**Automatic Identification System Compatible Turn-Rate and Heading Instrument**

Jayce A. Jocson 1, Neil Janviere T. Rovillos 2, Davy Brian L. Sorallo 3  
*School of Electrical, Electronics, and Computer Engineering*  
Mapúa University

Muralla St., Intramuros, Manila, 1002, Philippines

1 jaycejocson2@gmail.com, 2neiljanviere.rovillos@gmail.com, 3davybrians@gmail.com

*Abstract*—The Automatic Identification System (AIS) is an implemented instrument for maritime vehicles. It plays an essential role in securing maritime traffic due to its system capability to enable vessels to exchange information about the ship’s identity, real-time position, course and speed several times a minute automatically by the VHF band radio. This paves a way to this paper which focuses on the development of the AIS sensors with NMEA Code compatibility that would be integrated and attached inside the marine vehicle. This project specifically transmits its course in the rate of turn and true heading data. Hence, for the further part of the system which is out of the research’s scope, the message is then received and processed by each nearby vessel and share these data with different clients. Moreover, the shore-based stations help in the dissemination of the rate-of-turn and true heading data along with other AIS messages.

Keywords—AIS System, Arduino Nano, NMEA, Rate-of-Turn, True Heading, Magnetic Variation

# INTRODUCTION

Most nautical transportation is ensured by a few security gadgets, for example, the advancement of oceanic reconnaissance frameworks. Nowadays, maritime vessels consider an ever-increasing extent support to navigation systems [1].

Automatic Identification System (AIS) is an automated, self-governing tracking system which is broadly utilized by any maritime vessels and stations for trading navigational data between AIS-equipped workstations. Because of it, information of the vessel, including the identity, position, speed, destination, etc., can be electronically traded between AIS-receiving stations. This design project is a data gathering instrument which is compatible with AIS systems. The data being gathered are the Rate of Turn (ROT) and Heading (HDT) of a ship.

The controller initializes upon power up. The initialization includes the sensors, serial and LCD output, and the i2c interface. The data is gathered using a sensor and is interpreted using a microcontroller. The controller then runs a repeating instruction of reading data from the sensors then formats the data to NMEA sentence. This format allows marine electronics to send information to computers and to other marine equipment. The 16x2 LCD display will show the formatted data. The accuracy of the project design is proven using OpenCPN, a free software to create a chart plotter and navigation software.

# REVIEW OF RELATED LITERATURE

The build-up network of an Automatic Identification System (AIS) yields a rich source of vessel movement information. The AIS logs which are the information of the vessels are considered as maritime trajectory data. This vast amount of AIS trajectory data can be collected and be employed to achieve an awareness of maritime trafﬁc knowledge [1]. In this manner, the idea of the Automatic Identification System is then introduced.

**APPLICATION OF THE AIS TO MARITIME VESSELS**

In reference to Captain Edward Page, new technology lays a ground that would be helpful for its provision of dynamic speed limit signs based on current traffic and environmental conditions. This is the approach applied with AIS technology. Instead of implementing prescriptive Marine Protected Areas (MPAs) published in regulations that at times are irrelevant, AIS allows the transmission of the dynamic MPAs based on current observations, environmental sensitivities, and hazards. Similarly, the information provided by Coast Guard Notice to Mariners can be better disseminated to vessels when they are approaching the area of the notice through AIS. The location of whales, local hunting parties, walrus haul-out areas, ice, and other real-time relevant information can be digitally transmitted to vessels via AIS [2].

**NATIONAL MARINE ELECTRONICS ASSOCIATION STANDARDIZATION ORGANIZATION (NMEA)**

A marine vessel just like an ordinary ship requires a complex system that would manage and monitor a large amount of data that would be used in transmission. The NMEA sentence allows individual sensors to yield a building network connectivity to transmit information that is received to NMEA information displays.

There is no change in the understanding of the provision of information and the need for a radically different approach in sequence “sensor - data - indicator – user”. The ship can be considered as a complex technical system built of multiple mechanical and electrical systems as well as a complex information system. The latter requires the construction of the corresponding information structure including a transmission system, information processing units, extraction interfaces, and provision of data to and from external users/systems [4].

