

IoT Inertial Measurement Unit Monitoring

จัดทำโดย

นายธนวินท์ ศิริวรรณ 6203014610060

เสนอ

ผู้ช่วยศาสตราจารย์ ดร.สุพจน์ แก้วกรณ์

รายงานเล่มนี้เป็นส่วนหนึ่งของวิชาการโปรแกรมคอมพิวเตอร์สำหรับงานควบคุม

สาขาวิชาเทคโนโลยีวิศวกรรมแมคคาทรอนิกส์

ภาควิชาเทคโนโลยีวิศวกรรมเครื่องกล

วิทยาลัยเทคโนโลยีอุตสาหกรรม

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

พ.ศ.2565

**คำนำ**

รายงานเล่มนี้เป็นส่วนหนึ่งของวิชา 030143361 Computer Programming For Control โดยจัดทำขึ้นโดยเพื่อศึกษาการเขียนโปรแกรมคอมพิวเตอร์สำหรับการควบคุมโดยใช้โปรแกรม Delphi ซึ่งรายงานเล่มนี้มีเนื้อหาเกี่ยวกับการเขียนโปรแกรมติดต่อรับส่งข้อมูลกับคอนโทลเลอร์แสดงผล และบันทึกผลการทำงาน

ผู้จัดทำได้ทำรายงานเล่มนี้ขึ้นเพื่อแสดงการใช้โปรแกรม Delphi ในการเขียนโปรแกรมติดต่อสื่อสาร และแสดงผล โดยใช้ Arduino และคอมพิวเตอร์ร่วมกัน ผู้จัดทำหวังว่ารายงานเล่มนี้จะมีประโยชน์แก่ผู้ที่เข้ามาศึกษา หากรายงานเล่มนี้มีข้อผิดพลาดประการใดทางผู้จัดทำขออภัยมา ณ ที่นี้ด้วย

ผู้จัดทำ

ธนวินท์ ศิริวรรณ

**1. ที่มาและความสำคัญ**

เพื่อพัฒนาระบบแสดงสถานการทำงานของเซนเซอร์ GY-91 ที่เชื่อมต่อ Arduino และสื่อสารรับส่งข้อมูลกับคอมพิวเตอร์ด้วย TCP/IP โดยสามารถนำข้อมูลที่ได้จาก Arduino ที่ต่อกับเซนเซอร์ GY-91 (IMU) นำมาพล็อตกราฟ Acceleration, Gyro, Magnetometer, Temperature และบันทึกค่าไปยังฐานข้อมูล MQTT สามารถบันทึกเป็น Text ไฟล์ และสามารถแสดงสถานะการทำงานผ่าน Node-RED

**2. วัตถุประสงค์**

2.1 เพื่อศึกษาการเขียน VCL/FMX Application ด้วย Delphi

2.2 เพื่อศึกษาการส่งข้อมูลสื่อสารกับ Arduino ผ่าน TCP/IP ด้วย Delphi

2.3 เพื่อออกแบบระบบ IoT Inertia Measurement Unit Monitoring

**3. ขอบเขตการทำงาน**

3.1 สามารถรับส่งข้อมูลไปยัง Arduino ผ่าน TCP/IP

3.2 สามารถทำการแสดงค่าสัญญาณของ GY-91

3.3 สามารถส่งข้อมูลไปเก็บที่ฐานข้อมูล

3.4 สามารถบันทึกค่าที่อ่านจากเซนเซอร์ลง Text ไฟล์

3.5 สามารถควบคุมการเคลื่อนที่วัตถุ 3 มิติ

3.6. สามารถส่งข้อมูลไปยัง MQTT

3.7 สามารถทำการแสดงสถานการณ์ทำงานของระบบด้วย Node-RED

**4. ผลที่คาดว่าจะได้รับ**

4.1 ระบบสามารถรับส่งข้อมูลกับ Arduino ผ่าน TCP/IP ได้

4.2 ระบบสามารถส่งบันทึกข้อมูลไปยังฐานข้อมูลได้

4.3 ระบบสามารถส่งบันทึกข้อมูลในรูปแบบ Text ไฟล์ได้

4.4 ระบบสามารถรับส่งข้อมูลกับ MQTT ได้

**5. Block Diagram แสดงขั้นตอนการทำงาน**

Diagram

Description automatically generated

**6. การเชื่อมต่อ**

6.1 Arduino เชื่อมต่อกับโมดูล GY-19 ด้วย I2C

6.2 Arduino เชื่อมต่อ Arduino Ethernet Shield ด้วย SPI

6.3 คอมพิวเตอร์สื่อสารกับ Arduino ผ่าน TCP/IP โดย Arduino จะเป็น Server และคอมพิวเตอร์เป็น Client

**7. การดำเนินงาน**

7.1 ออกแบบระบบ

7.2 การเขียนโปรแกรมให้ Arduino อ่านค่าจาก GY-91

7.3 การเขียนโปรแกรมให้ Arduino เชื่อมต่อกับอินเทอร์เน็ต

7.4 การเขียนโปรแกรม Delphi สื่อสารกับ Arduino ผ่าน TCP/IP

7.5 การเขียนโปรแกรม Delphi ส่งข้อมูลไปยังฐานข้อมูล

7.6 การเขียนโปรแรกม Delphi ส่งและรับข้อมูลจาก MQTT

7.7 การเขียนโปรแกรมบันทึกค่าจาก GY-91 ลง text ไฟล์

7.8 การเขียนโปรแกรมควบคุมวัตถุ 3 มิติ

7.9 การเขียนโปรแกรม Delphi พล็อตกราฟ

7.10 การเขียนโปรแกรม Node-RED รับค่าจาก MQTT

**8. การออกแบบ Interface**

8.1 หน้าหลัก

A picture containing graphical user interface

Description automatically generated

8.2 การพล็อตกราฟ Acceleration X, Y, Z

Timeline

Description automatically generated

8.3 การพล็อตกราฟ Acceleration Norm

Chart

Description automatically generated

8.4 การพล็อตกราฟ Gyro X,Y,Z

Diagram

Description automatically generated

8.5 การพล็อตกราฟ Magnetometer X,Y,Z Axis

A picture containing text, black, mounted, silver

Description automatically generated

8.6 การพล็อตกราฟ Magnetometer Horizontal

Chart

Description automatically generated

8.7 การพล็อตกราฟ Temperature and pressure

Timeline

Description automatically generated

8.8 MySQL

Graphical user interface, application, table, Excel

Description automatically generated

8.9 Raw Data

Graphical user interface, application

Description automatically generated

8.10 MQTT

Graphical user interface, application, Word

Description automatically generated

**9. โปรแกรม Delphi**

unit Main;

interface

uses

System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,

FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.Edit,

FMX.Controls.Presentation, FMX.StdCtrls, IdBaseComponent, IdComponent,

IdTCPConnection, IdTCPClient, FMX.Memo.Types, FMX.ScrollBox, FMX.Memo,

FMX.TabControl, FMXTee.Engine, FMXTee.Procs, FMXTee.Chart,

System.Math.Vectors, FMX.Types3D, FMX.Objects3D, FMX.Controls3D,

FMX.Viewport3D, FMXTee.Series, FireDAC.Phys.FBDef, FireDAC.Stan.Intf,

FireDAC.Stan.Option, FireDAC.Stan.Error, FireDAC.UI.Intf, FireDAC.Phys.Intf,

FireDAC.Stan.Def, FireDAC.Stan.Pool, FireDAC.Stan.Async, FireDAC.Phys,

FireDAC.FMXUI.Wait, FireDAC.Stan.Param, FireDAC.DatS, FireDAC.DApt.Intf,

FireDAC.DApt, Data.DB, FireDAC.Comp.DataSet, FireDAC.Comp.Client,

FireDAC.Phys.IBBase, FireDAC.Phys.FB, FireDAC.Phys.MySQL,

FireDAC.Phys.MySQLDef, System.Rtti, FMX.Grid.Style, FMX.Grid,

Data.Bind.EngExt, Fmx.Bind.DBEngExt, Fmx.Bind.Grid, System.Bindings.Outputs,

Fmx.Bind.Editors, Data.Bind.Components, Data.Bind.Grid, Data.Bind.DBScope,

FMXTee.Control, FMXTee.Grid, FMX.TMSBaseControl, FMX.TMSGridCell,

FMX.TMSGridOptions, FMX.TMSGridData, FMX.TMSCustomGrid, FMX.TMSGrid,

FMX.Layouts, FMX.ListBox, TMS.MQTT.Global, TMS.MQTT.Client, Math, IniFiles,

FMX.WebBrowser, FMX.TMSWebBrowser, FMX.TMSTaskDialog;

const

GRAPH\_MAX\_NUM = 200;

LP\_K = 0.3;

HP\_K = 0.01;

SAMPLING\_TIME = 0.01; // 10mS.

