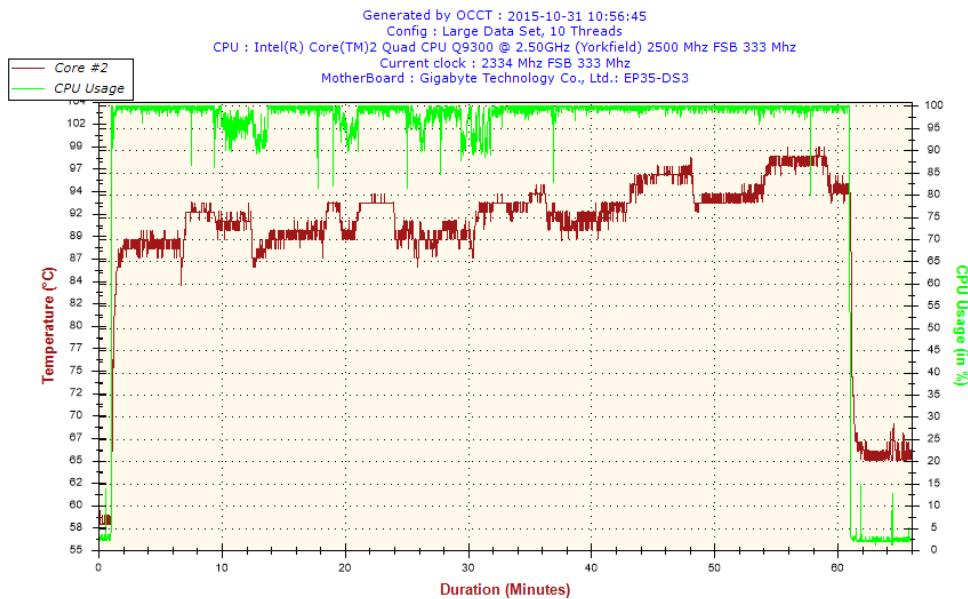
Rysunek 4.1. Wykres temperatur rdzenia 0 wykonany w OCCT



Rysunek 4.2. Wykres temperatur rdzenia 1 wykonany w OCCT

OCCT 4.4.1 - Core #2

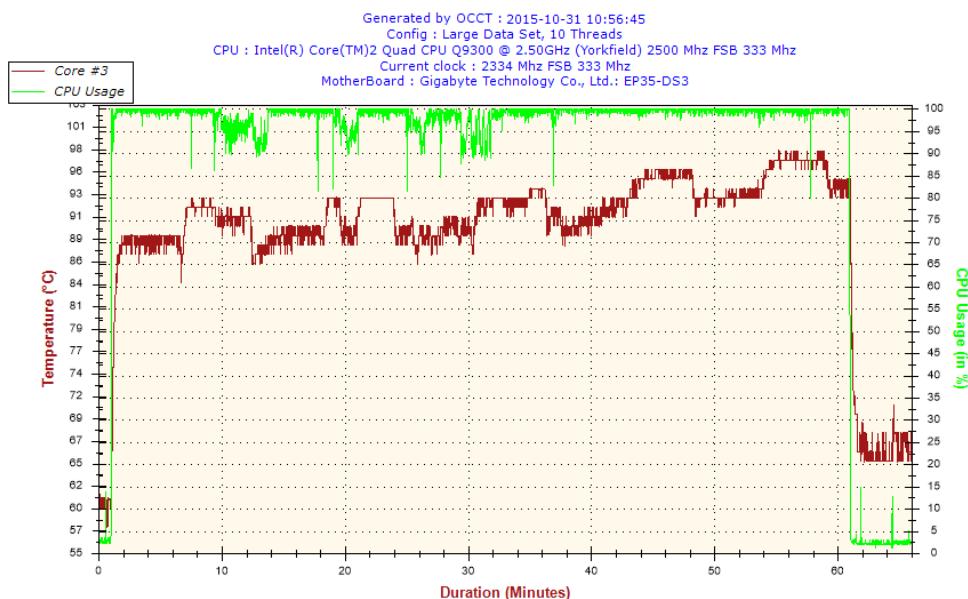
CPU:OCCT - Finished - No Error - 1 h 06m 00s



