**University College of Southeast Norway**

Faculty of technology

Bachelor of Science

**ATTACHMENT FOLDER FROM 6$^{TH}$ SEMESTER PROJECT SPRING 2016**

PRH612 Bachelor thesis

IA6-5-16

# Read, control and communication unit for manholes

**Address: Kjølnes ring 56, N-3918 Porsgrunn, Tel: +47 31 00 80 00, www.usn.no**

Bachelor's programmes – Master's programmes – Ph.D. programmes

# ATTACHMENTS

Høgskolen i Telemark

Fakultet for teknologiske fag

# PRH612 Bacheloroppgaven

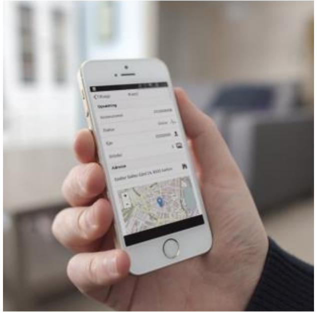
**Tittel**: **Utvikling av intelligente kumlokk**

**Hovedveileder**:

**Ekstern partner**: Ulefos Jernværk (willy.dorholt@ujv.no
97024791.

**Oppgavebeskrivelse**:

Enheten skal kunne ettermonteres i kummer, den består av 2 deler: En kommunikasjonsenhet og en sensor og styringsenhet. Begge enhetene har hver sin batteripakke. De skal være lette å ha service på. Kommunikasjonssentralen bør være rimelig å installere slik at inngangsbilletten til systemet er lav.



Videre er det viktig at datainnsamlingen og statusalarmer lett kan brukes av andre systemer. Det kan være i større alarmsystemer, melding til servicepersonell og planleggere. Det er også mulig at systemet kan samle inn data for analyser og dokumentasjoner ved skader.
Ved hjelp av kommunikasjonsenheten, vil det kunne være mulig å styre pumper og ventiler, samt avlese status på disse.

### Foreløpige ideer til produktet

1. Overvåke vannivå
2. Overvåke om lokket er rett plassert
3. Overvåke behov for tømming av slam

4. Overvåke om lokket er åpent
5. Mulig å søke etter lokket ved akutt behov.
6. Kamera kan ta situasjonsbilde av kummen
7. Status på vannstrømmer i rør
8. Status og styring av ventiler og pumper.
9. Måling av temperatur. Viktig dersom kummen har utstyr som er temperaturkritisk
10. Telling av overkjøringer
11. Vurdere ulyd ved overkjøringer

**Bakgrunn for oppgaven**:

Utvikling av tingenes internett er allerede på full fart inn i vår hverdag. Ulefos Jernværk var sammen med Xepto tidlig ute med å tilby overvåkning av overvann i kummer. Det har skjedd mye på teknologisiden og forventinger til hva slike system skal levere siden den gang. Ulefos-gruppen ønsker å videreutvikle dette produktet og derfor er det tatt initiativ til dette prosjektet.

I dag finnes det mange utviklingsfirmaer som kan designe slikt utstyr og bruke velutprøvde og åpne teknologier. Det som vil være avgjørende i fremtidig konkurranse kan være:

- Brukervennlighet
- Nytteverdi for kunden
- Tiltalende og robust design
- Lett å integrere i andre overvåkningssystemer
- Tar i bruk smarttelefoner og PC for å lette arbeidet til vedlikeholdspersonell og planleggere
- Høy opptid og god service
- Dyktige ved installasjon og igangsetting

**Studentkategori**:

IA

**Praktiske ordninger**:

Utstyr kan lånes fra Ulefos Jernværk

**Signaturer**:

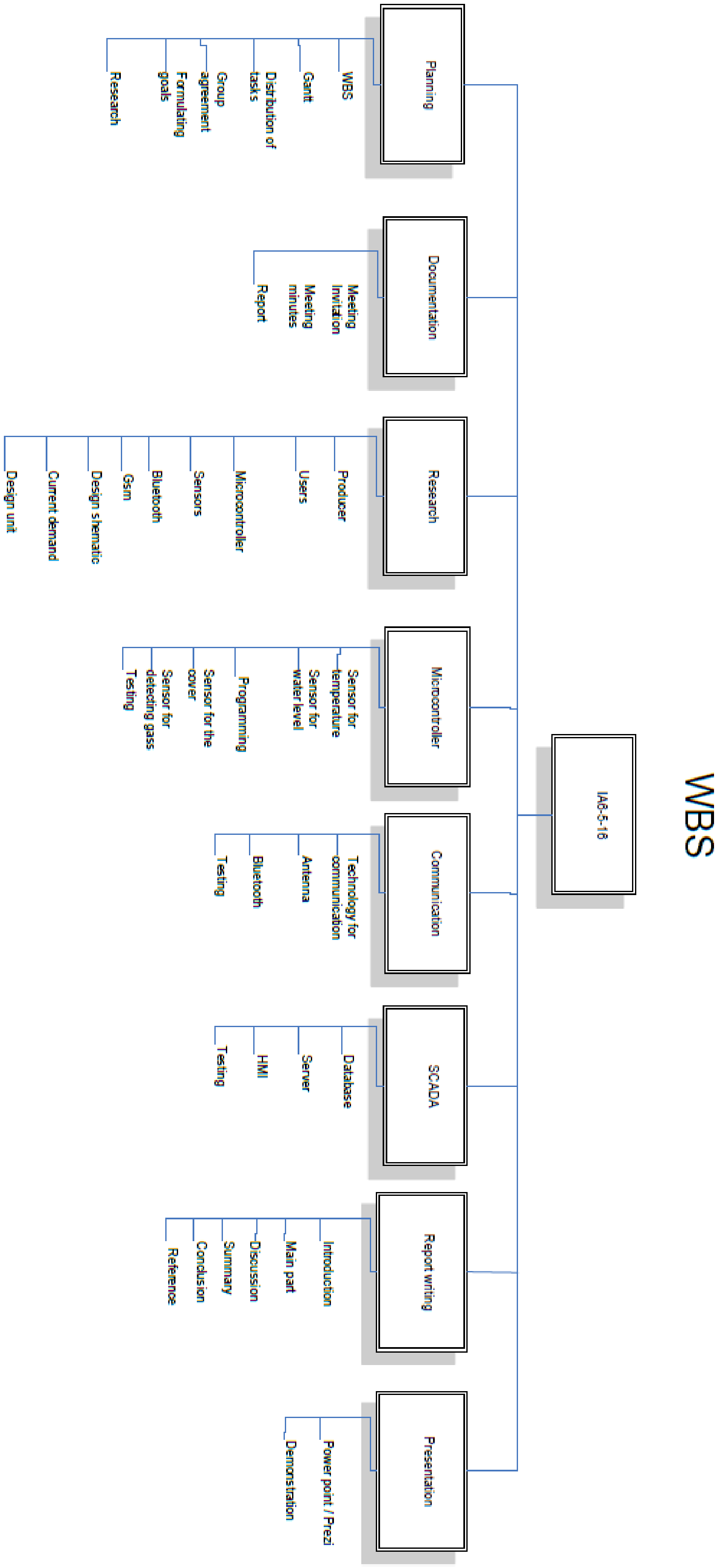
Student (dato og signatur):

Hovedveileder (dato og signatur):

# Formulation of goals

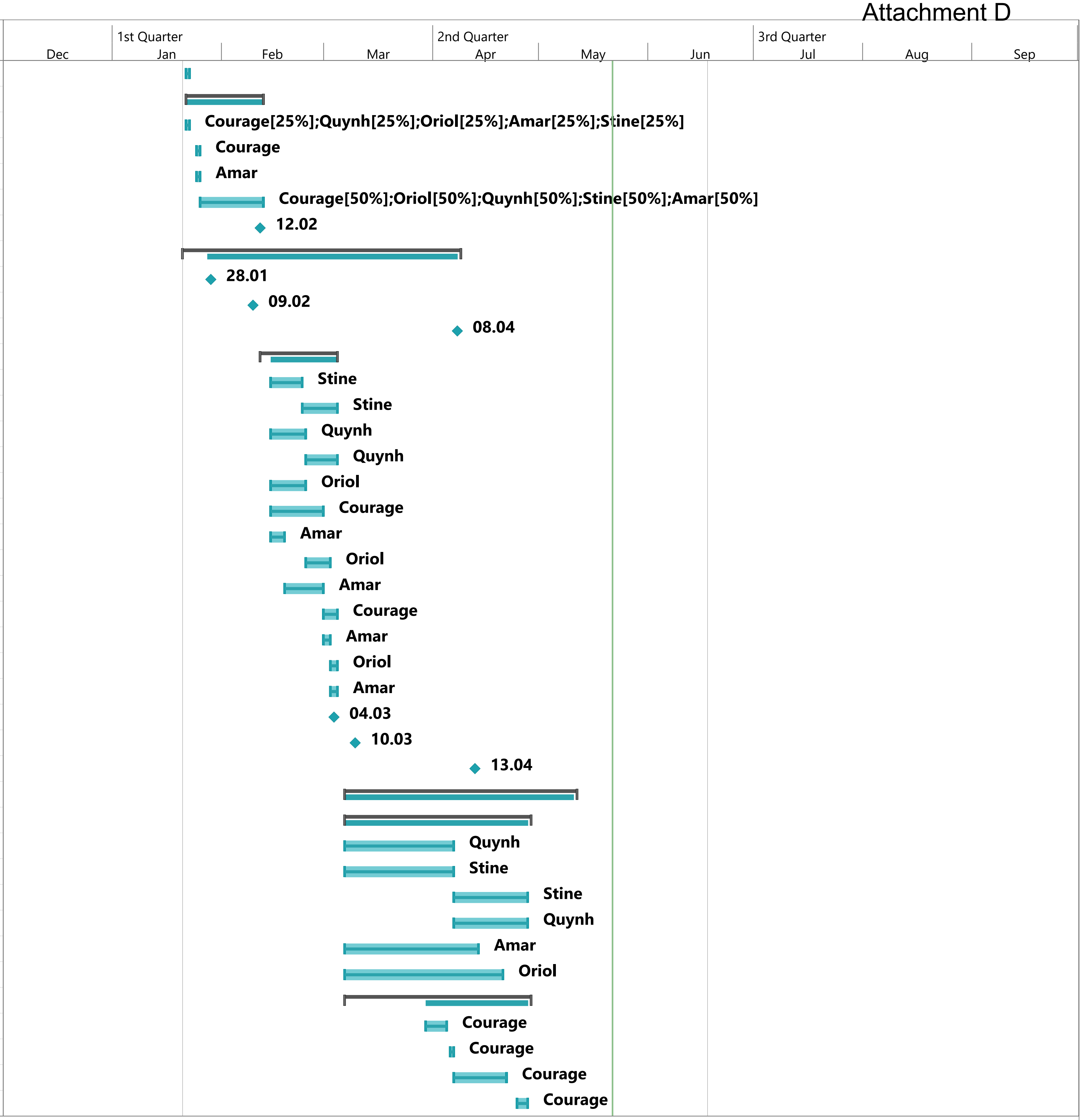
The goal with the project is to assess the current solution for intelligent manhole covers (kumlokk), look at the cost-efficiency and the functionality of the solution. The solution should be an improvement of the current solution. Meaning a cheaper product than today, has longer life-time, and a greater functionality. The current solution sends an alarm, only when the water flows over, and the solution can only send info, not receive. An improvement would be to read water-level when wanted and also have an option for the unit to receive a command to read current status. With the current solution, the whole manhole cover with seat, has to be changed. This is for the alignment of the cover-sensor. The cover sensor is a mechanical switch implemented in the cover, with the unit. This again gives the customer a higher cost (ca. 1000% more expensive). A better solution is to have a unit that can be added on the side of the manhole, or wherever the customer wants, with the possibility to add features later, and without the need to change the cover and seat, meaning the cover switch/sensor is implemented in the unit or on the side of the manhole, not in the cover, as it is in the current solution.

