1. Temat projektu i główne założenia projektu:

Temat: Chłodnica do akwarium

Założenia projektu: Zestaw wentylatorów komputerowych PWM wiejących na taflę wody, sterowanych przez mikrokontroler w zależności od wyniku pomiaru temperatury wody sondą wodoodporną. Wynik pomiaru oraz zadana temperatura mogą zostać wyświetlone na wyświetlaczu LCD. Dodatkowo można ustawić docelową temperaturę wody „w locie" bez konieczności przeprogramowania mikrokontrolera. Układ dodatkowo posiada czujnik poziomu wody. Przy niskim poziomie wody następuje awaryjne zwolnienie wentylatorów do prędkości minimalnej. Układ jest zasilany zasilaczem wpiętym do gniazdka sieciowego.

2. Analiza zadania oraz wybór elementów i narzędzi

2.1. Mikrokontroler AVR – Atmega328P-U DIP

Napięcie zasilania 1,8-5,5V
Pamięć Flash 32KB
23 linie we/wy
Dwa liczniki 8-bitowe, jeden licznik 16-bitowy
6 kanałów PWM
10-bitowy konwerter A/C
Cena: 12,90z;

Zalety:
- niska cena
- duża liczba materiałów edukacyjnych
- aż 6 kanałów PWM
- trzy liczniki
- wbudowany konwerter A/C

Alternatywy:
ATmega168A-PU DIP
- mniej kanałów PWM
- mniejsza pamięć Flash
- wyższa cena

ATmega88PA-PU DIP
- brak kanałów PWM
- mniejsza pamięć Flash
- identyczna cena

2.2. Wentylator SilentiumPC Sigma HP120mm PWM

Zasilanie 12V/280mA
Max. hałas 15dB
Max. RPM 2300
Żywotność 50000h
Złącze PWM
Cena: 22zł

Zalety:
- bardzo niska cena (najtańszy wentylator 120mm)
- złącze PWM
- renomowana firma
- cicha praca

Alternatywy:
Chieftec AF-0825
- mniejsza średnica (80mm)
- głośność aż 32dB
- mniejszy przepływ powietrza
- różnica w cenie raptem 3zł

2.3. Wyświetlacz LCD 2x16 znaków + konwerter I2C LCM1602

Matryca 2x16 znaków
Podświetlenie niebieskie
Zasilanie 5V
Cena: 17,90zł

Zalety:
- niska cena
- konwerter I2C

Alternatywy:
-wyświetlacz bez konwertera I2C, tańszy o kilka złotych, za to bardziej złożony w podłączeniu
-wyświetlacz 4x20 znaków – droższy od wyświetlacza 2x16 znaków, który jest w pełni wystarczająca

2.4. Sonda wodoodporna z czujnikiem temperatury DS18B20

Napięcie zasilania 3-5V
Zakres temperatury pracy -55C-125C
Dokładność ±0,5C w zakresie -10C-85C
Długość przewodu 1m
Cena 9,95zł

Zalety:
- bardzo niska cena
- istniejące biblioteki
- dobra dokładność
- 1-wire
- dostępność

Alternatywy:
Sonoff DS18B20
- złącze jack
- dwukrotnie wyższa cena
- identyczne pozostałe parametry

2.5. Czujnik poziomu cieczy - analogowy - Iduino SE045

Napięcie zasilania 5V
Zakres pomiarowy: do 40 mm (od 0 V do 3,5 V)
Cena: 2,90zł

Zalety:
- bardzo niska cena w porównaniu do innych czujników

Alternatywy:
Waveshare 9525

- większy zakres pomiarowy (48mm)
- czterokrotnie większa cena

CMW55 – magnetyczny
- pływak z magnesem
- sześciokrotnie wyższa cena
- wyższe napięcie zasilania (12V)
- brak kabla informacyjnego, czujnik albo przewodzi prąd, albo nie

2.6. Przetwornica step-down D-SUN 1,0V-17V 1,8A

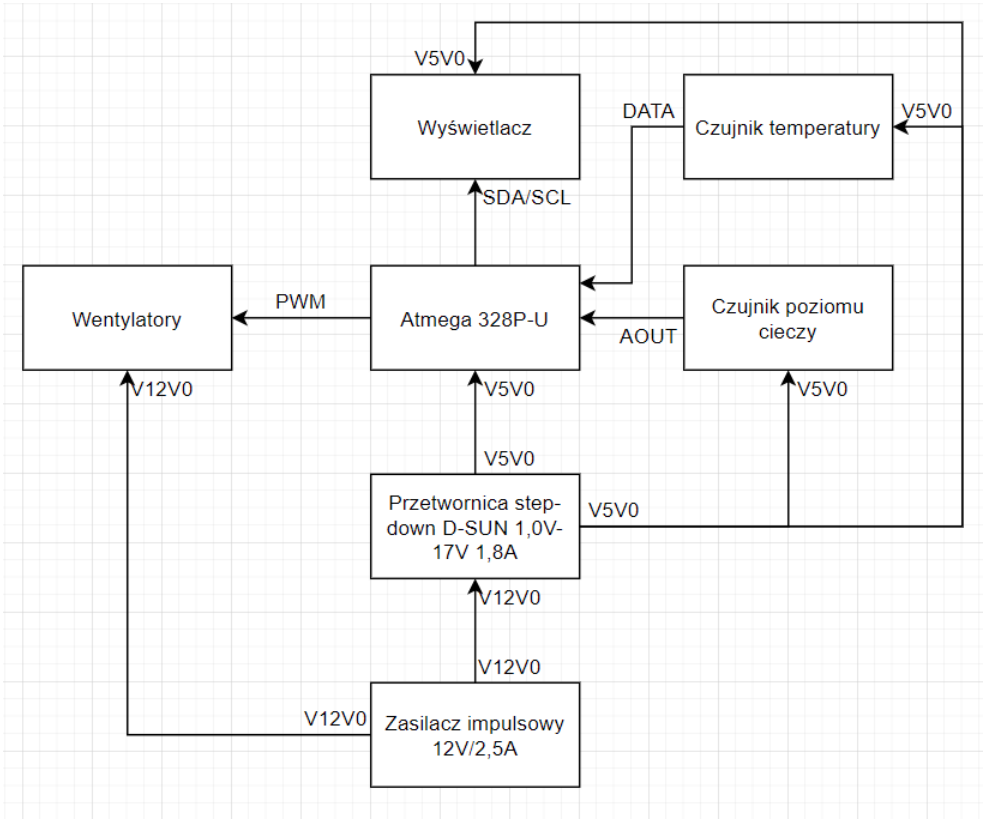
Napięcie wejściowe: od 1 V do 23 V
Napięcie wyjściowe regulowane w zakresie: od 1 V do 17 V
Maksymalny ciągły prąd wyjściowy: 1,8A (3 A przy zastosowaniu chłodzenia)
Cena: 5,90zł