MOVIVE\_AVG\_LEN = 10; // 10mS. \* 100 = 1000mS.

ACC\_NORM\_VAL = 101;

type

TClientRead = class(TThread)

private

protected

procedure Execute; override;

end;

type

TModel = class(TThread)

private

protected

procedure Execute; override;

end;

type

TMyThreadSQL = class(TThread)

private

LastData:String;

procedure DataToSQL(aX,aY,aZ,aN,gR,gP,gY,mgX,mgY,mgZ,mgH,temp,press,Alti:Single);

procedure ShowDataSQL();

protected

procedure Execute; override;

end;

type

Tmy3d = class(TThread)

private

procedure callSig();

protected

procedure Execute; override;

end;

type

TMyPlot = class(TThread)

private

protected

procedure Execute; override;

end;

type

TrecData = class(TThread)

private

protected

procedure Execute; override;

end;

type

TMoveAvg = record

Data:Array [1..10] of Single;

end;

type

TLowPass = record

LastLp:Single;

LastHp:Single;

end;

type

THighPass = record

LastHp:Single;

LastDataIn:Single;

end;

Type

tAcc = record

X:Single;

Y:Single;

Z:Single;

end;

type

TaccData = record

aX:String;

aY:String;

aZ:String;

rq:String;

end;

type

TgyroData = record

gyroX:String;

gyroY:String;

gyroZ:String;

end;

type

TmagData = record

magX:String;

magY:String;

magZ:String;

magH:String;

end;

type

TtempData = record

TempX:String;

PressX:String;

end;

type

Tsignal = record

accX:Single;

accY:Single;

accZ:Single;

accNorm:Single;

gyroR:Single;

gyroP:Single;

gyroY:Single;

end;

type

TsignalLP = record

accX:Single;

accY:Single;

accZ:Single;

accNorm:Single;

gyroR:Single;

gyroP:Single;

gyroY:Single;

end;

type

TMydata = record

accX:Single;

accY:Single;

accZ:Single;

accNorm:Single;

gyroR:Single;

gyroP:Single;

gyroY:Single;

magX:Single;

magY:Single;

magZ:Single;

end;

type

TGyro = record

X:Integer;

Y:Integer;

Z:Integer;

Norm:Single;

end;

type

TEmailDetail = record

User, Pass, FromAddr, ToAddr, Subject, Body, Info, FilePath:String;

end;

type

TMovingAvgBuffer = record

Data:array[1..MOVIVE\_AVG\_LEN] of Integer;

end;

type

TGyroForce = record

FX: Single;

FY: Single;

FZ: Single;

Fr:Single;

end;

type

TCalData = record

OldData: Single;

NewData: Single;

end;

type

TImuAngle = record

AngX: TCalData;

AngY: TCalData;

AngZ: TCalData;

end;

type

TForm1 = class(TForm)

IdTCPClient1: TIdTCPClient;

Z: TPanel;

Edit1: TEdit;

Edit2: TEdit;

Connect: TButton;

Disconnect: TButton;

Edit3: TEdit;

Stop: TButton;

Start: TButton;

TabControl1: TTabControl;

MonitoringData: TTabItem;

TableData: TTabItem;

Acc\_Data: TTabItem;

Acc\_Sqrt: TTabItem;

Gyro\_Data: TTabItem;

Meg\_Data: TTabItem;

MegDirection: TTabItem;

Temp\_Press: TTabItem;

RawData: TTabItem;

Memo1: TMemo;

Chart1: TChart;

Chart2: TChart;

Chart3: TChart;

Chart4: TChart;

Chart5: TChart;

Chart6: TChart;

AccX: TLineSeries;

GyroX: TLineSeries;

AccY: TLineSeries;

GyroY: TLineSeries;

AccZ: TLineSeries;

GyroZ: TLineSeries;

Viewport3D1: TViewport3D;

Camera1: TCamera;

Light1: TLight;

Model3D1: TModel3D;

FDPhysFBDriverLink1: TFDPhysFBDriverLink;

FDConnection1: TFDConnection;

FDTable1: TFDTable;

DataSource1: TDataSource;

FDTable1DateTime: TDateTimeField;

FDTable1accelerationX: TSingleField;

FDTable1accelerationY: TSingleField;

FDTable1accelerationZ: TSingleField;

FDTable1accelerationNorm: TSingleField;

FDTable1GyroX: TSingleField;

FDTable1GyroY: TSingleField;

FDTable1GyroZ: TSingleField;

FDTable1MagnetoX: TSingleField;

FDTable1MagnetoY: TSingleField;

FDTable1MagnetoZ: TSingleField;

FDTable1magHorizDirection: TSingleField;

FDTable1Temperature: TSingleField;

FDTable1Pressure: TSingleField;

FDTable1Altitude: TSingleField;

BindSourceDB1: TBindSourceDB;

BindingsList1: TBindingsList;

StringGrid1: TStringGrid;

LinkGridToDataSourceBindSourceDB12: TLinkGridToDataSource;

Chart7: TChart;

Chart8: TChart;

Chart9: TChart;

AccXD: TLineSeries;

AccYD: TLineSeries;

AccZD: TLineSeries;

Chart10: TChart;

Chart11: TChart;

Chart12: TChart;

Chart13: TChart;

Chart14: TChart;

AccSQRTD: TLineSeries;

GyroXD: TLineSeries;

GyroYD: TLineSeries;

GyroZD: TLineSeries;

Chart15: TChart;

Chart16: TChart;

Chart17: TChart;

Chart18: TChart;

Chart19: TChart;

magX: TLineSeries;

MagY: TLineSeries;

MagZ: TLineSeries;

MagH: TLineSeries;

TempS: TLineSeries;

PressS: TLineSeries;

Timer1: TTimer;

MqttData: TTabItem;

TMSMQTTClient1: TTMSMQTTClient;

SaveData: TButton;

WedNoedRed: TTabItem;

Label1: TLabel;

ListBox1: TListBox;

Memo2: TMemo;

TabItem3: TTabItem;

EditWeb: TEdit;

TMSFMXWebBrowser1: TTMSFMXWebBrowser;

ReFresh: TButton;

StopSave: TButton;

StopSQL: TButton;

TMSFMXTaskDialog1: TTMSFMXTaskDialog;

CheckBox1: TCheckBox;

Button1: TButton;

StartSQL: TButton;

SQLStat: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

procedure ConnectClick(Sender: TObject);

procedure DisconnectClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure TestClick(Sender: TObject);

procedure StopClick(Sender: TObject);

procedure StartClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure Timer1Timer(Sender: TObject);

procedure TMSMQTTClient1ConnectedStatusChanged(ASender: TObject;

const AConnected: Boolean; AStatus: TTMSMQTTConnectionStatus);

procedure TMSMQTTClient1PublishReceived(ASender: TObject; APacketID: Word;

ATopic: string; APayload: TArray<System.Byte>);

procedure SaveDataClick(Sender: TObject);

procedure FormKeyDown(Sender: TObject; var Key: Word; var KeyChar: Char;

Shift: TShiftState);

procedure ReFreshClick(Sender: TObject);

procedure IdTCPClient1Connected(Sender: TObject);

procedure IdTCPClient1Disconnected(Sender: TObject);

procedure StopSaveClick(Sender: TObject);

procedure StartSQLClick(Sender: TObject);

procedure StopSQLClick(Sender: TObject);

procedure FormDestroy(Sender: TObject);

procedure CheckBox1Change(Sender: TObject);

procedure IdTCPClient1Status(ASender: TObject; const AStatus: TIdStatus;

const AStatusText: string);

private

{ Private declarations }

accData:TaccData;

gyroData:TgyroData;

magData:TmagData;

MoveAvg:TMoveAvg;

LowPass:TLowPass;

HighPass:THighPass;

tempData:TtempData;

Accel:TAcc;

Signal:Tsignal;

signalLP:TsignalLP;

MyData:TMyData;

MyDataFile:TextFile;

MyDataPath:String;

myEmailDetail:TEmailDetail;

MyCleint:TClientRead;

MyModel:TModel;

ThreadSQL:TMYThreadSQL;

ThreadGrid:TGrid;

ThreadPlot:TMyPlot;

