DAFTAR PUSTAKA

[1] H. M. Manik *et al.*, "*Autonomous Underwater Vehicle* untuk Survei dan Pemantauan Laut," *J. Rekayasa Elektr.*, vol. 13, no. 1, p. 27, 2017, doi: 10.17529/jre.v13i1.5964.

[2] Y. Yuniati, "Perancangan Sistem Autopilot untuk Kontrol Kemudi Model Kapal Mengunakan Programable Automatic Controller Ni Compactrio," *Wave J. Ilm. Teknol. Marit.*, vol. 9, no. 2, pp. 65–70, 2018, doi: 10.29122/jurnalwave.v9i2.2659.

[3] National-Instruments, "LabVIEW User Manual - Getting Started with LabVIEW," no. June, p. 349, 2013, [Online]. Available: http://www.ni.com/getting-started/labview-basics/i/.

[4] N. S. Nise, *Control systems engineering*, 7th ed. John Wiley & Sons, 2020.

[5] E. Yuliza, "Invers Tergeneralisasi Matriks atas Z p," no. 1, pp. 1–6, 2016.

[6] Rahimuddin, H. Rivai, and H. Hasan, "Interacting Propeller Thrust for Navigating Omni-Directional Remotely Operated Vehicle," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 619, no. 1, 2019, doi: 10.1088/1757-899X/619/1/012048.

[7] S. A. Rofiq, J. T. Elektro, J. T. Perkapalan, F. T. Industri, and T. Kelautan, "Perancangan Sistem Pengaturan Kestabilan *Autonomous Underwater Vehicle* (AUV) untuk Gerak Lateral Menggunakan Sliding Mode Control (SMC)," vol. 3, no. 1, pp. 1–6, 2014.

[8] T. I. Fossen, "Guidance and control of ocean vehicles," *Univ. Trondheim, Norway, Print. by John Wiley Sons, Chichester, England, ISBN 0 471 94113 1, Dr. Thesis*, 1999.

LAMPIRAN

Lampiran 1: Surat Izin Penggunaan Laboratorium Hidrodinamika

Disiapkan oleh: A.Dian Eka Anggriani, S.T,M.T	Diperiksa oleh: Dr.Eng. Suandar Baso, S.T,M.T	Disahkan oleh: Dr.Eng. Suandar Baso, S.T,M.T
LABORATORIUM HIDRODINAMIKA Jln Poros Malino KM. 14.5 Kampus Teknik Unhas Gowa, Sulawesi Selatan, 92171 https://nasp.unhas.ac.id		
FORM IZIN UNTUK PENELITIAN TUGAS AKHIR MAHASISWA		
No. Dokumen		
Edisi	Juli 2020	

Form Izin Penggunaan Laboratorium Hidrodinamika untuk Penelitian Tugas Akhir Mahasiswa Fakultas Teknik UNHAS

Dengan Hormat,
Sehubungan dengan pelaksanaan Tugas Akhir, saya yang bertanda Tangan di bawah ini:

Nama : Muh. Nursyahrul Qadri
NIM : D33115503
No. Telepon : +62 852 4279 4030
Judul Tugas Akhir : Desain Kendali Gerak Vertikal - Horizontal Kendaraan Bawah Air Semi-Autonomous
Dosen Pembimbing : Rahimuddin, S.T,M.T,Ph.D

Tanggal Penggunaan : 9 September s/d 19 September 2020

Mohon diizinkan untuk menggunakan fasilitas Laboratorium Hidrodinamika Departemen Teknik Perkapalan Fakultas Teknik Universitas Hasanuddin dengan rincian alat sebagai berikut:

No.	Nama Alat / Fasilitas	Jumlah
1.	Kolam Uji	1
2.	Cran	1

Demikian surat permohonan ini saya buat, atas perhatian dan bantuan Bapak saya ucapkan terima kasih.

Mengetahui
Dosen Pembimbing

Rahimuddin, S.T,M.T,Ph.D
NIP 197108251999031002

Gowa, 8 September 2020
Pemohon

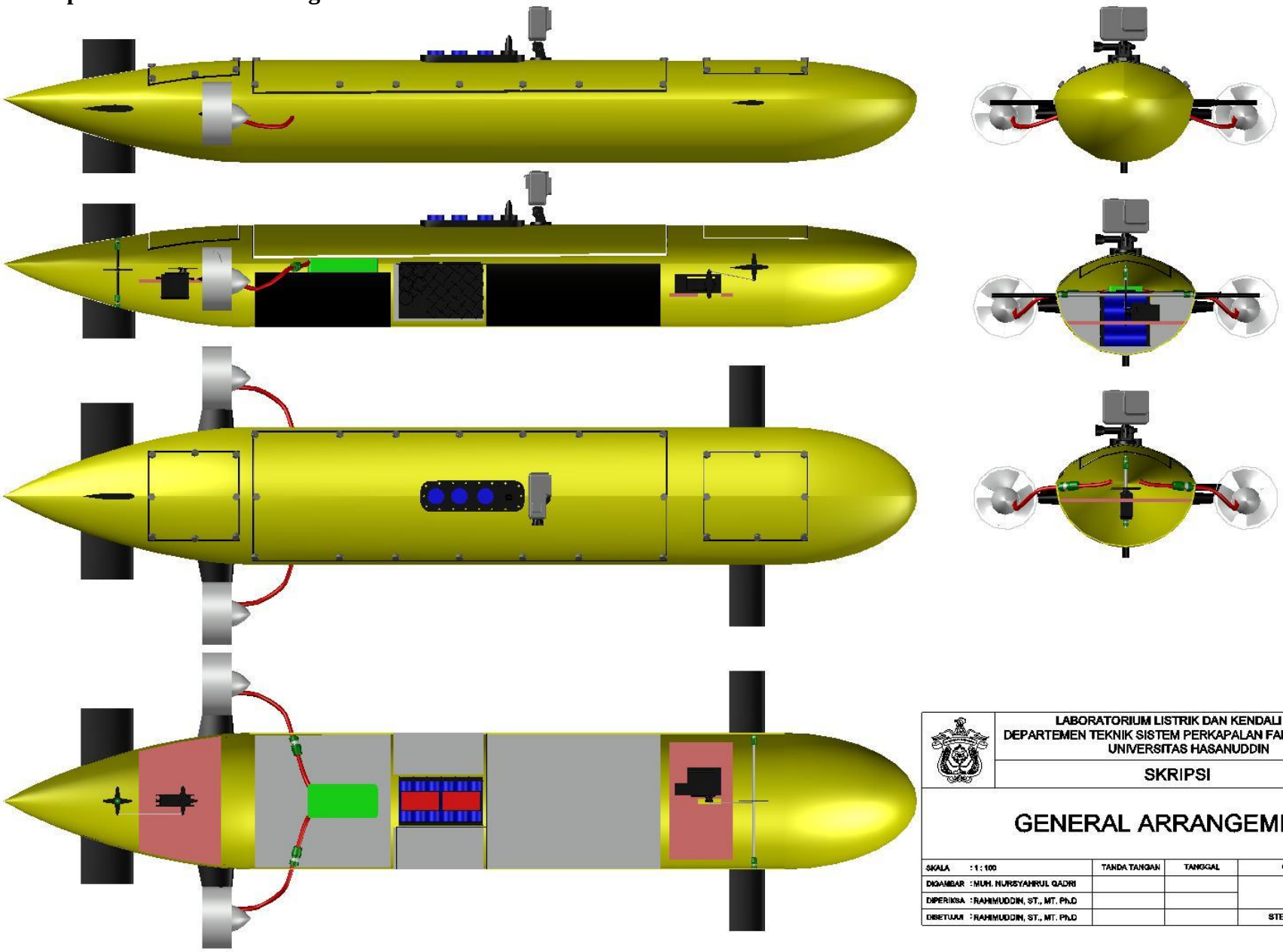
Muh. Nursyahrul Qadri
NIM D33115503

Menyetujui
Kepala Laboratorium Hidrodinamika

Dr. Eng. Suandar Baso, S.T., M.T.
NIP 197302062000121002

LABORATORIUM TERPADU FAKULTAS TEKNIK	Halaman: 1 dari 1

Lampiran 2: General Arrangement



LABORATORIUM LISTRIK DAN KENDALI KAPAL DEPARTEMEN TEKNIK SISTEM PERKAPALAN FAKULTAS TEKNIK UNIVERSITAS HASANUDDIN			
SKRIPSI			
GENERAL ARRANGEMENT			
SKALA : 1 : 100	TANDA TANGAN	TANGGAL	CATATAN
DIGAMBAR : MUH. NURSYAHRUL QADRI			
DIPERIKSA : RAHIMUDDIN, ST., MT. PhD			
DISETUJUI : RAHIMUDDIN, ST., MT. PhD			STB : D331 15 503