Zalety:
- niska cena
- możliwość ustawienia napięcia wejściowego i wyjściowego
- wystarczające parametry

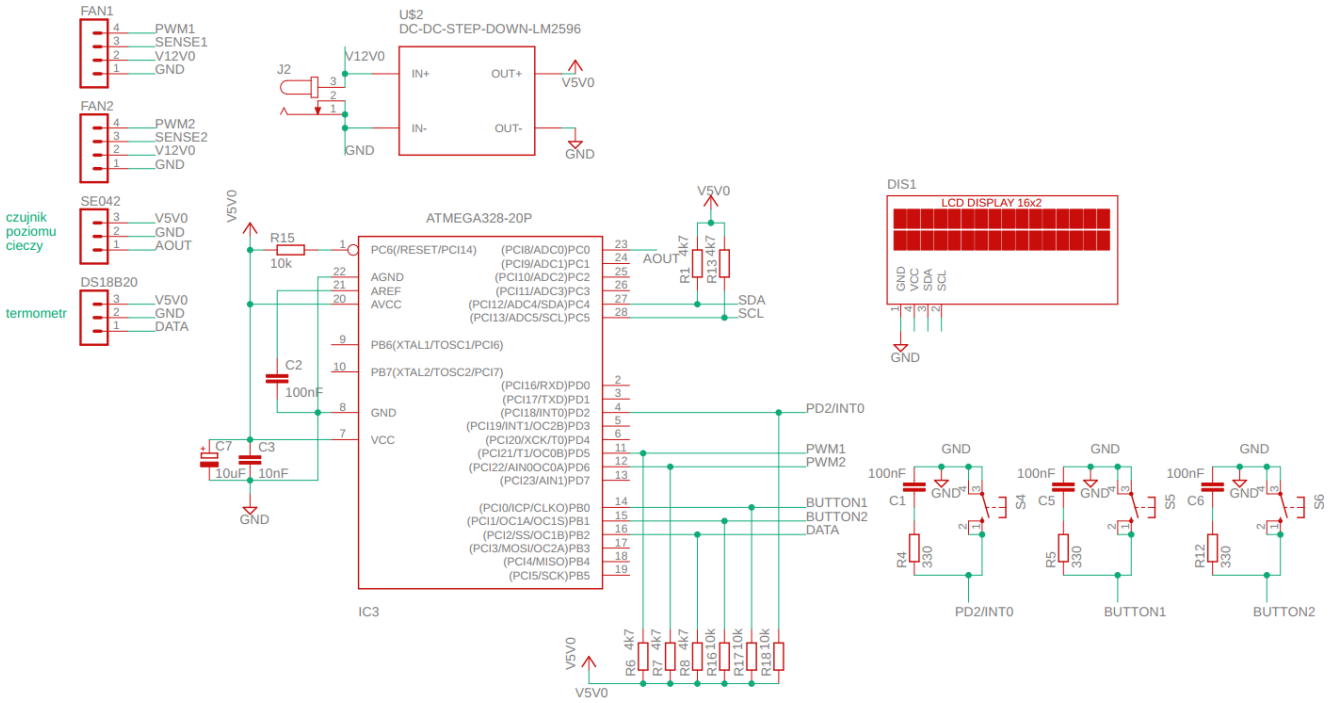
Alternatywa:
Przetwornica step-down LM2596 3,2V-35V 3A
- większy zakres napięcia wejściowego i wyjściowego
- większy maksymalny prąd wyjściowy
- większa cena

3. Specyfikacja wewnętrzna

3.1. Schemat blokowy



3.2. Schemat ideowy



3.3. Opis funkcji bloków układu

Zasilacz doprowadza napięcie do wentylatorów i **przetwornicy** step-down, która obniża napięcie z 12V do 5V i zasila pozostałe elementy układu (mikrokontroler, wyświetlacz, termometr, czujnik poziomu cieczy).

Sercem urządzenia jest **mikrokontroler** dokonujący pomiarów temperatury oraz poziomu wody za pomocą odpowiednich czujników, wyświetla temperaturę na wyświetlaczu oraz steruje pracą wentylatorów za pomocą kanału PWM.

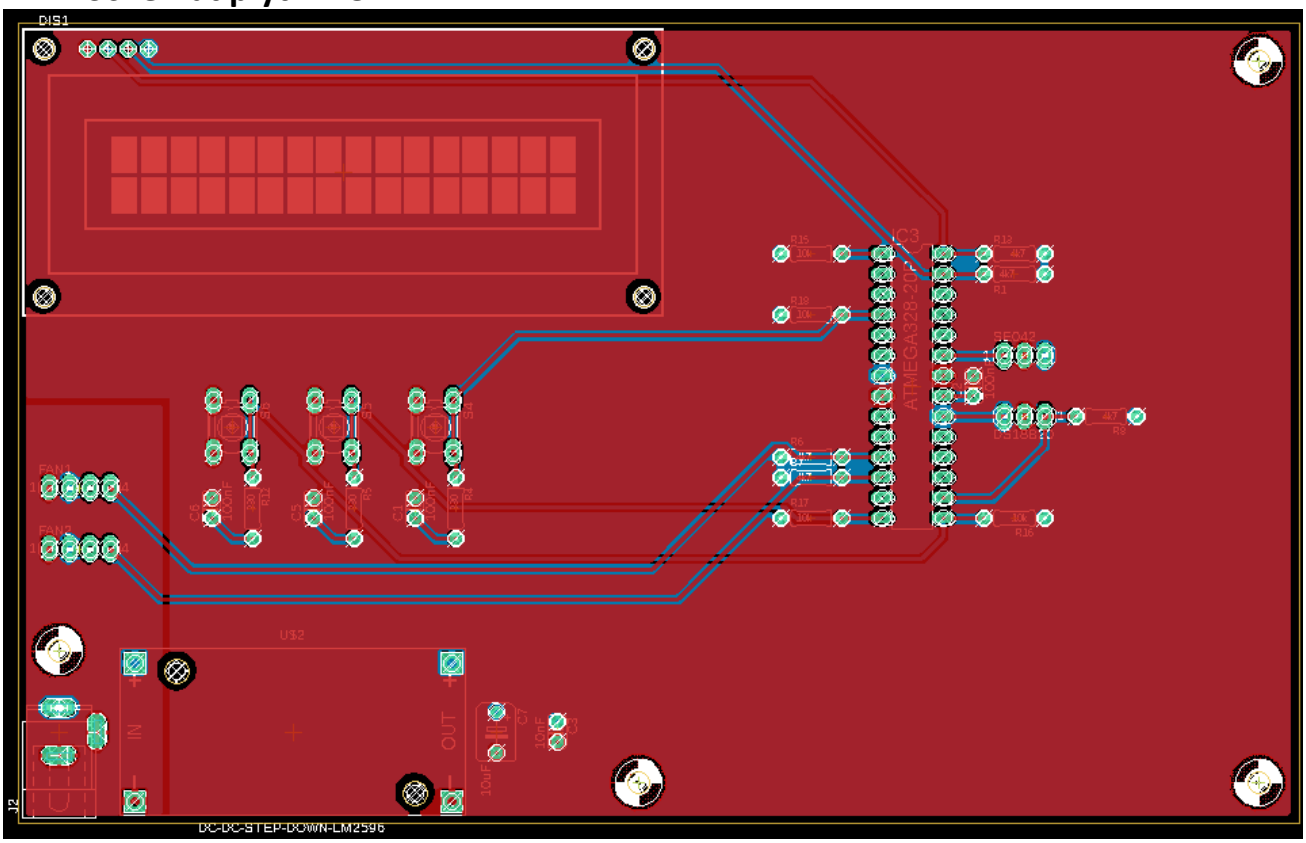
Termometr DS18B20 jest wodoodporną sondą zamocowaną na przewodzie i jest umieszczany w akwarium, mniej więcej na połowie głębokości.