MyImu:array[1..100] of TMydata;

MyRec:TrecData;

My3d:TMy3d;

GyroAng:TImuAngle;

iniFilePath:String;

procedure ReadIniCfg();

procedure WriteIniCfg();

procedure CreateCleint();

procedure PreModel();

procedure PlotAcc();

procedure PlotGyro();

procedure PlotAccD();

procedure PlotGyroD();

procedure PlotNorm();

procedure PlotMag();

procedure PlotMagH();

procedure Teamp\_Press();

procedure SumPlot();

procedure PlotLineGraph(Graph: TLineSeries; Data:Real);

procedure SetZero();

procedure startGrid();

procedure DataToSQL(aX,aY,aZ,aN,gR,gP,gY,mgX,mgY,mgZ,mgH,temp,press,Alti:Single);

procedure ShowDataSQl();

procedure ThreadSqlStart();

procedure EndThreadSQL();

procedure Thread3dStart();

procedure EndThread3d();

procedure ThreadPlotStart();

procedure EndThreadPlot();

procedure AddMessage(AMessage: string; AlignRight: boolean);

procedure RunMqtt();

procedure CallMqtt();

procedure recMyData();

procedure recDataStart();

procedure recDataStop();

function FloatToStr2(Data: Single): String;

procedure StartClent();

procedure StopClient();

procedure ThreadModelStop();

//-------------Cal Data-----------------------------//

function LpFilter(k, OldVal, InputVal:Single):Single;

function HpFilter(k, OldVal, InputVal:Single):Single;

function CalMovingAvg(DataIn:Integer; var DataBuff:TMovingAvgBuffer):Integer;

function CallGyroForce(NewData:TGyro):TGyroForce;

function ChkNoMove(OldVal, NewVal:Integer):Boolean;

function ChkNoMove2(AccNorm:Single):Boolean;

procedure AddLog(AMessage: string);

procedure callini();

procedure reCheck();

public

{ Public declarations }

end;

var

Form1: TForm1;

lastData:String;

ImuData:String;

Norm:String;

AccNoMoveCount:Integer;

implementation

const

ChatChannel = 'sensor1';

{$R \*.fmx}

procedure TForm1.Teamp\_Press; // Plot Temp

begin

PlotLineGraph(TempS,StrToFloat(tempData.TempX));

PlotLineGraph(PressS,StrToFloat(tempData.PressX));

end;

procedure TForm1.TestClick(Sender: TObject);

begin

recDataStart;

end;

procedure TForm1.Thread3dStart;

begin

if My3d = nil then

begin

My3D := TMy3d.Create(False);

end;

end;

procedure TForm1.ThreadModelStop;

begin

if MyModel <> nil then

begin

MyModel.Terminate;

MyModel.Free;

MyModel := nil;

end;

end;

procedure TForm1.ThreadPlotStart;

begin

if ThreadPlot = nil then

ThreadPlot := TMyPlot.Create(False);

end;

procedure TForm1.ThreadSqlStart;

begin

if ThreadSQL = nil then

ThreadSQL := TMyThreadSQL.Create(False);

end;

procedure TForm1.Timer1Timer(Sender: TObject);

var

Sl:TStringList;

Data:String;

i:Integer;

begin

Application.ProcessMessages;

RunMqtt;

Memo2.Lines.Add(','+accData.aX+','+accData.aY+','+accData.aZ+','+gyroData.gyroX+','+gyroData.gyroY+','+gyroData.gyroZ);

end;

procedure TForm1.TMSMQTTClient1ConnectedStatusChanged(ASender: TObject;

const AConnected: Boolean; AStatus: TTMSMQTTConnectionStatus);

begin

if AConnected then

begin

TMSMQTTClient1.Subscribe(ChatChannel);

end

else begin

case AStatus of

csConnectionRejected\_InvalidProtocolVersion,

csConnectionRejected\_InvalidIdentifier,

csConnectionRejected\_ServerUnavailable,

csConnectionRejected\_InvalidCredentials,

csConnectionRejected\_ClientNotAuthorized:; // the connection is rejected by broker

csConnectionLost:; // the connection with the broker is lost

csConnecting:; // The client is trying to connect to the broker

csReconnecting:; // The client is trying to reconnect to the broker

end;

end;

end;

procedure TForm1.TMSMQTTClient1PublishReceived(ASender: TObject;

APacketID: Word; ATopic: string; APayload: TArray<System.Byte>);

var

msg,orig: string;

vp: integer;

alright: boolean;

begin

msg := TEncoding.UTF8.GetString(APayload);

vp := pos('!', msg);

if vp > 0 then

begin

orig := copy(msg,1,vp-1);

alright := orig <> TMSMQTTClient1.ClientID;

msg := copy(msg, vp + 1, Length(msg));

AddMessage(msg, alright);

end;

end;

procedure TForm1.WriteIniCfg;

var

vIni:TIniFile;

begin

vIni:=TIniFile.Create(iniFilePath);

try

vIni.WriteBool(CheckBox1.ClassName, 'CheckBox1.Checked', CheckBox1.IsChecked);

finally

vIni.Free;

end;

end;

procedure TForm1.StartSQLClick(Sender: TObject);

begin

if Start.Enabled=True then

begin

ShowMessage('Please Enable Start');

end

else

begin

ThreadSqlStart;

StartSQL.Enabled:=False;

StopSQL.Enabled:=True;

end;

end;

procedure TForm1.StopClick(Sender: TObject);

begin

StopClient;

EndThreadSQL;

//EndThread3d;

Timer1.Enabled:=False;

Start.Enabled:=True;

Stop.Enabled:=False;

if CheckBox1.IsChecked then

begin

EndThreadSQL;

end;

end;

procedure TForm1.StopClient;

begin

if MyCleint <> nil then

begin

MyCleint.Terminate;

MyCleint.Free;

MyCleint:=nil;

end;

end;

procedure TForm1.StopSaveClick(Sender: TObject);

begin

recDataStop;

StopSave.Enabled:=False;

SaveData.Enabled:=True;

end;

procedure TForm1.StopSQLClick(Sender: TObject);

begin

StopSQL.Enabled:=False;

EndThreadSQL;

StartSQL.Enabled:=true;

SQLStat.Text:='SQL is Stop';

end;

procedure TForm1.SumPlot; // All Plot Data

begin

PlotAccD();

PlotGyroD();

PlotNorm();

PlotMag();

PlotMagH();

Teamp\_Press();

end;

procedure TForm1.SaveDataClick(Sender: TObject);

var

Buff:String;

I,x:integer;

S:String;

begin

if Start.Enabled=True then

begin

ShowMessage('Please Enable Start');

end

else

begin

recDataStart;

SaveData.Enabled:=False;

StopSave.Enabled:=True;

end;

end;

procedure TForm1.SetZero;

begin

ImuData:='0';

accData.aX:='0';

accData.aY:='0';

accData.aZ:='0';

accData.rq:='0';

gyroData.gyroX:='0';

gyroData.gyroY:='0';

gyroData.gyroZ:='0';

magData.magX:='0';

magData.magY:='0';

magData.magZ:='0';

magData.magH:='0';

tempData.TempX:='0';

tempData.PressX:='0';

Norm := '0';

lastData:='0';

MyData.accX:=0;

MyData.accY:=0;

MyData.accZ:=0;

MyData.accNorm:=0;

Mydata.gyroR:=0;

Mydata.gyroP:=0;

Mydata.gyroY:=0;

end;

procedure TForm1.ShowDataSQl;

begin

DataToSQL(StrToFloat(Form1.accData.aX),StrToFloat(Form1.accData.aY),StrToFloat(Form1.accData.aZ),StrToFloat(Form1.accData.rq),

StrToFloat(Form1.gyroData.gyroX),StrToFloat(Form1.gyroData.gyroY),StrToFloat(Form1.gyroData.gyroZ),

StrToFloat(Form1.magData.magX),StrToFloat(Form1.magData.magY),StrToFloat(Form1.magData.magZ),StrToFloat(Form1.magData.magH),

StrToFloat(Form1.tempData.TempX),StrToFloat(Form1.tempData.PressX),StrToFloat(lastData));

end;

procedure TForm1.StartClent;

begin

if MyCleint = nil then

MyCleint := TClientRead.create(false);

end;

procedure TForm1.StartClick(Sender: TObject);

begin

if IdTCPClient1.Connected=False then

begin

ShowMessage('Please Connect The Sever');

end

else

begin

Start.Enabled:=False;

Stop.Enabled:=True;