Attachment C



Attachment D

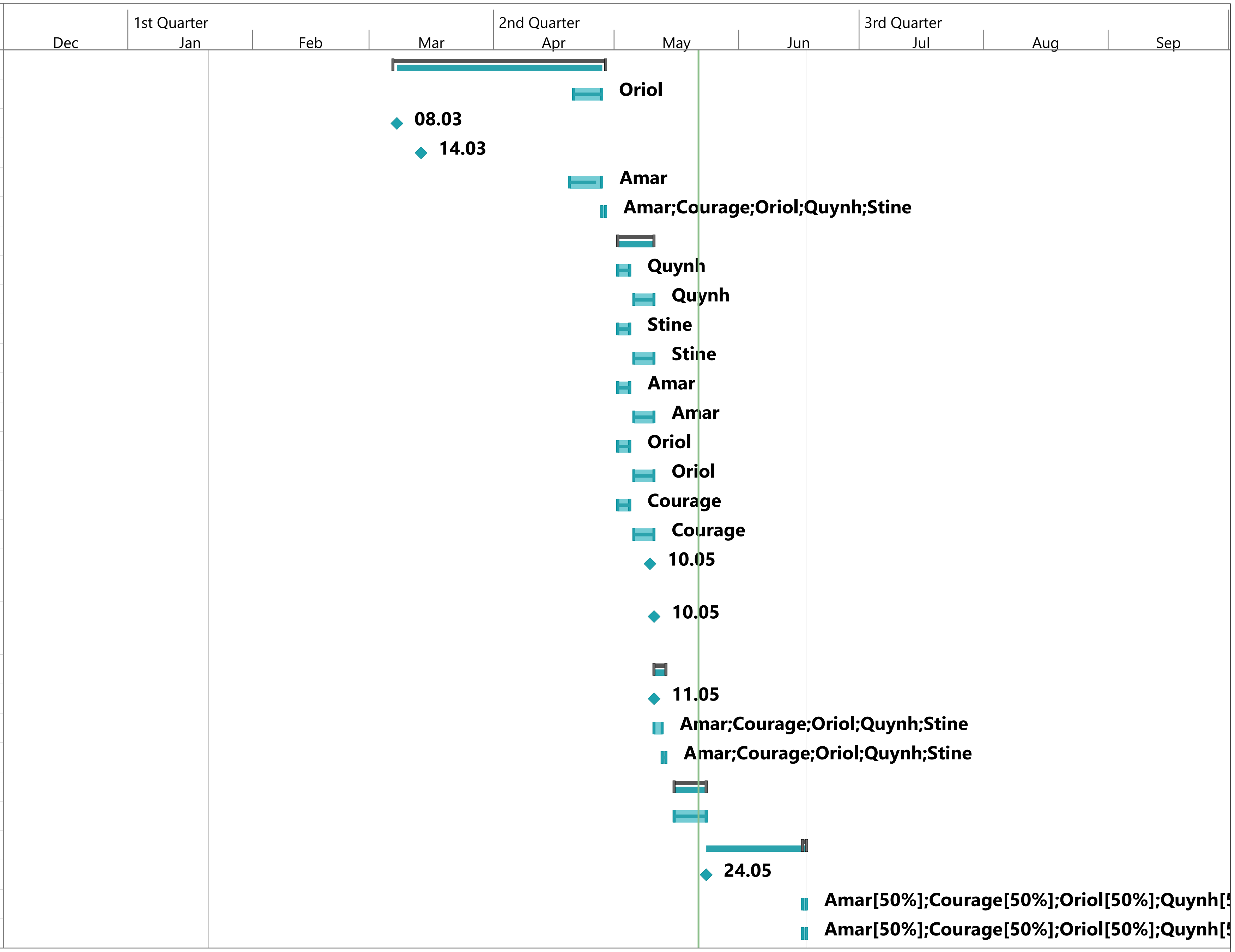
| ID | ℹ | Task Mode | Task Name | Duration | Start | Finish |
|---|---|---|---|---|---|---|
| 1 | ✓ | 📌 | Start | 1 day | Fri 22.01.16 | Fri 22.01.16 |
| 2 | ✓ | 📌 | **Planning** | **16 days** | **Fri 22.01.16** | **Fri 12.02.16** |
| 3 | ✓ | 📌 | Group agreement | 1 day | Fri 22.01.16 | Fri 22.01.16 |
| 4 | ✓ | 📌 | Gantt diagram | 1 day | Mon 25.01.16 | Mon 25.01.16 |
| 5 | ✓ | 📌 | Formulation of goals | 1 day | Mon 25.01.16 | Mon 25.01.16 |
| 6 | ✓ | 📌 | Research planning | 14 days | Tue 26.01.16 | Fri 12.02.16 |
| 7 | ✓ | 📌 | First formal meeting held | 0 days | Fri 12.02.16 | Fri 12.02.16 |
| 8 | ✓ | 📌 | **Research visits** | **42 days** | **Thu 21.01.16** | **Fri 08.04.16** |
| 9 | ✓ | 📌 | Ulefoss | 1 day | Thu 28.01.16 | Thu 28.01.16 |
| 10 | ✓ | 📌 | Skien kommune | 1 day | Tue 09.02.16 | Tue 09.02.16 |
| 11 | ✓ | 📌 | HSN Vestfold | 0 days | Fri 08.04.16 | Fri 08.04.16 |
| 12 | ✓ | 📌 | **Research** | **16 days** | **Fri 12.02.16** | **Fri 04.03.16** |
| 13 | ✓ | 📌 | Temp. sensor | 7 days | Mon 15.02.16 | Tue 23.02.16 |
| 14 | ✓ | 📌 | Gas sensor | 7 days | Wed 24.02.16 | Fri 04.03.16 |
| 15 | ✓ | 📌 | Level sensor | 8 days | Mon 15.02.16 | Wed 24.02.16 |
| 16 | ✓ | 📌 | Cover sensor | 6 days | Thu 25.02.16 | Fri 04.03.16 |
| 17 | ✓ | 📌 | Microcontroller | 8 days | Mon 15.02.16 | Wed 24.02.16 |
| 18 | ✓ | 📌 | Battery | 11 days | Mon 15.02.16 | Mon 29.02.16 |
| 19 | ✓ | 📌 | Bluetooth | 4 days | Mon 15.02.16 | Thu 18.02.16 |
| 20 | ✓ | 📌 | GSM | 4 days | Thu 25.02.16 | Wed 02.03.16 |
| 21 | ✓ | 📌 | SD-card | 7 days | Fri 19.02.16 | Mon 29.02.16 |
| 22 | ✓ | 📌 | Current demand for battery | 4 days | Tue 01.03.16 | Fri 04.03.16 |
| 23 | ✓ | 📌 | Total active schedule | 2 days | Tue 01.03.16 | Wed 02.03.16 |
| 24 | ✓ | 📌 | Design shematic | 2 days | Thu 03.03.16 | Fri 04.03.16 |
| 25 | ✓ | 📌 | Design of unit | 2 days | Thu 03.03.16 | Fri 04.03.16 |
| 26 | ✓ | 📌 | Research complete | 0 days | Fri 04.03.16 | Fri 04.03.16 |
| 27 | ✓ | 📌 | 2nd formal meeting | 0 days | Thu 10.03.16 | Thu 10.03.16 |
| 28 | ✓ | 📌 | 3rd formal meeting | 0 days | Wed 13.04.16 | Wed 13.04.16 |
| 29 |  | 📌 | **Programming & testing** | **30 days** | **Mon 07.03.16** | **Wed 11.05.16** |
| 30 | ✓ | 📌 | **Programming components** | **22 days** | **Mon 07.03.16** | **Thu 28.04.16** |
| 31 | ✓ | 📌 | Water level | 10 days | Mon 07.03.16 | Wed 06.04.16 |
| 32 | ✓ | 📌 | Temperature | 10 days | Mon 07.03.16 | Wed 06.04.16 |
| 33 | ✓ | 📌 | Gass content | 11 days | Thu 07.04.16 | Wed 27.04.16 |
| 34 | ✓ | 📌 | Cover sensor | 11 days | Thu 07.04.16 | Wed 27.04.16 |
| 35 | ✓ | 📌 | SD-card | 13 days | Mon 07.03.16 | Wed 13.04.16 |
| 36 | ✓ | 📌 | Microcontroller | 16 days | Mon 07.03.16 | Wed 20.04.16 |
| 37 | ✓ | 📌 | **System development** | **22 days** | **Mon 07.03.16** | **Thu 28.04.16** |
| 38 | ✓ | 📌 | SQL database | 3 days | Wed 30.03.16 | Mon 04.04.16 |
| 39 | ✓ | 📌 | GUI | 1 day | Wed 06.04.16 | Wed 06.04.16 |
| 40 | ✓ | 📌 | Coding | 7 days | Thu 07.04.16 | Thu 21.04.16 |
| 41 | ✓ | 📌 | Server | 3 days | Mon 25.04.16 | Wed 27.04.16 |