Czujnik poziomu cieczy SE042 jest czujnikiem rezystancyjnym, analogowym, przymocowanym do krawędzi akwarium.

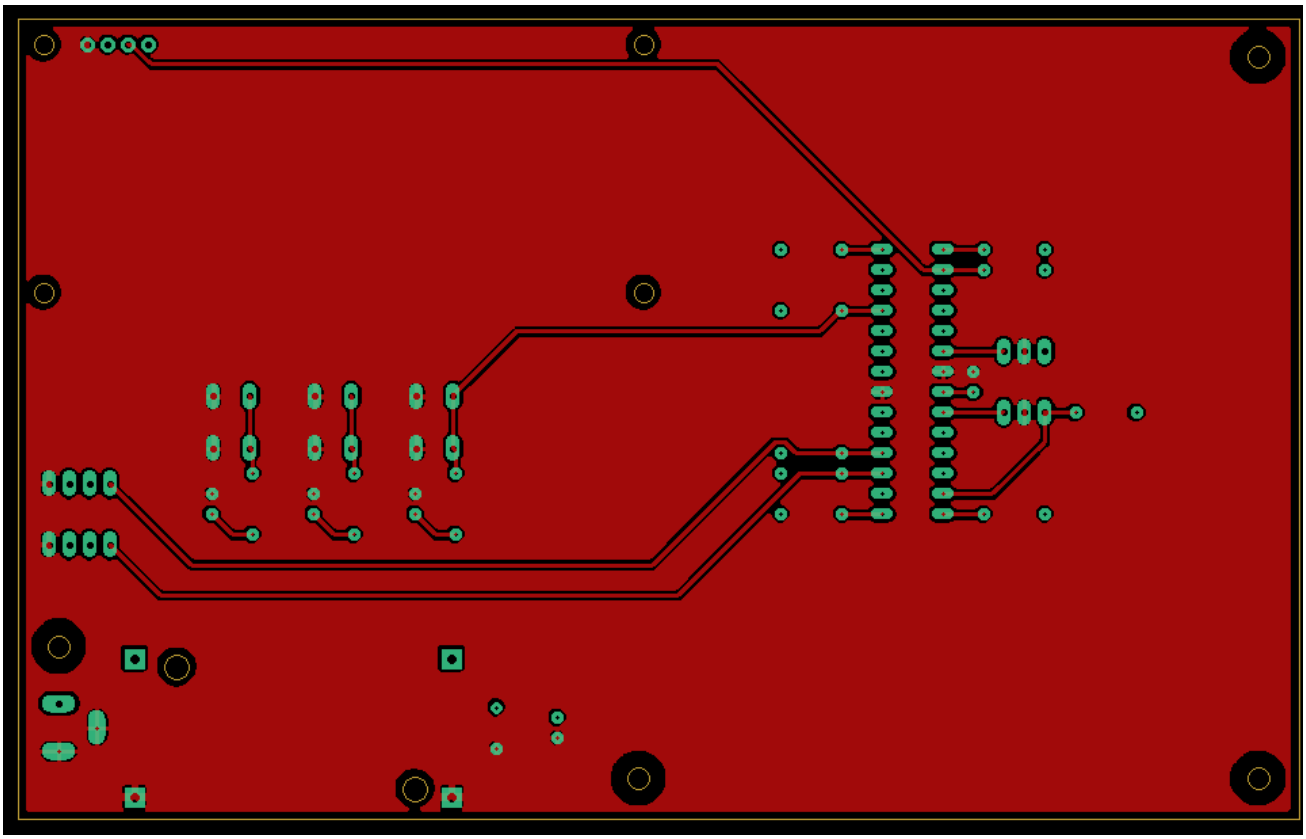
Wyświetlacz LCD 16x2 jest wyposażony w konwerter I2C upraszczający jego podłączenie do mikrokontrolera.

Wentylatory są sterowane kanałem PWM.

