**Functionality Outline**

Oil Well Simulation

**Program Assignment** 1

**Prepared By**

Jaiden Gann

[jmg0053@uah.edu](mailto:Jtw0014@uah.edu)

**Prepared for**  
**Mr. James Williamson**  
**CS 307, Object Oriented Programming**  
**Computer Science Department**  
**University of Alabama in Huntsville**

**1.0 System Overview…………………………………………………………………………………………….3**

**2.0 Relevant Terms and Acronyms………………………………………………………………………..3**

**3.0 Object Functionality………….…………………………………………………………………………….3**

**3.1 Main…………………..………………………………………….……………………………………3**

**3.2 OilFieldMonitor…………………………………………………..………………………………3**

**3.3 OilWell………………………………………………………………………………………………..4**

**3.4 Sensors………………….………………………………………………………………….………..4**

**3.5 DisplayReport………………………………………………………………………….………...5**

**3.6 UserInterruption……………………………………………….………………….…….…..…5**

**3.7 TimerLoop…………………………………………………………………………….…….………5**

**4.0 Notes………………………………………………………………………………………………………..……6**

## 1.0 System Overview

The purpose of the program is to simulate software used to monitor sensor readings on a number oil rigs. The sensors constantly output data to a display for works and managers to monitor the aspects of rig operations. This program will be developed with as much object-oriented functionality as possible so that it can be expanded and changed easily for the future.

## 2.0 Relevant Terms and Acronyms

*Rig* – Refers to a single oil well. Can be used interchangeably with “well”

*ID* – Refers to a number sequence used to Identify each well. Will succeed the name of the owning company

*Hole Depth* – Depth of the hole that has been drilled. Measured in feet (ft)

*Bit Depth* – Current depth(location) of the drill head. Measured in feet (ft)

*ROP* – rate of penetration. Used to indicate how fast the drill head is drilling through the crust. Measured in feet per hour (ft/hr)

*Off Bottom* – indicates the drill bit is not currently drilling and ROP is 0 ft/hr

*Pump Pressure* – Pressure in the lubricant(mud) pump. Measured in pounds per square inch (PSI)

*Casing Pressure* – Pressure in the drill bit casing itself. Measured in pounds per square inch (PSI)

*Flow out* – Mud(lubricant) flowing out of the bit casing. Measured in percentage of maximum flow (%)

*Torque Max* – Maximum torque that can be safely applied to the drill bit. Measured in kilo-foot pounds (kft-lbs)

*Mud Pit Volume* – Volume of mud(lubricant) available. Measured in barrels (BBL)

## 3.0 Object Functionality

### Main.cpp

Run timer loop //which I think unknowingly makes it the top file…..

Or

Call start() from oilfieldMonitor

### OilFieldMonitor.cpp

**Start()**

Ask user for data file

Give it to the parser

**GetdataWell() //**probably take the file name and or parser as input

Call dataparser and get well info

**//**other idea, enum the well name if possible and associate data that way

**Place id in a well vector**

**Place data in a vector**

GetdataSensor()

Call dataparser and get sensor info

//other idea, enum the sensor name if possible and associate data that way

Place names in a sensor vector

Place data in a vector

//implement timer loop back into this???

### Oil\_Well.cpp

List wellWatchlist // keep track of which well you’re watching

**Add\_remove() //**probably wanna pass the character from userinterruption

Call userinterruption to check for certain key hits

If(some character)

Add some well to the list

Update bool IsMonitored

Else if (some other character)

Delete some well from the list

Update bool IsMonitored

//alternative is a switch statement that checks if the character is equal to something and then for each case check IsMonitored and add or remove based on if its already added or not

**sendDisplayReport()**

send the sensor name and data to DisplayReport

probably use pointers or something so that DisplayReport can grab it

### Sensors.cpp

List sensorWatchlist //list to keep track of the sensors you want to watch

Add\_remove() //probably wanna pass the character from userinterruption

Call userinterruption to check for certain key hits

If(some character)

Add some sensor to the list

Update bool IsMonitored

Else if (some other character)

Delete some sensor from the list

Update bool IsMonitored

//alternative is a switch statement that checks if the character is equal to something and then for each case check IsMonitored and add or remove based on if its already added or not

sendDisplayReport()

send the sensor name and data to DisplayReport

probably use pointers or something so that DisplayReport can grab it

### DisplayReport.cpp

Access list of well and sensor info

printDisplay() //probably pass the lists

format output

### UserInterruption.cpp

Import conio and stdio

Set a variable to \_kbhit() to check key press

If(key hit)

Grab the character

### TimerLoop.cpp

import sys/types and sys/timeb and time

\_ftime(pointer to struct)

set up variables for current time and output

set up time struct

grab the start time

convert to double

set up next 5 sec interval

while() //setup internal loop

get current time

convert to double

if (next 5 sec interval)

printReport()

## Notes

I know that from looking at the example functionality outline there is probably a lot missing to make the actual simulation of the software functional. However, I was trying to stick with what I had outlined previously to the best of my ability because that is how I currently understand the interconnectedness of this simulation. This is a tentative outline that is subject to change when the coding the process actually starts.