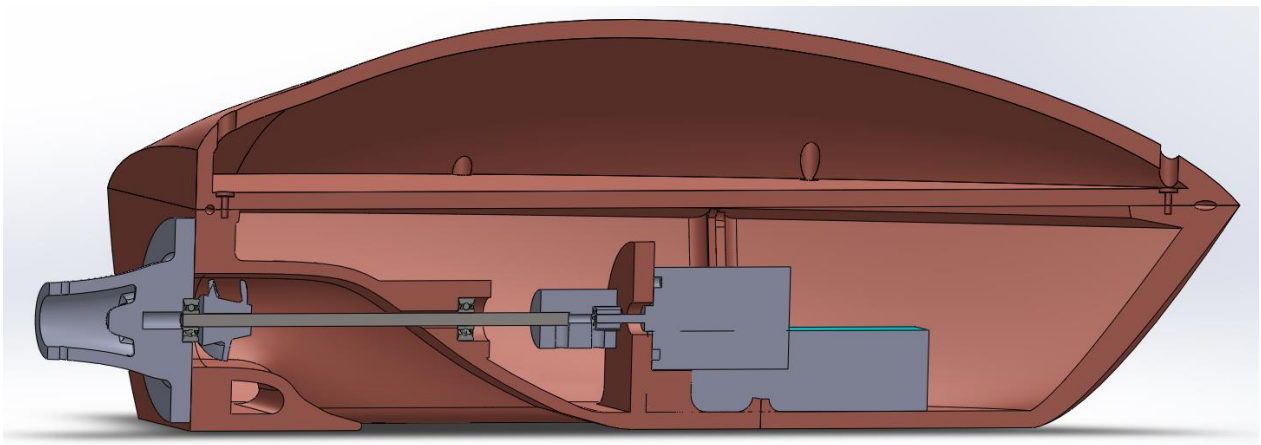
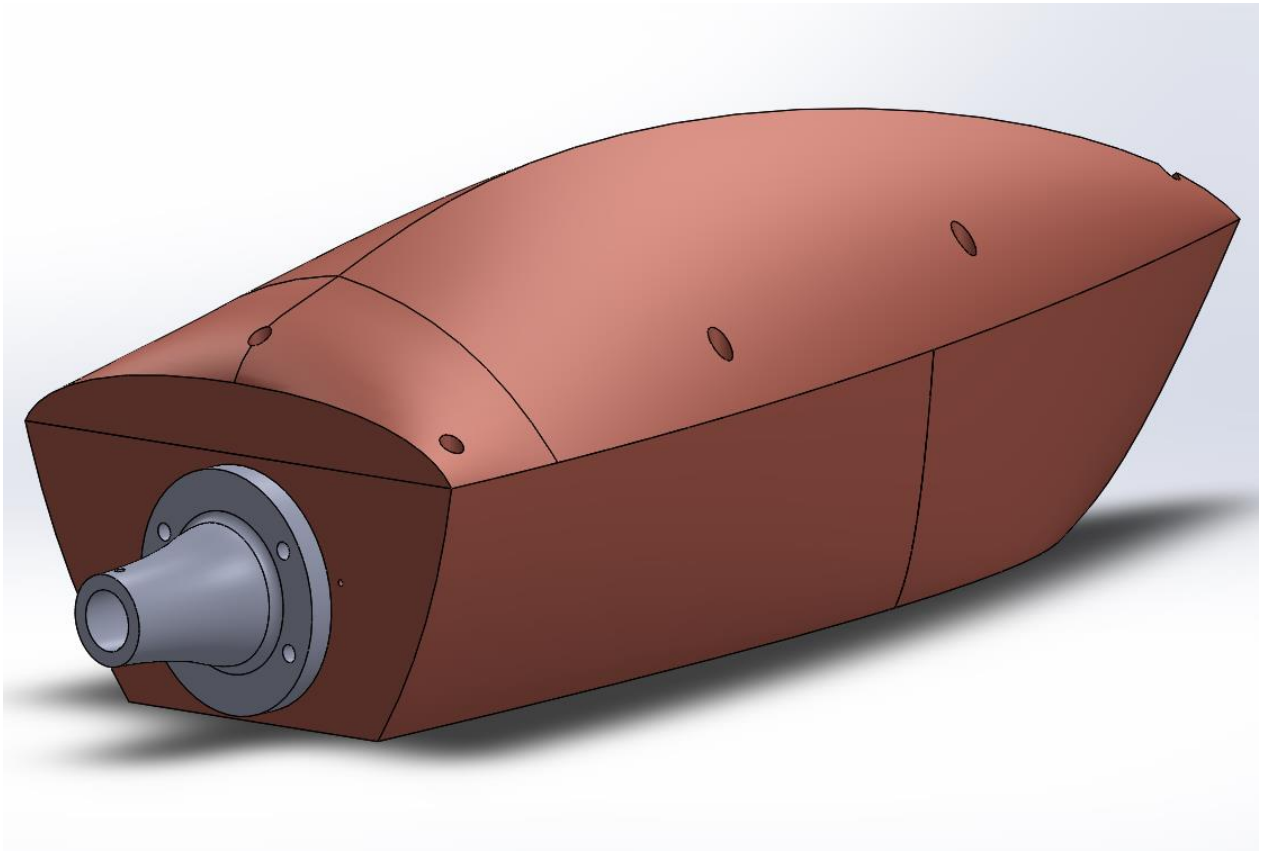
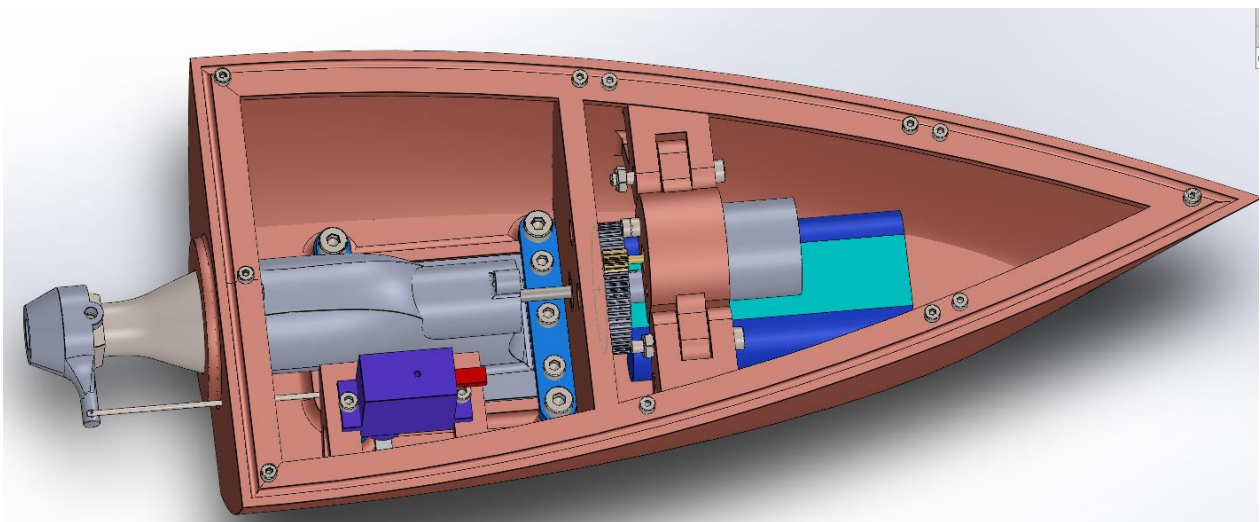
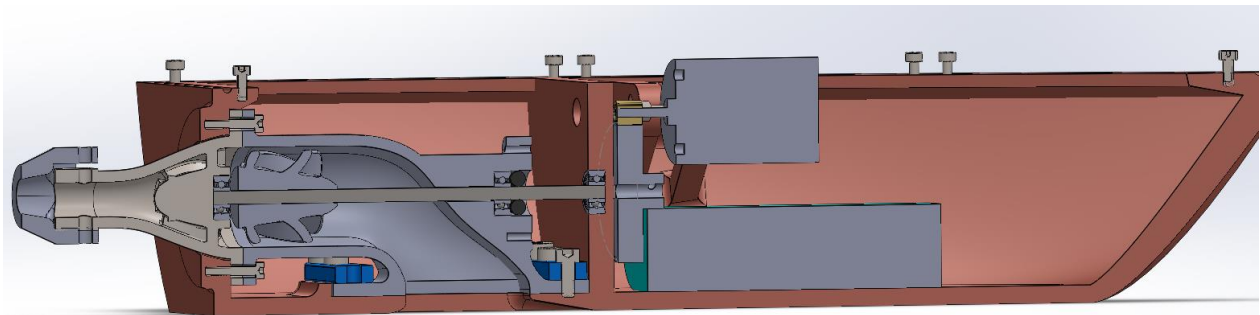
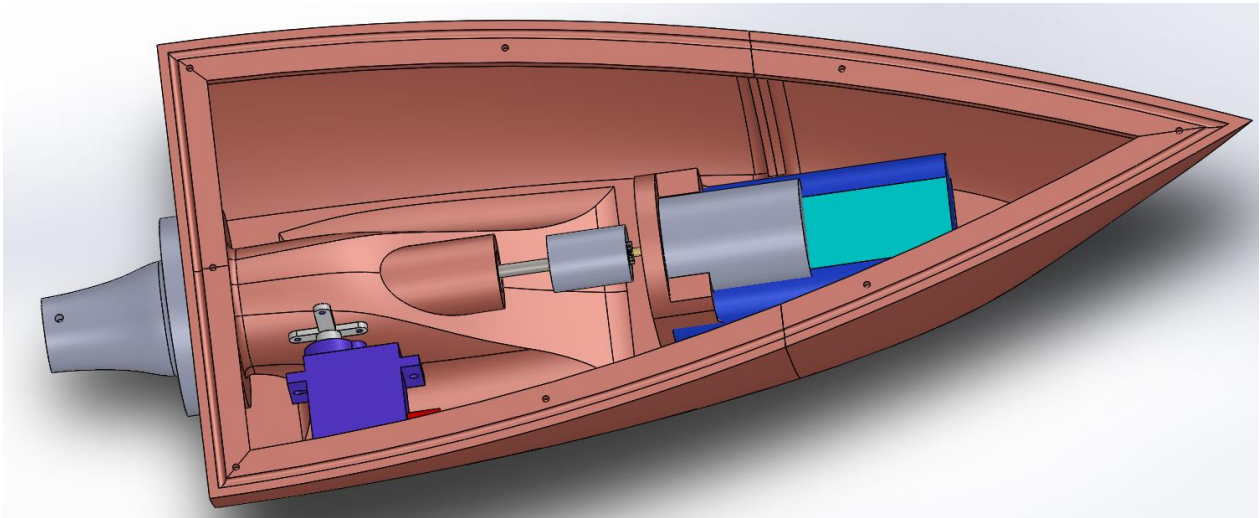


3D Printed Water Jet Boat





1. Opis ogólny projektu:

Celem projektu było skonstruowanie prototypu zdalnie sterowanej łodzi tzw. speed boat o napędzie turbinowym. Napęd ten jest zdecydowanie mniej popularny niż zwykły napęd śrubowy, jednak oferuje większą wydajność, większe przyspieszenie i lepszą manewrowość.