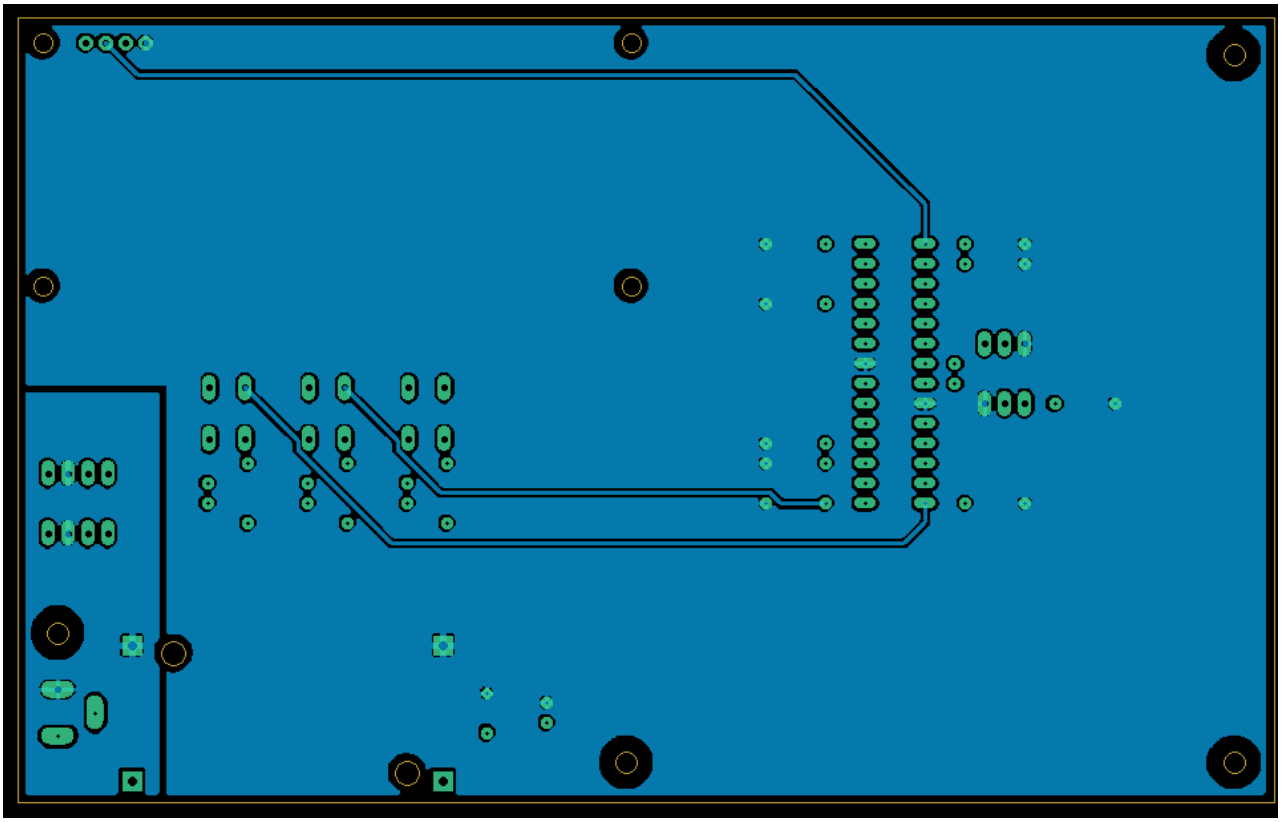
3.4. Schemat płytki PCB



all



top

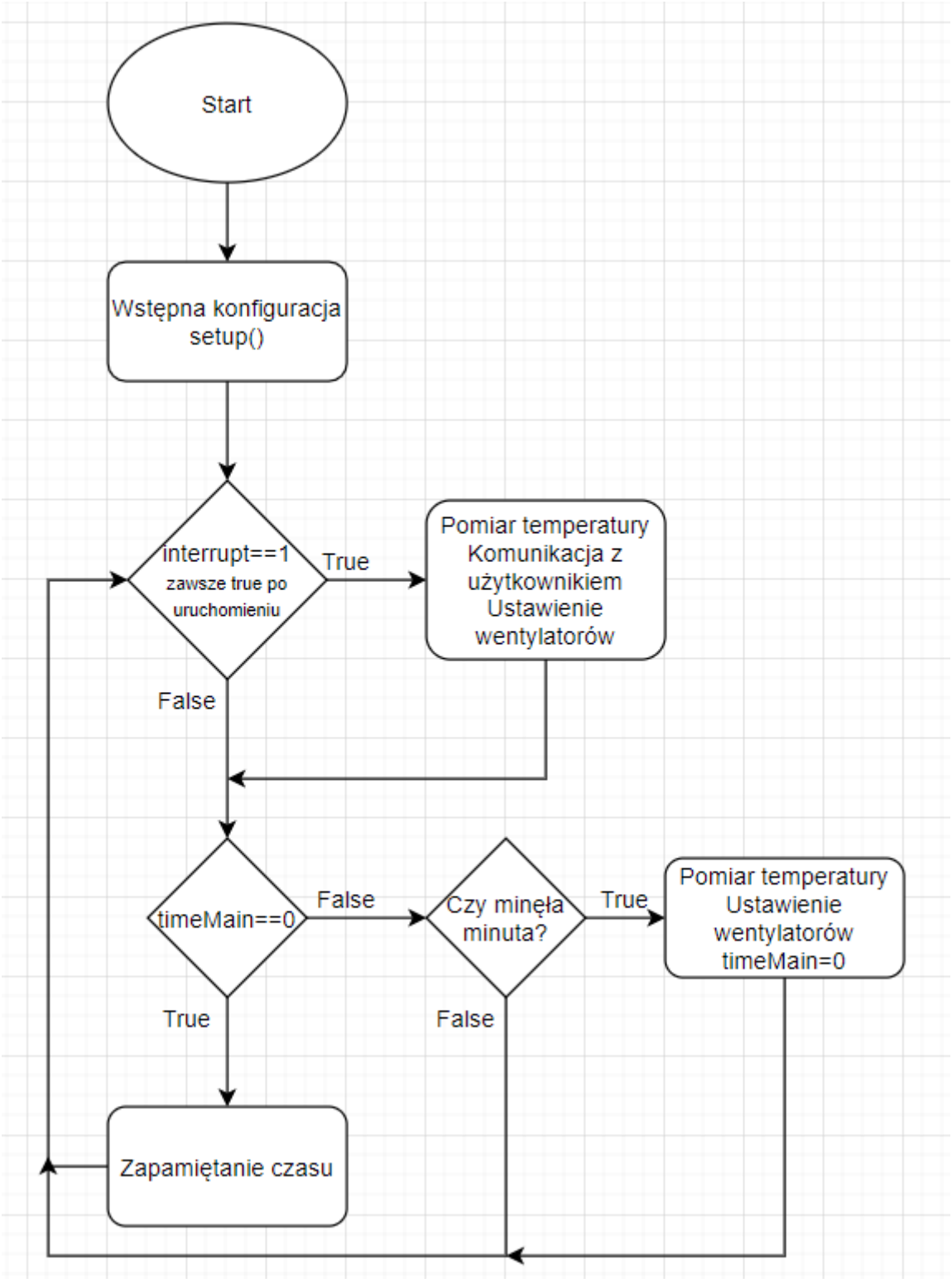


bottom

3.5. Lista elementów

- Mikrokontroler - ATMEGA328 20P
- Wyświetlacz LCD DISPLAY 16x2 – DIS1
- Przetwornica – DC-DC-STEP-DOWN-LM2596
- Kondensatory THT 10nF i 100nF
- Kondensator elektrolityczny THT 10uF
- Rezystory 330, 4k7 i 10k
- Złącze DC żeńskie
- Przyciski THT
- Goldpiny zewnętrznych elementów