**I2C SPECIFICATION**

The serial data communication protocols require a lot number of pin connections in the IC (Integrated Circuit). Hence, in able to produce a better and efficient device, the device is then decided to undergo reduction of size thus lesser number of pin connections for serial data transfer is implemented to physically cut down the size. The I2C which is an abbreviated term for Inter-Integrated Circuit is a serial data transfer bus, it is 8-bit oriented. By using two signal lines data transmission is done. The two signal lines are SCL and SDA. SCL is the Serial clock line, it is a unidirectional line and SDA is a Serial data line, it is a bidirectional line, both lines are used to transfer the data between Master and Slave. There are lots of slave devices are connected onto the I2C bus, Slave devices are identified by its unique 7-bit address. After each data transfer, there is acknowledgment bit sent by the slave to master or vice versa, to ensure the data is received [5]. In this paper, the MEMS sensors (gyroscope and electronic compass) is acting as the slave of the system.

# OBJECTIVES

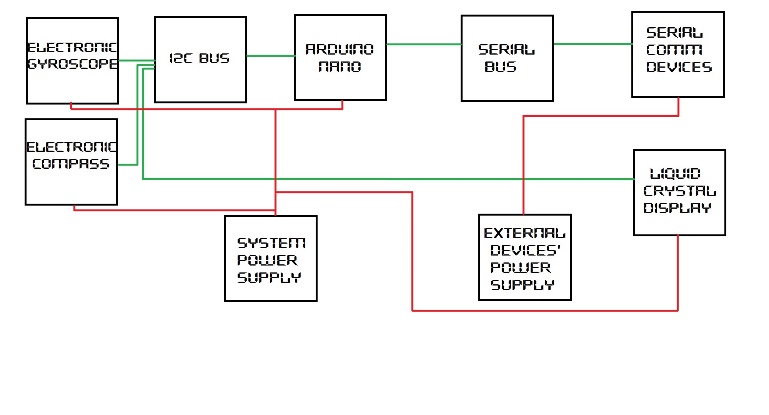
The main objective of this design is to design a project that would be integrated inside a marine vessel outputting rate-of turn and true heading.

Objectives:

a. To gather the rate of turn and the true heading of the ship using gyroscope

b. To format the data to NMEA sentence using a microcontroller

c. To display the output in a 16x2 i2c LCD display



# METHODOLOGY

## Block Diagram

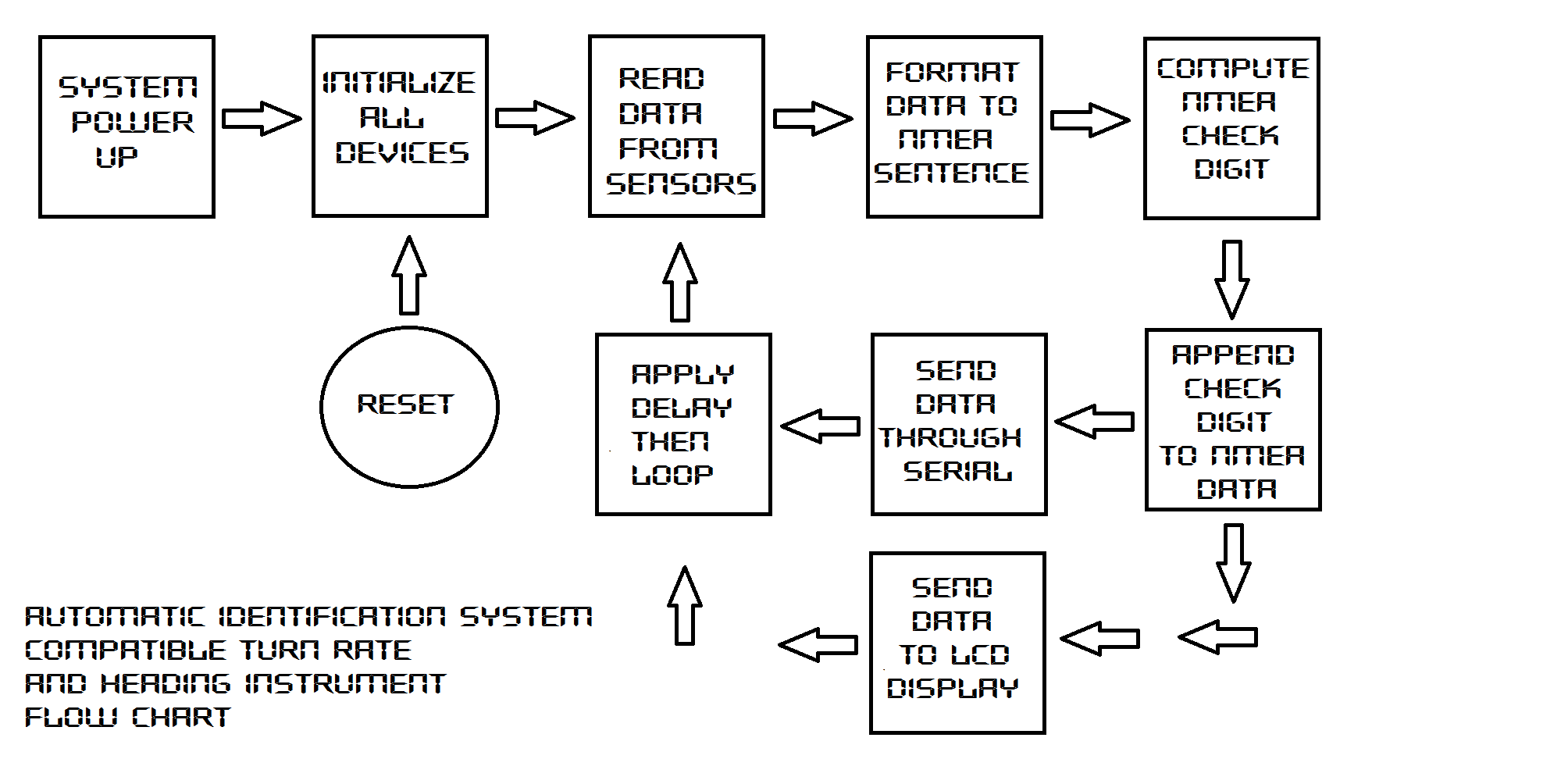
**Figure 4.1** Block Diagram

The main functional parts of the system are the electronic gyroscope, electronic compass, Arduino nano, serial communication devices, liquid crystal display, i2c bus, serial bus, system power supply, and external device power supply. External device power supply will power the external devices that are not part of the system but will communicate to the system serially. The system power supply powers all of the devices within the system like the e-gyroscope, e-compass, Arduino nano and LCD. The Arduino nano is the brain of the system which processes all incoming data and produces the output. Input devices like the e-compass and e-gyroscope and output devices like the LCD are communicating with the Arduino through i2c protocol. The input and output devices connect to the Arduino via an i2c bus which processes data before sending or receiving from the I/O device or sending or receiving from the Arduino nano. The serial bus provides a bidirectional serial data line for transmitting and receiving data and a unidirectional serial clock line for synchronization. Devices that connect through the serial peripheral will need to communicate with the serial bus built in the Arduino nano. This bus formats data in the same manner as the i2c bus but without addressing. This means that devices that connect to this bus will get the same data in reception. In sending data, the bus will receive it without knowing which device sent the data.

## Pins and Connections

**Figure 4.2** Circuit Diagram

A4 or analog 4 pin on the Arduino provides serial data line for i2c and A5 or analog pin 5 provides the serial clock line. The controller provides lines for 3.3V, 5V and ground.