Lampiran 3: Contoh Perhitungan Sistem Kendali

Teori

Dua representasi alternatif dari model Davidson dan Schiff (1946) diusulkan oleh Nomoto, Taguchi, Honda dan Hirano (1957). Model-model ini diperoleh dengan menghilangkan kecepatan sway v dari (2.8) untuk mendapatkan fungsi transfer Nomoto antara r dan δ_R , yaitu: [8]

$$\frac{r}{\delta_R}(s) = \frac{K_R(1+T_3 s)}{(1+T_1 S)(1+T_2 s)}$$

Parameter fungsi transfer terkait dengan turunan hidrodinamik sebagai:

$$T_1 T_2 = \frac{\det(M)}{\det(N)}$$

$$T_1 T_2 = \frac{n_{11}m_{22}+n_{22}m_{11}-n_{12}m_{21}-n_{21}m_{12}}{\det(N)}$$

$$K_R = \frac{n_{21}b_1-n_{11}b_2}{\det(N)}$$

$$K_R T_3 = \frac{m_{21}b_1 - m_{11}b_2}{\det(N)}$$

Di mana elemen m_{ij}, n_{ij} dan b_i $(i = 1{,}2\ dan\ j = 1{,}2)$ didefinisikan dalam) menentukan inersia dan matriks redaman dihitung sebagai:

$$\det(M) = (m - Y_{\dot{v}})(I_z - N_{\dot{r}}) - (mx_G - N_{\dot{v}})(mx_G - Y_{\dot{r}})$$

$$\det(N) = Y_v(N_r - mx_G u_0) - N_v(Y_r - mu_0)$$

$$\frac{\psi}{\delta}(s) = \frac{K(1+T_3 s)}{s(1+T_1 S)(1+T_2 s)} = \frac{K(1+T_3 s)}{(1+T_1 S)(1+T_2 s)}$$

$$\frac{r}{\delta}(s) = \frac{K(1+T_3 s)}{(1+T_1 S)(1+T_2 s)} = \frac{K(1+T_3 s)}{(1+T_1 S)(1+T_2 s)}$$

Untuk permukaan kapal kita dapat memperkirakan A_{22} dan A_{66} dengan memperlakukan bagian kapal yang terendam sebagai setengah silinder dengan massa tambahan:

$$A_{22}^{(2D)} = \frac{1}{2}\rho\pi D^2(x)$$

Di mana draft lambung D(x) diambil untuk menjadi radius silinder dan ρ adalah massa jenis air. Oleh karena itu, kumpulan rumus berikut dapat digunakan:

$$A_{11} = -X_{\dot{u}} \approx 0{,}05\ m$$

$$A_{22} = -Y_{\dot{v}} = \frac{1}{2}\int_{-\frac{L}{2}}^{L/2} \rho\pi D^2(x)dx \overset{D(x)=D}{=} \frac{1}{2}\rho\pi D^2 L$$

$$A_{66} = -N_{\dot{r}} = \frac{1}{2}\int_{-\frac{B}{2}}^{B/2} y^2 \frac{0{,}1m}{B} dy + \frac{1}{2}\int_{-\frac{L}{2}}^{L/2} x^2 \rho\pi D^2(x) dx$$

$$D(x) = D = \frac{1}{24}(0{,}1m\, B^2 + \rho\pi D^2 L^3)$$

Koefisien hidrodinamika didapat dari hasil turunan untuk kapal simetriks: [8]

$$Y_v' = \frac{Y_v}{\frac{1}{2}\rho LTU} = -(\frac{\pi T}{L} - C_{D0})$$

$$Y_r{}' = \frac{Y_r}{\frac{1}{2}\rho L^2 TU} = X_{\dot{u}}' + \frac{x_p}{L} Y_v$$

$$N_v{}' = \frac{N_v}{\frac{1}{2}\rho L^2 TU} = -(X_{\dot{u}}' + Y_{\dot{v}}')\frac{x_p}{L} Y_v{}'$$

$$N_r{}' = \frac{N_r}{\frac{1}{2}\rho L^3 TU} = \frac{1}{4} Y_v$$

$$Y_\delta{}' = \frac{Y_\delta}{\frac{1}{2}\rho LTU^2} = \rho \frac{\pi}{4}\frac{A_\delta}{LT}$$

$$N_\delta{}' = \frac{N_\delta}{\frac{1}{2}\rho L^2 TU^2} = -\frac{1}{2} Y_\delta{}'$$

Model AUV yang di Uji

Dimensi Utama

L = 1,3 m

B = 0,358 m

T = 0,15 m

U = 3,91 m/s

Δ = 24,07 Kg

π = 3,14

ρ = 1000 kg/m^3

Arudder = 0,0079 m^2

Maka didapatkan koefisien sebagai berikut:

$X_{\dot{u}}$ = -1,20	Y_v = -130,98	N_r = -166.46
$Y_{\dot{v}}$ = -45,92	Y_r = -613,53	Y_δ = 47268.94
$N_{\dot{r}}$ = -0,76	N_v = -802,46	N_δ = -30724.81

Nilai-nilai yang sudah didapatkan dimasukkan ke matriks sehingga didapatkan variabel yang ada pada persamaan Nomoto sebagai berikut:

$$M = \begin{bmatrix} 209.98 & 0 \\ 0 & 19.53 \end{bmatrix}, N = \begin{bmatrix} 130.98 & 707.65 \\ 802.46 & 166.46 \end{bmatrix}, b = \begin{bmatrix} 47268{,}94 \\ -30724{,}81 \end{bmatrix}$$

$\det(M) = 4101{,}64$, $\det(N) =$ -546055,63

$$T_1T_2 = \left|\frac{\det(M)}{\det(N)}\right| = 0{,}0075$$

$$T_1 + T_2 = \left|\frac{n_{11}m_{22} + n_{22}m_{11} - n_{12}m_{21} - n_{21}m_{12}}{\det(N)}\right| = 0{,}069$$

$$K_R = \left|\frac{n_{21}b_1 - n_{11}b_2}{\det(N)}\right| = 76{,}83$$

$$K_RT_3 = \left|\frac{m_{21}b_1 - m_{11}b_2}{\det(N)}\right| = 11{,}81$$

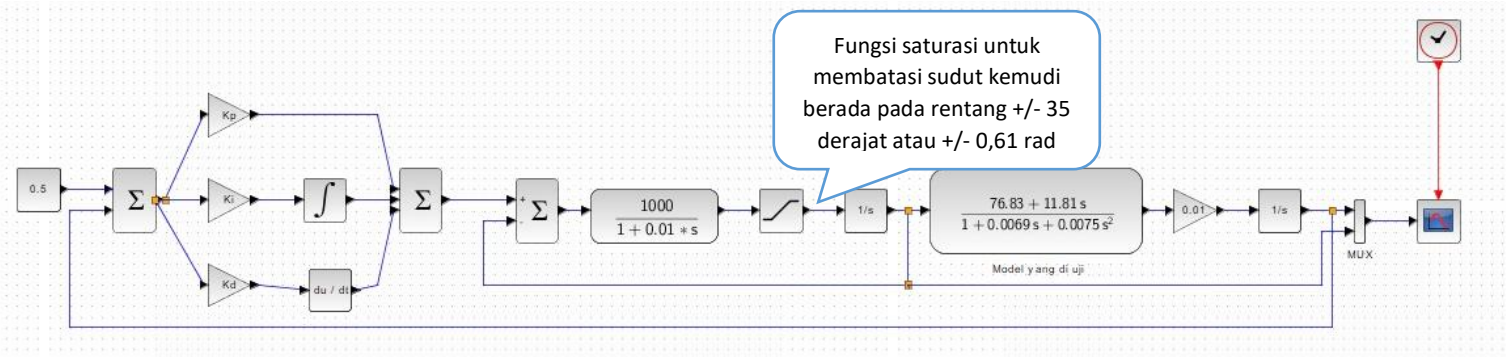
$$\frac{r}{\delta_R}(s) = \frac{K_R(1+T_3s)}{(1+T_1S)(1+T_2s)} = \frac{76{,}83+11{,}81s}{1+0.069s+0{,}0075s^2}$$

$$\frac{\psi}{\delta}(s) = \frac{K_R(1+T_3s)}{s(1+T_1S)(1+T_2s)} = \frac{76{,}83+11{,}81s}{s+0{,}069s^2+0{,}0075s^3}$$

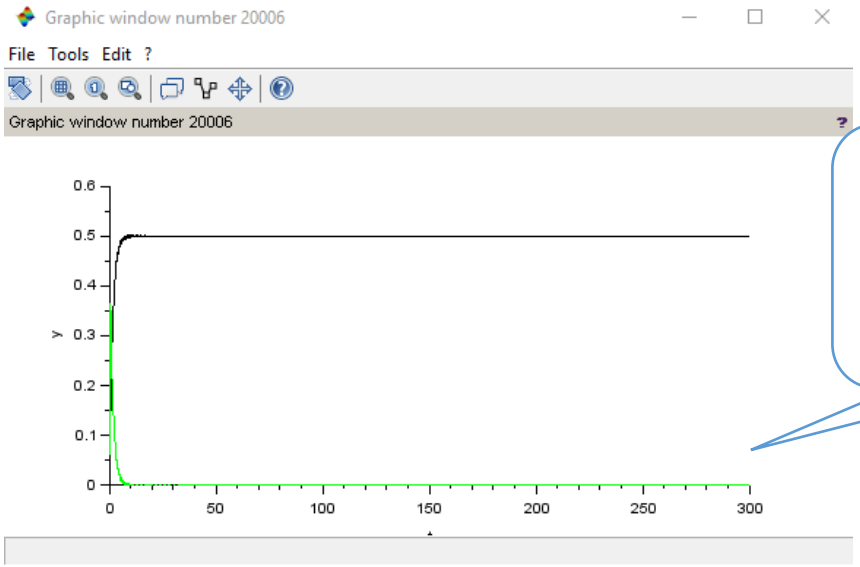
Catatan:

Grafik hijau menunjukkan sudut rudder

Grafik hitam menunjukkan sudut *heading*



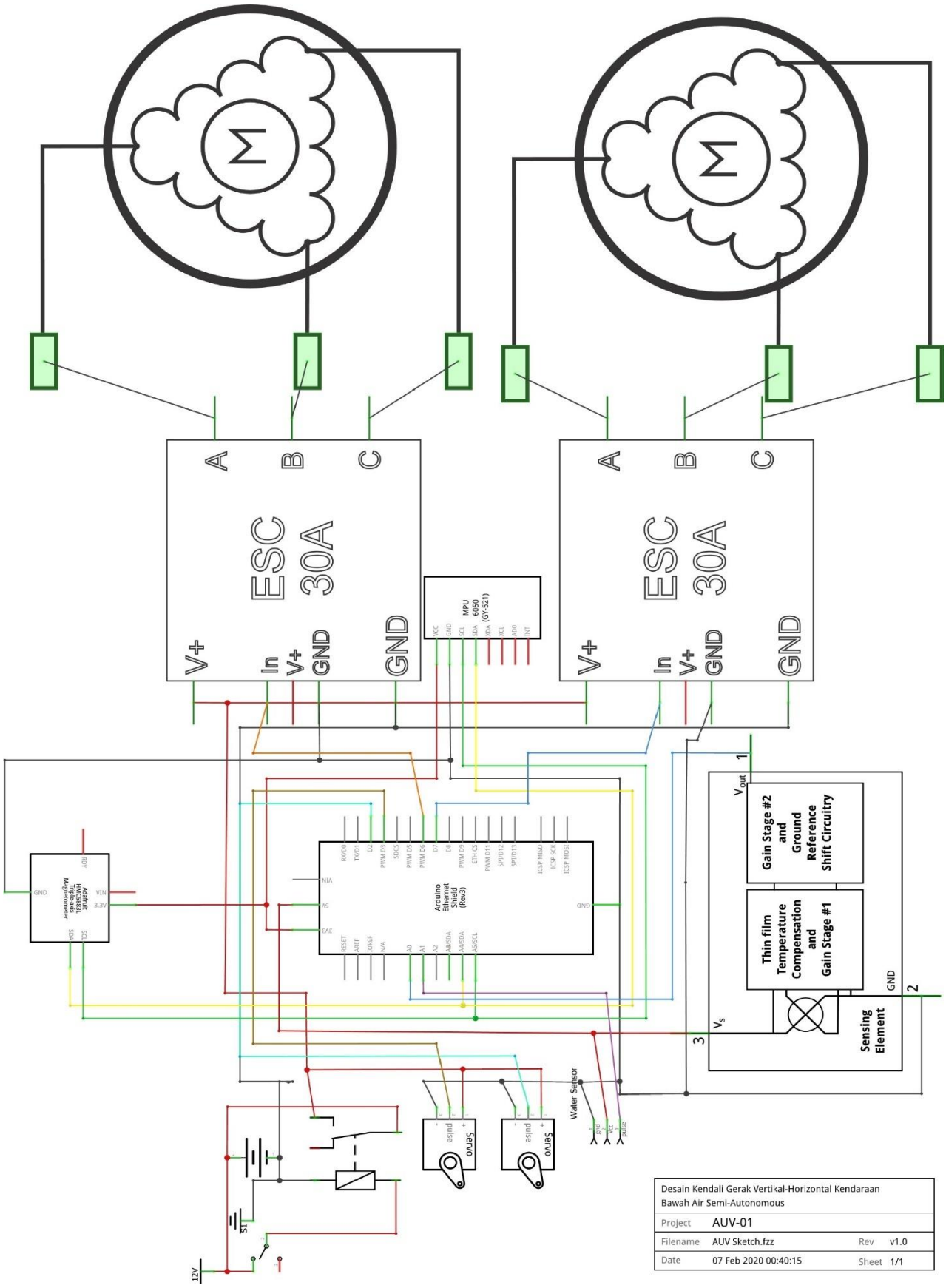
Gambar Block diagram model yag di uji



Kendali *heading* kapal berada pada sudut 0,5 rad atau pada 28,65 derajat. Sudut kemudi bergerak menuju nol. Sudut *heading* menuju set point dan bertahan pada sudut tersebut saat sudut kemudi 0 derajat.

Gambar Grafik Sudut terhadap waktu pada model yang di uji

Lampiran 4: Skema Rangkaian Listrik



Lampiran 5: Langkah Kerja Model

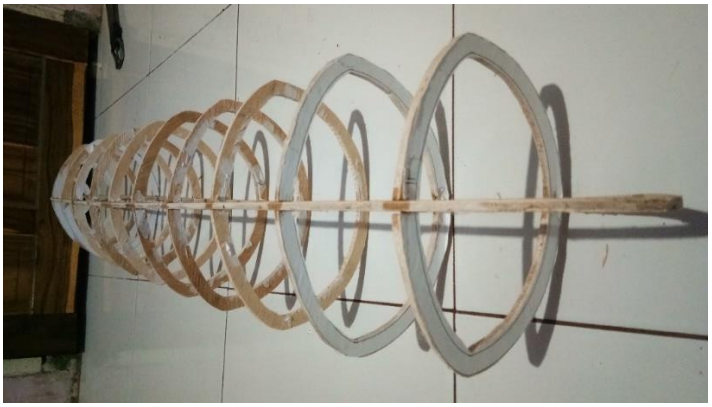
Alat dan bahan yang diperlukan adalah sebagai berikut:

Tabel 4.2 Alat dan bahan

No.	Alat	Bahan
1.	Gergaji/Pemotong Tripleks	Tripleks 2mm/3mm
2.	Bor	Kayu Balsa 1.5 mm
3.	Gerinda	Lem G
4.	Kuas	Dempul
5.	Cutter	Resin
6.		Katalis
7.		Pylox

Langkah kerja pembuatan model sebagai berikut:

1. Cetak gading sesuai bentuknya (dikurangi 0.2~0.3 cm sebagai tebal kulit) dengan bahan tripleks.
2. Susun gading tegak lurus



Penyusunan gading pada balok sebagai acuan

3. Potong-potong kayu balsah dengan lebar 0.2 cm agar dapat dengan mudah di bengkokkan sesuai bentuk lambung kapal
4. Tempelkan kayu balsah yang sudah dipotong-potong tadi dengan lem G secara bergantian antara bagian kiri dan kanan

Pemasangan kulit lambung kapal

5. Setelah semua kulit terpasang, lambung di dempul untuk menutupi bagian bagian yang berlubang, kemudian di amplas setelah kering
6. Untuk menjadikan cetakan agar dapat dibuat secara massal, dilanjutkan dengan fiberglass menggunakan metode kontak dengan cara menempelkan sejumlah resin dan diperkuat dengan mat.
7. Pada lapisan mat yang ketiga, diberi kayu agar cetakan nantinya mudah untuk dilepas.

Gambar 4.7 Pemasangan kayu pada lapisan ketiga

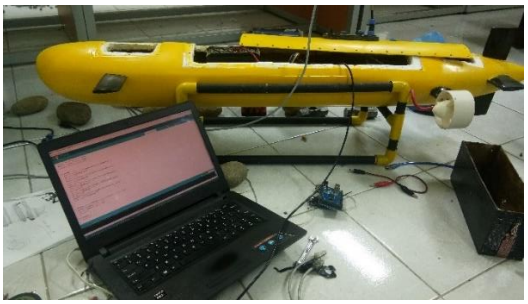
8. Setelah cetakan kering, maka cetakan dipotong menjadi bagian atas dan bawah dengan menggunakan gerinda. Inilah yang akan menjadi cetakan utama pada model AUV ini.
9. Bagian dalam pada setiap cetakan harus dibersihkan dengan MAA/ Mirror Glass (Wax) agar pada saat model ini selesai, model tidak akan lengket pada cetakan sehingga mudah untuk dilepas.
10. 30 menit setelah pelapisan MAA/ Mirror Glass, Campur resin+pewarna+ katalis lalu aplikasikan ke cetakan dengan menggunakan kuas.

Model bagian atas dan bagian bawah

11. Setelah semua bagian selesai, rakitlah setiap bagian dimulai dari baterai, Fin, dan komponen listrik lainnya.
12. Buatlah bukaan pada bagian atas lambung sebagai masuknya komponen-komponen elektronik, kemudian kapal di cat dengan warna kuning sebagai warna research pada umumnya.
13. Rangkai setiap komponen elektrik.