Project: gantt
Date: Sun 22.05.16

|  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|
| Task | ▬ | Project Summary | ┌─┐ | Manual Task | ▬ | Start-only | [ | Deadline | ⇩ |
| Split | ┄ | Inactive Task | ▭ | Duration-only | ▬ | Finish-only | ] | Progress | ─ |
| Milestone | ◆ | Inactive Milestone | ◇ | Manual Summary Rollup | ▬ | External Tasks | ▬ | Manual Progress | ─ |
| Summary | ┌─┐ | Inactive Summary | ┌─┐ | Manual Summary | ┌─┐ | External Milestone | ◇ |  |  |

| ID | ⓘ | Task Mode | Task Name | Duration | Start | Finish |
|---|---|---|---|---|---|---|
| 42 | | 📌 | **Communication** | **22 days** | **Mon 07.03.16** | **Thu 28.04.16** |
| 43 | ✓ | 📌 | GSM/GPRS/EDGE | 5 days | Thu 21.04.16 | Wed 27.04.16 |
| 44 | ✓ | 📌 | Simple test of GSM | 0 days | Tue 08.03.16 | Tue 08.03.16 |
| 45 | ✓ | 📌 | Test of GSM w/ results | 0 days | Mon 14.03.16 | Mon 14.03.16 |
| 46 | | 📌 | Bluetooth | 6 days | Wed 20.04.16 | Wed 27.04.16 |
| 47 | | 📌 | Deployment | 1 day | Thu 28.04.16 | Thu 28.04.16 |
| 48 | | 📌 | **Testing** | **6 days** | **Mon 02.05.16** | **Tue 10.05.16** |
| 49 | ✓ | 📌 | Water level | 3 days | Mon 02.05.16 | Wed 04.05.16 |
| 50 | ✓ | 📌 | Cover sensor | 3 days | Fri 06.05.16 | Tue 10.05.16 |
| 51 | ✓ | 📌 | Temperature | 3 days | Mon 02.05.16 | Wed 04.05.16 |
| 52 | ✓ | 📌 | Gass content | 3 days | Fri 06.05.16 | Tue 10.05.16 |
| 53 | ✓ | 📌 | SD-card | 3 days | Mon 02.05.16 | Wed 04.05.16 |
| 54 | ✓ | 📌 | Bluetooth | 3 days | Fri 06.05.16 | Tue 10.05.16 |
| 55 | ✓ | 📌 | Microcontroller | 3 days | Mon 02.05.16 | Wed 04.05.16 |
| 56 | ✓ | 📌 | GSM/Antenna | 3 days | Fri 06.05.16 | Tue 10.05.16 |
| 57 | ✓ | 📌 | System | 3 days | Mon 02.05.16 | Wed 04.05.16 |
| 58 | ✓ | 📌 | Power consumption | 3 days | Fri 06.05.16 | Tue 10.05.16 |
| 59 | | 📌 | Programming and testing of system done | 0 days | Tue 10.05.16 | Tue 10.05.16 |
| 60 | ✓ | 📌 | Programming and testing of unit done | 0 days | Tue 10.05.16 | Tue 10.05.16 |
| 61 | | 📌 | **Assembly of unit** | **3 days** | **Wed 11.05.16** | **Fri 13.05.16** |
| 62 | ✓ | 📌 | 4th formal meeting | 0 days | Wed 11.05.16 | Wed 11.05.16 |
| 63 | | 📌 | Connect and finalize | 2 days | Wed 11.05.16 | Thu 12.05.16 |
| 64 | ✓ | 📌 | User manual/Inst. Guide | 1 day | Fri 13.05.16 | Fri 13.05.16 |
| 65 | ✓ | 📌 | **Report writing** | **5 days** | **Mon 16.05.16** | **Mon 23.05.16** |
| 66 | ✓ | 📌 | Write and finalize report | 5 days | Mon 16.05.16 | Mon 23.05.16 |
| 67 | | 📌 | **Presentation** | **1 day** | **Fri 17.06.16** | **Fri 17.06.16** |
| 68 | | 📌 | Report delivered | 0 days | Tue 24.05.16 | Tue 24.05.16 |
| 69 | | 📌 | Demonstration | 1 day | Fri 17.06.16 | Fri 17.06.16 |
| 70 | | 📌 | Powerpoint/prezi | 1 day | Fri 17.06.16 | Fri 17.06.16 |

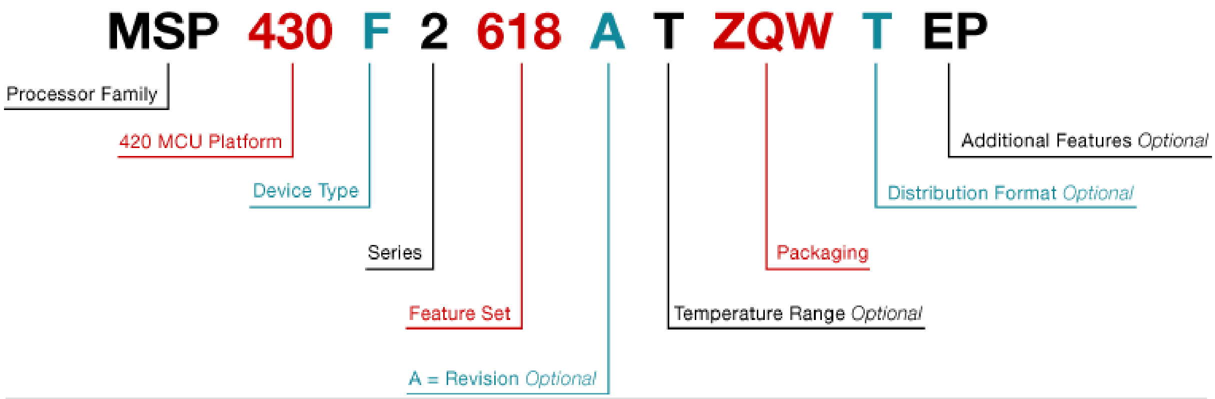


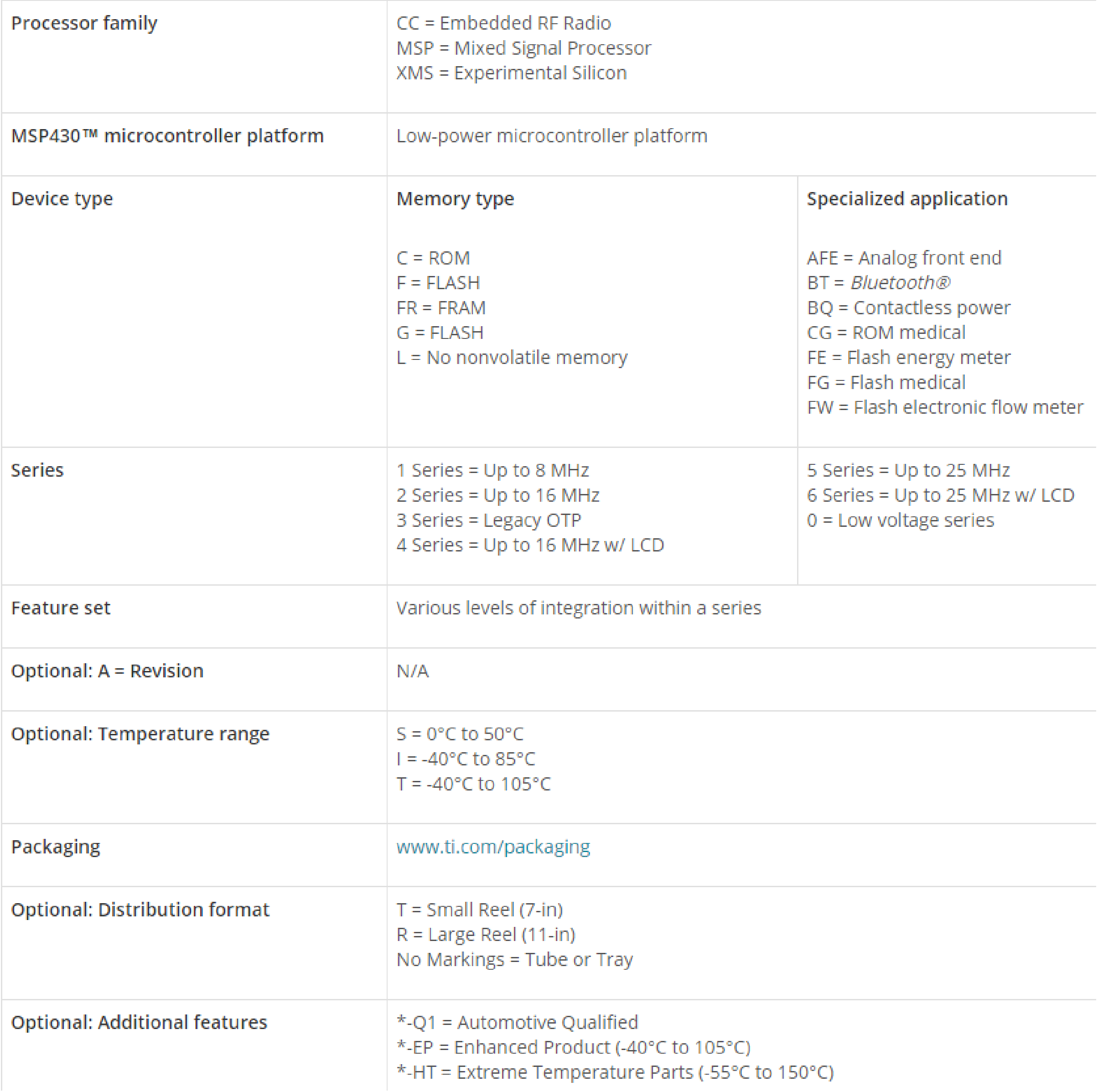
Project: gantt
Date: Sun 22.05.16

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | ▬ | Project Summary | ▬ | Manual Task | ▬ | Start-only | [ | Deadline | ⬇ |
| Split | ┄ | Inactive Task | ▭ | Duration-only | ▬ | Finish-only | ] | Progress | ▬ |
| Milestone | ◆ | Inactive Milestone | ◇ | Manual Summary Rollup | ▬ | External Tasks | ▬ | Manual Progress | ▬ |
| Summary | ▬ | Inactive Summary | ▭ | Manual Summary | ▬ | External Milestone | ◇ | | |