Największą wadą napędu turbinowego jest jego skomplikowana konstrukcja, oraz uzyskanie specyficznych warunków hydrodynamicznych. Przez konieczność wycięcia odpowiedniego doływu wody, który znajduje się na samym dnie łodzi, należy zapewnić pełną wodoszczelność całej konstrukcji (jak wiadomo otwory w dnie łodzi nie są typowym pomysłem na stworzenie obiektów unoszących się na wodzie).

Po wielu godzinach tworzenia coraz to nowych wersji modeli 3d, oraz testach w małych zbiornikach wodnych udało opracować się odpowiednią konstrukcję turbiny jak i całej łodzi, którą od tej pory należy optymalizować pod kątem wydajności i generowanej mocy.

Niestety z powodu zastosowania starego silnika szczotkowego DC ze starego modelu helikoptera RC, po wielu testach silnik ten przez wyboczony wał spowodował nieodwracalne szkody całego modelu i na tę chwilę nie udało się przetestować łodzi np. na jeziorze.

W następnej wersji, łódź zostanie wyposażona w dużo większy silnik BLDC, odpowiednio większy wał napędowy i łożyska, co przełoży się na zupełnie nową konstrukcję łodzi.

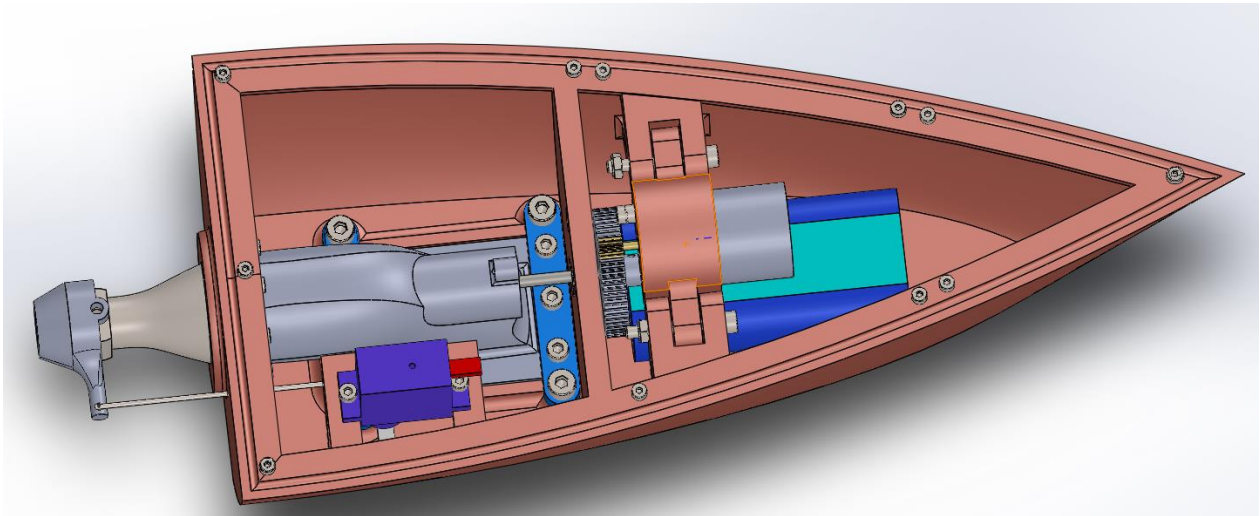
Aby opisać proces projektowania i testowania projektu, a także opisać wnioski i wskazówki co poprawić w następnych wersjach projektu, raport z projektu podzielono na 3 części odpowiadające 3 głównym dziedziną nad którymi należało się skupić podczas prac nad projektem:

1. Część mechaniczna – skupiająca się na zagadnieniach hydrodynamicznych, wyporności i zagadnieniom związanym z konstrukcją i projektowaniem okrętów wodnych, z konstrukcją zespołu odrzutowej turbiny wodnej, przeniesienia napędu i – najtrudniejszemu zagadnieniu – czyli zapewnieniu pełnej wodoszczelności łodzi.
2. Część elektroniczna – zapewniająca odpowiednie zasilanie silnika, układ mikroprocesorowy odbierający sygnały od operatora (kontroler radiowy/WiFi) i przetwarzający je na odpowiednie sterowanie mechanizmem sterującym łodzi.
3. Część programowa – zapewniająca funkcjonalność programową całego projektu, odczytywanie sygnałów elektrycznych z kontrolera, przetwarzanie i filtrowanie odczytanych wartości, wysyłanie ich zdalnie do modułu przetwarzania w łodzi, a tam sterowania odpowiednimi zadaniami w celu umożliwienia pełnej kontroli nad łodzią.

2. Część mechaniczna:

Podczas pierwszej wersji modelu założono, że zespół turbiny odrzutowej będzie wymienny tzn. przewidziano możliwość odkręcania i zmiany różnych parametrów turbiny, a następnie mocowania do dna łodzi za pomocą specjalnego systemu mocowań. System ten nie mógł korzystać z mocowań śrubowych, które będą bezpośrednio ingerować w elementy turbiny i dna łodzi, ze względu na to, że przy prędkości i ciśnieniu turbiny każdy otwór nawet gwintowany był potencjalnym miejscem przecieków wody do dna łodzi.

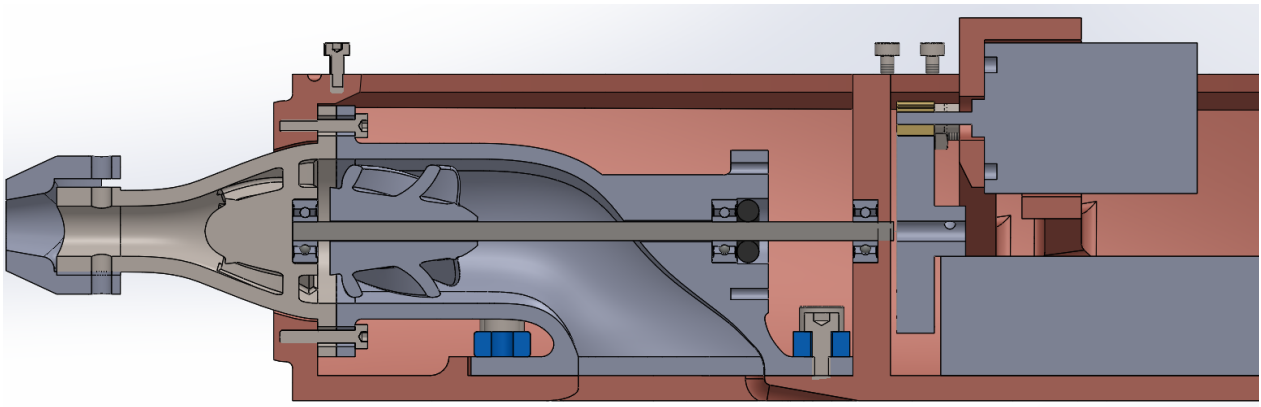
Mimo przemysłanej konstrukcji, łódź na wszelki wypadek wyposażono w specjalną grodz wodoszczelną, która miała oddzielać komorę z turbiną od komory tzw. suchej w której umieszczono silnik i całą elektronikę.



Do tego zdecydowano się użyć przekładni zębatej celem zredukowania prędkości obrotowej wału napędzającego śrubę turbiny – zbyt duża prędkość obrotu prowadzi do powstawania negatywnego zjawiska kawitacji, która zmniejsza sprawność napędu, generuje hałas i prowadzi do dużo szybszego zużycia łopatek turbiny. Przekładnia redukcyjna również pozwala uzyskać większy moment na wyjściu co umożliwia stosowanie większych turbin przy tym samym napędzie. Dzięki specjalnemu systemowi mocowania silnika istnieje możliwość regulacji pozycji wału silnika w celu wyeliminowania ewentualnych różnic w osiowaniu przekładni i wałów napędowych.

Oprócz napędu głównego, zastosowano dodatkowy napęd w postaci serwomechanizmu SG-90s, którego zadaniem jest obracanie za pomocą cięgna specjalnej tulei umieszczonej na końcu turbiny. Tuleja ta wpływa na kierunek wyrzucanej wody z turbiny, co zapewnia sterowanie kierunkiem ruchu łodzi.

Na obwodzie łodzi, w miejscu mocowania obudowy górnej przewidziano specjalne wycięcie na umieszczenie uszczelki w formie otwartego, gumowego wężyka, który można łatwo dopasować do kształtu kadłuba.