Proses Solder rangkaian listrik

14. Kapal siap di *setup*



Setup program Arduino

Lampiran 6: Coding Program Arduino

Tab 1: Main

```
/* INDEX DATA MODBUS
 *  0 Digital control by Remote
 *  1 joystick command motor 1    pin 2
 *  2 joystick command motor 2    pin 3
 *  3 joystick command motor 3    pin 6
 *  4 joystick command motor 4    pin 7
 *  5 joystick command motor 5    pin 8
 *  6 Kosong
 *  7 millis from Arduino
 *  8 Digital indicator by ROV
 *  9 accX
 *  10 accY
 *  11 accZ
 *  12 roll
 *  13 pitch
 *  14 yaw
 *  15 temperature
 *  16 Baterai voltage
 *  17 Salinity
 *  18 Water pressure
 *  19 Deep from water surface
 *  20 leaks indikator
 */
#include <SPI.h>
#include <Wire.h>
#include <Ethernet.h>
#include <Servo.h>
#include "MgsModbus.h"
#include "_defenitions.h"
int k=1;
long tick=0;
int16_t pos = 0;

void setup() {
  pinMode(9,OUTPUT);
  digitalWrite(9,HIGH);
  Serial.begin(57600);
  Wire.begin();                         // Join the line
  Wire.setClock(1000);
  Ethernet.begin(mac, ip, gateway, subnet);   // start ethernet interface
  initServo();
  initMPU6050();
  initHMC();
  tick = millis();
}
```

```
void loop() {
  //state_ROV();
  readMPU6050();
  delay(10);
  readHMC();
  delay(10);
  analogSensor();
  delay(5);
  digitalSensor();
  Mb.MbData[7] = (millis()-tick);
  Mb.MbsRun();
  delay(10);
  setPWM3();
  tick = millis();
}
```

Tab 2: MgsModbus.cpp

```
#include "MgsModbus.h"
EthernetServer MbServer(MB_PORT);
EthernetClient MbmClient;
#define DEBUG
MgsModbus::MgsModbus()  {
}

//******************** Recieve data for ModBusSlave ****************
void MgsModbus::MbsRun() {
//******************** Read from socket ****************
EthernetClient client = MbServer.available();
//******************** Read from serial ****************

 if(client.available()) {
   int i = 0;
   while(client.available()) { MbsByteArray[i] = client.read();   i++; }
     MbsFC = SetFC(MbsByteArray[7]);            //Byte 7 of request is FC
 }

 int Start, WordDataLength, ByteDataLength, CoilDataLength, MessageLength;
 //******************** Read Coils (1 & 2) **********************
 if(MbsFC == MB_FC_READ_COILS || MbsFC == MB_FC_READ_DISCRETE_INPUT) {
   Start = word(MbsByteArray[8],MbsByteArray[9]);
   CoilDataLength = word(MbsByteArray[10],MbsByteArray[11]);
   ByteDataLength = CoilDataLength / 8;
   if(ByteDataLength * 8 < CoilDataLength) ByteDataLength++;
     CoilDataLength = ByteDataLength * 8;
     MbsByteArray[5] = ByteDataLength + 3; //angka bytes setelah ini.
     MbsByteArray[8] = ByteDataLength;       //angka bytes setelah ini (atau angka bytes data).
     for(int i = 0; i < ByteDataLength ; i++) {
       MbsByteArray[9 + i] = 0; // To get all remaining not written bits zero
       for(int j = 0; j < 8; j++) {
         bitWrite(MbsByteArray[9 + i], j, GetBit(Start + i * 8 + j));
```

```
        }
       }
   MessageLength = ByteDataLength + 9;
   client.write(MbsByteArray, MessageLength);
   MbsFC = MB_FC_NONE;
  }
  //****************** Read Registers (3 & 4) ******************
  if(MbsFC == MB_FC_READ_REGISTERS || MbsFC == MB_FC_READ_INPUT_REGISTER) {
   Start = word(MbsByteArray[8],MbsByteArray[9]);
   WordDataLength = word(MbsByteArray[10],MbsByteArray[11]);
   ByteDataLength = WordDataLength * 2;
   MbsByteArray[5] = ByteDataLength + 3; //Number of bytes after this one.
   MbsByteArray[8] = ByteDataLength;    //Number of bytes after this one (or number of bytes of data).
   for(int i = 0; i < WordDataLength; i++)
    {
     MbsByteArray[ 9 + i * 2] = highByte(MbData[Start + i]);
     MbsByteArray[10 + i * 2] = lowByte(MbData[Start + i]);
    }
   MessageLength = ByteDataLength + 9;
   client.write(MbsByteArray, MessageLength);
   MbsFC = MB_FC_NONE;
  }
  //****************** Write Coil (5) ***********************
  if(MbsFC == MB_FC_WRITE_COIL) {
   Start = word(MbsByteArray[8],MbsByteArray[9]);
   if (word(MbsByteArray[10],MbsByteArray[11]) == 0xFF00){SetBit(Start,true);}
   if (word(MbsByteArray[10],MbsByteArray[11]) == 0x0000){SetBit(Start,false);}
   MbsByteArray[5] = 2; //Number of bytes after this one.
   MessageLength = 8;
   client.write(MbsByteArray, MessageLength);
   MbsFC = MB_FC_NONE;
  }

  //****************** Write Register (6) ******************
  if(MbsFC == MB_FC_WRITE_REGISTER) {
   Start = word(MbsByteArray[8],MbsByteArray[9]);
   MbData[Start] = word(MbsByteArray[10],MbsByteArray[11]);
   MbsByteArray[5] = 6; //Number of bytes after this one.
   MessageLength = 12;
   client.write(MbsByteArray, MessageLength);
   MbsFC = MB_FC_NONE;
  }

  //****************** Write Multiple Coils (15) ***********************
  if(MbsFC == MB_FC_WRITE_MULTIPLE_COILS) {
   Start = word(MbsByteArray[8],MbsByteArray[9]);
   CoilDataLength = word(MbsByteArray[10],MbsByteArray[11]);
   MbsByteArray[5] = 6;
```

```
    for(int i = 0; i < CoilDataLength; i++) {
       SetBit(Start + i,bitRead(MbsByteArray[13 + (i/8)],i-((i/8)*8)));
      }
    MessageLength = 12;
    client.write(MbsByteArray, MessageLength);
    MbsFC = MB_FC_NONE;
  }
  //****************** Write Multiple Registers (16) ******************
  if(MbsFC == MB_FC_WRITE_MULTIPLE_REGISTERS) {
    Start = word(MbsByteArray[8],MbsByteArray[9]);
    WordDataLength = word(MbsByteArray[10],MbsByteArray[11]);
    ByteDataLength = WordDataLength * 2;
    MbsByteArray[5] = 6;
    for(int i = 0; i < WordDataLength; i++) {
       MbData[Start + i] =  word(MbsByteArray[ 13 + i * 2],MbsByteArray[14 + i * 2]);
      }
    MessageLength = 12;
    client.write(MbsByteArray, MessageLength);
    MbsFC = MB_FC_NONE;
  }
}


//****************** ?? ******************
MB_FC MgsModbus::SetFC(int fc) {
  MB_FC FC;
  FC = MB_FC_NONE;
  if(fc == 1) FC = MB_FC_READ_COILS;
  if(fc == 2) FC = MB_FC_READ_DISCRETE_INPUT;
  if(fc == 3) FC = MB_FC_READ_REGISTERS;
  if(fc == 4) FC = MB_FC_READ_INPUT_REGISTER;
  if(fc == 5) FC = MB_FC_WRITE_COIL;
  if(fc == 6) FC = MB_FC_WRITE_REGISTER;
  if(fc == 15) FC = MB_FC_WRITE_MULTIPLE_COILS;
  if(fc == 16) FC = MB_FC_WRITE_MULTIPLE_REGISTERS;
  return FC;
}

word MgsModbus::GetDataLen() {
  return MbDataLen;
}
boolean MgsModbus::GetBit(word Number) {
  int ArrayPos = Number / 16;
  int BitPos = Number - ArrayPos * 16;
  boolean Tmp = bitRead(MbData[ArrayPos],BitPos);
  return Tmp;
}

boolean MgsModbus::SetBit(word Number,boolean Data) {
  int ArrayPos = Number / 16;
```

```
  int BitPos = Number - ArrayPos * 16;
  boolean Overrun = ArrayPos > MbDataLen * 16; // check for data overrun
  if (!Overrun){
    bitWrite(MbData[ArrayPos],BitPos,Data);
  }
  return Overrun;
}
```

Tab 3:

```
#include "Arduino.h"
#include <SPI.h>
#include <Ethernet.h>
#ifndef MgsModbus_h
#define MgsModbus_h
#define MbDataLen 21         // length of the MdData array
#define MB_PORT 502
enum MB_FC {
  MB_FC_NONE                  = 0,
  MB_FC_READ_COILS            = 1,
  MB_FC_READ_DISCRETE_INPUT   = 2,
  MB_FC_READ_REGISTERS        = 3,
  MB_FC_READ_INPUT_REGISTER   = 4,
  MB_FC_WRITE_COIL            = 5,
  MB_FC_WRITE_REGISTER        = 6,
  MB_FC_WRITE_MULTIPLE_COILS  = 15,
  MB_FC_WRITE_MULTIPLE_REGISTERS = 16  };
class MgsModbus {
public:
  //  general
  MgsModbus();
  word MbData[MbDataLen]; // memory block that holds all the modbus user data
  boolean GetBit(word Number);
  boolean SetBit(word Number,boolean Data); // returns true when the number is in the MbData
range
  // modbus master
  void Req(MB_FC FC, word Ref, word Count, word Pos);
  IPAddress remSlaveIP;
  // modbus slave
  void MbsRun();
  word GetDataLen();
private:
  //  general
  MB_FC SetFC(int fc);
  // modbus slave
  uint8_t MbsByteArray[260]; // send and recieve buffer
  MB_FC MbsFC;
};

#endif
```