**Attachment E**

# MSP430FR5969 Letters explanation


MSP 430 F 2 618 A T ZQW T EP
Processor Family
420 MCU Platform
Device Type
Series
Feature Set
A = Revision Optional
Temperature Range Optional
Packaging
Distribution Format Optional
Additional Features Optional



| | | |
|---|---|---|
| **Processor family** | CC = Embedded RF Radio<br>MSP = Mixed Signal Processor<br>XMS = Experimental Silicon | |
| **MSP430™ microcontroller platform** | Low-power microcontroller platform | |
| **Device type** | **Memory type**<br><br>C = ROM<br>F = FLASH<br>FR = FRAM<br>G = FLASH<br>L = No nonvolatile memory | **Specialized application**<br><br>AFE = Analog front end<br>BT = *Bluetooth®*<br>BQ = Contactless power<br>CG = ROM medical<br>FE = Flash energy meter<br>FG = Flash medical<br>FW = Flash electronic flow meter |
| **Series** | 1 Series = Up to 8 MHz<br>2 Series = Up to 16 MHz<br>3 Series = Legacy OTP<br>4 Series = Up to 16 MHz w/ LCD | 5 Series = Up to 25 MHz<br>6 Series = Up to 25 MHz w/ LCD<br>0 = Low voltage series |
| **Feature set** | Various levels of integration within a series | |
| **Optional: A = Revision** | N/A | |
| **Optional: Temperature range** | S = 0°C to 50°C<br>I = -40°C to 85°C<br>T = -40°C to 105°C | |
| **Packaging** | www.ti.com/packaging | |
| **Optional: Distribution format** | T = Small Reel (7-in)<br>R = Large Reel (11-in)<br>No Markings = Tube or Tray | |
| **Optional: Additional features** | *-Q1 = Automotive Qualified<br>*-EP = Enhanced Product (-40°C to 105°C)<br>*-HT = Extreme Temperature Parts (-55°C to 150°C) | |

Attachment F

**MARECHAL ELECTRIC S.A.S.**
Au capital de 5 207 500 €
5, avenue de Presles - F-94417 Saint-Maurice Cedex
Tél.: +33 (0)1 45 11 60 00 - Fax: +33 (0)1 45 11 60 60

SIRET 552 149 577 00058 - TVA n° FR16 552 149 577 - NAF 2733Z

**marechal.com**

MARECHAL ELECTRIC
Technor®

| For the attention of | Quotation |
|---|---|
| Mlle Quynh Nguyen | Reference : 54343/0 Must be mentioned in your P.O |
| UNIVERSITY COLLEGE OF SOUTHEAST NORWAY | **Subject : USN - Request PNCX for Manholes** |
| Høgskolen i Sørøst-Norge Postboks 235 | Created on : 02/03/2016 |
| 3603 Oslo | Entered by : Anwar MOUHASSINE |
| NORVEGE | |
| Tel : +47 46274252<br>email : thaoquynh86@yahoo.com | Validity date : 01/04/2016 |

| Your contacts | |
|---|---|
| Commercial Engineer | Customer Service |
| Anwar MOUHASSINE | Nathalie BLAYAC |
| Tel : +33 6 76 20 75 87 | Tel :+33 1 45 11 60 00 |
| Fax : +33 1 45 11 60 60 | Fax : +33 1 45 11 60 60 |
| a.mouhassine@marechal.com | n.blayac@marechal.com |

Dear Ms Nguyen,

Please find attached our offer as per your requirement

Please check Compatibility of your customer’s cable with our handles and wall boxes entrance.

Feel free to contact us if you have any further requirement.

Best Regards,

Anwar Mouhassine
Export Sales Engineer

**MARECHAL ELECTRIC S.A.S.**
Au capital de 5 207 500 €
5, avenue de Presles - F-94417 Saint-Maurice Cedex
Tél.: +33 (0)1 45 11 60 00 - Fax: +33 (0)1 45 11 60 60

SIRET 552 149 577 00058 - TVA n° FR16 552 149 577 - NAF 2733Z

**marechal.com**

MARECHAL ELECTRIC
Technor®

| | | |
|---|---|---|
| Subject : USN - Request PNCX for Manholes | | |
| Entered by : Anwar MOUHASSINE | Quotation reference : 54343/0 | On 02/03/2016 |
| Company : University College of Southeast Norway | Validity date : 01/04/2016 | |

| N° | Code | Description | Qty | Unit price excl. VAT | Total |
|---|---|---|---|---|---|
| 1 | 06E3007 | PNCX SOCKET II2GD Ex e IIC Gb - Ex tb IIIC Db T6 POLY BLACK +HANDLE IP66/67 5C 5A 250V M20 7-14MM | 3 | 42,00 | 126,00 € |
| 2 | 06E1007 | PNCX INLET II2GD Ex e IIC Gb - Ex tb IIIC Db T6 POLY BLACK +HANDLE IP66/67 5C 5A 250V M20 7-14MM | 3 | 31,50 | 94,50 € |
| | | | | Subtotal excl. VAT | 220,50 € |
| | | | | Freight costs | |
| | | | | VAT 0.0% | 0,00 € |
| | | | | Total incl. VAT | 220,50 € |

All prices are quoted in €uro (EUR) excluding Taxes
Incoterm (delivery terms) : EXW, Maromme, France (postal code : 76150)
Materials of French origin and manufacture
Harmonized System Code : 85366990
Lead-time : shipment within 1 week to 10 days upon order confirmation
Payment at order placement before start of production
Minimum order value : 250.00 €

WARNING ! Delivered products are non-returnable and will not be refunded or exchanged. Please check the product part numbers and/or the compatibility of the products with those used by you. We are at your service for any assistance you may require.

University College of Southeast Norway
Faculty of technology
Bachelor of Science

**REPORT FROM 6$^{TH}$ SEMESTER PROJECT SPRING 2016**

PRH612 Bachelor thesis

IA6-5-16

# Attachment G

# SOFTWARE REQUIREMENTS SPECIFICATIONS AND SOFTWARE DESIGN DOCUMENT

**Address: Kjølnes ring 56, N-3918 Porsgrunn, Tel: +47 31 00 80 00, www.usn.no**

Bachelor's programmes – Master's programmes – Ph.D. programmes

# TABLE OF CONTENT

# 1 INTRODUCTION

This document tells about the usefulness of the software and the technical requirements needed to meet the acceptable criteria. It also describes the conditions and constrains required to successfully operate the program.

# 2 PURPOSE AND BENEFITS

The software is to enable users to interact with data from Read, Control and Communication unit for manholes in an easy and friendly manner.  It is to inform users about normal and abnormal situations in the manholes and display useful data from manholes for planning, safety and maintenance.

# 3 TECHNICAL REQUIREMENTS

This aspect sets out and describe the systems functions and services. It also describes how the system is expected to react to a particular input.

## 3.1 Functions

This is how the software is required to react to inputs from the users. The program shall have two type of users, ordinary users and administrators. It should be able to display data from the manholes and inform users about situations in the manhole. Users should be able to search for relevant data through a given search criteria. There should be possibility to save and print data. ‘

Administrators should be able to register and edit information about manholes. They should also have the capability to change administrator’s password and company information.

Normal and abnormal situations shall be displayed continuously on the home page as alarms. Both manhole details, Alarm history, Data from manholes and company’s information shall be displayed in less than a second when users navigate to their various pages from the home page. Admin page should be displayed in less than three seconds after login. Changes made by an administrator should be saved in the database immediately.

## 3.2 User Interface

This is the user’s interactive section with the application.