Jak widać na powyższym przekroju, wał napędowy jest podparty oraz łożyskowany w 3 miejscach. Dodatkowo, zastosowano uszczelnienie jednego z łożysk turbiny, aby uniemożliwić ewentualny wyciek wody podczas obrotu wału napędowego.

Całość modelu, przez swoje duże gabaryty, została podzielona na dwie części. Było to związane z chęcią wydruku całego kadłuba na drukarce 3D, której pole robocze było zbyt małe by wydrukować kadłub jako jedną część. To stwarzało dodatkowy problem, a mianowicie w jaki sposób połączyć ze sobą dwie części kadłuba aby utrzymać pełną wodoszczelność.

Postanowiono wykonać odpowiednie fazowanie części łączonych, aby po sklejeniu modelu, w powstałe przestrzenie wprowadzić uszczelniacz silikonowy. Obie części sklejało klejem cyjanoakrylowym.



Dodatkową rzeczą jaką należało przetestować było zagadnienie wodoszczelności materiału i techniki druku 3D całego kadłuba. Jak wiadomo druk 3D FFF/FDM polega na łączeniu ze sobą cienkich warstw ciekłego tworzywa sztucznego (filamentu). Dlatego też przez większość materiałów stosowanych do druku 3D po pewnym czasie zaczyna przeciekać woda – albo przez mikroprzestnienie na granicy warstw, albo

przez tzw. infill czyli tylko częściowe wypełnienie przestrzeni np. ściany kadłuba plastikiem, podczas gdy resztę ścianki stanowi pusta przestrzeń (powoduje to oszczędność materiału i czasu wydruku).

Jednak materiał, który zdecydowano się zastosować do wytworzenia kadłuba i innych podzespołów to PET-G, który znany jest z dobrej szczelności i doskonałej spójności warstw.

Przed wydrukowaniem całego modelu w jego rzeczywistej skali, postanowiono zacząć od wydrukowania mniejszego modelu kadłuba, żeby sprawdzić czy konstrukcja jest wodoszczelna (po 2 dniach unoszenia się na wodzie kadłub nie przeciekał) oraz czy ma odpowiedni hydrodynamiczny kształt. Przetestowano również poszczególne części pod względem technologicznym (niektóre kształty i gabaryty są trudne do wydrukowania i uzyskania dobrej dokładności na drukarce 3D).

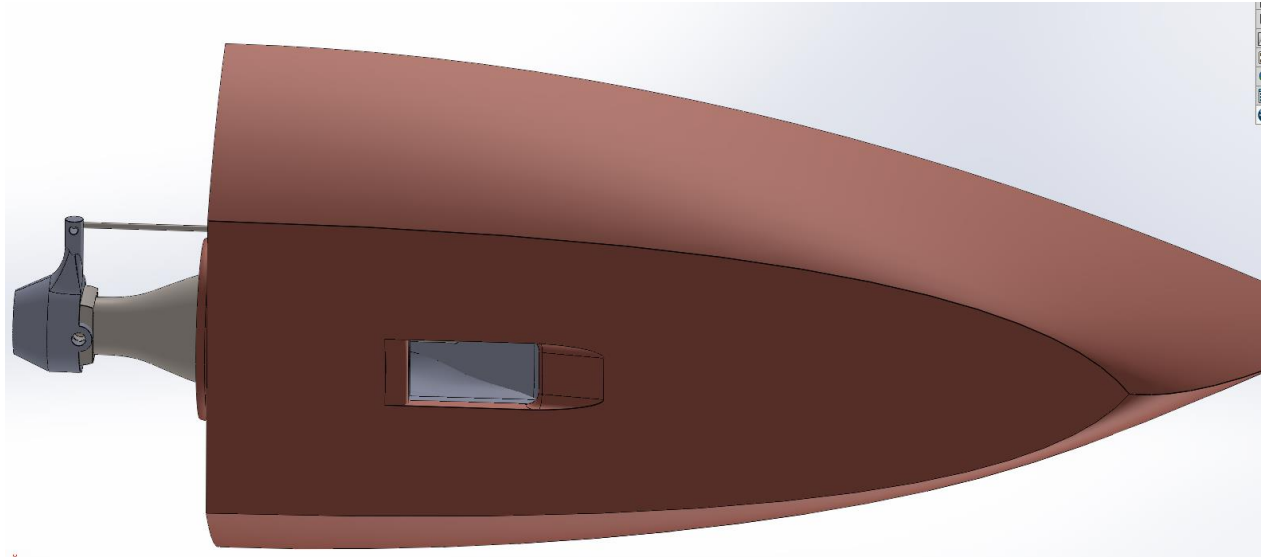


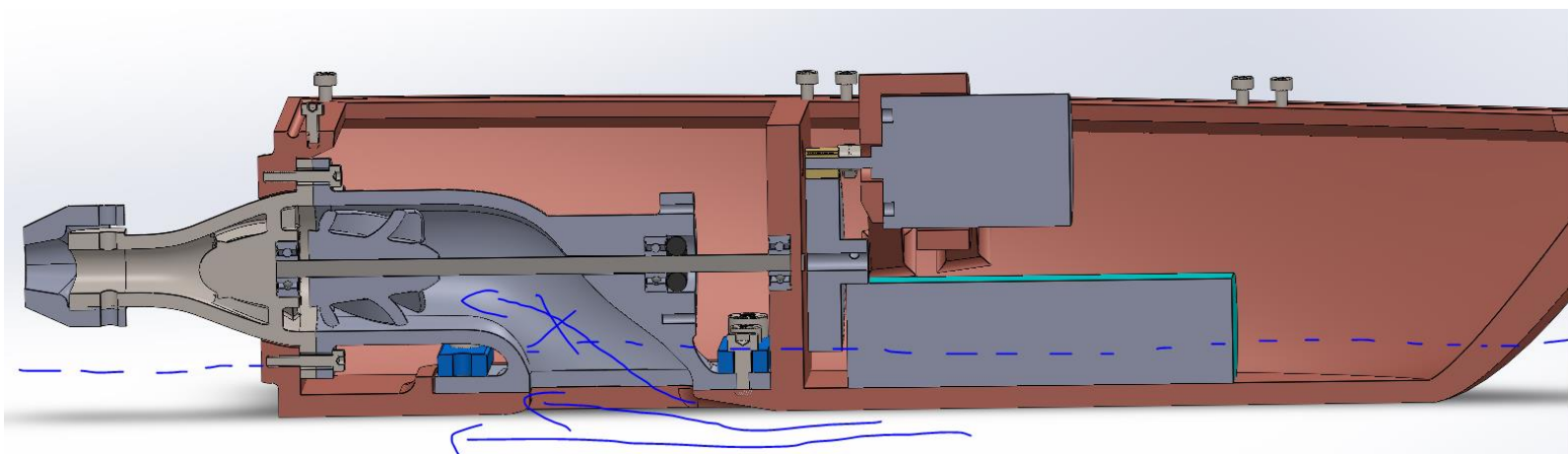


3. Wnioski z części mechanicznej – wersja 1.0:

Po wydrukowaniu całego kadłuba oraz złożeniu wszystkich części rozpoczęto testy zespołu turbiny. Okazało się, że przeniesienie napędu działa bez zarzutów, zastosowanie łożysk pozwala na ciche i płynne działanie układu, nie ma zbyt dużych oporów ruchu ze względu na tarcie.

Niestety po pierwszych testach na wodzie okazało się, że do turbiny nie dopływa odpowiednia ilość wody. Turbina jest osadzona zbyt wysoko względem poziomu lustra wody, oraz wlot turbiny jest zbyt mały i zbyt daleko od łopatek śruby napędowej.





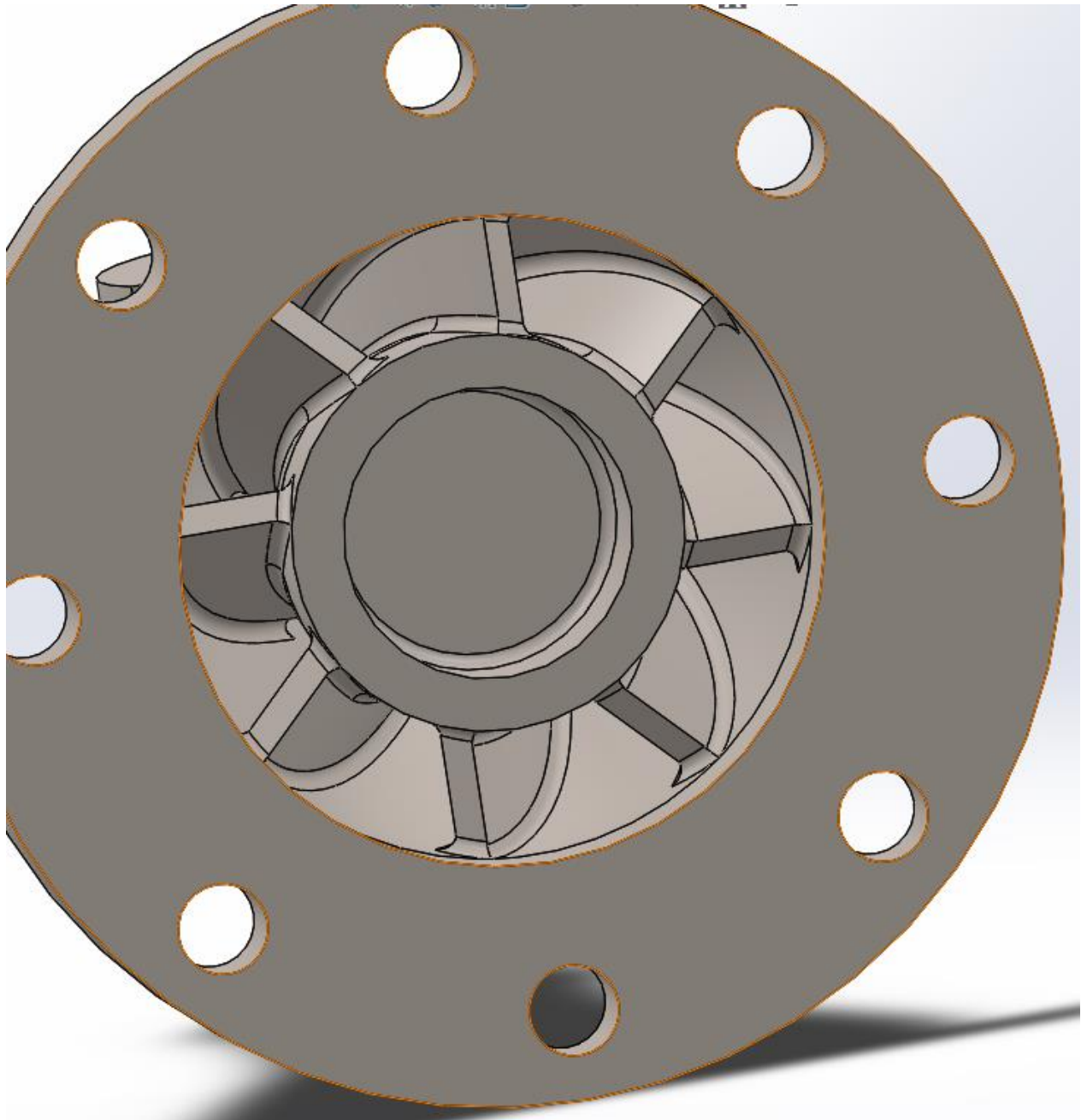
Błąd ten mógł zostać łatwo wyeliminowany poprzez zastosowanie symulacji komputerowej przepływów (CFD simulation) np. w programie Ansys. Jednak z powodu braku czasu zdecydowano o nieprzeprowadzaniu takiej analizy i od razu przejściu do testów rzeczywistych.