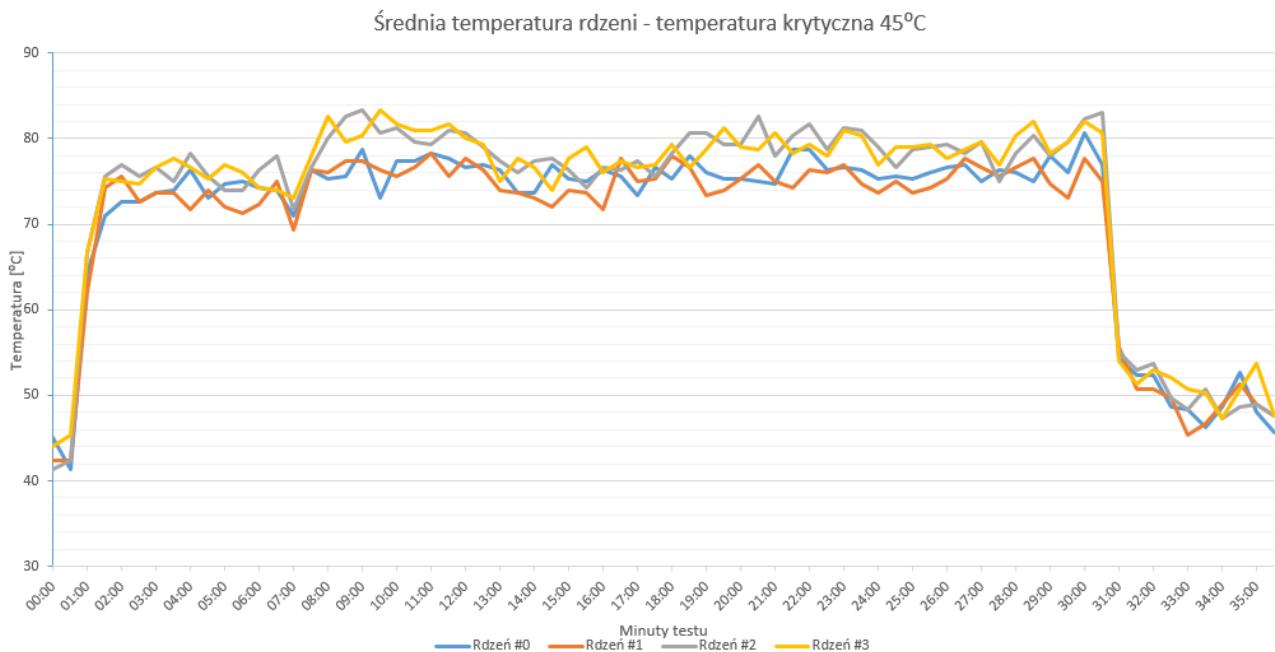
Rysunek 4.3. Wykres temperatur rdzenia 2 wykonany w OCCT

OCCT 4.4.1 - Core #3

CPU:OCCT - Finished - No Error - 1 h 06m 00s



Rysunek 4.4. Wykres temperatur rdzenia 3 wykonany w OCCT



Rysunek 4.5. Wykres temperatur poszczególnych rdzeni dla temperatury krytycznej 45°C wykonany w Excel 2013