Tab 4: _definition.h

```
// ================================================MPU6050 Parameters ========================================
#define grav 9.80665
int   gyro_error=0;                    //menggunakan variabel untuk menghitung data gyro yang error
float Gyr_rawX, Gyr_rawY, Gyr_rawZ;        //menyimpan pembacaan data mentah
float Gyro_angle_x, Gyro_angle_y;          //menyimpan nilai sudut yang didapatkan pada data gyro
float Gyro_raw_error_x, Gyro_raw_error_y;  //menyimpan data awal gyro yang error

//Acc Variables
int   acc_error=0;                     //We use this variable to only calculate once the Acc data error
float rad_to_deg = 180/3.141592654;         //This value is for pasing from radians to degrees values
float Acc_rawX, Acc_rawY, Acc_rawZ;        //Here we store the raw data read
float Acc_rawX_error, Acc_rawY_error, Acc_rawZ_error;    //Here we store the raw data read
float Acc_angle_x, Acc_angle_y;            //Here we store the angle value obtained with Acc data
float Acc_angle_error_x, Acc_angle_error_y; //Here we store the initial Acc data error
float Total_angle_x, Total_angle_y;

// Create a compass HMC5883L
#define HMCAdd  0x1E
#define HMCRegA 0x10    // RegA='00010000'
#define HMCRegB 0xC0   // RegB='00100000' Mode = 10Hz, RegB='11000000' Mode = 50Hz
#define HMCRead 0x03    // RegAddres DataOut 0x03 --> 0x08
#define HMCWrite 0x3C   // RegAddres DataOut 0x03 --> 0x08
int16_t mag[3];
const float declinationAngle = 0.71666667;    // untuk kota Makassar
const float  PI_F =  3.14159265F;
const float  G    =  9.817F;

// membuat class pada pwm motor
// Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

#define PWMAdd 0x40            // pwm addres default
#define PWMRegMode 0x00        //
#define PWMPrescal 0xFE        //
#define PWMLED0_ON_L 0x20      // 0x06
const uint8_t nMtr = 5;
uint8_t  pwmPin[] = {2,3,6,7,8}; // {3,5,6,9,10};
//uint16_t pwmON[]  = {0,0,0,0,0}, pwmOFF[nMtr];
//int      PWM[5];
//int      initPWM  = 1500;
//int      minPWM   = 700;
//int      maxPWM   = 2300;
//uint16_t freq = 1000;
uint16_t count;
```

```
// membuat modbus
MgsModbus Mb;
// Modbus register idx = 0 is identifier and output coils
// Modbus register idx = 1 is identifier and input coils
// ============================================== PARAMETER ETHERNET
==============================================
// Ethernet  settings  (Tergantung pada  MAC dan Local  network)
byte  mac[]  =  {0x90,  0xA2,  0xDA,  0x0E,  0x94,  0xB5  };
IPAddress  ip(192,168,1,3);
IPAddress  gateway(192,168,1,3);
IPAddress  subnet(255,  255,  255,  0);

//==============================================
byte mode;
//==============================================                PARAMETER
SOFTWARE SERIAL ==========================================

Servo myservo1;  // membuat object servo untuk mengontrol servo
Servo myservo2;
Servo myservo3;
Servo myservo4;
Servo myservo5;

// pin 4,10,11,12,13   digunakan oleh ethernet shield dan SD Card
const  int  pinVolt  =  A3;
const  int  pinSalt  =  A2;
const  int  pinPres  =  A0;
const  int  pinLeaks  = A1;      // pin  leak;  , leak  case FloatBox,  Lamp
//const   int   pinSCL     =  A4;          // SCL   --> Sensors: MPU6050(Accl,  Gyro,  Temp),
ServoDrv(Motor(5),Lamp(2))
//const  int  pinSDA    = A5;       // SDA  --> Address: MPU6050 = 0x68, ServoDrive= 0x40,
HMC = 0x1E;
const  int  pinLamp1  = 22, pinLamp2 = 23;


// Coef  konversi  analog  sensor
float  kVolt   =  1.0;      //Coef  Konversi  tegangan per 1024
float  kPres   =  1.0;      //Coef  Konversi  tekanan per 1024
float  kDeep   =  1.0;      //Coef  Konversi  kedalaman per 1024
float  kSalt   =  1.0;     //Coef  Konversi  Salinity per 1024
float  Kvolt   =  1.0;      //Coef  konversi  tegangan / volt
float  kLeaks  =  1.0;
```

Tab 5: _WritereadByte

```
// Penulisan data byte untuk penggunaan I2C

void writeByte(uint8_t address, uint8_t regAdd, uint8_t data) {
  Wire.beginTransmission(address);  // Initialize the Tx buffer
```

```
  Wire.write(regAdd);              // Put slave register address in Tx buffer
  Wire.write(data);                // Put data in Tx buffer
  Wire.endTransmission();          // Send the Tx buffer
}

uint8_t readByte(uint8_t address, uint8_t regAdd) {
  uint8_t data; // `data` will store the register data
  Wire.beginTransmission(address);         // Initialize the Tx buffer
  Wire.write(regAdd);                      // Put slave register address in Tx buffer
  Wire.endTransmission(false);          // Send the Tx buffer, but send a restart to keep connection
alive
  Wire.requestFrom(address, (uint8_t)1, true);  // Read one byte from slave register address
  data = Wire.read();                      // Fill Rx buffer with result
  //Wire.endTransmission();                //
  return data;                             // Return data read from slave register
}

void writeBytes(uint8_t address, uint8_t regAdd, uint8_t n, uint8_t * data) {
  Wire.beginTransmission(address);  // Initialize the Tx buffer
  Wire.write(regAdd);               // Put slave register address in Tx buffer
  for(int i=0;i<n;i++) {
    Wire.write(data[i]);            // Put data in Tx buffer
  }
  Wire.endTransmission();           // Send the Tx buffer
}

uint8_t readBytes1(uint8_t address, uint8_t regAdd) {
  uint8_t data; // `data` will store the register data
  Wire.beginTransmission(address);     // Initialize the Tx buffer
  Wire.write(regAdd);                  // Put slave register address in Tx buffer
  Wire.endTransmission(false);         // Send the Tx buffer, but send a restart to keep connection
alive
  Wire.requestFrom(address, 2, true);  // Read two bytes from slave register address
  return Wire.read()<<8 | Wire.read();                    // Fill Rx buffer with result
}

word convert2word(int16_t data) {
  if(data<0) data=data+32768;
  return data;
}
```

Tab 6: a_magFunctions

```
// code untuk membaca sensor magnetometer
void initHMC(void){
  writeByte(HMCAdd,0x02,0x00);
  delay(100);
}

void readHMC(){
  float angle[3];
```

```
  Wire.beginTransmission(HMCAdd);            //Initiate a transmission with HMC5883 (Write address).
  Wire.write(HMCRead);                       //Place the Mode Register Address in send-buffer.
  Wire.endTransmission(false);               //Send the send-buffer to HMC5883 and end the I2C transmission.
  Wire.requestFrom(HMCAdd,6,true);           //Request 6 bytes of data from the address specified.
                                 // read high byte before low byte, its different to accelerometer and gyro
    mag[0] = (int16_t)(Wire.read()<<8 | Wire.read());
    mag[1] = (int16_t)(Wire.read()<<8 | Wire.read());
    mag[2] = (int16_t)(Wire.read()<<8 | Wire.read());
    angle[0] = mag[0] / 1100;
    angle[1] = mag[1] / 1100;
    angle[2] = mag[2] / 980;

 if(mag[0]==0) {(mag[1]<0) ? angle[2] = 90 : angle[2] = 0;}
   else {
   angle[2] = (atan2(mag[1], mag[0])) * 360 / (2*PI);
   if(angle[2]>=360) angle[2]=angle[2]-360;
   if(angle[2]<0) angle[2]=angle[2]+360;
   }
   Mb.MbData[14] = convert2word(angle[2]);
```