Dodatkowo nie sprawdzono mocowania turbiny jako osobnego elementu łodzi, oraz konieczne było stosowanie silikonu by uszczelnić drobne szczeliny między turbiną a dnem łodzi. Również mocowanie specjalnej statycznej zwężki (stator) turbiny, której zadaniem jest przyspieszenie wyrzucanej wody z turbiny, poprzez odpowiednie modyfikacje przekroju przepływu, było zbyt uciążliwe przez konieczność przykręcania 8 śrub (początkowo pomysł ten miał zapewnić dobry docisk elementu do kadłuba i uniemożliwić ewentualne nieszczelności).

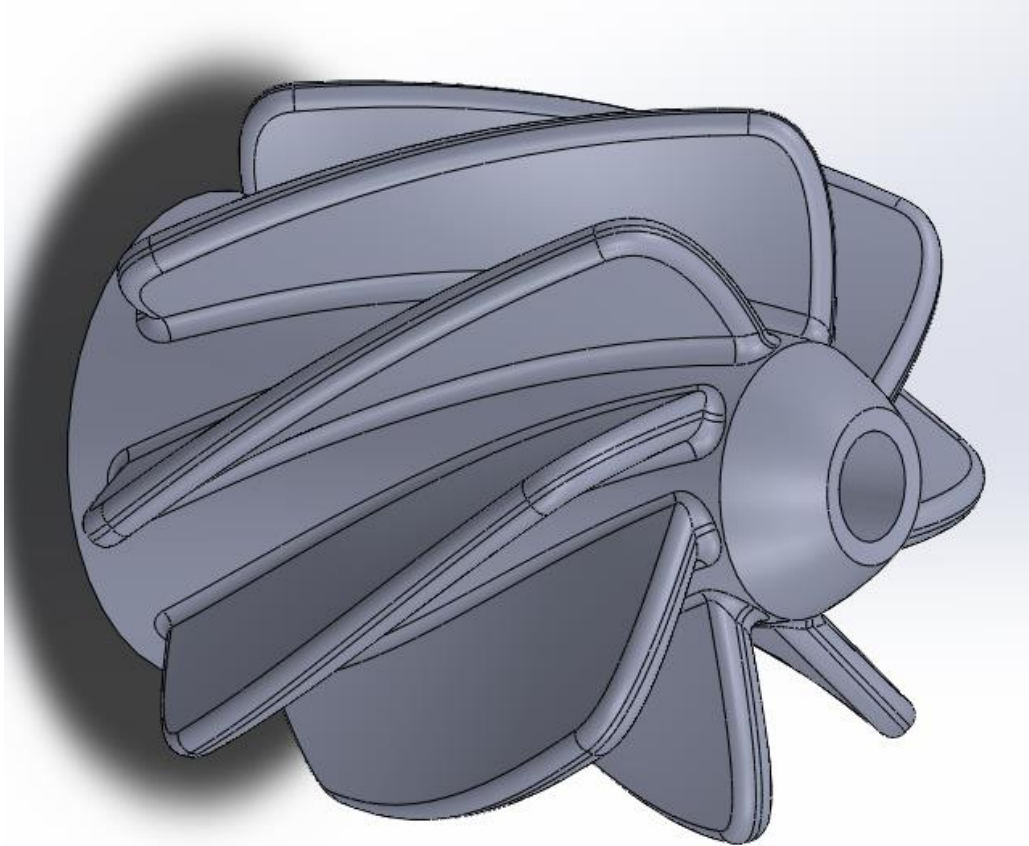
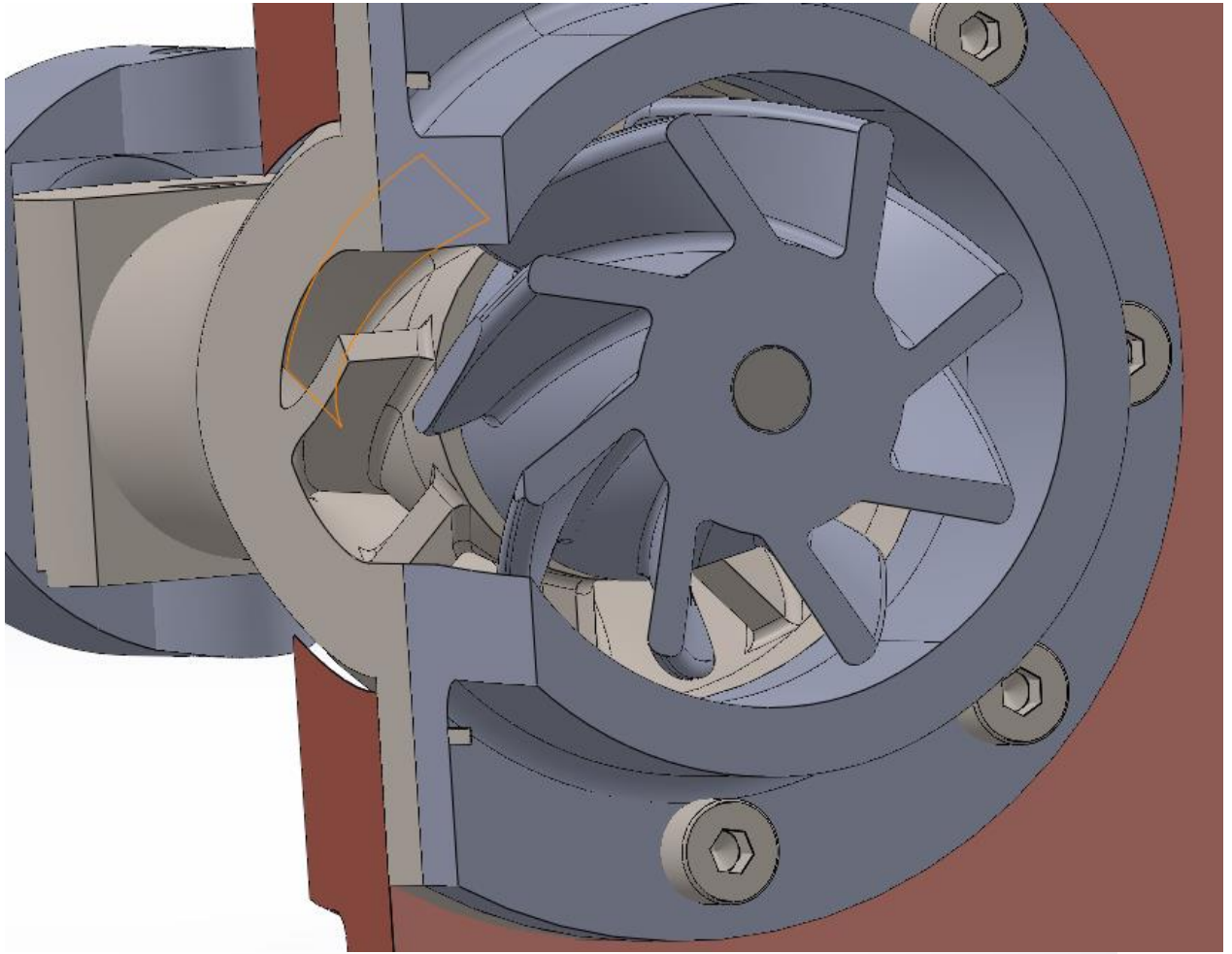
Po wielu testach uznano, że należy stworzyć całkowicie nową wersję kadłuba łodzi jak i turbiny. W nowej wersji należałoby zmodyfikować otwór w dnie łodzi, który doprowadza wodę do turbiny, zintegrować na stałe zespół turbiny z kadłubem, zrezygnować z zespołu przekładni zębatej oraz zmodyfikować element zwężki.

Warto w tym miejscu dodać ważną rolę zwężki (stator) w działaniu turbiny. Zadaniem tego elementu oprócz umiejscowienia w nim łożyska podpierające końcówkę wału, oraz przyspieszanie wypływającego z turbiny medium napędowego, jest również prostowanie/stabilizowanie przepływu. Przepływ wody za obracającymi się łopatkami śruby napędowej jest wysoce burzliwy, a co za tym idzie odrzut generowany przez taki przepływ jest mało efektywny i nie ma tak dużej energii jak czysty przepływ laminarny.