StartClent;

//ThreadSqlStart;

Timer1.Enabled:=True;

//Thread3dStart;

if CheckBox1.IsChecked then

begin

ThreadSqlStart;

end

else

begin

EndThreadSql;

end;

end;

end;

procedure TForm1.startGrid;

begin

end;

procedure TForm1.AddLog(AMessage: string);

begin

if Memo1.Lines.Count > 100 then

begin

Memo1.Lines.Clear;

Application.ProcessMessages; //without this, scrollbars will show wrong size/position

end;

Memo1.Lines.Add(AMessage);

Memo1.GoToTextEnd;

end;

procedure TForm1.AddMessage(AMessage: string; AlignRight: boolean);

var

li: Tlistboxitem;

begin

li := Tlistboxitem.Create(self);

li.StyledSettings := li.StyledSettings - [TStyledSetting.Other];

li.Text := AMessage;

li.Height := 22;

li.VertTextAlign := TTextAlign.Trailing;

if AlignRight then

li.TextAlign := TTextAlign.Trailing

else

li.TextAlign := TTextAlign.Leading;

listbox1.AddObject(li);

end;

function TForm1.CallGyroForce(NewData: TGyro): TGyroForce;

var

NewAngle:TImuAngle;

NewForce:TGyroForce;

begin

NewAngle.AngX.NewData:=GyroAng.AngX.OldData+

(NewData.X\*SAMPLING\_TIME );

//GyroAng.AngX.OldData:=NewAngle.AngX.NewData;

NewAngle.AngY.NewData:=GyroAng.AngY.OldData+

(NewData.Y\*SAMPLING\_TIME);

//GyroAng.AngY.OldData:=NewAngle.AngY.NewData;

NewAngle.AngZ.NewData:=GyroAng.AngZ.OldData+

(NewData.Z\*SAMPLING\_TIME );

//GyroAng.AngZ.OldData:=NewAngle.AngZ.NewData;

NewAngle.AngX.NewData:=DegToRad(NewAngle.AngX.NewData);

NewAngle.AngY.NewData:=DegToRad(NewAngle.AngY.NewData);

NewAngle.AngZ.NewData:=DegToRad(NewAngle.AngZ.NewData);

NewForce.FX:=1/

(Sqrt(1+(Cot(NewAngle.AngX.NewData)\*Cot(NewAngle.AngX.NewData))\*

(Sec(NewAngle.AngY.NewData)\*Sec(NewAngle.AngY.NewData))));

if NewAngle.AngX.NewData<0 then

NewForce.FX:=(-1)\*NewForce.FX;

NewForce.FY:=1/

Sqrt(1+(Cot(NewAngle.AngY.NewData)\*Cot(NewAngle.AngY.NewData))\*

((Sec(NewAngle.AngX.NewData)\*Sec(NewAngle.AngX.NewData))));

if NewAngle.AngY.NewData<0 then

NewForce.FY:=(-1)\*NewForce.FY;

NewForce.FZ:=Sqrt(1-(NewForce.FX\*NewForce.FX)-(NewForce.FY\*NewForce.FY));

Result:=NewForce;

end;

procedure TForm1.callini;

begin

iniFilePath:=ExtractFilePath(ParamStr(0)) + 'cfg.ini'; // application exe

if not FileExists(iniFilePath) then

WriteIniCfg // Set Object Config as Default

else

ReadIniCfg;

end;

procedure TForm1.CallMqtt;

begin

TMSMQTTClient1.BrokerHostName := 'broker.mqttdashboard.com';

TMSMQTTClient1.Connect();

memo2.Lines.Add('Connect');

end;

function TForm1.CalMovingAvg(DataIn: Integer;

var DataBuff: TMovingAvgBuffer): Integer;

var

i, SumIn:Integer;

begin

SumIn:=0;

for i := 1 to MOVIVE\_AVG\_LEN do

begin

if i<MOVIVE\_AVG\_LEN then

DataBuff.Data[i]:=DataBuff.Data[i+1] // Shift Data

else

DataBuff.Data[i]:=DataIn; // Last Data = Data input

SumIn:=SumIn+DataBuff.Data[i];

end;

Result:=Round(SumIn/MOVIVE\_AVG\_LEN);

end;

procedure TForm1.CheckBox1Change(Sender: TObject);

begin

if CheckBox1.IsChecked =true then

begin

StartSQL.Enabled:=False;

StopSQL.Enabled:=False;

end

else

begin

StartSQL.Enabled:=True;

StopSQL.Enabled:=False;

EndThreadSQL;

end

end;

function TForm1.ChkNoMove(OldVal, NewVal: Integer): Boolean;

var

AbsDiff:Integer;

begin

AbsDiff:=abs(NewVal-OldVal);

if AbsDiff<=1 then // Stationary State threshold

begin

AccNoMoveCount:=AccNoMoveCount+1; // Waiting for long time

if 20<AccNoMoveCount then // wait for 20\*Sampling time

begin

AccNoMoveCount:=0;

Result:=True;

end

else

Result:=False;

end

else

begin

AccNoMoveCount:=0;

Result:=False;

end;

end;

function TForm1.ChkNoMove2(AccNorm: Single): Boolean;

var

AbsNorm:Single;

begin

AbsNorm:=abs(AccNorm-ACC\_NORM\_VAL);

if AbsNorm<3 then // Stationary State threshold

begin

AccNoMoveCount:=AccNoMoveCount+1; // Waiting for long time

if 20<AccNoMoveCount then // wait for 20\*Sampling time

begin

AccNoMoveCount:=0;

Result:=True;

end

else

Result:=False;

end

else

begin

AccNoMoveCount:=0;

Result:=False;

end;

end;

procedure TForm1.ConnectClick(Sender: TObject);

begin

try

IdTCPClient1.Connect;

Except

ShowMessage('Fail Connection');

end;

end;

procedure TForm1.CreateCleint;

begin

if MyCleint = nil then

MyCleint := TClientRead.create(false);

end;

procedure TForm1.DataToSQL(aX, aY, aZ, aN, gR, gP, gY, mgX, mgY, mgZ, mgH, temp,

press, Alti: Single);

begin

Form1.FDTable1.Append; // Insert at last

Form1.FDTable1DateTime.Value:=Now;

Form1.FDTable1accelerationX.Value:=aX;

Form1.FDTable1accelerationY.Value:=aY;

Form1.FDTable1accelerationZ.Value:=aZ;

Form1.FDTable1accelerationNorm.Value:=aN;

//

Form1.FDTable1GyroX.Value:=gR;

Form1.FDTable1GyroY.Value:=gP;

Form1.FDTable1GyroZ.Value:=gY;

//

Form1.FDTable1MagnetoX.Value:=mgX;

Form1.FDTable1MagnetoY.Value:=mgY;

Form1.FDTable1MagnetoZ.Value:=mgZ;

Form1.FDTable1magHorizDirection.Value:=mgH;

//

Form1.FDTable1Temperature.Value:=temp;

Form1.FDTable1Pressure.Value:=press;

//

Form1.FDTable1Altitude.Value:=Alti;

Form1.FDTable1.Post;

end;

procedure TForm1.DisconnectClick(Sender: TObject);

begin

IdTCPClient1.Disconnect;

end;

procedure TForm1.EndThread3d;

begin

if My3d <> nil then

begin

My3d.Terminate;

My3d.Free;

My3d := nil;

end;

end;

procedure TForm1.EndThreadPlot;

begin

if ThreadPlot <> nil then

begin

ThreadPlot.Terminate;

ThreadPlot.Free;

ThreadPlot := nil;

end;

end;

procedure TForm1.EndThreadSQL;

begin

if ThreadSQL <> nil then

begin

ThreadSQL.Terminate;

ThreadSQL.Free;

ThreadSQL:=nil;

SQLStat.Text:='SQL is Stop';

end;

end;

function TForm1.FloatToStr2(Data: Single): String;

begin

Result:= Format('%0.2f', [Data]);

end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);

begin

ThreadModelStop;

EndThreadSQL;

EndThreadPlot;

recDataStop;

EndThread3d;

StopClient;

Timer1.Enabled:=False;

Application.Terminate;

end;

procedure TForm1.FormCreate(Sender: TObject);

var

I:Integer;

begin

IdTCPClient1.Host := Edit1.Text;

IdTCPClient1.Port := StrToInt(Edit2.Text);

PreModel;

SetZero;

ThreadPlotStart;

callMqtt;

Thread3dStart;

MyDataPath:=ExtractFilePath(Application.Name)+'MyFile.txt';