3.6. Schemat blokowy



3.7. Opis wszystkich ważniejszych zmiennych

```
float temperatureSet = 25.0f; //zmienna przechowująca zadaną temperaturę
float tempC = 0.0f; //zmienna przechowująca zmierzoną temperaturę
unsigned long timeMain = 0; //główna zmienna wykorzystywana przy pomiarach czasu
bool interrupt = 1; //flaga sygnalizująca konieczność wykonania funkcji przerwania
int liquidLevel = 0; //funkcja przechowująca zmierzony poziom wody (0-1024)
```

3.8. Opis funkcji wszystkich procedur

```
void lcdInit() //funkcja inicjalizująca pracę wyświetlacza
float tempCheck() //funkcja dokonująca pomiaru temperatury i zwracająca wartość pomiaru
```

```
wyrażoną w stopniach Celsjusza
void tempCheckManual() //funkcja uruchamiana po wciśnięciu przycisku wyświetlająca
wynik pomiaru na wyświetlaczu oraz obsługująca zmianę zadanej temperatury
void setupTimer2() //funkcja konfigurująca timer2, który jest wykorzystywany przy
sterowaniu PWM
void setPWM2(float f) //funkcja ustawiająca wypełnienie PWM zgodnie z przekazanym
parametrem
void pwmSet() //funkcja obliczająca wypełnienie PWM i wywołująca funkcję setPWM2()
void buttonInterrupt() //funkcja ustawiająca flagę przerwania
void setup() //funkcja inicjalizująca pracę układu m.in. ustawiająca tryb pracy
poszczególnych pinów
void loop() //główna funkcja programu, wykonująca się cyklicznie przez cały czas pracy
mikrokontrolera
```

3.9. Opis interakcji oprogramowania z układem elektronicznym

Czujnik poziomu cieczy SE045 – rezystancyjny czujnik wyposażony w trzy piny (Vcc, GND, Data). Linię data należy podłączyć do pinu analogowego mikrokontrolera, który odczytuje wartość z przedziału 0-1024 gdzie 0 oznacza brak wody, 1024 pełne zanurzenie.

Czujnik temperatury DS18B20 posiada trzy przewody (czerwony Vcc, czarny GND, żółty Data). Sygnał Data jest sygnałem cyfrowym, a pin do którego zostanie podłączony należy podciągnąć rezystorem do zasilania. Przesłaną wartość mikrokontroler musi przekonwertować na stopnie Celsjusza.

Wyświetlacz LCD 16x2 z konwerterem I2C posiada cztery piny (Vcc, GND, SDA, SCL), ostatnie dwa należy podciągnąć przez rezystor do zasilania.

3.10. Szczegółowy opis działania ważniejszych procedur

```
void setup() {
  //===PWM===
  pinMode(3,OUTPUT);
  setupTimer2();
  setPWM2(0.0f); //set duty to 0% on pin 3
  //Buttons
  pinMode(8,INPUT);
  pinMode(9,INPUT);
  //Liquid Sensor
  pinMode(14,INPUT);

  lcdInit();
  attachInterrupt(digitalPinToInterrupt(2), buttonInterrupt, LOW); //Przerwanie na
pinie 2
}
```

Główna funkcja konfigurująca. Funkcja ta ustawia odpowiednie tryby na wskazanych pinach do obsługi kanału PWM, przycisków, odczytu pomiaru poziomu wody, konfiguruje zewnętrzne przerwanie wywoływane przez przycisk oraz uruchamia funkcję inicjalizującą wyświetlacz.

```
void lcdInit(){
  Wire.setClock(10000);
  lcd.begin(16,2);   // Inicjalizacja LCD 2x16
  lcd.backlight(); // zalaczenie podwietlenia
  lcd.clear();
  lcd.setCursor(0,0); // Ustawienie kursora w pozycji 0,0 (pierwszy wiersz, pierwsza
kolumna)
  lcd.print("LCD OK...");
  delay(2000);
  lcd.noBacklight();
}
```

Funkcja inicjalizująca wyświetlacz oraz wyświetlająca próbną wiadomość.

```
void loop() {
```

Główna funkcja programu

```
  if(interrupt == 1){
    do{
      tempC = tempCheck();
```

```
    }while(tempC<10.0f);
    tempCheckManual();
    pwmSet();
    interrupt = 0;
  }
```

Ponieważ flaga *interrupt* ma domyślną wartość 1, ta część funkcji wykona się zawsze po pierwszym uruchomieniu. W trakcie działania programu flaga ta jest ustawiana przez funkcję wywołaną przez zewnętrzne przerwanie. Temperatura zwrócona przez funkcję *tempCheck()* jest przekazywana do zmiennej *tempC*. Ta część programu jest umieszczona w pętli *do while*, aby pomiar został powtórzony gdy poprzedni nie wykonał się poprawnie (zwrócona wartość 0). Następnie jest uruchamiana funkcja *tempCheckManual()* która wyświetla temperaturę na wyświetlaczu oraz pozwala na inkrementację bądź dekrementację zadanej temperatury. Kolejno wywoływana zostaje funkcja obliczająca wypełnienie kanału PWM oraz flaga *interrupt* jest ustawiana na 0.

```
  if(timeMain == 0) timeMain = millis();
  else if(timeMain + 60000< millis()){
    do{
      tempC = tempCheck();
    }while(tempC<10.0f);
    pwmSet();
    timeMain = 0;
  }
}
```