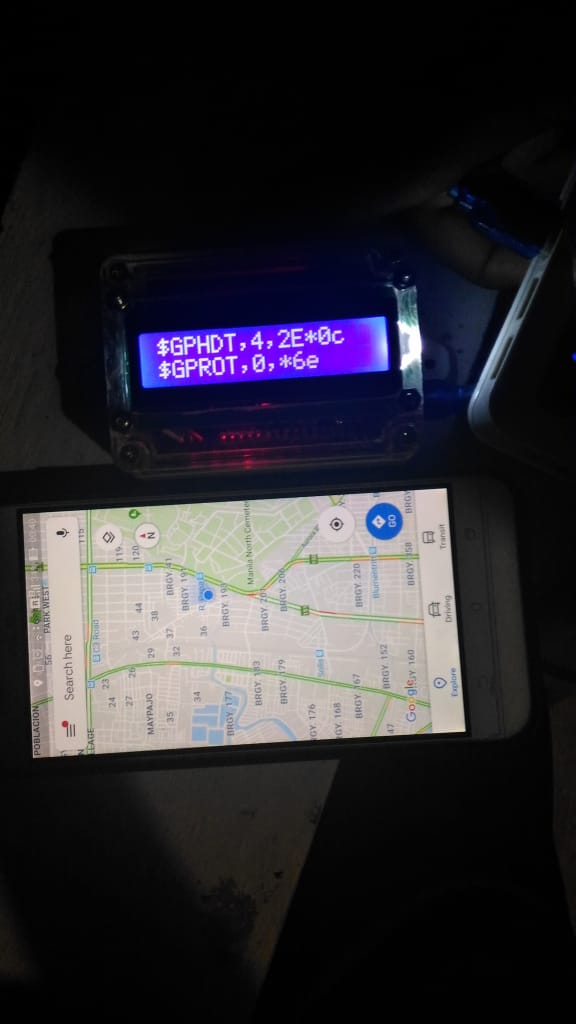
**Figure 4.3** Process Flowchart

Upon powering the system up, the Arduino will run its initialization protocol. This includes initialization of serial communication, i2c communication, input/output device initialization, and pin initializations for the Arduino. Initialization will include setting communication baud rate, resetting registers, preparing sensors for data gathering, and pin configuration settings whether pins are output or input. After initialization, the controller will request for data from the two sensors upon which the sensors will comply and send the data requested. The controller then processes the data and formats them to nmea. The check digit of the nmea data will be computed and appended to the main formatted data. This new data will then be sent to both serial output and i2c output devices. The liquid crystal display will take the data from the i2c bus and display the data. Serial devices will receive data from the serial bus and use the data for producing an output depending on what serial device is connected. After the controller sends data through output lines, it will apply a delay functions which will stop processes for a brief time in milliseconds, depending on the set value. This will ensure the stability of the system and produce consistency. After the delay function, the controller will start executing instructions again at the point where the initialization ended.

# RESULTS

**Table 5.1** Device Heading (Referenced to a compass)

|  |  |
| --- | --- |
| Direction | Heading(Degrees) |
| North | 3 |
| East | 87 |
| South | 183 |
| West | 269 |

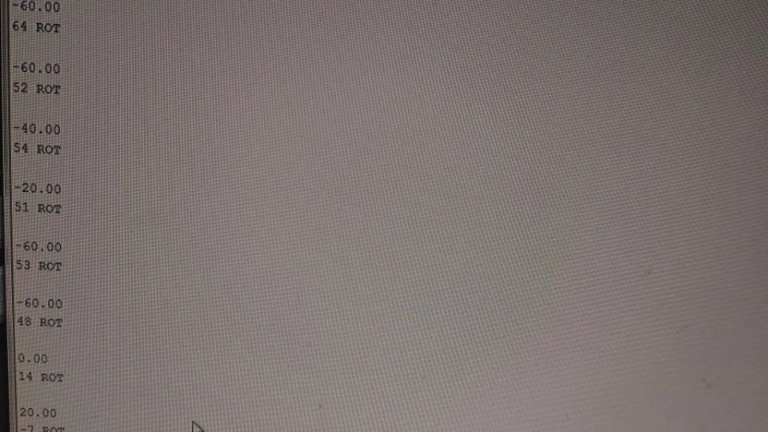


**Figure 5.1** Heading Sample Test

The sample test for the heading involves aligning the device with a compass and pointing north, east, south and west. Ideally this should produce heading values of 0, 90, 180, and 270 degrees respectively. The output heading was recorded in a table.

**Table 5.2** Gyroscope Test Relative to Compass Differential

|  |  |
| --- | --- |
| Gyroscope Rate(Deg/Sec) | Differential Rate from Compass(Deg/sec) |
| 64 | 60 |
| 52 | 60 |
| 54 | 40 |
| 53 | 60 |



**Figure 5.2** Rate of Turn Multiple Sample Test

Testing rate of turn involves aligning the system with a compass whose data can be recorded in a computer. The differential rate of the compass is taken. Rate can be computed using the difference between two samples and the time it took to between them. Both the compass and gyroscope were spun simultaneously at random directions. 5 samples were taken from the recorded data and was recorded.