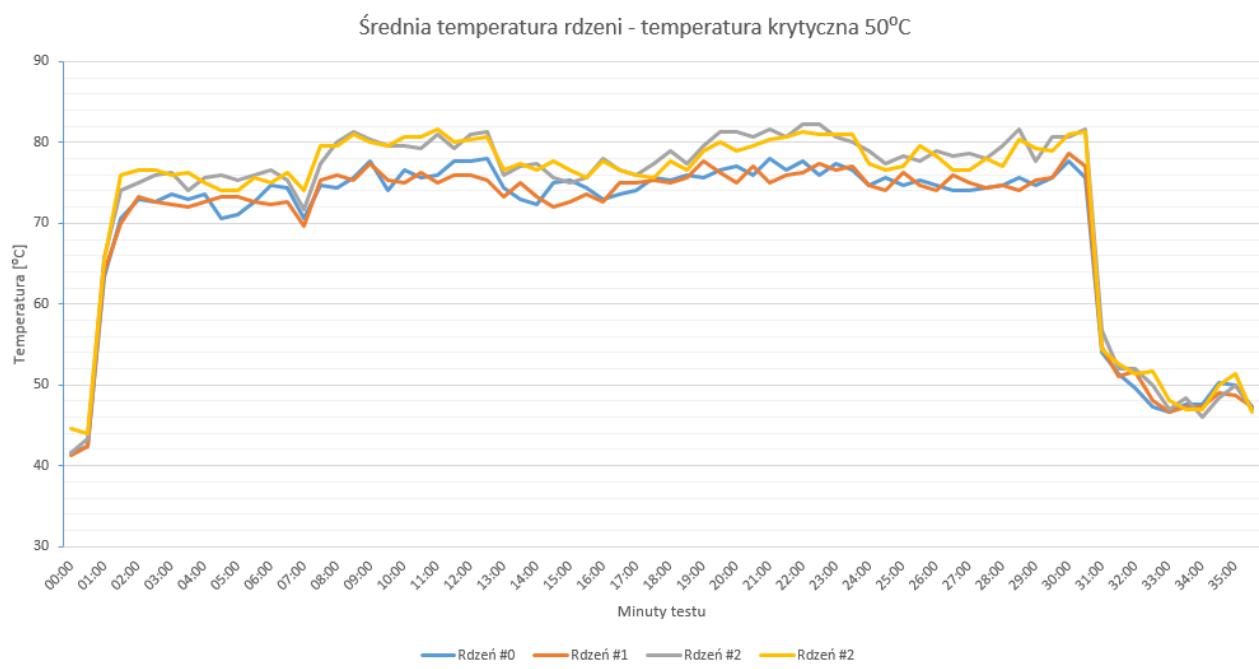
wentylatora przez system. Pomimo tych chwilowych wzrostów temperatury zauważać można tendencję wzrostową średniej temperatury. Na początku obciążenia procesora temperatura waha się w okolicach 88°C, natomiast po upływie około godziny testu temperatura wzrosła do okolic 98°C, a nawet momentami zbliżała się do 100°C co jest niebezpieczne dla procesora.

Po przeprowadzeniu testów zbudowanego układu chłodzącego wyniki zaprezentowano na wykresach.