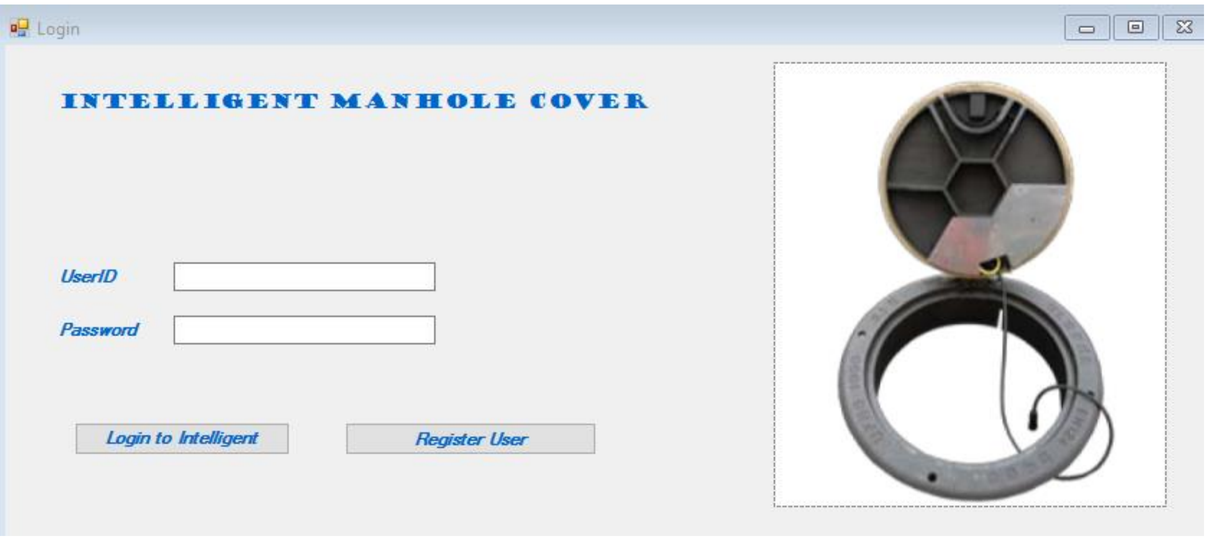


Figure 3.2.1 User Interface

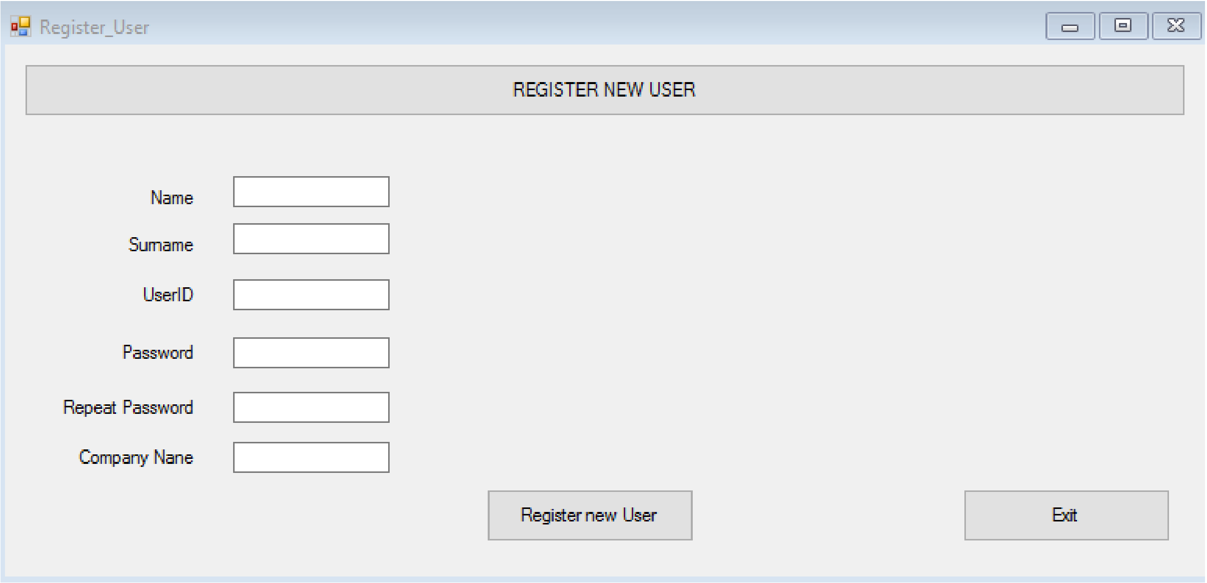


Figure 3.2.2 Register new User

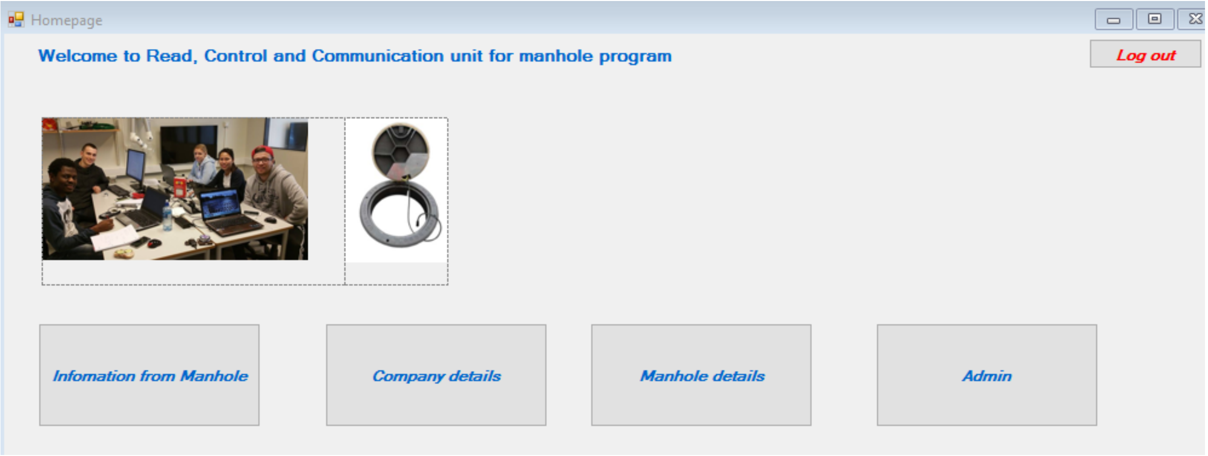
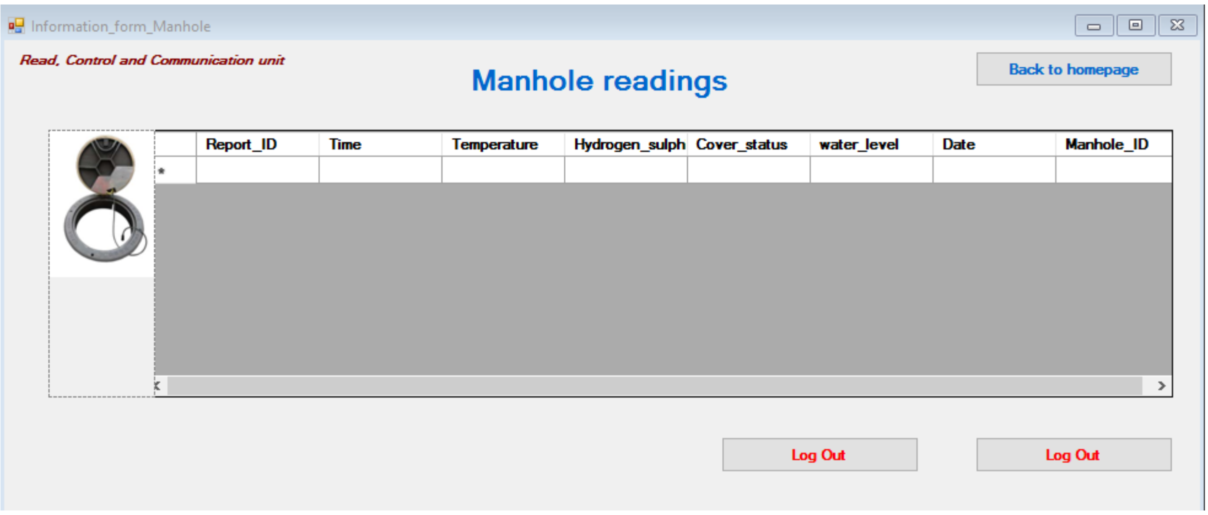
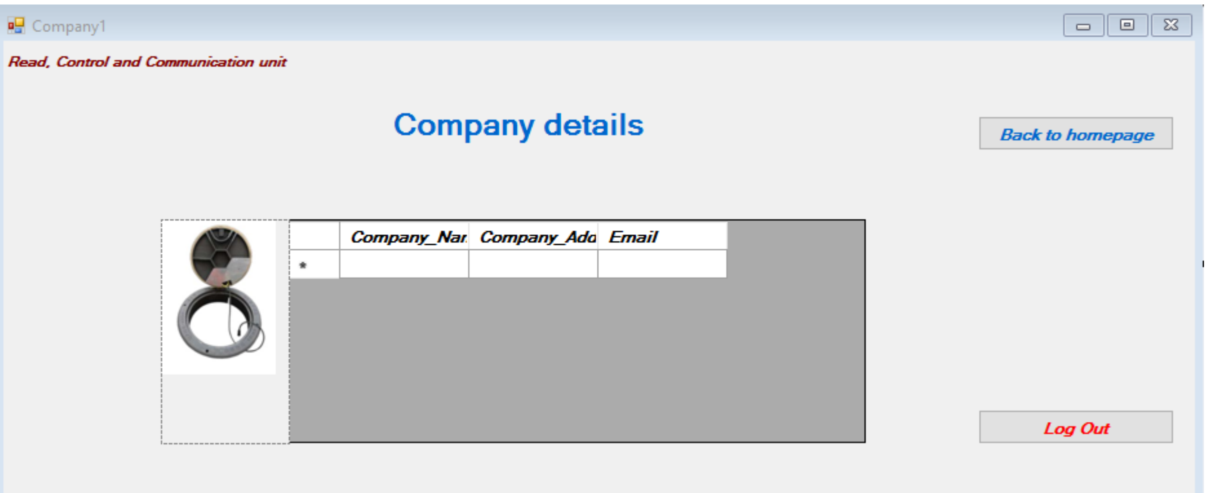


Figure 3.2.3 Home page



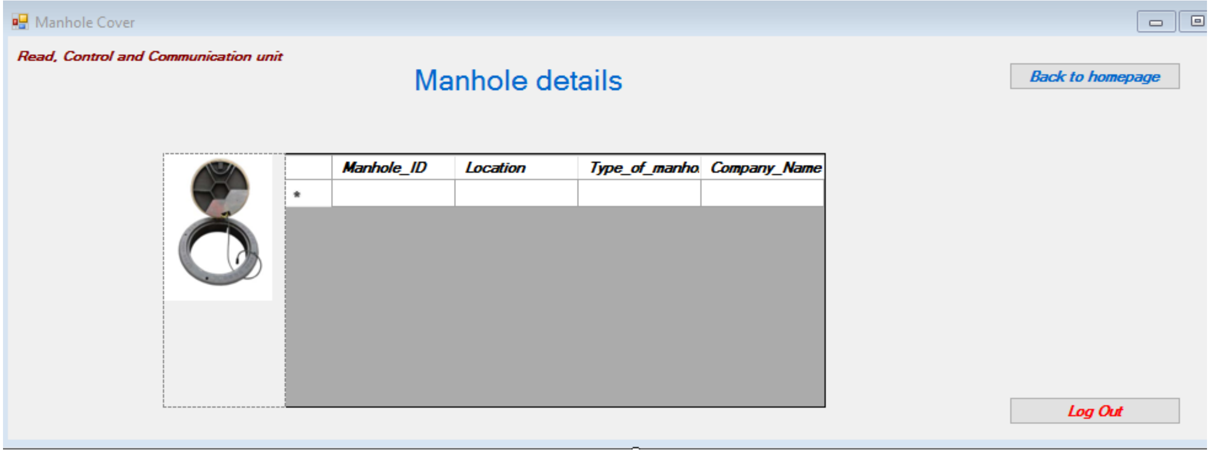
| | Report_ID | Time | Temperature | Hydrogen_sulph | Cover_status | water_level | Date | Manhole_ID |
|---|---|---|---|---|---|---|---|---|
| * | | | | | | | | |