//WebBrowser1.Navigate(EditWeb.Text);

Disconnect.Enabled:=False;

Stop.Enabled:=False;

StopSQL.Enabled:=False;

StopSave.Enabled:=False;

callini;

reCheck;

end;

procedure TForm1.FormDestroy(Sender: TObject);

begin

WriteIniCfg;

end;

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; var KeyChar: Char;

Shift: TShiftState);

begin

if (Key = 27) and (ssShift in Shift) then

begin

Application.Terminate;

end;

end;

//---------------------cal HP---------------------------------//

function TForm1.HpFilter(k, OldVal, InputVal: Single): Single;

var

Ans:Single;

begin

Ans:=(k\*OldVal)+((1-k)\*InputVal);

Result:=Ans;

end;

procedure TForm1.IdTCPClient1Connected(Sender: TObject);

begin

Connect.Enabled:=False;

Disconnect.Enabled:=True;

end;

procedure TForm1.IdTCPClient1Disconnected(Sender: TObject);

begin

Connect.Enabled:=True;

Disconnect.Enabled:=False;

end;

procedure TForm1.IdTCPClient1Status(ASender: TObject; const AStatus: TIdStatus;

const AStatusText: string);

begin

end;

//---------------------cal LP---------------------------------//

function TForm1.LpFilter(k, OldVal, InputVal: Single): Single;

var

Ans:Single;

begin

Ans:=(k\*InputVal)+((1-k)\*OldVal);

Result:=Ans;

end;

procedure TForm1.PlotAcc; //Plot Acc

begin

PlotLineGraph(AccX, StrToFloat(accData.aX));

PlotLineGraph(AccY, StrToFloat(accData.aY));

PlotLineGraph(AccZ, StrToFloat(accData.aZ));

end;

procedure TForm1.PlotAccD; //Plot Acc

begin

PlotLineGraph(AccXD, StrToFloat(accData.aX));

PlotLineGraph(AccYD, StrToFloat(accData.aY));

PlotLineGraph(AccZD, StrToFloat(accData.aZ));

end;

procedure TForm1.PlotGyro; //Plot Gyro

begin

PlotLineGraph(GyroX, StrToFloat(gyroData.gyroX));

PlotLineGraph(GyroY, StrToFloat(gyroData.gyroY));

PlotLineGraph(GyroZ, StrToFloat(gyroData.gyroZ));

end;

procedure TForm1.PlotGyroD; //Plot Gyro

begin

PlotLineGraph(GyroXD, StrToFloat(gyroData.gyroX));

PlotLineGraph(GyroYD, StrToFloat(gyroData.gyroY));

PlotLineGraph(GyroZD, StrToFloat(gyroData.gyroZ));

end;

procedure TForm1.PlotLineGraph(Graph: TLineSeries; Data: Real);

var

tmpX:Double;

begin

with Graph do

begin

if XValues.Count<100 then

begin

Add(Data);

end

else

begin

tmpX:=XValues[1]-XValues[0];

Delete(0);

AddXY( XValues.Last+tmpX, Data,'',clTeeColor);

end;

end;

end;

procedure TForm1.PlotMag; // Plot Mag

begin

PlotLineGraph(MagX, StrToFloat(magData.magX));

PlotLineGraph(MagY, StrToFloat(magData.magY));

PlotLineGraph(MagZ, StrToFloat(magData.magZ));

end;

procedure TForm1.PlotMagH; // Plot MagH

begin

PlotLineGraph(MagH, StrToFloat(magData.magH));

end;

procedure TForm1.PlotNorm; // Plot AccSQRT

begin

PlotLineGraph(AccSQRTD, StrToFloat(accData.rq));

end;

procedure TForm1.PreModel;

begin

if MyModel = nil then

MyModel := TModel.Create(False);

end;

procedure TForm1.ReadIniCfg;

var

vIni:TIniFile;

begin

vIni:=TIniFile.Create(iniFilePath);

try

CheckBox1.IsChecked:=vIni.ReadBool(CheckBox1.ClassName, 'CheckBox1.Checked', False);

finally

vIni.Free;

end;

end;

procedure TForm1.recDataStart;

begin

if MyRec = nil then

MyRec := TrecData.Create(False);

end;

procedure TForm1.recDataStop;

begin

if MyRec <> nil then

begin

Myrec.Free;

Myrec.Terminate;

Myrec := nil;

end;

end;

procedure TForm1.reCheck;

begin

if CheckBox1.IsChecked then

begin

StartSQL.Enabled:=False;

StopSQL.Enabled:=False;

ShowMessage('Start With SQL');

end

else

begin

StartSQL.Enabled:=True;

StopSQL.Enabled:=False;

end;

end;

procedure TForm1.recMyData;

var

i:Integer;

begin

for i := 1 to 100 do

begin

// sleep(100);

MyImu[i].accX:=StrToFloat(accData.aX);

MyImu[i].accY:=StrToFloat(accData.aY);

MyImu[i].accZ:=StrToFloat(accData.aZ);

MyImu[i].accNorm:=StrToFloat(accData.rq);

MyImu[i].gyroR:=StrToFloat(gyroData.gyroX);

MyImu[i].gyroP:=StrToFloat(gyroData.gyroY);

MyImu[i].gyroY:=StrToFloat(gyroData.gyroZ);

MyImu[i].magX:=StrToFloat(magData.magX);

MyImu[i].magY:=StrToFloat(magData.magY);

MyImu[i].magZ:=StrToFloat(magData.magZ);

end;

end;

procedure TForm1.ReFreshClick(Sender: TObject);

begin

TMSFMXWebBrowser1.Navigate(EditWeb.Text);

end;

procedure TForm1.RunMqtt;

begin

TMSMQTTClient1.Publish(ChatChannel, TMSMQTTClient1.ClientID+'!'+ memo2.Lines.Text);

memo2.Lines.Clear;

end;

{ TClientRead }

procedure TClientRead.Execute;

var

Sl:TStringList;

Data:String;

begin

inherited;

try

repeat

Application.ProcessMessages;

Form1.IdTCPClient1.Socket.WriteLn('b'+#13#10);

Form1.Edit3.Text:=Form1.IdTCPClient1.Socket.ReadLn;

Form1.AddLog(Form1.Edit3.text);

Sl:=TStringList.Create;

Sl.CommaText:=Form1.Edit3.Text; //Extrax string

if Sl.Count = 14 then

begin

Form1.accData.aX:=Sl[0];

Form1.accData.aY:=Sl[1];

Form1.accData.aZ:=Sl[2];

Form1.accData.rq:=Sl[3];

Form1.gyroData.gyroX:=Sl[4];

Form1.gyroData.gyroY:=Sl[5];

Form1.gyroData.gyroZ:=Sl[6];

Form1.magData.magX:=Sl[7];

Form1.magData.magY:=Sl[8];

Form1.magData.magZ:=Sl[9];

Form1.magData.magH:=Sl[10];

Form1.tempData.TempX:=Sl[11];

Form1.tempData.PressX:=Sl[12];

lastData:=Sl[13];

end;

if Assigned(Sl) then

FreeAndNil(Sl);

Form1.recMyData;

until Terminated;

Except

ShowMessage('Please Connecte The Sever!');

end;

end;

{ TModel }

procedure TModel.Execute;

begin

inherited;

repeat

Application.ProcessMessages;

Form1.PlotAcc();

Form1.PlotGyro();

Sleep(100);

until Terminated;

end;

{ TMyThreadSQL }

procedure TMyThreadSQL.DataToSQL(aX, aY, aZ, aN, gR, gP, gY, mgX, mgY, mgZ, mgH,

temp, press, Alti: Single);

begin

Form1.FDTable1.Append; // Insert at last

Form1.FDTable1DateTime.Value:=Now;

Form1.FDTable1accelerationX.Value:=aX;

Form1.FDTable1accelerationY.Value:=aY;

Form1.FDTable1accelerationZ.Value:=aZ;

Form1.FDTable1accelerationNorm.Value:=aN;

//

Form1.FDTable1GyroX.Value:=gR;

Form1.FDTable1GyroY.Value:=gP;

Form1.FDTable1GyroZ.Value:=gY;

//

Form1.FDTable1MagnetoX.Value:=mgX;

Form1.FDTable1MagnetoY.Value:=mgY;

Form1.FDTable1MagnetoZ.Value:=mgZ;

Form1.FDTable1magHorizDirection.Value:=mgH;

//

Form1.FDTable1Temperature.Value:=temp;