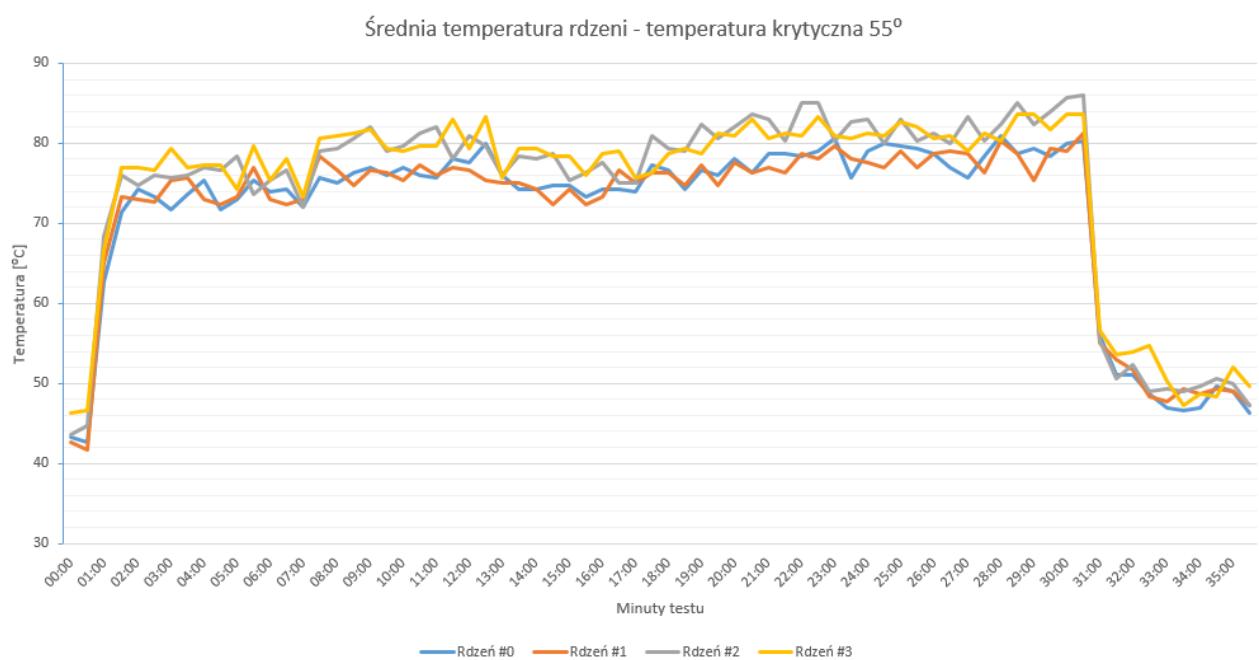
Temperatura na wszystkich wykresach oscyluje głównie w granicach 70°C - 80°C. Na pierwszym wykresie (Rysunek 4.1) w okolicach 12 minut trwania testu zauważać można spadek temperatury na wszystkich czterech rdzeniach. Jest to moment załączenia pompy. Podobny spadek temperatury widać na schemacie (Rysunek 4.2) odpowiadającym za reprezentowanie wyników dla temperatury krytycznej ustawionej na 50°C. W tym przypadku pompa jednak załączyła się jednak około 10 minut później, w okolicach 24 minut trwania testu. Podczas trwania testów dla temperatury załączenia pompy ustawionej na 55°C nie zauważono załączenia pompy, co widać na wykresie (Rysunek 4.3).

Kolejny wykres (Rysunek 4.4) przedstawia średnią temperaturę wszystkich rdzeni zaobserwowaną dla poszczególnych ustawień temperatury krytycznej.

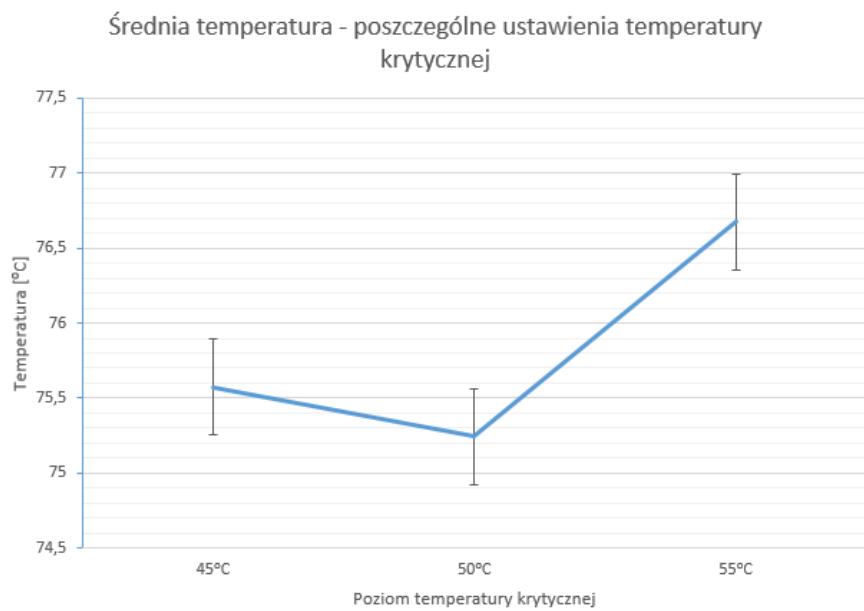
Różnica pomiędzy średnią temperaturą dla 45°C wynoszącą 75,6°C a 50°C, której wartość wynosi około 75,3 jest niewielka (jedynie 0,3°C) i mieści się w zaznaczonym błędzie. Niższy poziom średniej temperatury dla 50°C jest wynikiem błędu w pomiarze oraz zaokrąglenia nanoszonego przez komputer. Tak niewielką różnicę można pominąć i traktować wyniki jako zbliżone do siebie. Dopiero średnia temperatura dla temperatury krytycznej ustawionej na poziomie 55°C, jest wyższa niż temperatury w dwóch pozostałych przypadkach. Wynosi ona 76,7°C. Wywołane jest to brakiem załączenia pompy, ustawienie to nie załączyło pompy w czasie 30 minut trwania testu.