Figure 3.2.4 Readings from manhole



|  | Company_Nar | Company_Adc | Email |
|---|---|---|---|
| * |  |  |  |

Figure 3.2.5 Company details



|  | Manhole_ID | Location | Type_of_manho | Company_Name |
|---|---|---|---|---|
| * |  |  |  |  |

Figure 3.2.6 Manhole details

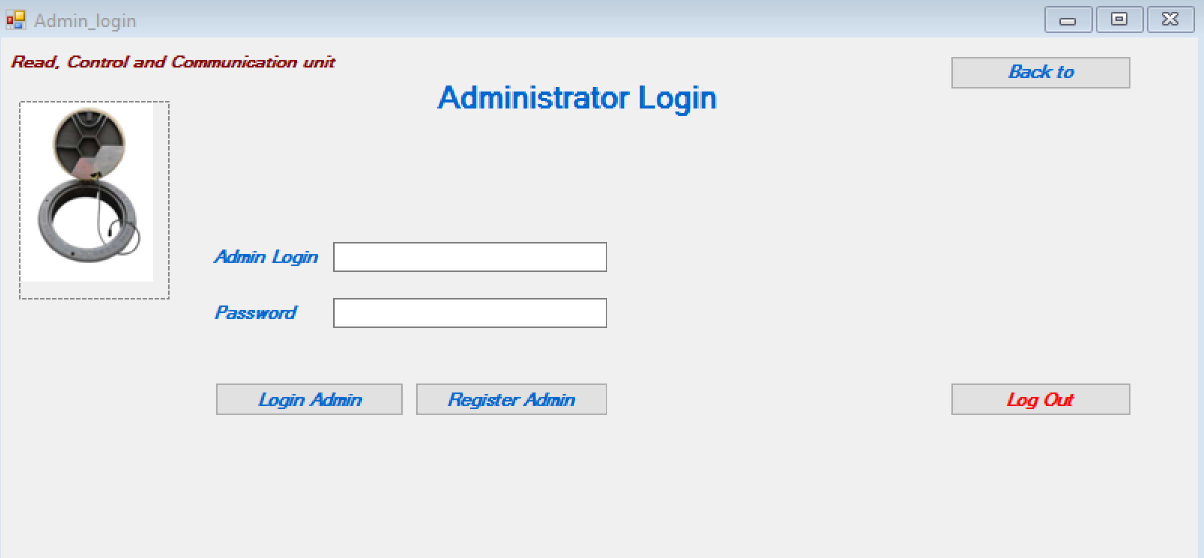


Figure 3.2.7 Admin Login

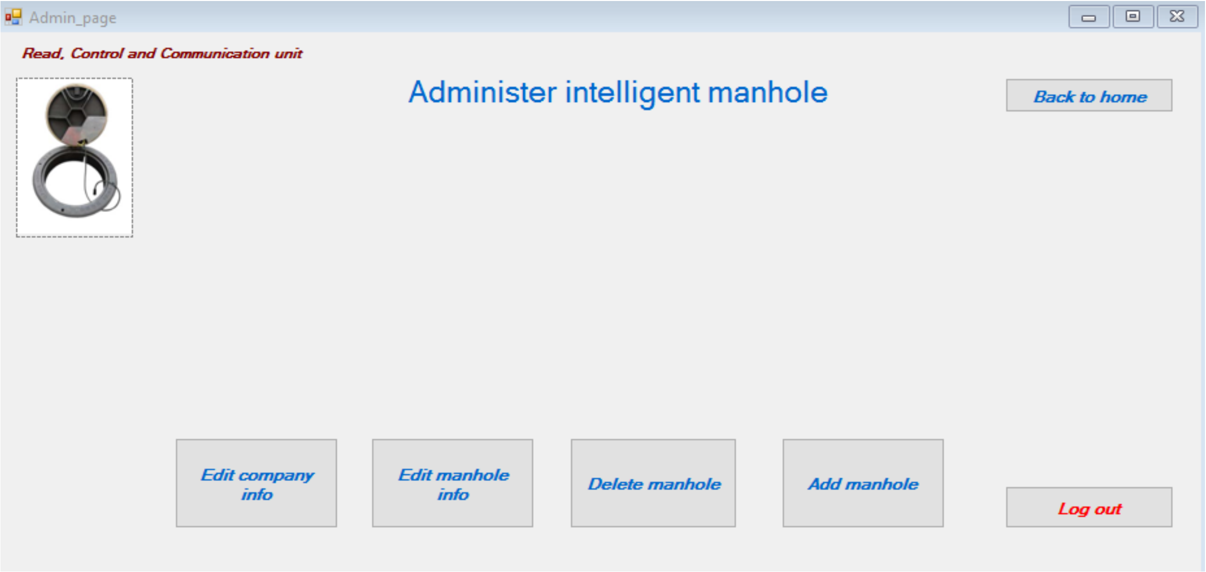


Figure 3.2.8 Admin page

## 3.3 User Task Flow

This shows how users navigate through the program templates to view data from the Database and register/ edit data about manholes and administrators.

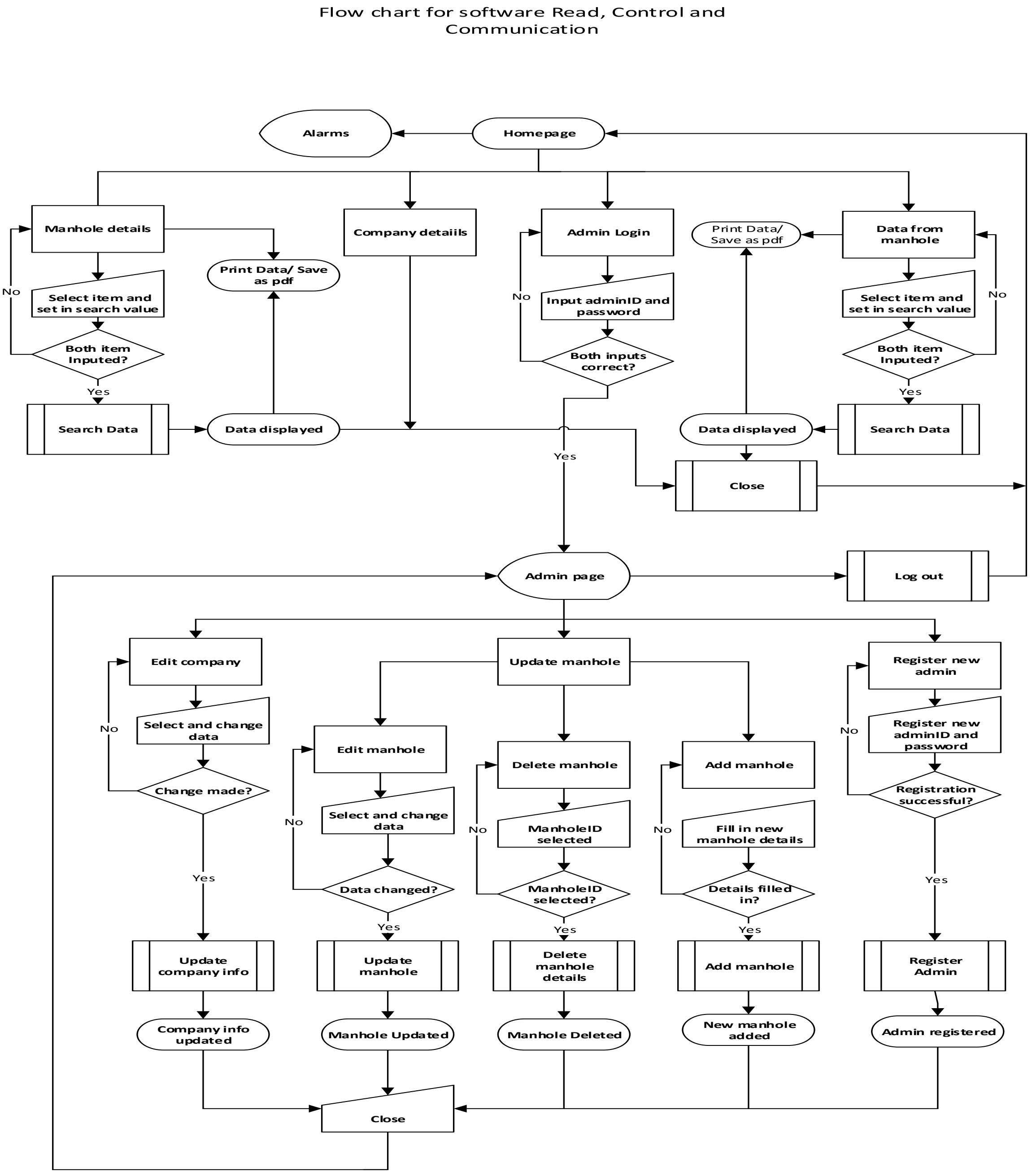


Figure 3.3.1Flow chart

## 3.4 Software module

The application has two main module, the visual studio application and the Database. Visual studio has been used to develop the Graphical Users Interface (GUI) as well as the codes. While the Database helps to save both data from the manholes and manhole’s information.

Data from the manhole are sent from the system unit via GSM/GPRS to the Database. These data are displayed on the GUI with the help of the programming codes for users view. Data are also sent from the GUI to the database by an administrator. Administrator send data to the database both when new manholes are registered or when an editing task is performed. In other words, both GUI and the Database work hand-in-hand to execute the software task with the help of the programming codes.