Form1.FDTable1Pressure.Value:=press;

//

Form1.FDTable1Altitude.Value:=Alti;

Form1.FDTable1.Post;

end;

procedure TMyThreadSQL.Execute;

begin

inherited;

inherited;

Form1.SQLStat.Text:='SQL is Start';

lastData:='0';

repeat

Application.ProcessMessages;

Synchronize(ShowDataSQL);

Sleep(5000);

until Terminated;

end;

procedure TMyThreadSQL.ShowDataSQL;

begin

DataToSQL(StrToFloat(Form1.accData.aX),StrToFloat(Form1.accData.aY),StrToFloat(Form1.accData.aZ),StrToFloat(Form1.accData.rq),

StrToFloat(Form1.gyroData.gyroX),StrToFloat(Form1.gyroData.gyroY),StrToFloat(Form1.gyroData.gyroZ),

StrToFloat(Form1.magData.magX),StrToFloat(Form1.magData.magY),StrToFloat(Form1.magData.magZ),StrToFloat(Form1.magData.magH),

StrToFloat(Form1.tempData.TempX),StrToFloat(Form1.tempData.PressX),StrToFloat(lastData));

end;

{ TMyPlot }

procedure TMyPlot.Execute;

begin

inherited;

repeat

Application.ProcessMessages;

Sleep(100);

Synchronize(Form1.SumPlot);

until Terminated;

end;

{ TrecData }

procedure TrecData.Execute;

var

Buff:String;

I,x:integer;

begin

inherited;

i:=0;

AssignFile(Form1.MyDataFile,'RecordIMU\'+FormatDateTime('hh\_nn\_ss', Now)+Form1.MyDataPath); {Assigns the Filename}

ReWrite(Form1.MyDataFile); {Create a new file }

Buff:='No, Time, AccX, AccY, AccZ, AccNorm, GyroY, GyroZ';

Writeln(Form1.MyDataFile, Buff);

repeat

Application.ProcessMessages;

Inc(i);

Sleep(100);

Form1.MyData.accX:=StrToFloat(Form1.accData.aX);

Form1.MyData.accY:=StrToFloat(Form1.accData.aY);

Form1.MyData.accZ:=StrToFloat(Form1.accData.aZ);

Form1.MyData.accNorm:=StrToFloat(Form1.accData.rq);

Form1.MyData.gyroR:=StrToFloat(Form1.gyroData.gyroX);

Form1.MyData.gyroP:=StrToFloat(Form1.gyroData.gyroY);

Form1.MyData.gyroY:=StrToFloat(Form1.gyroData.gyroZ);

// Form1.Memo3.Lines.Add(FloatToStr(Form1.MyData.accX)+','+FloatToStr(Form1.MyData.accY)+''+FloatToStr(Form1.MyData.accZ));

Buff := IntToStr(i)+','+FormatDateTime('dd-mm-yyyy hh:nn:ss', Now)+','+Form1.FloatToStr2(Form1.MyImu[i].accX)+','+Form1.FloatToStr2(Form1.MyImu[i].accY)+','+Form1.FloatToStr2(Form1.MyImu[i].accy)+','+Form1.FloatToStr2(Form1.MyImu[i].accNorm)

+','+Form1.FloatToStr2(Form1.MyImu[i].gyroR)+','+Form1.FloatToStr2(Form1.MyImu[i].gyroP)+','+Form1.FloatToStr2(Form1.MyImu[i].gyroY);

Writeln(Form1.MyDataFile, Buff);

until Terminated;

Closefile(Form1.MyDataFile); {Closes file } //save

end;

{ Tmy2d }

procedure Tmy3d.callSig;

var

Sl:TStringList;

Data:String;

i:Integer;

begin

Application.ProcessMessages;

Form1.Model3D1.Position.X := StrToFloat(Form1.accData.aX);

Form1.Model3D1.Position.Y := StrToFloat(Form1.accData.aY);

Form1.Model3D1.Position.Z := StrToFloat(Form1.accData.aZ);

Form1.Model3D1.RotationAngle.X := StrToFloat(Form1.gyroData.gyroX);

Form1.Model3D1.RotationAngle.Y := StrToFloat(Form1.gyroData.gyroY);

Form1.Model3D1.RotationAngle.Z := StrToFloat(Form1.gyroData.gyroZ);

end;

procedure Tmy3d.Execute;

begin

inherited;

repeat

Application.ProcessMessages;

Sleep(100);

Synchronize(callSig);

until Terminated;

end;

end.

10. โปรแกรม Arduino

#include "SPI.h"

#include "Ethernet.h"

/\*

by MohammedDamirchi

Home

\*/

#include <MPU9250\_asukiaaa.h>

#include <Adafruit\_BMP280.h>

//#include <Wire.h>

//#include <SPI.h>

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

EthernetServer server(80);

int v;

float voltage;

int trigPin = 2; //Assign D2 as trigPin

int echoPin = 4; //Assign D4 as echoPin

long duration;

float distance;

Adafruit\_BMP280 bme; // I2C

MPU9250\_asukiaaa mySensor;

float aX, aY, aZ, aSqrt, gX, gY, gZ, mDirection, mX, mY, mZ;

void setup() {

// pinMode(trigPin, OUTPUT);

// pinMode(echoPin, INPUT);

Serial.begin(115200); //Set Serial Communication

Ethernet.begin(mac); //

server.begin(); //Start Arduino as Server role

Serial.print("Arduino as Server Role IPaddress: "); Serial.println(Ethernet.localIP());

// Serial.begin(115200);

while (!Serial);

#ifdef \_ESP32\_HAL\_I2C\_H\_ // For ESP32

Wire.begin(SDA\_PIN, SCL\_PIN);

mySensor.setWire(&Wire);

#else

Wire.begin();

mySensor.setWire(&Wire);

#endif

bme.begin();

mySensor.beginAccel();

mySensor.beginGyro();

mySensor.beginMag();

// You can set your own offset for mag values

// mySensor.magXOffset = -50;

// mySensor.magYOffset = -55;

// mySensor.magZOffset = -10;

}

void loop() {

EthernetClient client = server.available(); //Wait connection from TCP/IP

if (client) {

Serial.println("Hi...New Client");

while (client.connected()) {

while (client.available()) {

char data = client.read();

Serial.println(data);

switch (data) {

case 'a': // data='a'--> Do nothing

Serial.print("");

client.println(""); //Arduino as Server send data to LabVIEW as Client

break;

case 'b': // data='b' Read voltage from A0 and send back to LabVIEW

if (mySensor.accelUpdate() == 0) {

aX = mySensor.accelX();

aY = mySensor.accelY();

aZ = mySensor.accelZ();

aSqrt = mySensor.accelSqrt();

Serial.print(String(aX)); Serial.print(",");

Serial.print(String(aY)); Serial.print(",");

Serial.print(String(aZ)); Serial.print(",");

Serial.print(String(aSqrt)); Serial.print(",");

//---------------------------------------------------//

client.print(String(aX)); client.print(",");

client.print(String(aY)); client.print(",");

client.print(String(aZ)); client.print(",");

client.print(String(aSqrt)); client.print(",");

}

if (mySensor.gyroUpdate() == 0) {

gX = mySensor.gyroX();

gY = mySensor.gyroY();

gZ = mySensor.gyroZ();

client.print(String(gX)); client.print(",");

client.print(String(gY)); client.print(",");

client.print(String(gZ)); client.print(",");

//-------------------------------------------------//

Serial.print(String(gX)); Serial.print(",");

Serial.print(String(gY)); Serial.print(",");

Serial.print(String(gZ)); Serial.print(",");

}

if (mySensor.magUpdate() == 0) {

mX = mySensor.magX();

mY = mySensor.magY();

mZ = mySensor.magZ();

mDirection = mySensor.magHorizDirection();

Serial.print(String(mX)); Serial.print(",");

Serial.print(String(mY)); Serial.print(",");

Serial.print(String(mZ)); Serial.print(",");

Serial.print(String(mDirection)); Serial.print(",");

//--------------------------------------------------//

client.print(String(mX)); client.print(",");

client.print(String(mY)); client.print(",");

client.print(String(mZ)); client.print(",");

client.print(String(mDirection)); client.print(",");

}

// Serial.print("\tTemperature(\*C): ");

Serial.print(bme.readTemperature()); Serial.print(",");

//-----------------------------------------------------//

client.print(bme.readTemperature()); client.print(",");

// Serial.print("\tPressure(Inches(Hg)): ");

Serial.print(bme.readPressure() / 3377); Serial.print(",");