Dlatego stator wyposażony jest w specjalne łopatki nieruchome, których zadaniem jest „wyprostowanie” przepływu.

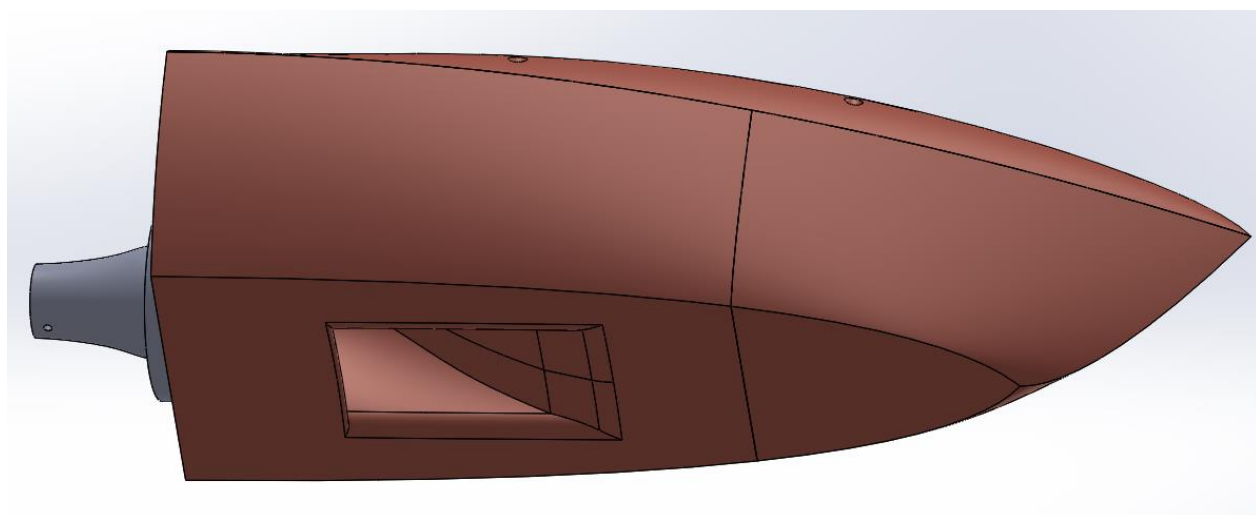
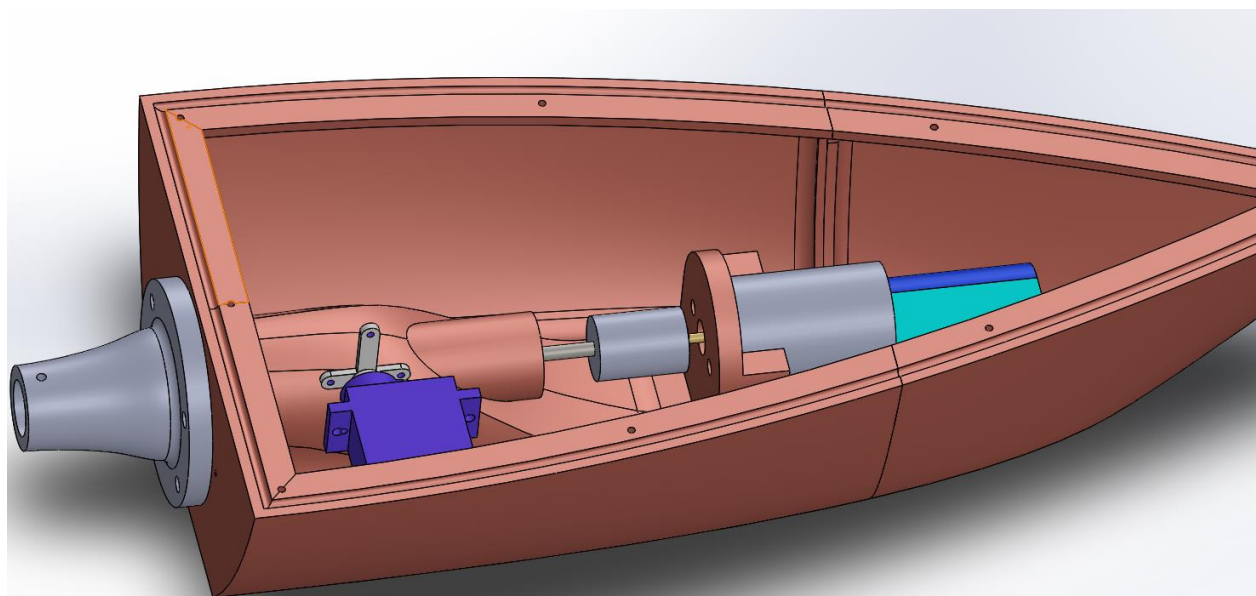
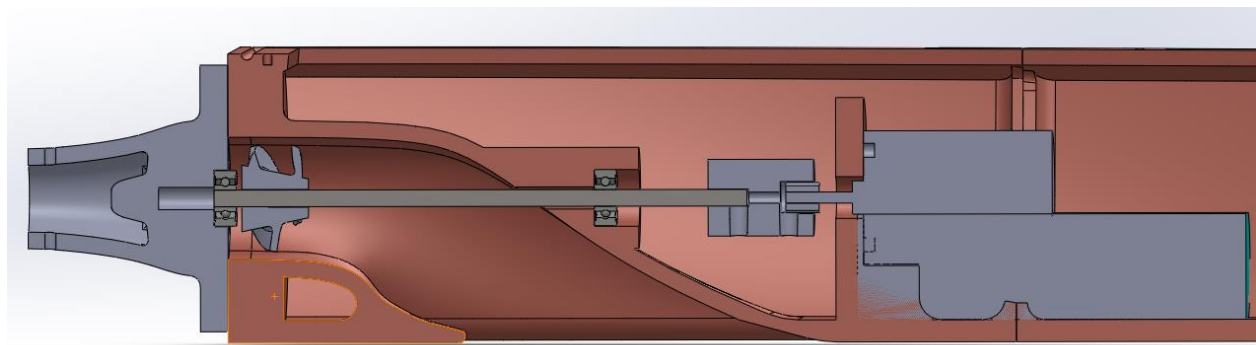


W pierwszej wersji modelu, kształt łopatek nieruchomych miał być zbliżony do kształtu łopatek śruby napędowej. W kolejnej jednak wersji konieczne będzie zmodyfikowanie tego założenia i zaprojektowanie zupełnie prostych łopatek, które jeszcze lepiej będą wprowadzać przepływ turbulentny w laminarny.



4. Część mechaniczna wersja 2.0:

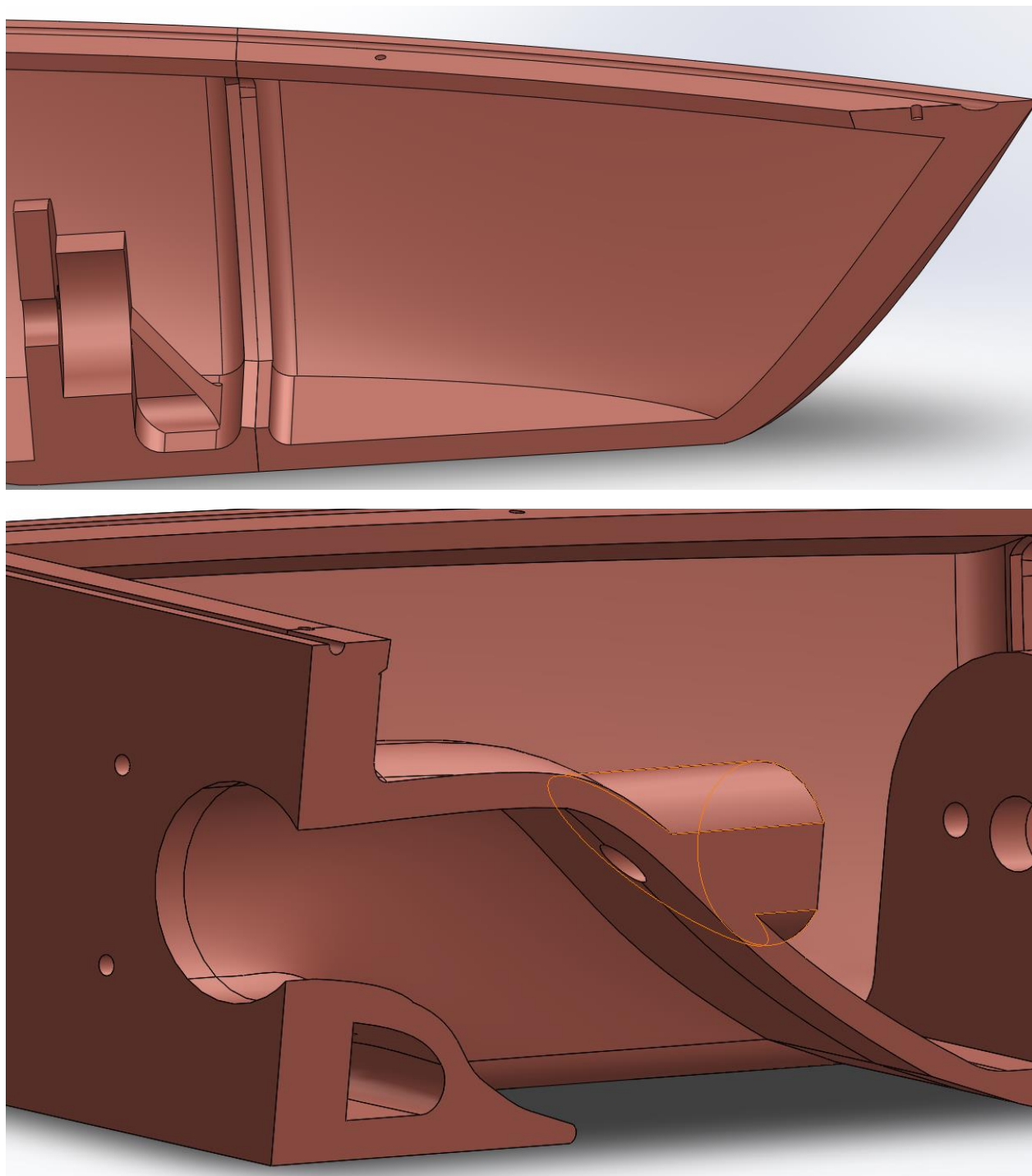
W nowej wersji poprawiono wszystkie błędy i nieudane pomysły z poprzedniej wersji, oraz zrezygnowano z kilku niepotrzebnych elementów.







