```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package seafarm;

import de.re.easymodbus.modbusclient.ModbusClient;
import java.util.Observable;
import java.util.Observer;
import java.util.logging.Level;
import java.util.logging.Logger;
   //modbus client, set to the same client as the sender in constructor//

/**
 *
 * @author Platform
 */
public class ModbusSender implements Observer,Runnable {

     private ModbusClient client;
     private Thread t;
     private boolean connected=false;
     private int timeBetweenSending =200;
     private long timeVar=0;
     private boolean varHasBeenUpdated=false;
     private boolean timeElapsed=false;

    //platform movement commands//
     int thrusterSpeed=0;
     boolean fwdMotion = false;
     boolean bckMotion = false;
     boolean rightMotion = false;
     boolean leftMotion = false;
     boolean clockWMotion = false;


     boolean counterClocMotion = false;
     boolean enableLight=false;
     boolean enableFlute=false;
     boolean platformEnable=false;
     boolean enableAuto=false;
     boolean enableManual=false;
```

```
//DpMode commands//
boolean dpModeEnable=false;
float latitude=0;
float longitude=0;



//winch commands//
boolean winchUp=false;
boolean winchDown=false;
int winchSpeed=0;
boolean winchLockOn=false;
boolean winchLockOff=false;

//Pump commands//
boolean startPump = false;

//ROV data for logging on plc
 //rov data for logging
float rovDpt=0;
float rovTmpInrov=0;
float rovTmpInSea=0;
float rovHeading=0;
float rovOxygen=0;

// var for plc to check if gui is not connected
boolean guiConcheckVar=false;
//Modbus Registers to write on///
static int thrusterSpeerReg=32040;
static int fwdMotionReg=32769;
static int bckMotionReg=32770;
static int rightMotionReg=32771;
static int leftMotionReg=32772;
static int clockMotionReg=32773;
static int counterClMotionReg=33057;
static int enableLightReg=32774;
static int enableFluteReg=32775;
static int platfromEnableReg=32776;
static int enableAturoReg=32777;
static int enableManualReg=32778;
static int enableDpModeReg=32779;
static int longitudeReg=32048;
```

```
    static int latitudeReg=32044;
    static int startPumpReg=33056;
    static int winchUpReg=32780;
    static int winchDownReg=32781;
    static int winchLockOnReg=32782;
    static int winchLockOffReg=32783;
    static int winchSpeedReg=32042;
    static int guiConcheckVarReg= 33058;
    //rov
    static int rovHeadReg= 32034;
    static int rovInternalTempReg= 32022;
    static int rovExternalTempReg= 32030;
    static int rovDeptReg= 32028;
    static int rovOxygenReg= 32032;
public ModbusSender(ModbusClient modbusClient){
    this.client=modbusClient;

}

    public void start() {
            t = new Thread(this, "MODBUSreceiverThread");
            t.start();
            System.out.println("Starting");
        }


     public void run() {


    while(true){
        try {
            Thread.sleep(8000);
        } catch (InterruptedException ex) {
            Logger.getLogger(ModbusReciever.class.getName()).log(Level.SEVE
RE, null, ex);
        }


        while(!connected){
            try {
            System.out.println("sender Trying to connect to server");
            client.Connect();
            connected=true;
                }
```

```
            catch (Exception e) {
                connected=false;
                System.out.println(e);
                continue;

             }


        }


            while (connected) {

                if (System.currentTimeMillis()>= timeVar){
                    timeElapsed=true;
                }else{
                    timeElapsed=false;
                }


                if (varHasBeenUpdated || timeElapsed){
                try{

                    //sends true and false concheck so the plc can check if
it changes,
                    if(guiConcheckVar){
                        guiConcheckVar=false;
                    }
                    else{
                       guiConcheckVar=true;
                    }

                    sendData();
                   // System.out.println("Data sent to PLC");
                    varHasBeenUpdated=false;
                    timeVar=System.currentTimeMillis()+timeBetweenSending;
                     }

            catch (Exception e)
            {
                connected=false;
                System.out.println(e);
            }
            }
```

```
    }
     }



}



    public void sendData(){
        try {

            ///writing coils////booleans///
            //todoo: endre til wirte multiple coils.