Rysunek 4.6. Wykres temperatur poszczególnych rdzeni dla temperatury krytycznej 50°C wykonany w Excel 2013



Rysunek 4.7. Wykres temperatur poszczególnych rdzeni dla temperatury krytycznej 55°C wykonany w Excel 2013



Rysunek 4.8. Wykres średniej temperatury rdzeni dla różnych temperatur krytycznych wykonany w Excel 2013

Zastosowanie zbudowanego systemu chłodzącego zaowocowało spadkiem średniej temperatury rdzeni procesora o około 15°C. Po modernizacji systemu chłodzenia temperatura rdzeni procesora, nie zbliżyła się ani razu do niebezpiecznego poziomu 100°C, co więcej ledwo przekroczyła 80°C. Porównując to do wyników sprzed modernizacji jest dobrym wynikiem, który dowodzi skuteczności zastosowanego systemu.

Rozdział 5

Podsumowanie

Celem powyższej pracy dyplomowej było zaprojektowanie i zbudowanie automatycznego systemu chłodzącego. Rolę sterownika w projekcie przyjęła platforma Arduino. Wykorzystana platforma spełniła swoje zadanie i wykonuje wszystkie funkcje przewidziane dla niej w projekcie algorytmu głównego. Analiza wyników testu dowiodła iż zbudowany automatyczny system chłodzenia spełnia swoją rolę. Temperatura rdzeni procesora zmniejszyła się o około 15°C co jest dobrym wynikiem. Pomimo maksymalnego obciążenia procesora, temperatura na jego rdzeniach nie zbliżyła się do niebezpiecznego poziomu 100°C. Zapewnia to bezpieczne korzystanie z komputera nawet przy bardzo wymagających zadaniach.

Niezbędne okazało się przetłaczanie cieczy przez chłodnicę. Chłodziwo bez obiegu chłodzącego odbierało ciepło z podzespołów, lecz nie miało jak oddać go do otoczenia. Pomimo zauważalnej skuteczności systemu stwierdzono, iż zastosowanie wentylatora chłodzącego chłodnicę, zwiększyłoby skuteczność zastosowanego systemu chłodzącego. Jak wiadomo skuteczność zbudowanego układu chłodzącego zależy w dużej mierze od chłodzenia cieczy poza zbiornikiem z podzespołami komputera. W projekcie nie przewidziano funkcji wskazywania przepływu cieczy. Funkcja ta mogłaby być wskaźnikiem wystąpienia nieszczelności w układzie obiegu chłodzącego, lub uszkodzenia pompy.

Wybór cieczy zastosowanej jako chłodziwo stawał na bezpieczeństwo. Głównym czynnikiem była przewodność prądu, a właściwie jej brak oraz dostępność na rynku. Rozwijając projekt można by zastanowić się nad właściwościami termicznymi różnych cieczy. Przeprowadzenie szeregu badań z zastosowaniem różnych cieczy mogłoby zwiększyć skuteczność zbudowanego systemu oraz pokazać jego skuteczność podczas dłuższego obciążenia.