//-------------------------------------------------------//

client.print(bme.readPressure() / 3377); client.print(",");

// Serial.print("\tApproxAltitude(m): ");

Serial.print(bme.readAltitude(1013.25)); // this should be adjusted to your local forcase

//-----------------------------------------//

client.print(bme.readAltitude(1013.25)); // this should be adjusted to your local forcase

Serial.println(""); // Add an empty line

//-----------------------------------------//

client.println("");

break;

}

}

}

}

}

**12. Dashboard Node-RED**

**Graphical user interface, application

Description automatically generated**

**13 Flow Program Node-RED**

Diagram

Description automatically generated

**14. โปรแกรม Node-RED Json**

[

{

"id": "3f59ad1f848847bf",

"type": "mqtt in",

"z": "93d7d386ad906e5f",

"name": "Node-Sub",

"topic": "sensor1",

"qos": "0",

"datatype": "auto-detect",

"broker": "b6494bf324dc4470",

"nl": false,

"rap": true,

"rh": 0,

"inputs": 0,

"x": 240,

"y": 260,

"wires": [

[

"05253856ddb7d8ac",

"582e7228efeec111"

]

]

},

{

"id": "05253856ddb7d8ac",

"type": "split",

"z": "93d7d386ad906e5f",

"name": "",

"splt": ",",

"spltType": "str",

"arraySplt": "1",

"arraySpltType": "len",

"stream": false,

"addname": "topic",

"x": 490,

"y": 200,

"wires": [

[]

]

},

{

"id": "582e7228efeec111",

"type": "function",

"z": "93d7d386ad906e5f",

"name": "Extrax1",

"func": "var outputMsgs = [];\nvar words = msg.payload.split(\",\");\n// for (var w in words) {\n// outputMsgs.push({ payload: words[w] });\n// }\nvar text = { payload: words[0] };\nvar accx = { payload: words[1] };\nvar accy = { payload: words[2] };\nvar accz = { payload: words[3] };\nvar gyrox = { payload: words[4] };\nvar gyroy = { payload: words[5] };\nvar gyroz = { payload: words[6] };\n// var out7 = { payload: words[7] };\nreturn [accx, accy, accz, gyrox, gyroy, gyroz];",

"outputs": 6,

"noerr": 0,

"initialize": "",

"finalize": "",

"libs": [],

"x": 560,

"y": 360,

"wires": [

[

"dc8c1a544e5d054c",

"11dbd3299518a2dc",

"db28527f1995ea3d"

],

[

"701e22f6b6d89691",

"12947342af2887e5",

"3b88359673e778be"

],

[

"4d2a63447a04406b",

"6a700b94e4945a6c",

"8358be5c055d2cd5"

],

[

"03770943bf5b0886",

"99442b9bb65d3d21",

"16a87ab2cafbf5db"

],

[

"f4582acc9fcdf523",

"7804686d71a76c1d",

"0f7a39d3204d89b3"

],

[

"549c883b57d8e198",

"ea62f7ded1aaded9",

"c3ce7391b9cca130"

]

]

},

{

"id": "dc8c1a544e5d054c",

"type": "debug",

"z": "93d7d386ad906e5f",

"name": "AccX",

"active": true,

"tosidebar": true,

"console": false,

"tostatus": false,

"complete": "payload",

"targetType": "msg",

"statusVal": "",

"statusType": "auto",

"x": 790,

"y": 240,

"wires": []

},

{

"id": "701e22f6b6d89691",

"type": "debug",

"z": "93d7d386ad906e5f",

"name": "AccY",

"active": true,

"tosidebar": true,

"console": false,

"tostatus": false,

"complete": "payload",

"targetType": "msg",

"statusVal": "",

"statusType": "auto",

"x": 790,

"y": 280,

"wires": []

},

{

"id": "4d2a63447a04406b",

"type": "debug",

"z": "93d7d386ad906e5f",

"name": "AccZ",

"active": true,

"tosidebar": true,

"console": false,

"tostatus": false,

"complete": "payload",

"targetType": "msg",

"statusVal": "",

"statusType": "auto",

"x": 790,

"y": 320,

"wires": []

},

{

"id": "03770943bf5b0886",

"type": "debug",

"z": "93d7d386ad906e5f",

"name": "GyroX",

"active": true,

"tosidebar": true,

"console": false,

"tostatus": false,

"complete": "payload",

"targetType": "msg",

"statusVal": "",

"statusType": "auto",

"x": 790,

"y": 360,

"wires": []

},

{

"id": "f4582acc9fcdf523",

"type": "debug",

"z": "93d7d386ad906e5f",

"name": "GyroY",

"active": true,

"tosidebar": true,

"console": false,

"tostatus": false,

"complete": "payload",

"targetType": "msg",

"statusVal": "",

"statusType": "auto",

"x": 790,

"y": 400,

"wires": []

},

{

"id": "549c883b57d8e198",

"type": "debug",

"z": "93d7d386ad906e5f",

"name": "GyroZ",

"active": true,

"tosidebar": true,

"console": false,

"tostatus": false,

"complete": "payload",

"targetType": "msg",

"statusVal": "",

"statusType": "auto",

"x": 790,

"y": 440,

"wires": []

},

{

"id": "ce85ab19d67a4932",

"type": "inject",

"z": "93d7d386ad906e5f",

"name": "test",

"props": [

{

"p": "payload"

},

{

"p": "topic",

"vt": "str"

}

],

"repeat": "",

"crontab": "",

"once": false,

"onceDelay": 0.1,

"topic": "",

"payload": "0,1,2,3,4,5,6",

"payloadType": "str",

"x": 250,

"y": 420,

"wires": [

[

"582e7228efeec111"

]

]

},

{

"id": "11dbd3299518a2dc",

"type": "ui\_gauge",

"z": "93d7d386ad906e5f",

"name": "",

"group": "762a9c64e9042f0e",

"order": 1,

"width": 0,

"height": 0,

"gtype": "gage",

"title": "AccX",

"label": "units",

"format": "{{value}}",

"min": "-3",

"max": "3",

"colors": [

"#00b500",

"#e6e600",

"#ca3838"

],

"seg1": "",

"seg2": "",

"className": "",

"x": 970,

"y": 120,

"wires": []

},

{

"id": "12947342af2887e5",

"type": "ui\_gauge",

"z": "93d7d386ad906e5f",

"name": "",

"group": "d45b44600d92c5d7",

"order": 2,

"width": 0,

"height": 0,

"gtype": "gage",

"title": "AccY",

"label": "units",

"format": "{{value}}",

"min": "-3",

"max": "3",

"colors": [

"#00b500",

"#e6e600",

"#ca3838"

],

"seg1": "",

"seg2": "",

"className": "",

"x": 970,

"y": 160,

"wires": []

},

{

"id": "6a700b94e4945a6c",

"type": "ui\_gauge",

"z": "93d7d386ad906e5f",

"name": "",

"group": "18af62378bfc9340",

"order": 3,

"width": 0,

"height": 0,

"gtype": "gage",

"title": "AccZ",

"label": "units",

"format": "{{value}}",

"min": "-3",

"max": "3",

"colors": [

"#00b500",

"#e6e600",

"#ca3838"

],

"seg1": "",

"seg2": "",

"className": "",

"x": 970,

"y": 200,

"wires": []

},

{

"id": "99442b9bb65d3d21",

"type": "ui\_gauge",

"z": "93d7d386ad906e5f",

"name": "",

"group": "3f7ab6be951f07ef",

"order": 4,

"width": 0,

"height": 0,

"gtype": "gage",

"title": "GyroX",

"label": "units",

"format": "{{value}}",

"min": "-500",

"max": "500",

"colors": [

"#00b500",

"#e6e600",

"#ca3838"

],

"seg1": "",

"seg2": "",

"className": "",

"x": 970,

"y": 240,

"wires": []

},

{

"id": "7804686d71a76c1d",

"type": "ui\_gauge",

"z": "93d7d386ad906e5f",

"name": "",

"group": "67da9534adb32455",

"order": 5,

"width": 0,

"height": 0,

"gtype": "gage",

"title": "GyroY",

"label": "units",

"format": "{{value}}",

"min": "-500",

"max": "500",

"colors": [

"#00b500",

"#e6e600",

"#ca3838"

],

"seg1": "",

"seg2": "",

"className": "",

"x": 970,

"y": 280,

"wires": []