Ponieważ zmienna *timeMain* ma domyślną wartość 0 po uruchomieniu, warunek zostanie spełniony i do zmiennej zostanie przypisany aktualny czas zwrócony przez funkcję *millis()*. W kolejnym przebiegu funkcji *loop()* zostanie sprawdzone czy od zapamiętany czas plus minuta (60000ms) jest mniejszy od aktualnego czasu. Jeśli tak jest, zostanie wykonany pomiar, ponownie zostanie obliczone wypełnienie PWM, a zapamiętany czas zostanie wyzerowany pozwalając na odmierzenie następnej minuty w kolejnym przebiegu pętli.

```
float tempCheck() {//returns temperature in celcious (float)
  byte i;
  byte present = 0;
  byte type_s;
  byte data[12];
  byte addr[8];
  float celsius;

  if ( !ds.search(addr)) {
    ds.reset_search();
    delay(250);
    return;
  }

  if (OneWire::crc8(addr, 7) != addr[7]) {
      return;
  }

  // the first ROM byte indicates which chip
  switch (addr[0]) {
    case 0x10:
      //Serial.println("  Chip = DS18S20");  // or old DS1820
      type_s = 1;
      break;
    case 0x28://
      //Serial.println("  Chip = DS18B20");
      type_s = 0;
      break;
    case 0x22:
      //Serial.println("  Chip = DS1822");
      type_s = 0;
      break;
    default:
```

```
      //Serial.println("Device is not a DS18x20 family device.");
      return;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44, 1);        // start conversion, with parasite power on at the end

  delay(1000);
  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);         // Read Scratchpad

  for ( i = 0; i < 9; i++) {           // we need 9 bytes
    data[i] = ds.read();
  }

  int16_t raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3; // 9 bit resolution default
    if (data[7] == 0x10) {
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
    else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
    else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
    // default is 12 bit resolution, 750 ms conversion time
  }
  celsius = (float)raw / 16.0;
  return celsius;
 }
```

Funkcję wykonującą pomiar temperatury i konwertującą otrzymaną wartość na stopnie zaczerpnięto za [strony produktu] oraz zmodyfikowano usuwając elementy odpowiedzialne za komunikację przez port COM oraz konwersję na stopnie Farenheita oraz Kelwiny.

```
void tempCheckManual(){
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temp:");
  lcd.setCursor(7,0);
  lcd.print(tempC);

  lcd.setCursor(0,1);
  lcd.print("Set temp:");
  lcd.setCursor(10,1);
  lcd.print(temperatureSet);

  unsigned long time1 = millis();
  while(true){
    if(!digitalRead(8)) {//dekrementacja zadanej temperatury
      temperatureSet-=0.2f;
      lcd.setCursor(10,1);
      lcd.print(temperatureSet);
      delay(250);
      time1 = millis();
    }
    if(!digitalRead(9)) {//inkrementacja zadanej temperatury
      temperatureSet+=0.2f;
      lcd.setCursor(10,1);
      lcd.print(temperatureSet);
      delay(250);
      time1 = millis();
    }
```

```
    if(time1+5000<millis()) break;
  }
  lcd.noBacklight();
}
```

Funkcja ta jest wywoływana przy pierwszym uruchomieniu oraz po wciśnięciu przycisku przerwania i odpowiada za komunikację z użytkownikiem. Wyświetla ona zmierzoną temperaturę oraz zadaną oraz pozwala na zmianę tej drugiej za pomocą dwóch przycisków. Każde wciśnięcie zmienia temperaturę o 0,2^{o}C. Funkcja jest aktywna przez 5 sekund, ale każde wciśnięcie przycisku resetuje ten czas, więc nie ma potrzeby zdążenia ze zmianą w 5 sekund. Przytrzymanie przycisku powoduje cykliczną zmianę temperatury w odstępach 250ms. Na koniec swojego działania funkcja wygasza wyświetlacz w celu oszczędzania energii.

```
void setupTimer2(){
    //Set PWM frequency to about 25khz on pin 3 (timer 2 mode 5, prescale 8, count to
79)
    TIMSK2 = 0;
    TIFR2 = 0;
    TCCR2A = (1 << COM2B1) | (1 << WGM21) | (1 << WGM20);
    TCCR2B = (1 << WGM22) | (1 << CS21);
    OCR2A = 79;
    OCR2B = 0;
}
//equivalent of analogWrite on pin 3
void setPWM2(float f){
    f=f<0?0:f>1?1:f;
    OCR2B = (uint8_t)(79*f);
}
```

Obie funkcje zostały zaczerpnięte ze strony https://fdossena.com/?p=ArduinoFanControl/i.md i odpowiadają za działanie kanału PWM. Pierwsza z nich konfiguruje timer2 mikrokontrolera, który jest niezbędny do generowania sygnału prostokątnego na wyjściu. Druga funkcja ustawia wypełnienie sygnału zgodnie z przekazaną jej wartością (typ float, zakres 0.0-1.0).

```
void pwmSet(){
  liquidLevel=analogRead(14);
  delay(200);
  if(liquidLevel<550) setPWM2(0.0f);
  else if(tempC>temperatureSet) setPWM2(max(min((tempC-
TemperatureSet)*2.0f,0.8f),0.2f)) ;//pwm duty min 0.2, max 0.8
  else if(tempC>temperatureSet-0.5f) setPWM2(0.2f);
  else setPWM2(0.0f);
}
```

Funkcja ta oblicza wypełnienie sygnału PWM w zależności od zmierzonego poziomu wody i temperatury. Wartości zwrócone przez funkcję *liquidLevel* mniejsze od 550 oznaczają, że poziom wody spadł poniżej mniej więcej połowy wysokości czujnika. Przy tak niskim poziomie wypełnienie jest ustawiane na 0.0 co powoduje zatrzymanie bądź spowolnienie do minimalnej prędkości wentylatorów (w zależności od konstrukcji tychże, część z nich jest zabezpieczona przed zatrzymaniem przy zerowym wypełnieniu z uwagi na ich krytyczną rolę w chłodzeniu komputera). Wypełnienie obliczane jest ze wzoru (temperaturaZmierzona-TemperaturaZdana)*2 przy czym najmniejsza wartość to 0.2, a największa to 0.8 (w celu uniknięcia rozchlapywania wody przez zbyt silny strumień powietrza). Woda jest chłodzona aż jej temperatura spadnie o pół stopnia poniżej temperatury zdanej. Chłodzenie zostaje wznowione po przekroczeniu temperatury zadanej. Takie rozwiązanie zostało przyjęte w celu wprowadzenia histerezy zapobiegającej ciągłemu włączaniu i wyłączaniu urządzenia przy małych zmianach temperatury w okolicach temperatury zdanej, wynikających m.in. z błędów pomiaru.

```
void buttonInterrupt(){
  interrupt = 1;
}
```

Funkcja ta jest wywoływana zewnętrznym przerwaniem poprzez wciśnięcie przycisku i jej jedynym zadaniem jest ustawienie flagi, która pozwoli na wykonanie właściwej funkcji (*tempChec()*, *tempCheckManual()* i *pwmSet()*). Takie rozwiązanie zostało przyjęte z uwagi na brak możliwości wywołania funkcji *delay()* oraz *millis()* w przerwaniu.