Bibliografia

- [1] Arduino, "Introduction." <https://www.arduino.cc/en/Guide/Introduction#>, 2016. Dostępne: 27-styczeń-2016.
- [2] Wikipedia, "Wiring, development platform." [https://en.wikipedia.org/wiki/Wiring_\(development_platform\)](https://en.wikipedia.org/wiki/Wiring_(development_platform)), 2015. Dostępne: 27-styczeń-2016.
- [3] Wikipedia, "Processing, programming language." [https://en.wikipedia.org/wiki/Processing_\(programming_language\)](https://en.wikipedia.org/wiki/Processing_(programming_language)), 2016. Dostępne: 27-styczeń-2016.
- [4] CadSoft, "What is eagle." <http://www.cadsoftusa.com/eagle-pcb-design-software/about-eagle/>, 2011. Dostępne: 27-styczeń-2016.
- [5] Wikipedia, "Autocad." <https://pl.wikipedia.org/wiki/AutoCAD#Historia>, 2015. Dostępne: 27-styczeń-2016.
- [6] dobreprogramy, "Occt 4.4.1." <http://www.dobreprogramy.pl/OCCT,Program,Windows,28567.html>, 2014. Dostępne: 27-styczeń-2016.
- [7] P. Hempowicz, R. Kiełsznia, A. Piłatowicz, J. Szymczyk, T. Tomborowski, A. Wąsowski, A. Zielińska, and W. Żurawski, *Elektrotechnika i elektronika dla niewielokrotników*. Wydawnictwa Naukowo-Techniczne, 2004.
- [8] M. Riley, *Inteligentny dom. Automatyzacja mieszkania za pomocą platformy Arduino, systemu Android i zwykłego komputera*. Helion, 2013.
- [9] P. Bastian, G. Schuberth, O. Spielvogel, H.-J. Steil, K. Kotz, and K. Ziegler, *Praktyczna elektrotechnika ogólna*. REA, 2003.

Spis rysunków

1.1.	Gotowy projekt	5
2.1.	Arduino UNO	6
2.2.	Arduino IDE	7
2.3.	Program EAGLE - wszystkie okna	8
2.4.	Program Fritzing - edytor schematów poglądowych	9
3.1.	Schemat poglądowy układu chłodzenia wykonany w AutoCAD 2016	12
3.2.	Rzut izometryczny bryły pojemnika wraz z wymiarami wykonany w AutoCAD 2016	13
3.3.	Elementy obudowy wykorzystane w projekcie	14
3.4.	Rama z zamontowanymi podzespołami komputera	14
3.5.	Zmontowane akwarium z chłodnicą i pompą	15
3.6.	Schemat poglądowy wykonany we Fritzing	16
3.7.	Schemat elektryczny układu współpracującego z Arduino wykonany w EAGLE	17
3.8.	Zmontowany moduł komunikacji	18
3.9.	Podłączony układ elektryczny współpracujący z Arduino	19
3.10.	Diagram przepływu pracy sterownika	20
4.1.	Wykres temperatur rdzenia 0 wykonany w OCCT	27
4.2.	Wykres temperatur rdzenia 1 wykonany w OCCT	27
4.3.	Wykres temperatur rdzenia 2 wykonany w OCCT	28
4.4.	Wykres temperatur rdzenia 3 wykonany w OCCT	28
4.5.	Wykres temperatur poszczególnych rdzeni dla temperatury krytycznej 45°C wykonany w Excel 2013	29
4.6.	Wykres temperatur poszczególnych rdzeni dla temperatury krytycznej 50°C wykonany w Excel 2013	30
4.7.	Wykres temperatur poszczególnych rdzeni dla temperatury krytycznej 55°C wykonany w Excel 2013	30
4.8.	Wykres średniej temperatury rdzeni dla różnych temperatur krytycznych wykonany w Excel 2013	31

Lista algorytmów

1	Inicjalizacja urządzenia Arduino	21
2	Funkcja setup()	22
3	Funkcja loop()	23

Spis tablic

4.1. Wykaz podzespołów testowanego komputera	26
--	----