            this.client.WriteSingleCoil(fwdMotionReg, fwdMotion);
            this.client.WriteSingleCoil(bckMotionReg, bckMotion);
            this.client.WriteSingleCoil(rightMotionReg, rightMotion);
            this.client.WriteSingleCoil(leftMotionReg, leftMotion);
            this.client.WriteSingleCoil(clockMotionReg, clockWMotion);
            this.client.WriteSingleCoil(counterClMotionReg,
counterClocMotion);
            this.client.WriteSingleCoil(enableLightReg, enableLight);
            this.client.WriteSingleCoil(enableFluteReg, enableFlute);
            this.client.WriteSingleCoil(platfromEnableReg, platformEnable);
            this.client.WriteSingleCoil(enableAturoReg, enableAuto);
            this.client.WriteSingleCoil(enableManualReg, enableManual);
            this.client.WriteSingleCoil(enableDpModeReg, dpModeEnable);
            this.client.WriteSingleCoil(winchUpReg, winchUp);
            this.client.WriteSingleCoil(winchDownReg, winchDown);
            System.out.println(winchDown);
            this.client.WriteSingleCoil(winchLockOnReg, winchLockOn);
            this.client.WriteSingleCoil(winchLockOffReg, winchLockOff);
            this.client.WriteSingleCoil(startPumpReg, startPump);
            this.client.WriteSingleCoil(guiConcheckVarReg, guiConcheckVar);

            ///writing numeric values///
            //send float to two registers
            this.client.WriteMultipleRegisters(latitudeReg,
ModbusClient.ConvertFloatToTwoRegisters((float) latitude));
            this.client.WriteMultipleRegisters(longitudeReg,
ModbusClient.ConvertFloatToTwoRegisters((float) longitude));
            //rov values for logging on plc
            this.client.WriteMultipleRegisters(rovOxygenReg,
```

```
ModbusClient.ConvertFloatToTwoRegisters((float) rovOxygen));
            this.client.WriteMultipleRegisters(rovDeptReg,
ModbusClient.ConvertFloatToTwoRegisters((float) rovDpt));
            this.client.WriteMultipleRegisters(rovExternalTempReg,
ModbusClient.ConvertFloatToTwoRegisters((float) rovTmpInSea));
            this.client.WriteMultipleRegisters(rovInternalTempReg,
ModbusClient.ConvertFloatToTwoRegisters((float) rovTmpInrov));
            this.client.WriteMultipleRegisters(rovHeadReg,
ModbusClient.ConvertFloatToTwoRegisters((float) rovHeading));
            //System.out.println(longitude +"longSetInmodbusSend");
            // send int values to registers//
            this.client.WriteSingleRegister(thrusterSpeerReg,
thrusterSpeed);
            this.client.WriteSingleRegister(winchSpeedReg, winchSpeed);




        } catch (Exception e) {

            System.out.println("cant send to mobusregisters");
            connected=false;
            run();
        }


     }




    @Override
    public void update(Observable o, Object arg) {
        if(o instanceof SendDataObserver){
           // System.out.println("Update has been called ");
            SendDataObserver sendOb = (SendDataObserver) o;
            this.fwdMotion=sendOb.isFwdMotion();
            this.bckMotion=sendOb.isBckMotion();
            this.leftMotion=sendOb.isLeftMotion();
            this.rightMotion=sendOb.isRightMotion();
            this.clockWMotion=sendOb.isClockWMotion();
```

```
            this.counterClocMotion=sendOb.isCounterClockWMotion();
            this.dpModeEnable=sendOb.isDpModeEnable();
            this.enableManual=sendOb.isEnableManual();
            this.enableAuto=sendOb.enableAuto;
            this.thrusterSpeed=sendOb.getThrusterSpeed();
            this.winchSpeed=sendOb.getWinchSpeed();
            this.longitude=sendOb.getLongitude();
            this.latitude=sendOb.getLatitude();
            this.enableFlute=sendOb.isEnableFlute();
            this.winchDown=sendOb.isWinchDown();
            this.winchUp=sendOb.isWinchUp();

            //rov data
            this.rovDpt=sendOb.getDpt();
            this.rovHeading=sendOb.getHeadrov();
            this.rovOxygen=sendOb.getOxygen();
            this.rovTmpInSea=sendOb.getTmpInSea();
            this.rovTmpInrov=sendOb.getTmpInrov();
            varHasBeenUpdated=true;
        }

    }
}
```