Tab 7: b_mpuFunctions

```
void initMPU6050(){
  Wire.begin();                        //begin the wire comunication
  Wire.beginTransmission(0x68);        //begin, Send the slave adress (in this case 68)
  Wire.write(0x6B);                    //make the reset (place a 0 into the 6B register)
  Wire.write(0x00);
  Wire.endTransmission(true);          //end the transmission
  //Gyro config
  Wire.beginTransmission(0x68);        //begin, Send the slave adress (in this case 68)
  Wire.write(0x1B);                    //We want to write to the GYRO_CONFIG register (1B hex)
  Wire.write(0x10);                    //Set the register bits as 00010000 (1000dps full scale)
  Wire.endTransmission(true);          //End the transmission with the gyro
  //Acc config
  Wire.beginTransmission(0x68);        //Start communication with the address found during search.
  Wire.write(0x1C);                    //We want to write to the ACCEL_CONFIG register
  Wire.write(0x10);                    //Set the register bits as 00010000 (+/- 8g full scale range)
  Wire.endTransmission(true);

  /*Here we calculate the acc data error before we start the loop
 * I make the mean of 200 values, that should be enough*/
  if(acc_error==0)
  {
    for(int a=0; a<200; a++)
    {
      Wire.beginTransmission(0x68);
```

```
    Wire.write(0x3B);                  //Ask for the 0x3B register- correspond to AcX
    Wire.endTransmission(false);
    Wire.requestFrom(0x68,6,true);
    Acc_rawX=(Wire.read()<<8|Wire.read())/4096.0 ; //each value needs two registres
    Acc_rawY=(Wire.read()<<8|Wire.read())/4096.0 ;
    Acc_rawZ=(Wire.read()<<8|Wire.read())/4096.0 ;
    Acc_rawX_error += Acc_rawX;
    Acc_rawY_error += Acc_rawY;
    Acc_rawZ_error += Acc_rawZ;

    /*---X---*/
    Acc_angle_error_x = Acc_angle_error_x + ((atan((Acc_rawY)/sqrt(pow((Acc_rawX),2) + pow((Acc_rawZ),2)))*rad_to_deg));
    /*---Y---*/
    Acc_angle_error_y = Acc_angle_error_y + ((atan(-1*(Acc_rawX)/sqrt(pow((Acc_rawY),2) + pow((Acc_rawZ),2)))*rad_to_deg));

    if(a==199)
    {
      Acc_angle_error_x = Acc_angle_error_x/200;
      Acc_angle_error_y = Acc_angle_error_y/200;

      Acc_rawX_error = Acc_rawX_error/200;
      Acc_rawY_error = Acc_rawY_error/200;
      Acc_rawZ_error = Acc_rawZ_error/200;
      acc_error=1;
    }
   }
  }//end of acc error calculation
  delay(100);
}

void readMPU6050() {
  static long prevTime;
  static float preAccX, preAccY, preAccZ, dX, dY, dZ;
  long nowTime = millis();                  // actual time read
  float elapsedTime = (nowTime - prevTime) / 1000; //divide by 1000 in order to obtain seconds
  float totalAcc, NormAccX, NormAccY, NormAccZ;

  ////////////////////////////////////Gyro read//////////////////////////////////
   Wire.beginTransmission(0x68);          //begin, Send the slave adress (in this case 68)
   Wire.write(0x43);                  //First adress of the Gyro data
   Wire.endTransmission(false);
   Wire.requestFrom(0x68,4,true);         //We ask for just 4 registers

   Gyr_rawX=Wire.read()<<8|Wire.read();     //Once again we shif and sum
   Gyr_rawY=Wire.read()<<8|Wire.read();
   /*Now in order to obtain the gyro data in degrees/seconds we have to divide first
   the raw value by 32.8 because that's the value that the datasheet gives us for a 1000dps range*/
```

```
  /*---X---*/
  Gyr_rawX = (Gyr_rawX/32.8) - Gyro_raw_error_x;
  /*---Y---*/
  Gyr_rawY = (Gyr_rawY/32.8) - Gyro_raw_error_y;

  /*Now we integrate the raw value in degrees per seconds in order to obtain the angle
  * If you multiply degrees/seconds by seconds you obtain degrees */
  /*---X---*/
  Gyro_angle_x = Gyr_rawX*elapsedTime;
  /*---X---*/
  Gyro_angle_y = Gyr_rawY*elapsedTime;
 ////////////////////////////////////Acc read/////////////////////////////////////
 Wire.beginTransmission(0x68);     //begin, Send the slave adress (in this case 68)
 Wire.write(0x3B);                 //Ask for the 0x3B register- correspond to AcX
 Wire.endTransmission(false);      //keep the transmission and next
 Wire.requestFrom(0x68,6,true);    //We ask for next 6 registers starting withj the 3B
 /*We have asked for the 0x3B register. The IMU will send a brust of register.
 * The amount of register to read is specify in the requestFrom function.
 * In this case we request 6 registers. Each value of acceleration is made out of
 * two 8bits registers, low values and high values. For that we request the 6 of them
 * and just make then sum of each pair. For that we shift to the left the high values
 * register (<<) and make an or (|) operation to add the low values.
 If we read the datasheet, for a range of+-8g, we have to divide the raw values by 4096*/
 Acc_rawX=(Wire.read()<<8|Wire.read())/4096.0 ; //each value needs two registres
 Acc_rawY=(Wire.read()<<8|Wire.read())/4096.0 ;
 Acc_rawZ=(Wire.read()<<8|Wire.read())/4096.0 ;
/*Now in order to obtain the Acc angles we use euler formula with acceleration values
 after that we substract the error value found before*/
 /*---X---*/
 Acc_angle_x              =              (atan((Acc_rawY)/sqrt(pow((Acc_rawX),2)              +
pow((Acc_rawZ),2)))*rad_to_deg) - Acc_angle_error_x;
 /*---Y---*/
 Acc_angle_y              =              (atan(-1*(Acc_rawX)/sqrt(pow((Acc_rawY),2)              +
pow((Acc_rawZ),2)))*rad_to_deg) - Acc_angle_error_y;

 ////////////////////////////////////Total angle and filter/////////////////////////////////////
 /*---X axis angle---*/
 Total_angle_x = 0.98 *(Total_angle_x + Gyro_angle_x) + 0.02*Acc_angle_x;
 /*---Y axis angle---*/
 Total_angle_y = 0.98 *(Total_angle_y + Gyro_angle_y) + 0.02*Acc_angle_y;
                  // the previous time is stored before the actual time read

// ================================ ===== SPEED ======================

//Serial.print(nowTime - prevTime);
//elapsedTime = (nowTime-prevTime);
totalAcc = sqrt((Acc_rawX*Acc_rawX + Acc_rawY*Acc_rawY + Acc_rawZ*Acc_rawZ));
NormAccX = Acc_rawX / totalAcc;
NormAccY = Acc_rawY / totalAcc;
NormAccZ = Acc_rawZ / totalAcc;
```

```
//dX += (NormAccX - preAccX) * elapsedTime;
//dY += (NormAccY - preAccY) * elapsedTime;
//dZ += (NormAccZ - preAccZ) * elapsedTime;
//preAccX = NormAccX;   preAccY = NormAccY;    preAccZ = NormAccZ;
//Serial.print(NormAccX); Serial.print("   ");
//Serial.print(NormAccY); Serial.print("   ");
//Serial.print(NormAccZ); Serial.print("   ");
//Serial.print(dX); Serial.print("   ");
//Serial.print(dY); Serial.print("   ");
//Serial.print(dZ); Serial.print("   ");

 Mb.MbData[12] = convert2word(Total_angle_x);  // + 180;                    // Sudut gerak
putar roll
// Serial.print("Tot.angle.x: ");
// Serial.print(Mb.MbData[12]);
// Serial.print("     ");
 Mb.MbData[13] = convert2word(Total_angle_y);  // + 180);                   // Sudut gerak
putar pitch
// Serial.print("Tot.angle.y: ");
// Serial.print(Mb.MbData[13]);
// Serial.print("     ");
 Mb.MbData[9]  = convert2word(NormAccX*1000);     //*1000+1000;             // Percepatan
pada sumbu_X +/- 2g
// Serial.print("NormAccX: ");
// Serial.print(Mb.MbData[9]);
// Serial.print("     ");
 Mb.MbData[10] = convert2word(NormAccY*1000);      //NormAccY*1000+1000;      //
Percepatan pada sumbu_Y
// Serial.print("NormAccY: ");
// Serial.print(Mb.MbData[10]);
// Serial.print("     ");
 Mb.MbData[11]  =  convert2word(NormAccZ*1000);          //*1000,-10000,10000,0,65535);
//NormAccZ*1000+1000;                    // Percepatan pada sumbu_Z Mb.MbData[17] =
(readBytes1(0x68, 0x41) / 340. + 36.53) * 100; // temperature, Temperature in degrees
Centigrade
// Serial.print("NormAccZ: ");
// Serial.print(Mb.MbData[11]);
// Serial.print("     ");
 Mb.MbData[15]  =  convert2word((readBytes1(0x68,0x41)/340.+36.53)*1000);          //
temperature, Temperature in degrees Centigrade
// Serial.print("Temp.in.degree: ");
// Serial.print(Mb.MbData[12]);
// Serial.print("     ");
prevTime = nowTime;
}
```

Tab 8: c_pwmFunctions

```
void initServo() {
  myservo1.attach(pwmPin[0]);  // attaches the servo on pin 2 to the servo object
  myservo2.attach(pwmPin[1]);  // attaches the servo on pin 3 to the servo object
```

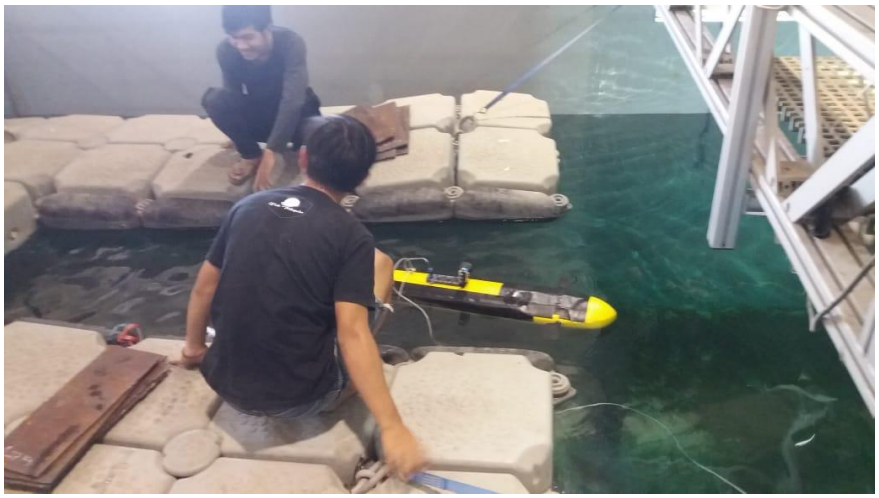
```
  myservo3.attach(pwmPin[2]);  // attaches the servo on pin 6 to the servo object
  myservo4.attach(pwmPin[3]);  // attaches the servo on pin 7 to the servo object
  myservo5.attach(pwmPin[4]);  // attaches the servo on pin 8 to the servo object
  delay(50);
//  myservo1.writeMicroseconds(initPWM);              // tell servo to go to position in variable
'pos'
//  myservo2.writeMicroseconds(initPWM);
//  myservo3.writeMicroseconds(initPWM);
//  myservo4.writeMicroseconds(initPWM);
//  myservo5.writeMicroseconds(initPWM);
//  delay(50);
}

void setPWM3() {
//    myservo1.writeMicroseconds(minmax(minPWM,maxPWM,Mb.MbData[1]));
//    myservo2.writeMicroseconds(minmax(minPWM,maxPWM,Mb.MbData[2]));
//    myservo3.writeMicroseconds(minmax(minPWM,maxPWM,Mb.MbData[3]));
//    myservo4.writeMicroseconds(minmax(minPWM,maxPWM,Mb.MbData[4]));
//    myservo5.writeMicroseconds(minmax(minPWM,maxPWM,Mb.MbData[5]));
    myservo1.writeMicroseconds(Mb.MbData[1]);
    myservo2.writeMicroseconds(Mb.MbData[2]);
    myservo3.writeMicroseconds(Mb.MbData[3]);
    myservo4.writeMicroseconds(Mb.MbData[4]);
    myservo5.writeMicroseconds(Mb.MbData[5]);
}
```