},

{

"id": "ea62f7ded1aaded9",

"type": "ui\_gauge",

"z": "93d7d386ad906e5f",

"name": "",

"group": "78825f875df1d4cd",

"order": 6,

"width": 0,

"height": 0,

"gtype": "gage",

"title": "GyroZ",

"label": "units",

"format": "{{value}}",

"min": "-500",

"max": "500",

"colors": [

"#00b500",

"#e6e600",

"#ca3838"

],

"seg1": "",

"seg2": "",

"className": "",

"x": 970,

"y": 320,

"wires": []

},

{

"id": "db28527f1995ea3d",

"type": "ui\_chart",

"z": "93d7d386ad906e5f",

"name": "",

"group": "762a9c64e9042f0e",

"order": 7,

"width": 0,

"height": 0,

"label": "AccX",

"chartType": "line",

"legend": "false",

"xformat": "HH:mm:ss",

"interpolate": "linear",

"nodata": "",

"dot": false,

"ymin": "-3",

"ymax": "3",

"removeOlder": 1,

"removeOlderPoints": "1000",

"removeOlderUnit": "60",

"cutout": 0,

"useOneColor": false,

"useUTC": true,

"colors": [

"#1f77b4",

"#aec7e8",

"#ff7f0e",

"#2ca02c",

"#98df8a",

"#d62728",

"#ff9896",

"#9467bd",

"#c5b0d5"

],

"outputs": 1,

"useDifferentColor": false,

"className": "",

"x": 970,

"y": 380,

"wires": [

[]

]

},

{

"id": "3b88359673e778be",

"type": "ui\_chart",

"z": "93d7d386ad906e5f",

"name": "",

"group": "d45b44600d92c5d7",

"order": 8,

"width": 0,

"height": 0,

"label": "AccY",

"chartType": "line",

"legend": "false",

"xformat": "HH:mm:ss",

"interpolate": "linear",

"nodata": "",

"dot": false,

"ymin": "-3",

"ymax": "3",

"removeOlder": 1,

"removeOlderPoints": "1000",

"removeOlderUnit": "60",

"cutout": 0,

"useOneColor": false,

"useUTC": true,

"colors": [

"#1f77b4",

"#aec7e8",

"#ff7f0e",

"#2ca02c",

"#98df8a",

"#d62728",

"#ff9896",

"#9467bd",

"#c5b0d5"

],

"outputs": 1,

"useDifferentColor": false,

"className": "",

"x": 970,

"y": 420,

"wires": [

[]

]

},

{

"id": "8358be5c055d2cd5",

"type": "ui\_chart",

"z": "93d7d386ad906e5f",

"name": "",

"group": "18af62378bfc9340",

"order": 9,

"width": 0,

"height": 0,

"label": "AccZ",

"chartType": "line",

"legend": "false",

"xformat": "HH:mm:ss",

"interpolate": "linear",

"nodata": "",

"dot": false,

"ymin": "-3",

"ymax": "3",

"removeOlder": 1,

"removeOlderPoints": "1000",

"removeOlderUnit": "60",

"cutout": 0,

"useOneColor": false,

"useUTC": true,

"colors": [

"#1f77b4",

"#aec7e8",

"#ff7f0e",

"#2ca02c",

"#98df8a",

"#d62728",

"#ff9896",

"#9467bd",

"#c5b0d5"

],

"outputs": 1,

"useDifferentColor": false,

"className": "",

"x": 970,

"y": 460,

"wires": [

[]

]

},

{

"id": "16a87ab2cafbf5db",

"type": "ui\_chart",

"z": "93d7d386ad906e5f",

"name": "",

"group": "3f7ab6be951f07ef",

"order": 10,

"width": 0,

"height": 0,

"label": "GyroX",

"chartType": "line",

"legend": "false",

"xformat": "HH:mm:ss",

"interpolate": "linear",

"nodata": "",

"dot": false,

"ymin": "-500",

"ymax": "500",

"removeOlder": 1,

"removeOlderPoints": "1000",

"removeOlderUnit": "60",

"cutout": 0,

"useOneColor": false,

"useUTC": true,

"colors": [

"#1f77b4",

"#aec7e8",

"#ff7f0e",

"#2ca02c",

"#98df8a",

"#d62728",

"#ff9896",

"#9467bd",

"#c5b0d5"

],

"outputs": 1,

"useDifferentColor": false,

"className": "",

"x": 970,

"y": 500,

"wires": [

[]

]

},

{

"id": "0f7a39d3204d89b3",

"type": "ui\_chart",

"z": "93d7d386ad906e5f",

"name": "",

"group": "67da9534adb32455",

"order": 11,

"width": 0,

"height": 0,

"label": "GyroY",

"chartType": "line",

"legend": "false",

"xformat": "HH:mm:ss",

"interpolate": "linear",

"nodata": "",

"dot": false,

"ymin": "-500",

"ymax": "500",

"removeOlder": 1,

"removeOlderPoints": "1000",

"removeOlderUnit": "60",

"cutout": 0,

"useOneColor": false,

"useUTC": true,

"colors": [

"#1f77b4",

"#aec7e8",

"#ff7f0e",

"#2ca02c",

"#98df8a",

"#d62728",

"#ff9896",

"#9467bd",

"#c5b0d5"

],

"outputs": 1,

"useDifferentColor": false,

"className": "",

"x": 970,

"y": 540,

"wires": [

[]

]

},

{

"id": "c3ce7391b9cca130",

"type": "ui\_chart",

"z": "93d7d386ad906e5f",

"name": "",

"group": "78825f875df1d4cd",

"order": 12,

"width": 0,

"height": 0,

"label": "GyroZ",

"chartType": "line",

"legend": "false",

"xformat": "HH:mm:ss",

"interpolate": "linear",

"nodata": "",

"dot": false,

"ymin": "-500",

"ymax": "500",

"removeOlder": 1,

"removeOlderPoints": "1000",

"removeOlderUnit": "60",

"cutout": 0,

"useOneColor": false,

"useUTC": true,

"colors": [

"#1f77b4",

"#aec7e8",

"#ff7f0e",

"#2ca02c",

"#98df8a",

"#d62728",

"#ff9896",

"#9467bd",

"#c5b0d5"

],

"outputs": 1,

"useDifferentColor": false,

"className": "",

"x": 970,

"y": 580,

"wires": [

[]

]

},

{

"id": "b6494bf324dc4470",

"type": "mqtt-broker",

"name": "hiveMqtt-pub",

"broker": "broker.mqttdashboard.com",

"port": "1883",

"clientid": "",

"autoConnect": true,

"usetls": false,

"protocolVersion": "4",

"keepalive": "60",

"cleansession": true,

"birthTopic": "",

"birthQos": "0",

"birthPayload": "",

"birthMsg": {},

"closeTopic": "",

"closeQos": "0",

"closePayload": "",

"closeMsg": {},

"willTopic": "",

"willQos": "0",

"willPayload": "",

"willMsg": {},

"userProps": "",

"sessionExpiry": ""

},

{

"id": "762a9c64e9042f0e",

"type": "ui\_group",

"name": "AccelerationX",

"tab": "4b11d5cd7d4d9975",

"order": 2,

"disp": true,

"width": "6",

"collapse": false,

"className": ""

},

{

"id": "d45b44600d92c5d7",

"type": "ui\_group",

"name": "accelerationY",

"tab": "4b11d5cd7d4d9975",

"order": 3,

"disp": true,

"width": "6",

"collapse": false,

"className": ""

},

{

"id": "18af62378bfc9340",

"type": "ui\_group",

"name": "accelerationZ",

"tab": "4b11d5cd7d4d9975",

"order": 4,

"disp": true,

"width": "6",

"collapse": false,

"className": ""

},

{

"id": "3f7ab6be951f07ef",

"type": "ui\_group",

"name": "GyroScopeX",

"tab": "4b11d5cd7d4d9975",

"order": 5,

"disp": true,

"width": "6",

"collapse": false,

"className": ""

},

{

"id": "67da9534adb32455",

"type": "ui\_group",

"name": "GyroScopeY",

"tab": "4b11d5cd7d4d9975",

"order": 6,

"disp": true,

"width": "6",

"collapse": false,

"className": ""

},

{

"id": "78825f875df1d4cd",

"type": "ui\_group",

"name": "GyroScopeZ",

"tab": "4b11d5cd7d4d9975",

"order": 7,

"disp": true,

"width": "6",

"collapse": false,

"className": ""

},

{

"id": "4b11d5cd7d4d9975",

"type": "ui\_tab",

"name": "IMU",

"icon": "dashboard",

"disabled": false,

"hidden": false

}

]