VI. CONCLUSION, ASSESSMENT, AND RECOMMENDATION

The device was able to yield the data from its sensors. The rate of turn in degrees/min. relative to the vertical axis was extracted through the gyroscope and true heading in degrees was given by an electronic compass with compensation for magnetic variation. The microcontroller was able to extract all the data and then format it to NMEA sentence. The microcontroller was then able to compute for the NMEA check digit for both output data and append it. The controller was able to send the data to the serial and i2c busses to be prepared for output. Followed by the system that was to yield an output for a visual display and serial communication with two of the data formatted in NMEA sentences.

In assessing the group’s overall design, the objectives were met, and the output was displayed correctly. Materials required were also utilized and instructions of the adviser were followed accordingly. However, the turn rate can still be improved, especially for ships turning at a slower rate, and magnetic variation since it depends on the location of the ship.

Out of 10, the group’s rating is 9.

# APPENDIX

APPENDIX I: ARDUINO CODES

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <MechaQMC5883.h>

#include <string.h>

MechaQMC5883 qmc;

#define GYRO 0x68

#define G\_SMPLRT\_DIV 0x15

#define G\_DLPF\_FS 0x16

#define G\_INT\_CFG 0x17

#define G\_PWR\_MGM 0x3E

#define G\_TO\_READ 8

LiquidCrystal\_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

int indexH, indexR;

char incharH, incharR;

char HEADINGtemp[80], ROTtemp[80];

byte start\_withH = 0, start\_withR = 0;

byte end\_withH = 0, end\_withR = 0;

byte CRCH = 0, CRCR;

int g\_offx = 120;

int g\_offy = 20;

int g\_offz = 15;

int hx, hy, hz, turetemp;

void initGyro()

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* ITG 3200

\* power management set to:

\* clock select = internal oscillator

\* no reset, no sleep mode

\* no standby mode

\* sample rate to = 125Hz

\* parameter to +/- 2000 degrees/sec

\* low pass filter = 5Hz

\* no interrupt

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

writeTo(GYRO, G\_PWR\_MGM, 0x00);

writeTo(GYRO, G\_SMPLRT\_DIV, 0x07);

writeTo(GYRO, G\_DLPF\_FS, 0x1E);

writeTo(GYRO, G\_INT\_CFG, 0x00);

}

void getGyroscopeData(int \* result)

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Gyro ITG-3200 I2C

registers:

temp MSB = 1B, temp LSB = 1C

x axis MSB = 1D, x axis LSB = 1E

y axis MSB = 1F, y axis LSB = 20

z axis MSB = 21, z axis LSB = 22

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int regAddress = 0x1B;

int temp, x, y, z;

byte buff[G\_TO\_READ];

readFrom(GYRO, regAddress, G\_TO\_READ, buff);

result[0] = ((buff[2] << 8) | buff[3]) + g\_offx;

result[1] = ((buff[4] << 8) | buff[5]) + g\_offy;

result[2] = ((buff[6] << 8) | buff[7]) + g\_offz;

result[3] = (buff[0] << 8) | buff[1];

}

//

void setup() {

Wire.begin();

Serial.begin(9600);

qmc.init();

initGyro();

lcd.begin(16,2);

lcd.backlight();

}

void loop(){

String HEADING, ROT, STATUS;

int x, y, z, COMP;

String VAR;

int azimuth;

byte addr;

int gyro[4];

getGyroscopeData(gyro);

hx = gyro[0] / 14.375;

hy = gyro[1] / 14.375;

hz = gyro[2] / 14.375;

turetemp = 35+ ((double) (gyro[3] + 13200)) / 280;

qmc.read(&x, &y, &z,&azimuth);

if(azimuth>=0 && azimuth< 30)

{ COMP = 2;

VAR="E";

}

else if(azimuth >= 30 &&azimuth<60)

{

COMP = 3;

VAR="E";

}

else if(azimuth >= 60 &&azimuth<90)

{

COMP = 4;

VAR="E";