4. Specyfikacja zewnętrzna urządzenia

4.1. Opis funkcji elementów sterujących urządzeniem

- Czujnik temperatury – pozwala na sprawdzenie temperatury w akwarium przez mikrokontroler
- Czujnik poziomu cieczy – pozwala na pomiar poziomu wody w akwarium przez mikrokontroler
- Mikrokontroler – odpowiada za dokonywanie pomiarów oraz sterowanie wyświetlaczem i wentylatorami

4.2. Opis funkcji elementów wykonawczych

- Wyświetlacz LCD – wyświetla zmierzoną oraz zadaną temperaturę

4.3. Opis reakcji oprogramowania na zdarzenia zewnętrzne

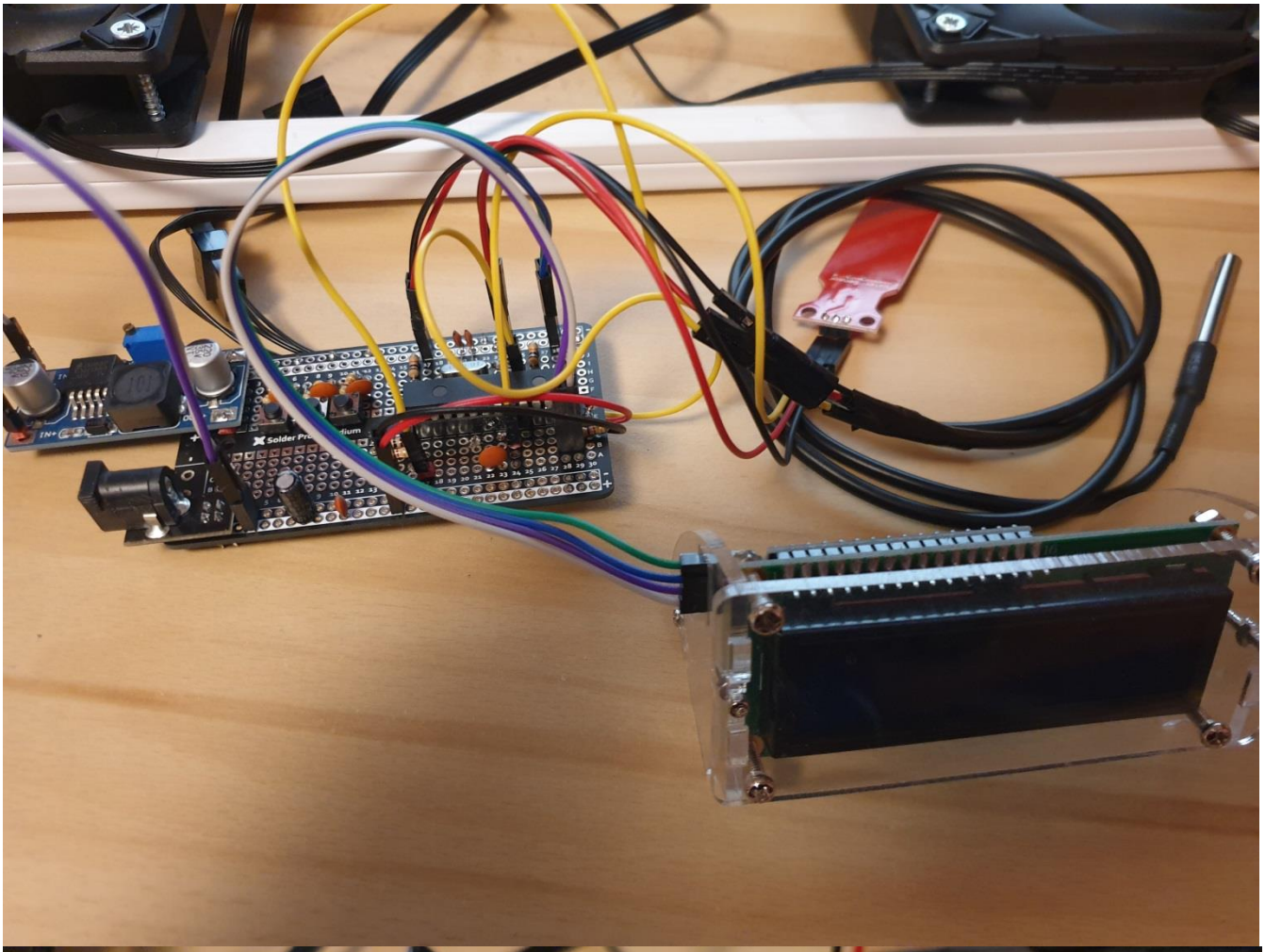
Możliwe zdarzenia zewnętrzne obejmują: wysoki lub niski poziom wody oraz temperatura wody większa bądź mniejsza od temperatury zadanej. Urządzenie chłodzi wodę tylko gdy temperatura jest większa od zadanej, a poziom wody jest wystarczający. Gdy jeden z tych warunków nie jest spełniony następuje zatrzymanie bądź wyhamowanie do prędkości minimalnej wentylatorów (w zależności od modelu wentylatora).