Aby połączyć dwie części kadłuba tym razem nie robiono dużej fazy i nie wpuszczano silikonu do środka. Zrobiono tylko małe pogrubienia wewnątrz kadłuba w celu zwiększenia powierzchni klejenia, a całość kadłuba pokryto lakierem okrętowym. Lakier ten, oprócz tworzenia wodoodpornej bariery, pozwala również uzyskać gładką powierzchnię kadłuba, oraz zniwelować ewentualne niedoskonałości w turbinie po wydruku na drukarce 3D.





Ciekawym wyzwaniem było również zaprojektowanie odpowiedniego sprzęgła, które również miało być wydrukowane na drukarce 3D. Sprzęgło to miało zapewnić dobre przeniesienie napędu (odpowiednia sztywność) ale również pewną elastyczność mogącą skompensować ewentualną nieosiowość wału silnika i wału napędowego turbiny.



Problemem okazała się sama specyfika powstawania wyrobów za pomocą druku 3D. Poprzez łączenie ze sobą poszczególnych warstw plastycznego tworzywa, miejsca zespolenia tych warstw były najbardziej narażone na rozwarstwianie pod wpływem momentów skrętnych oraz momentów generowanych przez wkręcanie śrub w sprzęgła, które miały umocować wały napędowe w sprzęgle,

Aby rozwiązać ten problem, zaprojektowano sprzęgła, które zamiast być drukowane pionowo (oś wałów do góry) drukowane były poziomo. Dzięki temu warstwy materiału były łączone prostopadle a nie równolegle do działania sił niszczących w sprzęgle.



5. Wnioski z części mechanicznej wersja 2.0:

Dzięki poprawie wlotu (intake) turbiny wodnej, zintegrowaniu jej z kadłubem łodzi, oraz zmianie statora oraz śruby napędowej uzyskano zadziwiająco dobre rezultaty. Moc turbiny pozwala na uzyskanie dobrego odrzutu, a co za tym idzie poruszania całą łodzią. Szczelność kadłuba po wydrukowaniu na drukarce 3D też jest bez zarzutów. Wytworzenie kadłuba jak i wszystkich innych części metodą druku 3D z wykorzystaniem filamentu PET-G sprawdza się doskonale.

Niestety po wielu testach problemem nie do rozwiązania na tę chwilę okazał się silnik. Napęd został wymontowany ze starego helikoptera RC i po długim użytkowaniu wał silnika wyboczył się co spowodowało powstanie potężnych wibracji, które uszkodziły mocowanie silnika, zakleszczyły wał w turbinie i przez tak duże obciążenie, a co za tym idzie prąd pobierany przez silnik, cały napęd spalił się, a łódka została bezpowrotnie zniszczona.