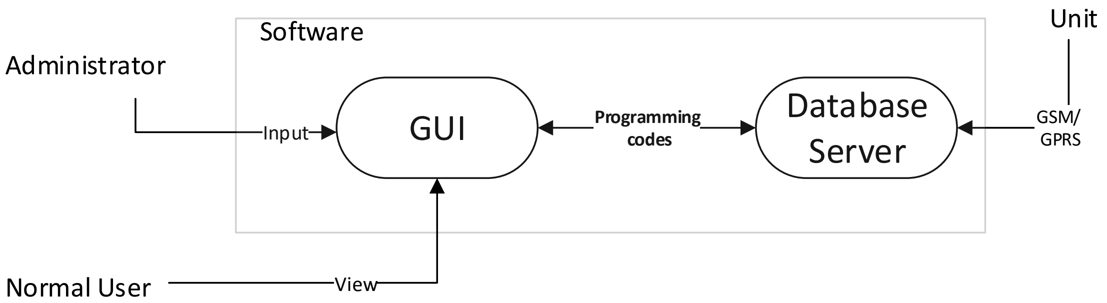


Figure 3.4.1 System Modul

## 3.5 ER- Diagram

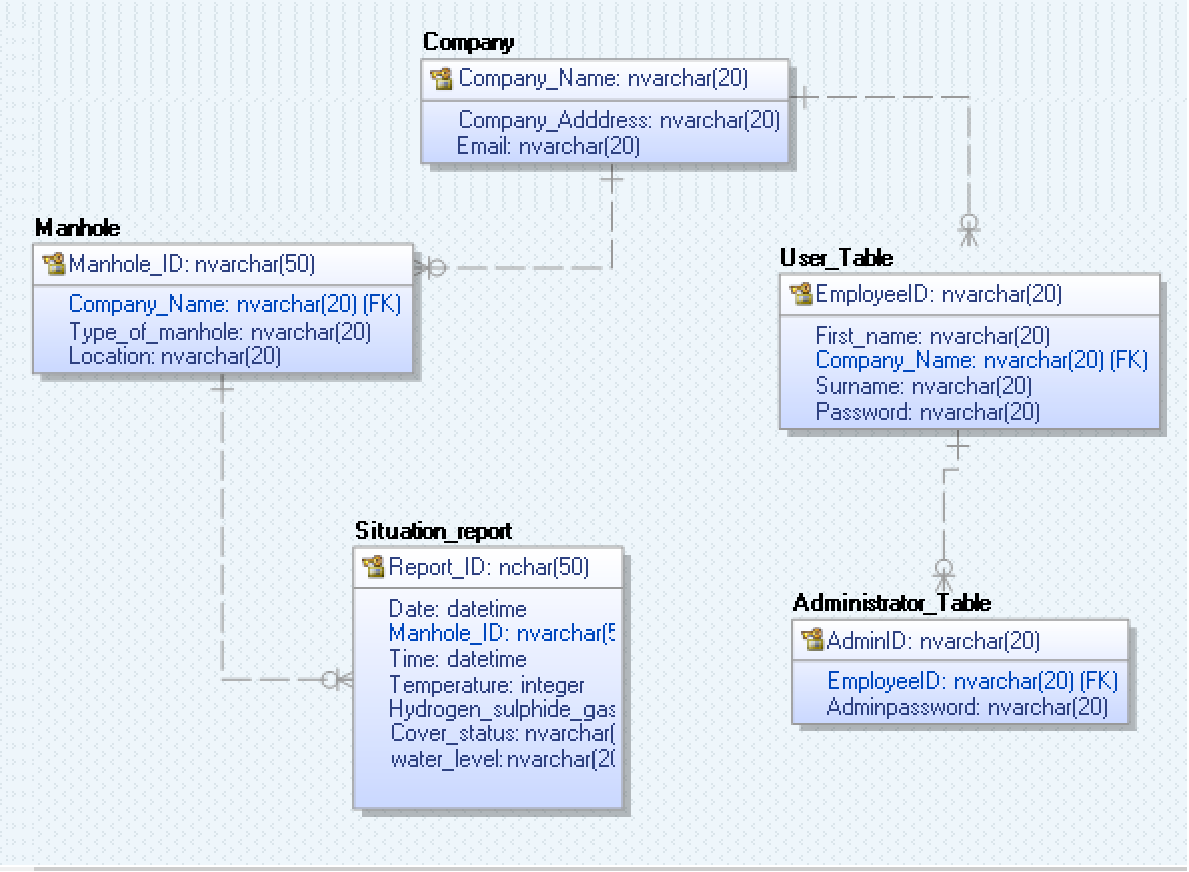


Figure 3.5.1 ER - Diagram

## 3.6 Class Diagram

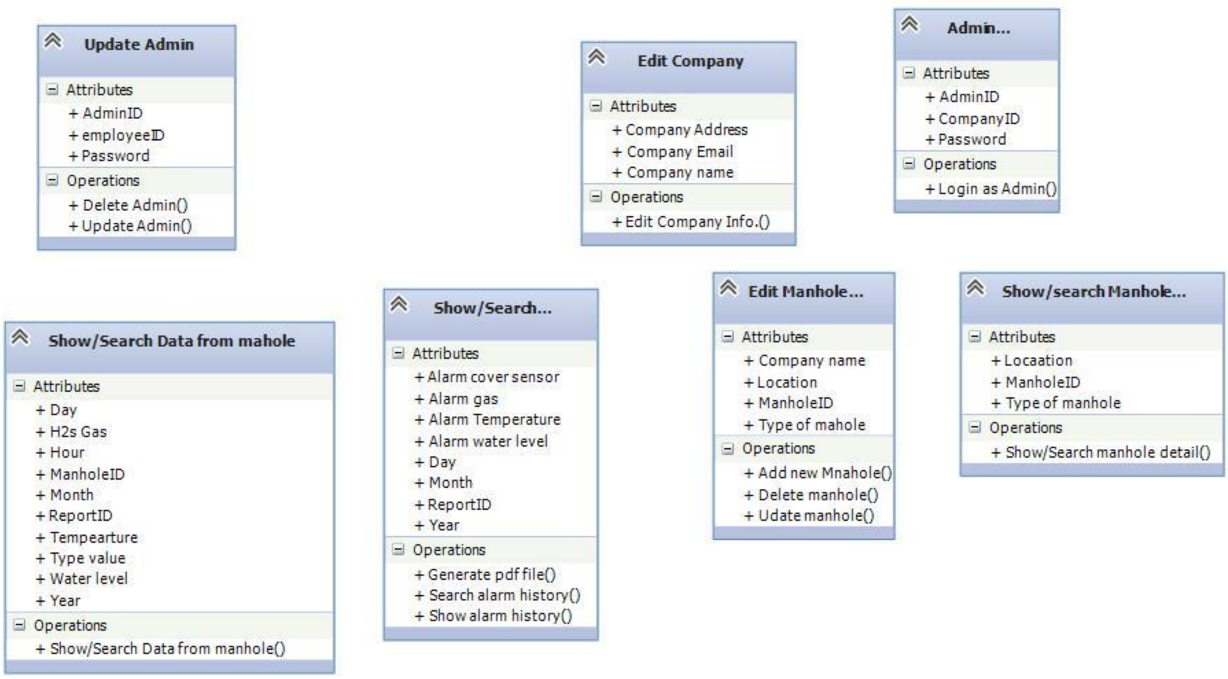


Figure 3.6.1 Class Diagram

## 3.7 Use Case Diagram

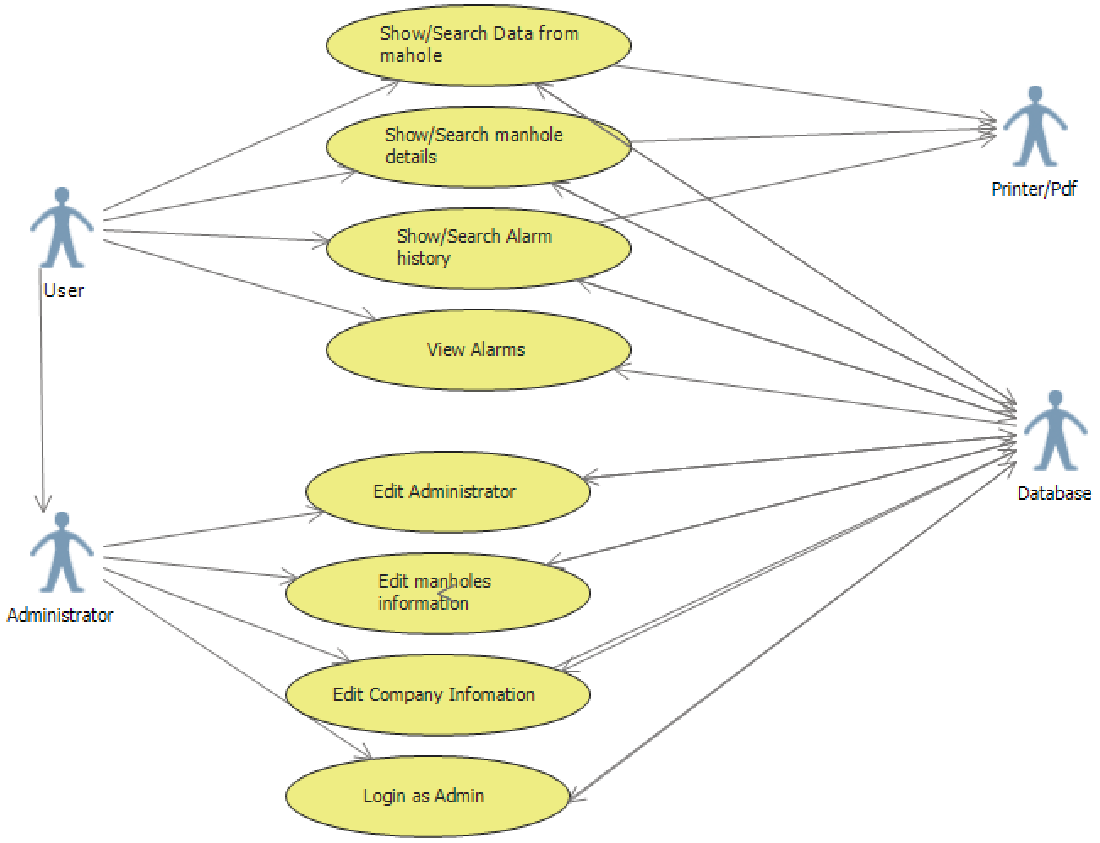

Show/Search Data from mahole
Show/Search manhole details
Show/Search Alarm history
View Alarms
Edit Administrator
Edit manholes information
Edit Company Infomation
Login as Admin
User
Administrator
Printer/Pdf
Database

Figure 3.7.1 Use Case

# 4 ACCEPTANCE CRITERIA

The application is expected to work properly without any form of error and should be easy to use by both regular users and administrator.

# 5 REQUIREMENT CONSIDERATIONS

This explains the condition and constrains needed to achieve the purpose and benefits of the Software.

## 5.1 Assumptions made about the software

This software application shall be unbounded. It should be possible to develop an updated version by the software developers to satisfy end users requirement.