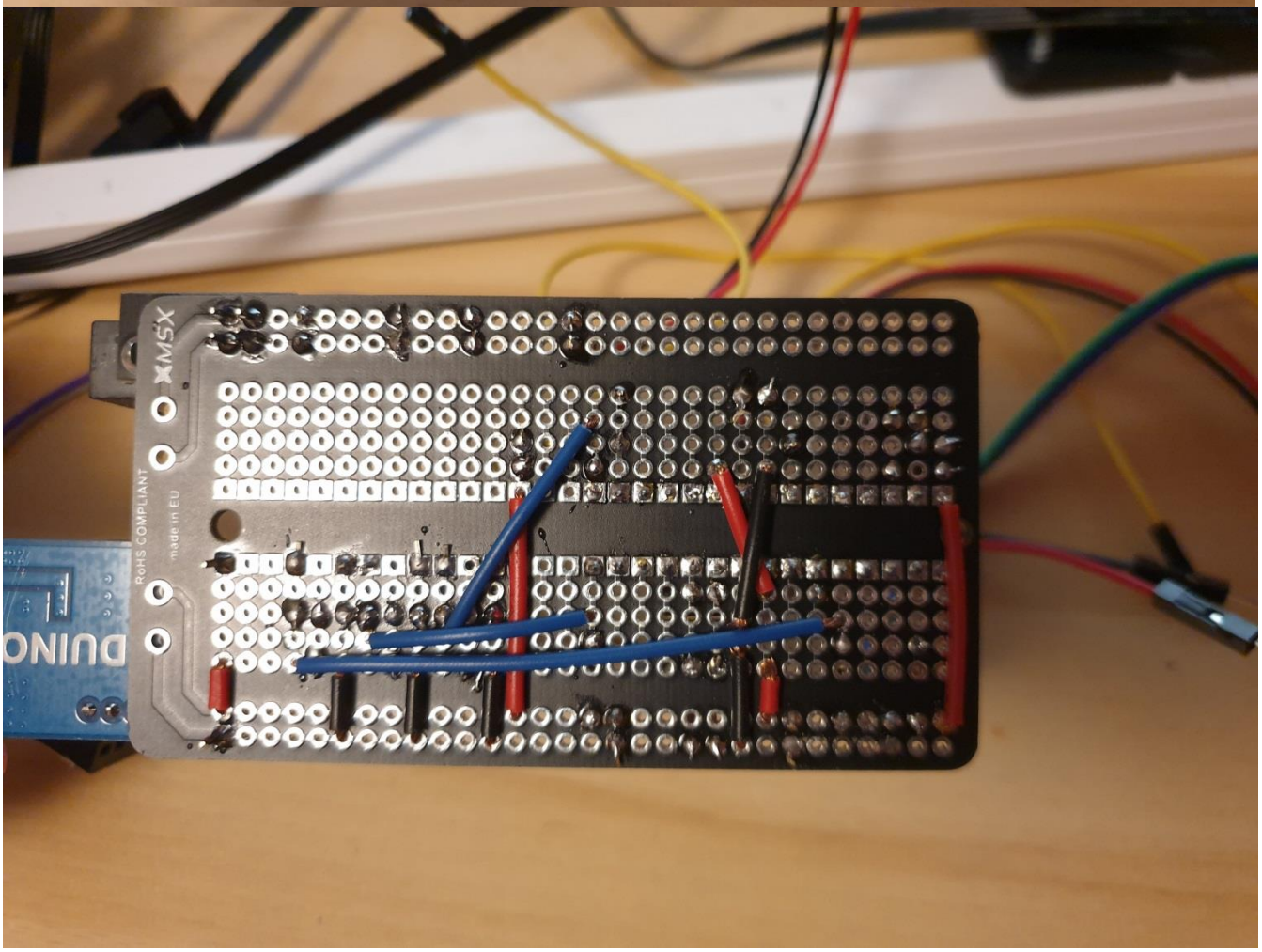
4.4. Instrukcja obsługi urządzenia

Po podłączeniu do prądu urządzenie zaczyna pracę, a użytkownik może w trakcie 5 sekundowego okna czasowego sprawdzić zmierzoną i zadaną temperaturę oraz zacząć zmieniać zmienić tą drugą. Aby sprawdzić aktualną temperaturę należy wcisnąć przycisk, który znajduje się najbliżej złącza zasilacza. Spowoduje to wyświetlenie aktualnej oraz zadanej temperatury. Aby zmienić zadaną temperaturę należy użyć pozostałych dwóch przycisków po wciśnięciu tego najbliżej złącza zasilacza. Środkowy powoduje inkrementację zadanej temperatury o $0{,}2^{o}C$, a ten znajdujący się najdalej od złącza zasilacza ją dekrementuje.

4.5. Opis złączy i/lub schematu okablowania

Układ został zmontowany na płytce uniwersalnej 420 pól. Bezpośrednio na niej znajdują się: mikrokontroler, przetwornica, przyciski, złącze DC, kondensatory, rezystory oraz złącza elementów zewnętrznych (wentylatory, wyświetlacz, termometr, czujnik poziomu cieczy). Połączenia zostały wykonanie zgodnie ze schematem ideowym przedstawionym we wcześniejszej części raportu.




MSX
RoHS COMPLIANT
made in EU

4.6. Opis montażu układu

Urządzenie zostało samodzielnie zlutowane na uniwersalnej płytce prototypowej. Elementy zewnętrzne (wyświetlacz, termometr, czujnik poziomu cieczy, wentylatory) są podłączone do przylutowanych goldpinów by umożliwić ich bezproblemową wymianę w razie awarii. Mikrokontroler znajduje się na przylutowanej podstawce z tego samego powodu.

4.7. Opis sposobu programowania układu

Kod programu był pisany w programie ArduinoIDE, następnie wgrywany na mikrokontroler zamontowany w Arduino Uno, który był przekładany do urządzenia docelowego wyposażonego w podstawkę, co umożliwiało łatwą podmianę mikrokontrolerów.

4.8. Opis sposobu uruchamiania oraz testowania

Urządzenie było pierwotnie uruchamiane oraz testowane na etapie tworzenia urządzenia na breadboardzie. Po zlutowaniu urządzenia było ono testowane w okresie czerwiec-lipiec na akwarium.

5. Wnioski i uwagi z przebiegu pracy ze szczególnym uwzględnieniem:

5.1. Jakie problemy wystąpiły podczas montażu i uruchamiania i jak zostały rozwiązane

W trakcie realizacji projektu niemożliwym okazało się na zaprogramowanie mikrokontrolera za pomocą zewnętrznego programatora USBasp. Zdecydowano się na zakup Arduino i programowanie mikrokontrolera podmieniając go z fabrycznym mikrokontrolerem Arduino, a następnie umieszczenie już zaprogramowanego kontrolera w chłodnicy. Drugim problemem okazało się przypadkowe spalenie termometru poprzez podłączenie zasilania do masy, a masy do zasilania, konieczna okazała się wymiana.

5.2. Jakie przeprowadzono testy poprawności działania urządzenia

Przetestowano jak urządzenie reaguje na zmiany temperatury i poziomu cieczy, sprawdzono poprawność działania przycisków oraz prawidłowe wykonywanie się pomiarów w odstępach 60 sekund.

6. Wnioski końcowe

Najwięcej czasu pochłonęły próby programowania urządzenia przez programator USBasp. Lutowanie elementów wymagało pewnej wprawy przez co pierwsze luty nie są tak dobrze wykonane jak te ostatnie, jednakże testy przeprowadzone multimetrem nie wykazały problemów w przewodzeniu. Projekt okazał się bardzo rozwijający, zarówno w kontekście wiedzy z elektroniki, oprogramowania (ArduinoIDE

7. Literatura

https://fdossena.com/?p=ArduinoFanControl/i.md
https://botland.com.pl/content/134-arduino-w-polaczeniu-z-czujnikiem-temperatury-ds18b20
https://forbot.pl/blog/kursy
http://www.learningaboutelectronics.com/Articles/Arduino-liquid-level-sensor-circuit.php
https://www.microchip.com/en-us/product/atmega328