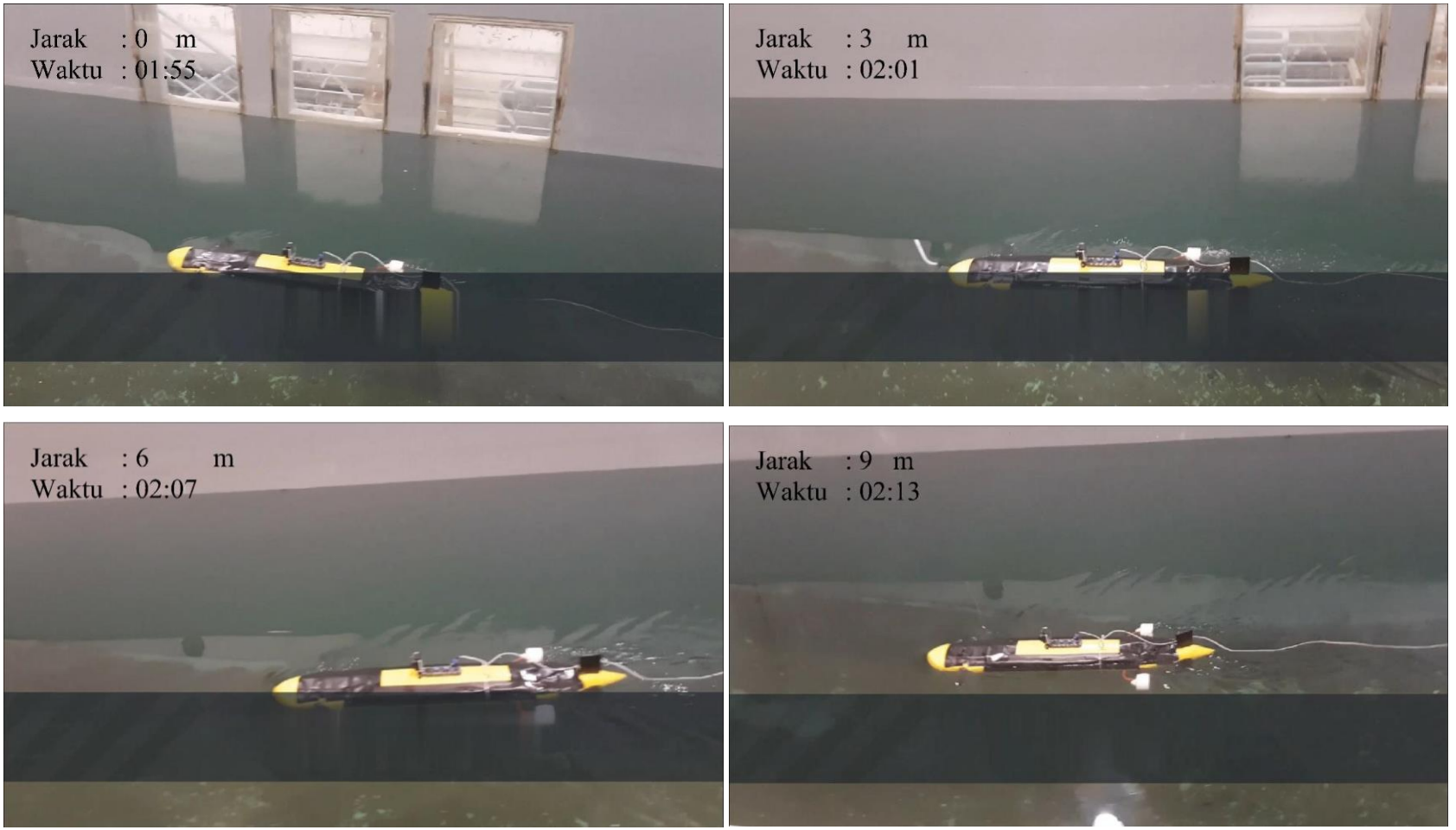
Tab 9 : d_analogDigitalModule

```
void analogSensor(){
  // read the baterai voltage
  Mb.MbData[16] = convert2word(analogRead(pinVolt));    // Tegangan baterai maksimum 16
volt
 // read salinity water
  Mb.MbData[17] = convert2word(analogRead(pinSalt));    // Salinity Scale maksimum 10
 // read water pressure
  Mb.MbData[18] = convert2word(analogRead(pinPres));    // Pressure Scale maksimum 100
Bar
 // read deep
  Mb.MbData[19] = convert2word(analogRead(pinPres));    // Pressure Scale maksimum 100
draft
  // read water leaks of tank
  Mb.MbData[20] = convert2word(analogRead(pinLeaks));   // Pressure Scale maksimum 100
draft
}

void digitalSensor() {
  // digital indikator diletakkan pada  Mb.MbData[8]
  uint16_t bitOutput = 0;
  uint16_t bitInput  = Mb.MbData[0];
  // blinking led indicator for power problem
  // digitalRead(pinPower) == HIGH ? bitWrite(bitOutput, 0, 1) : bitWrite(bitOutput, 0, 0);
```

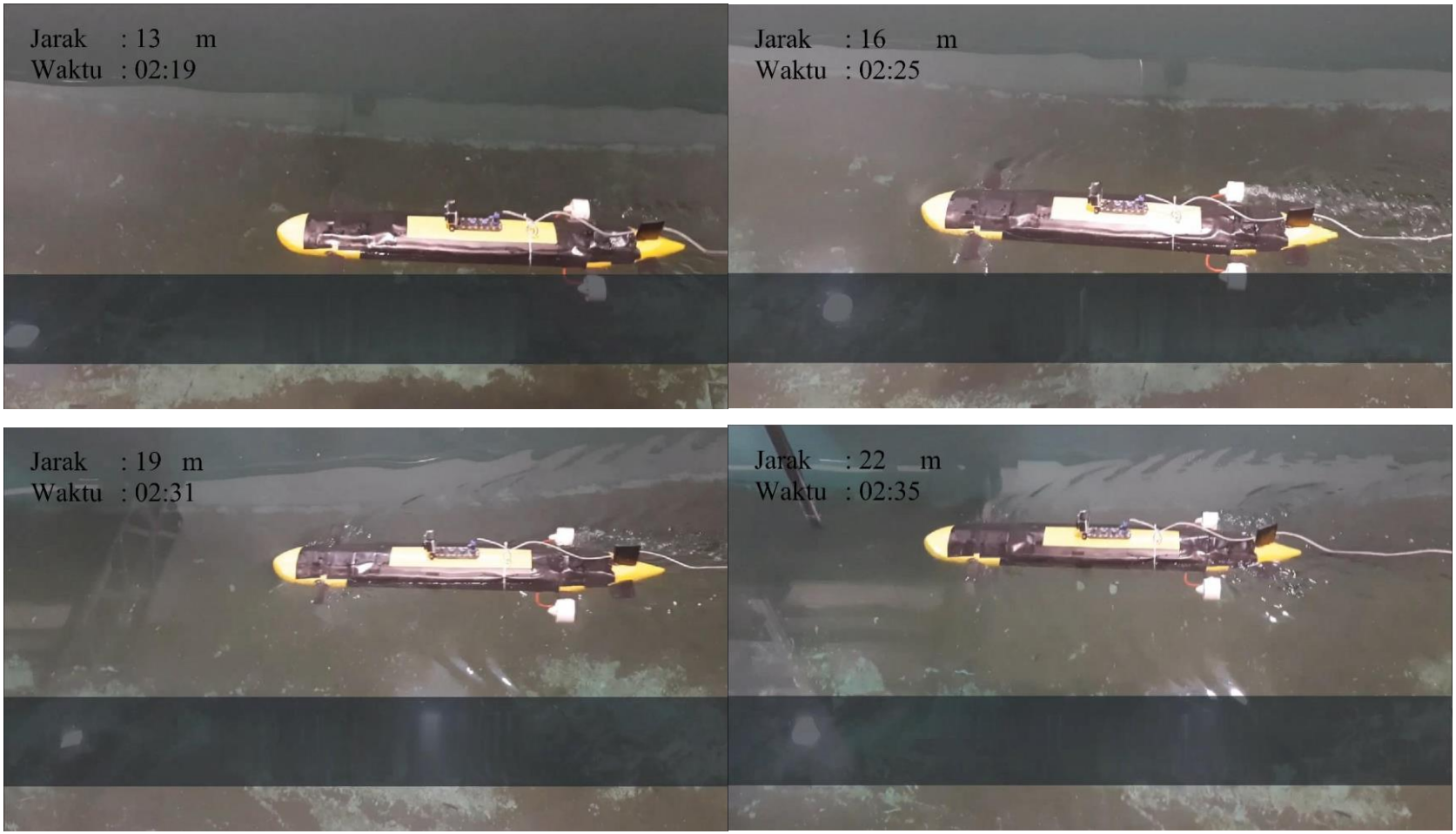
```
  // blinking led indicagor for leaking problem
  // digitalRead(pinLeak) == HIGH ? bitWrite(bitOutput, 1, 1) : bitWrite(bitOutput, 1, 0);

  // blinking led indicator for running process

  // blinking led indicator for initialization

  // bit indicator for underwater ligthting
     bitRead(bitInput,0) == 1 ? digitalWrite(pinLamp1,HIGH) : digitalWrite(pinLamp1,LOW);
     bitRead(bitInput,1) == 1 ? digitalWrite(pinLamp2,HIGH) : digitalWrite(pinLamp2,LOW);
    Mb.MbData[0] = bitOutput;
}

void counter() {
   count++;
}
```