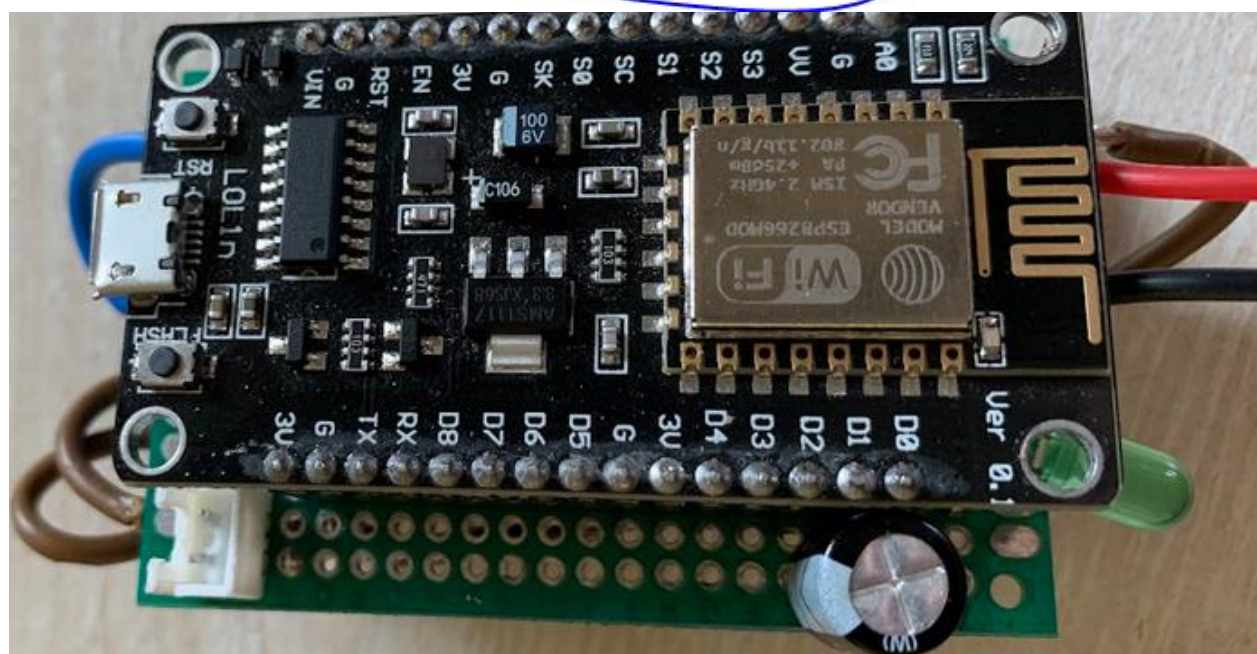


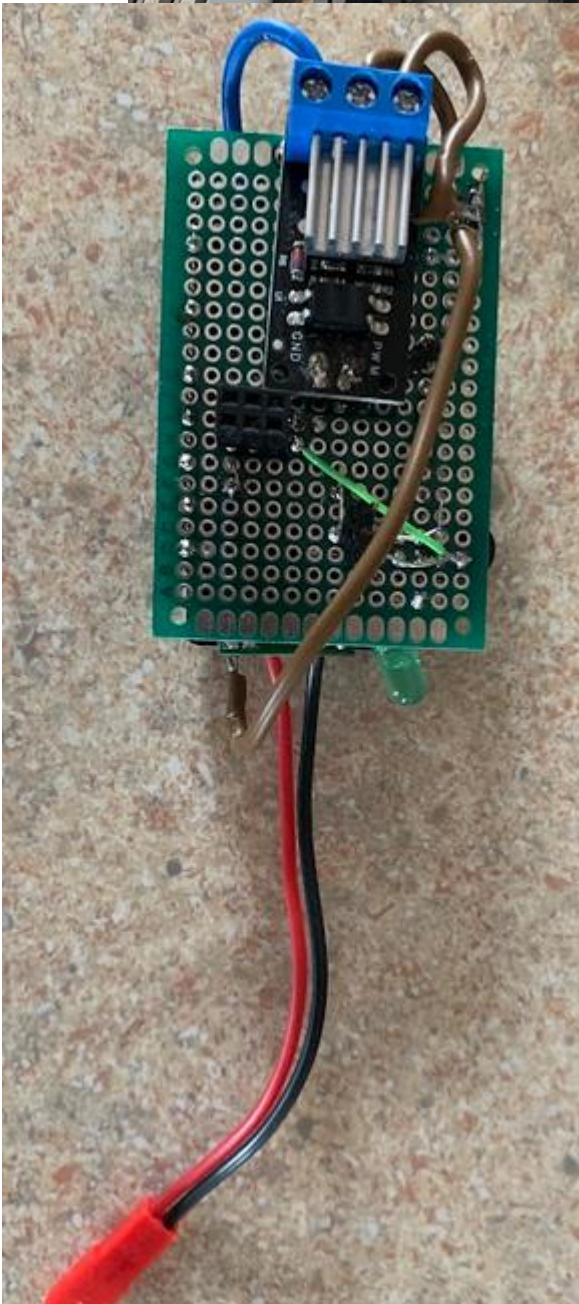
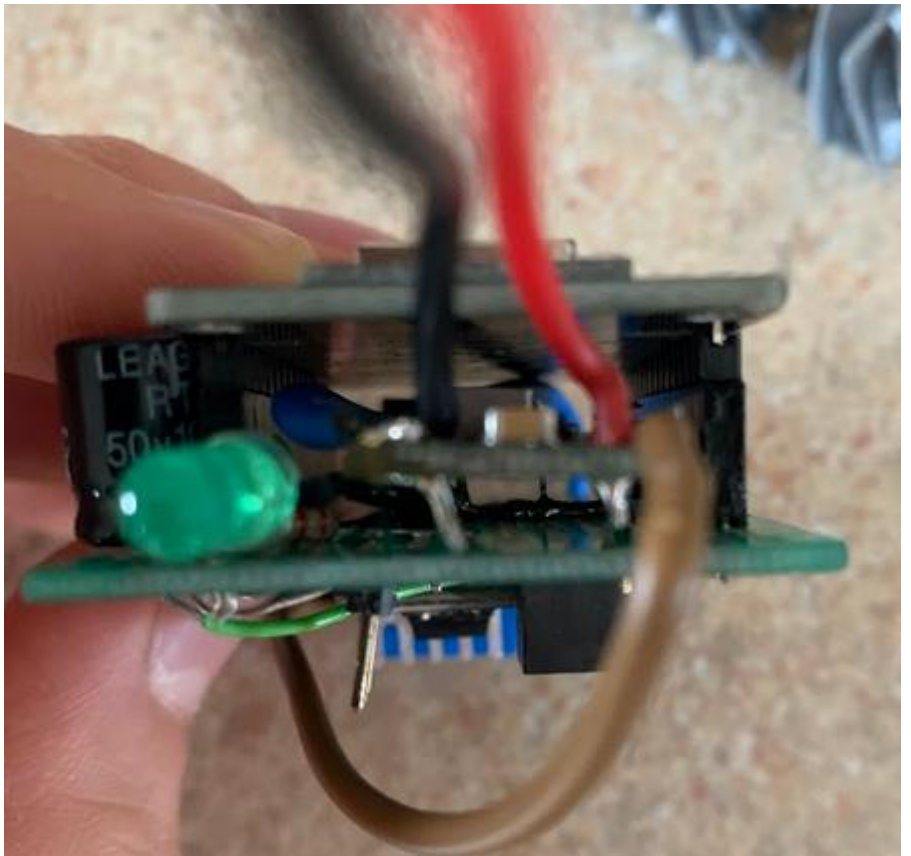
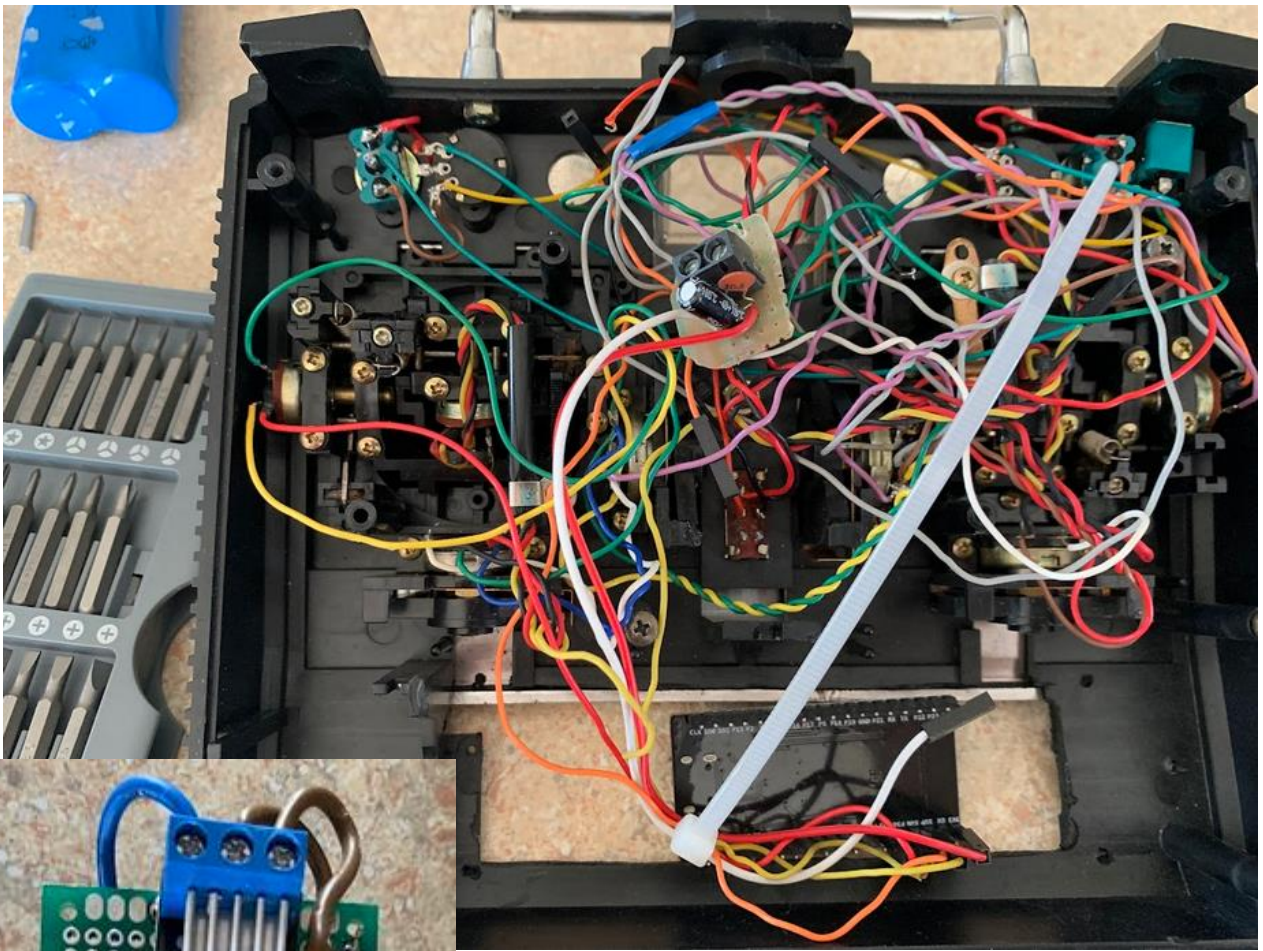
Z tego powodu należy stworzyć trzecią już wersję łodzi z nowym, dużo większym silnikiem. Silnik ten zamiast silnika DC szczotkowego, powinien być typu BLDC, dzięki czemu będzie on niewrażliwy na wodę (budowa tych silników, a zwłaszcza zastosowanie izolowanego uzwojenia uniemożliwia powstawanie zwarcia w środowisku wody). Do tego silnik taki należy wyposażyć w chłodzenie wodne oraz odpowiednio mocny tzw. ESC (electric speed contoler). Chłodzenie pozwoli na dużo efektywniejszy oraz dłuższy czas użytkowania silnika (podczas testów silnik szczotkowy był w stanie pracować tylko przez niecałą 1min zanim zaczynał się niebezpiecznie grzać).

Do mocniejszego silnika należy zastosować dużo grubszy wał napędowy, oraz większą śrubę napędową. Można również zastanowić się nad zaprojektowaniem dwu-stopniowej turbiny tzw. two-stage turbine.

Jest ona dużo bardziej efektywna przez możliwość zaciągania wody do turbiny przy niepełnym zanurzeniu co jest bardzo ważne przy użytkowaniu łodzi na wzburzonych wodach, gdy fale powodują nieregularne dostarczanie wody do turbiny.

6. Część elektroniczna:





Do sterowania zdalnego łodzią wykorzystano stary kontroler RC do helikoptera. Do każdego potencjometru przylutowano przewody: zasilające, masy oraz sygnałowe. Dzięki temu odczytywanie napięcia na potencjometrze jako wartości sygnału elektrycznego, pozwala ocenić pozycję joysticka. Odczytanie i przetworzenie danych następuje z wykorzystaniem ESP32, który następnie przesyła dane do drugiego, identycznego ESP32 znajdującego się w łodzi.

Do zasilenia silnika (oraz odprowadzenia energii z silnika, który działa jak prądnica podczas hamowania silnikiem), zasilenia ESP32 oraz serwomechanizmu i potrzebnych komponentów, stworzono wielowarstwową płytkę prototypową. Układ elektryczny został polutowany na dwuwarstwowej płycie z otworami do montażu THT.

8. Część programistyczna:

Oprogramowanie obu ESP32 zostało napisane z wykorzystaniem środowiska Arduino IDE i dodatkowych bibliotek dedykowanych płytkom ESP32 NodeMCU.

Kod ESP32 znajdującego się w kontrolerze sterowanym przez operatora ma za zadanie:

- odczytywać dane z potencjometrów
- filtrować te dane za pomocą filtrów medianowych
- inicjalizować komunikację bezprzewodową jako główny HOST. Mimo że układ ma możliwość tworzenia hotspota WiFi oraz komunikacji przez WiFi z innymi urządzeniami, nie zdecydowano się na wykorzystanie zwykłej komunikacji WiFi. Korzystając z tego samego chipu 2.4GHz, którym generować można zwykły sygnał WiFi 2.4GHz, dane przesyłane są zmodyfikowanym protokołem za pomocą specjalnej biblioteki ESP-NOW. Działa ona dużo szybciej niż zwykła komunikacja między hostami Wifi, oraz nie obciąża tak mocno procesora ESP32.
- tworzyć paczkę danych (struct) zawierającą wszystkie dane, które chcemy przesłać do esp znajdującego się w łodzi
- wysyłać paczkę danych

Kod ESP32 znajdującego się w łodzi ma za zadanie:

- podłączyć się zdalnie do głównego esp za pomocą komunikacji ESP-NOW
- odebrać paczkę danych oraz ją przetworzyć
- na podstawie otrzymanych danych odpowiednioysterować silnik i serwomechanizm aby zapewnić bezproblemową kontrolę nad łodzią

Poniżej znajduje się wciąż modyfikowany program:

ESP32 w kontrolerze:

```
#include <esp_now.h>
```

```
#include <WiFi.h>
```

```
// Variables for test data
```

```
const int Pot1X = 36;
```

```
const int Pot1Y = 39;
```

```
const int Pot2X = 34;
```

```
const int Pot2Y = 35;
```

```
int int_value;
```

```
// MAC Address of responder - edit as required
```

```
uint8_t broadcastAddress[] = {0xA4, 0xCF, 0x12, 0x45, 0x5B, 0xF8};
```

```
// Define a data structure
```

```
typedef struct struct_message {
```

```
    byte median1;
```

```
    byte median2;
```

```
    byte median3;
```

```
    byte median4;
```

```
    int b;
```

```
} struct_message;
```

```
// Create a structured object
```

```
struct_message myData;
```

```
// Peer info
```

```
esp_now_peer_info_t peerInfo;
```

```

// Callback function called when data is sent
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
    Serial.print("\r\nLast Packet Send Status:\t");
    Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
}

void setup() {

    // Set up Serial Monitor
    Serial.begin(115200);

    // Set ESP32 as a Wi-Fi Station
    WiFi.mode(WIFI_STA);

    // Initilize ESP-NOW
    if (esp_now_init() != ESP_OK) {
        Serial.println("Error initializing ESP-NOW");
        return;
    }

    // Register the send callback
    esp_now_register_send_cb(OnDataSent);

    // Register peer
    memcpy(peerInfo.peer_addr, broadcastAddress, 6);
    peerInfo.channel = 0;
    peerInfo.encrypt = false;