}

else if(azimuth >= 90 &&azimuth<120)

{ COMP = 2;

VAR ="E";

}

else if(azimuth >= 120 &&azimuth<150)

{

}

for (byte x = start\_withH + 1; x < end\_withH; x++)

{

CRCH = CRCH ^ HEADINGtemp[x];

}

//ROT

ROT.toCharArray(ROTtemp, 80);

for ( indexR = 0; indexR < 80; indexR++) {

incharR = ROTtemp[indexR];

if ( incharR == '$') {

start\_withR = indexR;

}

if (incharR == '\*') {

end\_withR = indexR;

}

}

for (byte x = start\_withR + 1; x < end\_withR; x++)

{

CRCR = CRCR^ ROTtemp[x];

}

lcd.setCursor(0,0);

if(CRCH < 16)

lcd.print(HEADING +"0" + String(CRCH, HEX));

else

lcd.print(HEADING + String(CRCH, HEX));

lcd.setCursor(0,1);

lcd.print(ROT + String(CRCR, HEX));

if(CRCH < 16)

{Serial.println(HEADING +"0" + String(CRCH, HEX));}

else

Serial.println(HEADING + String(CRCH, HEX));

Serial.println(ROT + String(CRCR, HEX));

CRCR = 0;

CRCH = 0;

delay(750);

lcd.clear();

}

void writeTo(int DEVICE, byte address, byte val) {

Wire.beginTransmission(DEVICE);

Wire.write(address);

Wire.write(val);

Wire.endTransmission();

}

void readFrom(int DEVICE, byte address, int num, byte buff[]) {

Wire.beginTransmission(DEVICE);

Wire.write(address);

Wire.endTransmission();

COMP = 1;

VAR="E";

}

else if(azimuth >= 150 &&azimuth<180)

{

COMP = 1;

VAR="W";

}

else if(azimuth >= 180 &&azimuth<210)

{

COMP = 2;

VAR="W";

}

else if(azimuth >= 210 &&azimuth<240)

{

COMP = 3.5;

VAR="W";

}

else if(azimuth >= 240 &&azimuth<270)

{

COMP = 3;

VAR="W";

}

else if(azimuth >= 270 &&azimuth<300)

{

COMP = 1.5;

VAR="W";

}

else if(azimuth >= 300 &&azimuth<330)

{

COMP = 0;

VAR="W";

}

else if(azimuth >= 330 &&azimuth<=360)

{

COMP = 1.5;

VAR="E";

}

HEADING = "$GPHDT," + (String)azimuth +","+ (String)COMP + (VAR)+ "\*";

ROT= "$GPROT," + String(hz\*60)+"," + STATUS +"\*";

//HEADING

HEADING.toCharArray(HEADINGtemp, 80);

for ( indexH = 0; indexH < 80; indexH++) {

incharH = HEADINGtemp[indexH];

if ( incharH == '$') {

start\_withH = indexH;

}

if (incharH == '\*') {

end\_withH = indexH;

}

Wire.beginTransmission(DEVICE);

Wire.requestFrom(DEVICE, num);

int i = 0;

while(Wire.available())

{

buff[i] = Wire.read();

i++;

}

Wire.endTransmission();

}

# REFERENCES

[1] Ronan Kerbiriou, Laurent Lévêque, Aboozar Rajabi, Arnaud Serry, “The Automatic Identification System (AIS) as a Data Source for Studying Maritime Traffic”, 2017,p.1

[2] *P.* Lei, T. Tsai and W. Peng, "Discovering Maritime Traffic Route from AIS network," *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Kanazawa, 2016, pp. 1-6.

[3] C. E. Page, "Maximizing maritime safety and environmental protection with AIS: (Automatic identification system)," *OCEANS 2017 - Anchorage*, Anchorage, AK, 2017, pp. 1-4.

[4] *Y. Sivkov, "Transformation of NMEA ship network from sensor-based to information-based model,"*2018 20th International Symposium on Electrical *Apparatus and Technologies (SIELA)*, Bourgas, 2018, pp. 1-4.

[5] R. S. S. Kumari and C. Gayathri, "Interfacing of MEMS motion sensor with FPGA using I2C protocol," *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, Coimbatore, 2017, pp. 1-5.