Lampiran 7: Dokumentasi Pengujian



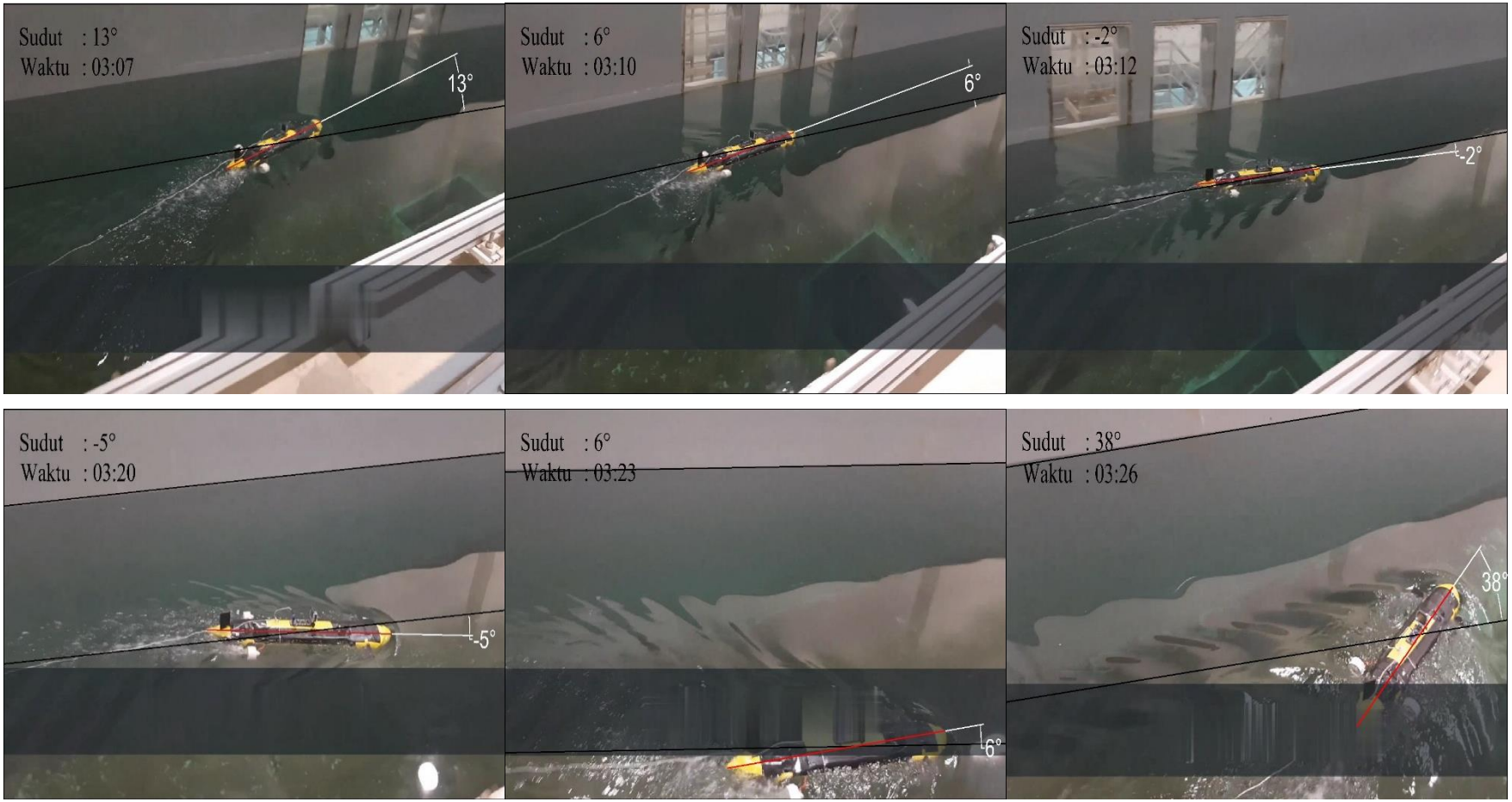


Gambar kondisi saat bergerak longitudinal

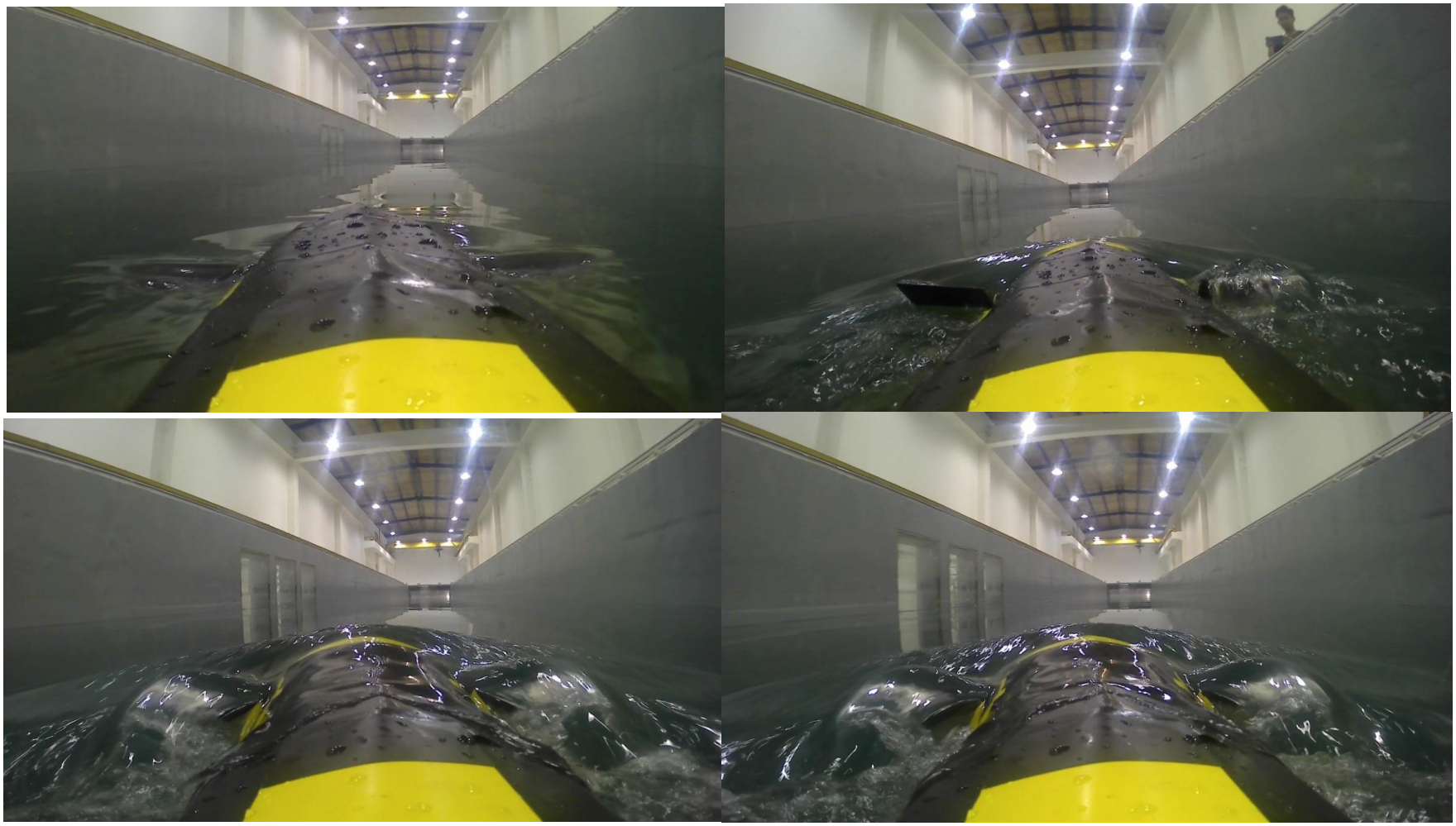


Gambar kondisi saat bergerak longitudinal

Gambar kondisi saat gerak horizontal



Gambar kondisi saat gerak horizontal



Gambar kondisi saat gerak vertikal

Lampiran 8: Biodata Diri

Data Pribadi:

Nama Lengkap : Muh. Nursyahrul Qadri

Nama Panggilan : Syahrul

Tempat/Tanggal Lahir : Takalar, 14 Februari 1996

Pekerjaan : Mahasiswa

Agama : Islam

Jenis Kelamin : Laki-laki

Gol. Darah : B

Nama Orang Tua

- Ayah : M. Syahrir S.
- Ibu : Subaedah

Pekerjaan Orang Tua

- Ayah : Pegawai Negeri Sipil
- Ibu : Ibu Rumah Tangga

Anak ke : 1 dari 3 bersaudara

Alamat saat ini : BTN Bontomate'ne B3 No. 14C

No. HP : 085242794030

Email : syahrulqadri14@gmail.com

Riwayat Pendidikan Formal

Periode	Sekolah/Institusi/Universitas	Jurusan
2002-2008	SDN Inp. 234 Takalar Kota	
2008-2011	MTs. Manongkoki	
2011-2014	SMA Neg. 1 Takalar	IPA
2015-Sekarang	Fakultas Teknik, Universitas Hasanuddin	T. Sistem Perkapalan