```

```

// Add peer
if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
}
}

void loop() {

    // Generate a random integer
    int_value = random(1,20);

    // Format structured data
    myData.b = int_value;
    readAllADCs(myData.median1, myData.median2, myData.median3, myData.median4);

    // Send message via ESP-NOW
    esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData, sizeof(myData));

    if (result == ESP_OK) {
        Serial.println("Sending confirmed");
    }
    else {
        Serial.println("Sending error");
    }
}

```

```
//-----Functions-----  
-----
```

```
// Function to calculate the median of 10 ADC readings
```

```
int getMedian(int pin) {
```

```
    int adcValues[11];
```

```
    unsigned long previousMillis = 0; // Variable to store the previous time
```

```
    const unsigned long interval = 5; // Interval between ADC readings in milliseconds
```

```
    // Read the ADC values and store them in an array
```

```
    for (int i = 0; i < 11; i++) {
```

```
        // Wait for the specified interval to elapse
```

```
        while (millis() - previousMillis < interval) {
```

```
            // Do nothing
```

```
        }
```

```
        // Record the current time as the new previous time
```

```
        previousMillis = millis();
```

```
        // Read the ADC value and store it in the array
```

```
        adcValues[i] = analogRead(pin);
```

```
    }
```

```
    // Calculate the median of the ADC values
```

```
    for (int i = 0; i < 11; i++) {
```

```
        int minVal = adcValues[i];
```

```
        int minIndex = i;
```

```
        for (int j = i+1; j < 11; j++) {
```

```

    if (adcValues[j] < minVal) {
        minVal = adcValues[j];
        minIndex = j;
    }
}

int temp = adcValues[i];
adcValues[i] = adcValues[minIndex];
adcValues[minIndex] = temp;
}

```

```

int median = adcValues[6];

```

```

return median;

```

```

}

```

// Function to calculate the medians of all ADC inputs

```

void readAllADCs(byte &median1, byte &median2, byte &median3, byte &median4) {
    median1 = map(getMedian(Pot1X), 1390, 2385, 0, 255);
    median2 = map(getMedian(Pot1Y), 1530, 2265, 0, 255);
    median3 = map(getMedian(Pot2X), 1390, 2430, 0, 255);
    median4 = map(getMedian(Pot2Y), 1390, 2410, 0, 255);
}

```

Kod ESP32 w łodzi:

```
#include <esp_now.h>
```

```
#include <WiFi.h>
```

```
#include <ESP32Servo.h>
```

```
#include <Arduino.h>
```

```
#include <analogWrite.h>
```

```
//-----Motors Pins-----
```

```
int R_EN_1 = 26;
```

```
int R_PWM_1 = 32;
```

```
int L_EN_1 = 25;
```

```
int L_PWM_1 = 33;
```

```
int R_EN_2 = 19;
```

```
int R_PWM_2 = 14;
```

```
int L_EN_2 = 18;
```

```
int L_PWM_2 = 27;
```

```
//-----PWM setup
```

```
int PWM_FREQUENCY = 10000;
```

```
int PWM_CHANNEL_R1 = 0; //pwm channel to control movement on X-axis
```

```
int PWM_CHANNEL_L1 = 1; //pwm channel to control movement on X-axis
```

```
int PWM_CHANNEL_R2 = 2; //pwm channel to control movement on X-axis
```

```
int PWM_CHANNEL_L2 = 3; //pwm channel to control movement on X-axis
```

```
int PWM_RESOLUTION = 8;
```

```
// int GPIOPIN1 = 27; LED for testing
```

```
// int GPIOPIN2 = 14; LED for testing
```

```
Servo myservo;
```

```
// Define a data structure
```

```
typedef struct struct_message {
```

```
    byte median1; //Pot1X
```

```
    byte median2; //Pot1Y
```

```
    byte median3; //Pot2X
```

```
    byte median4; //Pot2Y
```

```
    int b;
```

```
} struct_message;
```

```
// Create a structured object
```

```
struct_message myData;
```

```
// Callback function executed when data is received
```

```
void OnDataRecv(const uint8_t * mac_addr, const uint8_t *incomingData, int len) {
```

```
    memcpy(&myData, incomingData, sizeof(myData));
```

```
// Code for analyzing data received from esp32 in rc controller
```

```
// Serial.print("Pot1X: ");
```

```
// Serial.println(myData.median1);
```

```
// Serial.print("Pot1Y ");
```

```
// Serial.println(myData.median2);
```

```

// Serial.print("Pot2X: ");
// Serial.println(myData.median3);
// Serial.print("Pot2Y: ");
// Serial.println(myData.median4);
}

void setup() {
  // Set up Serial Monitor
  Serial.begin(115200);

  // Set ESP32 as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  // Initilize ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Register callback function
  esp_now_register_recv_cb(OnDataRecv);

  myservo.attach(32); // attach the servo to pin 32

  //-----Motors setup-----

  pinMode(R_EN_1, OUTPUT);
  pinMode(R_PWM_1, OUTPUT);

```

```
pinMode(L_EN_1, OUTPUT);  
pinMode(L_PWM_1, OUTPUT);  
pinMode(R_EN_2, OUTPUT);  
pinMode(R_PWM_2, OUTPUT);  
pinMode(L_EN_2, OUTPUT);  
pinMode(L_PWM_2, OUTPUT);
```

```
digitalWrite(R_EN_1, HIGH);  
digitalWrite(L_EN_1, HIGH);  
digitalWrite(R_EN_2, HIGH);  
digitalWrite(L_EN_2, HIGH);
```

```
//-----PWM setup-----
```

```
ledcSetup(PWM_CHANNEL_R1, PWM_FREQUENCY, PWM_RESOLUTION); // channel setup for controlling  
PWM in X-axis movement
```

```
ledcSetup(PWM_CHANNEL_L1, PWM_FREQUENCY, PWM_RESOLUTION); // channel setup for controlling  
PWM in X-axis movement
```

```
ledcSetup(PWM_CHANNEL_R2, PWM_FREQUENCY, PWM_RESOLUTION); // channel setup for controlling  
PWM in X-axis movement
```

```
ledcSetup(PWM_CHANNEL_L2, PWM_FREQUENCY, PWM_RESOLUTION); // channel setup for controlling  
PWM in X-axis movement
```

```
ledcAttachPin(L_PWM_1, PWM_CHANNEL_R1);  
ledcAttachPin(R_PWM_1, PWM_CHANNEL_L1);  
ledcAttachPin(L_PWM_2, PWM_CHANNEL_R2);  
ledcAttachPin(R_PWM_2, PWM_CHANNEL_L2);
```

```
// ledcAttachPin(GPIOPIN1, PWM_CHANNEL_F); LED for testing
```

```
// ledcAttachPin(GPIOPIN2, PWM_CHANNEL_B); LED for testing
```

```
}
```

```
void loop() {
```

```
    int X_joystickValue = myData.median3; // get X joystick value
```

```
    int Y_joystickValue = myData.median4; // get Y joystick value
```

```
    int motorSpeedF = map(Y_joystickValue, 130, 255, 0, 255); // map joystick value to PWM range (motor speed)
```

```
    int motorSpeedB = map(Y_joystickValue, 120, 0, 0, 255); // map joystick value to PWM range (motor speed)
```

```
    int motorSpeedR = map(X_joystickValue, 130, 255, 0, 255); // map joystick value to PWM range (motor speed)
```

```
    int motorSpeedL = map(X_joystickValue, 120, 0, 0, 255); // map joystick value to PWM range (motor speed)
```

```
// Forward and Backward motors control loop
```

```
if (Y_joystickValue > 130) { // move Forward
```

```
    ledcWrite(PWM_CHANNEL_L1, motorSpeedF);
```

```
    ledcWrite(PWM_CHANNEL_R1, 0);
```

```
    ledcWrite(PWM_CHANNEL_R2, 0);
```

```
    Serial.println("Backward");
```

```
    Serial.println(motorSpeedF);
```

```
} else if (Y_joystickValue < 120) { // move Backward
```

```
    ledcWrite(PWM_CHANNEL_L1, 0);
```

```
    ledcWrite(PWM_CHANNEL_R1, motorSpeedB);
```

```
    ledcWrite(PWM_CHANNEL_L2, 0);
```

```
    ledcWrite(PWM_CHANNEL_R2, motorSpeedB);  
    Serial.println("Forward");  
    Serial.println(motorSpeedB);  
}
```

```
// Left and Right motors control loop
```

```
if (X_joystickValue > 130) { // move Right  
    ledcWrite(PWM_CHANNEL_L1, motorSpeedR);  
    ledcWrite(PWM_CHANNEL_R1, 0);  
    ledcWrite(PWM_CHANNEL_L2, 0);  
    ledcWrite(PWM_CHANNEL_R2, motorSpeedR);  
    Serial.println("Backward");  
    Serial.println(motorSpeedR);  
} else if (X_joystickValue < 120) { // move Left  
    ledcWrite(PWM_CHANNEL_L1, 0);  
    ledcWrite(PWM_CHANNEL_R1, motorSpeedL);  
    ledcWrite(PWM_CHANNEL_L2, motorSpeedL);  
    ledcWrite(PWM_CHANNEL_R2, 0);  
    Serial.println("Forward");  
    Serial.println(motorSpeedL);  
} else if (121<X_joystickValue<129 && 121<Y_joystickValue<129) { // stop  
    ledcWrite(PWM_CHANNEL_R1, 0);  
    ledcWrite(PWM_CHANNEL_L1, 0);  
    ledcWrite(PWM_CHANNEL_R2, 0);  
    ledcWrite(PWM_CHANNEL_L2, 0);  
    Serial.println("Stop");  
}  
delay(100);  
}
```