## 5.2 End User

After the software has been fully developed, installed and tested; Users who download the program to their working computer can see data from the manholes. Only users who have access to the admin ID and password can register new manhole and edit both manhole details, admin and company information.

## 5.3 Existing System

No external or existing system is required for the software to function properly.

## 5.4 Environment

Language Environment - This software language will be in English.

Operating Environment – PC operating Microsoft Windows 10.

## 5.5 Limitations

- ✓ Software cannot support Multilanguage system
- ✓ Software operates only when the program is installed in the working computer and a database is developed.

ATTACHMENT H
SOFTWARE CODE

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;



namespace Smart_manhole_cover
{
    public partial class Homepage : Form
    {

             public Homepage()
        {
            InitializeComponent();

        }

        private void button2_Click(object sender, EventArgs e)
        {

            Information_form_Manhole page4 = new Information_form_Manhole(); // make object of class
            page4.Show();// show page 4

        }

        private void button1_Click(object sender, EventArgs e)
        {
            Manhole_Details page5 = new Manhole_Details();// make object of class
            page5.Show(); // show page 5

        }

        private void button3_Click(object sender, EventArgs e)
        {
            Admin_login page9 = new Admin_login(); // make object of class
            page9.Show(); // show page 9


        }

        private void button4_Click(object sender, EventArgs e)
        {
            Company1 page7 = new Company1(); // make object of class
            page7.Show(); // show page 7

        }



        private void Homepage_Load(object sender, EventArgs e)
        {


            try
            {
               // Connection string
                SqlConnection Conn = new SqlConnection(Properties.Settings.Default.DBCS);

                // SQL Querry
                string SelectQuerry = @"select Alarm_Cover, Alarm_water_level, Alarm_Temp, Alarm_Gas from↙
     Situation_report where Report_ID = (select max(Report_ID) from Situation_report)";
                SqlCommand SelectCmd = new SqlCommand(SelectQuerry, Conn);

                // Open Connection
                Conn.Open();

                // Read  and execute Querry
                SqlDataReader reader = SelectCmd.ExecuteReader();

                // While statement
                while (reader.Read())
                {

                    // Assign database value to textbox
                    TxtAlarm.Text = (reader["Alarm_Cover"].ToString());
                    TxtWater.Text = (reader["Alarm_water_level"].ToString());
                    TxtGas.Text = (reader["Alarm_Gas"].ToString());
                    TxtTemp.Text = (reader["Alarm_Temp"].ToString());

                    //if statement to assign color to bottons
                    if (TxtAlarm.Text == "1")
                    {
                        BtnCover.BackColor = Color.Red;
                    }
                    else
                    {
                        BtnCover.BackColor = Color.Green;

                    }
                     if (TxtWater.Text == "1")
                    {
                        BtnWater.BackColor = Color.Red;

                    }
                          else
                    {
                        BtnWater.BackColor = Color.Green;

                    }

                    if (TxtGas.Text == "1")
                    {
                        BtnH2s.BackColor = Color.Red;

                    }
                         else
                    {
                        BtnH2s.BackColor = Color.Green;

                    }
                    if (TxtTemp.Text == "1")
                    {
                        BtnTemp.BackColor = Color.Red;

                    }
                    else
                    {

                        BtnTemp.BackColor = Color.Green;

                    }
                }


                reader.Close(); // Close reader

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.   ↙
    Error);

            }


        }


        private void button6_Click(object sender, EventArgs e)
        {
           Alarm yy = new Alarm(); // Operate object of a class
           yy.Show(); // Show yy

```

```
        }

        private void dataFromManholesToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Information_form_Manhole page4 = new Information_form_Manhole(); // Operate object of a class
            page4.Show(); // Show page4
        }

        private void userToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Alarm yy = new Alarm();  // Operate object of a class
            yy.Show(); // Show yy
        }

        private void companyDetailsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Company1 page7 = new Company1(); // Operate object of a class
            page7.Show();  // Show page 7

        }

        private void manholeDetailsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Manhole_Details page5 = new Manhole_Details(); // Operate object of a class
            page5.Show(); // Show page5
        }

        private void adminToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            Admin_login page9 = new Admin_login(); // Operate object of a class
            page9.Show(); //Show page9
        }







    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;


namespace Smart_manhole_cover
{
    public partial class Information_form_Manhole : Form
    {

        // Variable
        SqlDataAdapter sda;
        DataTable dt;
        SqlCommandBuilder scb;


        public Information_form_Manhole()
        {
            InitializeComponent();


        }

        private void Information_form_Manhole_Load(object sender, EventArgs e)
        {

            try {

                //Connection string
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);

             //SQL Querry statement
            string CompQuery = @"SELECT Report_ID, Temperature, Hydrogen_sulphide_gass, water_level, Manhole_ID, Hour, Month, Day, Month, Year   from Situation_report ";

                // Assign parameters to object of class

            sda = new SqlDataAdapter(CompQuery, con);
            dt = new DataTable();

                // fiil datagridview
            sda.Fill(dt);
            dataGridView1.DataSource = dt;



            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }



        private void button4_Click(object sender, EventArgs e)
        {

            // objects of classes
            Document doc = new Document(iTextSharp.text.PageSize.LETTER, 10, 10, 42, 35);
            PdfWriter wri = PdfWriter.GetInstance(doc, new FileStream("Test.pdf", FileMode.Create));

            //Open document to write
            doc.Open();

            //write some content)
            Paragraph paragraph = new Paragraph("  ......................................................↙
    ....... Control and Communication Device for manhole  ..... .........................................↙
    ... \n\n\n  ");
            doc.Add(paragraph);

            // creating list in pdf file
            List list = new List(List.UNORDERED);

            // list starts with the space of 30f
            list.IndentationLeft = 30f;
            list.Add(new ListItem("............................................................ Data    ↙
    from manholeRead ............................................................  \n\n\n"));
            //list.Add(" \n\n\n");
            doc.Add(list);



            //  make object of class
            PdfPTable table = new PdfPTable(dataGridView1.Columns.Count);

            for (int j = 0; j < dataGridView1.Columns.Count; j++)
            {
                table.AddCell(new Phrase(dataGridView1.Columns[j].HeaderText));

            }

            table.HeaderRows = 1;

            for (int i = 0; i < dataGridView1.Rows.Count; i++)
            {
                for (int k = 0; k < dataGridView1.Columns.Count; k++)
                {
                    if (dataGridView1[k, i].Value != null)
                    {
                        table.AddCell(new Phrase(dataGridView1[k, i].Value.ToString()));
                    }
                }
            }

            doc.Add(table);

            //close document
            doc.Close();

            // show message
            MessageBox.Show("Page is sucussfully saved as pdf in Test file");


            PrintDialog pd = new PrintDialog();
            pd.ShowDialog();
        }

        private void SearchTxt_TextChanged(object sender, EventArgs e)
        {
            if ( SearchTxt.Text == "" )
            {
                try
                {
                   // Connection string
                    SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);

                    // SQL Querry statement
                    string CompQuery = @"SELECT Report_ID, Temperature, Hydrogen_sulphide_gass,        ↙
    water_level, Manhole_ID, Hour, Month, Day, Month, Year   from Situation_report ";

                    // assign parameters to class sda
                    sda = new SqlDataAdapter(CompQuery, con);
                    dt = new DataTable();

                    // fill datagridview
                    sda.Fill(dt);
                    dataGridView1.DataSource = dt;



                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }


        }

        private void button1_Click(object sender, EventArgs e)
        {


            try
            {
               // SqlConnection con = new SqlConnection(ConString);
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);

                con.Open(); //open connection

                // if statement
                if (comboBox3.Text == "Manhole cover status (input 1 for open status,  input 2 for Close staus)")
                {

                    // Querry statement
                    string CompQuery = @"SELECT Report_ID, Date, Time, Cover_status, Manhole_ID FROM Situation_report WHERE Alarm_Cover LIKE '" + SearchTxt.Text + "%'";

                    sda = new SqlDataAdapter(CompQuery, con);
                    dt = new DataTable();
                    sda.Fill(dt);
                    dataGridView1.DataSource = dt;


                }

                else if (comboBox3.Text == "ManholeID")
                {
                    // Querry statement
                    string CompQuery = @"SELECT * FROM Situation_report WHERE Manhole_ID LIKE '" + SearchTxt.Text + "%'";

                    //assign parameters to object of class SqlDataAdapter
                    sda = new SqlDataAdapter(CompQuery, con);
                    dt = new DataTable();

                    //Fill datagridview
                    sda.Fill(dt);
                    dataGridView1.DataSource = dt;

                }

                else if (comboBox3.Text == "Temperature")
                {

                    string CompQuery = @"SELECT * FROM Situation_report WHERE Temperature LIKE '" + SearchTxt.Text + "%'";

                    sda = new SqlDataAdapter(CompQuery, con);
                    dt = new DataTable();
                    sda.Fill(dt);
                    dataGridView1.DataSource = dt;

                }

                else if (comboBox3.Text == "ReportID")
```

```
                {
                    // SQL Querry statement
                    string CompQuery = @"SELECT * FROM Situation_report WHERE Report_ID LIKE '" + SearchTxt.Text + "%'";


                    //assign parameters to object of class SqlDataAdapter
                    sda = new SqlDataAdapter(CompQuery, con);
                    dt = new DataTable();

                    // fill datagridview
                    sda.Fill(dt);
                    dataGridView1.DataSource = dt;

                }

                else if (comboBox3.Text == "Hydrogen sulphide gas")
                {
                    //  SQL Querry statement
                    string CompQuery = @"SELECT * FROM Situation_report WHERE Hydrogen_sulphide_gass LIKE '" + SearchTxt.Text + "%'";


                    //assign parameters to object of class SqlDataAdapter
                    sda = new SqlDataAdapter(CompQuery, con);
                    dt = new DataTable();

                    //Fill datagridview
                    sda.Fill(dt);
                    dataGridView1.DataSource = dt;

                }

                else if (comboBox3.Text == "Water level")
                {

                    // SQL Querry statement
                    string CompQuery = @"SELECT * FROM Situation_report WHERE water_level LIKE '" + SearchTxt.Text + "%'";

                    //assign parameters to object of class SqlDataAdapter
                    sda = new SqlDataAdapter(CompQuery, con);
                    dt = new DataTable();

                    //Fill datagridview
                    sda.Fill(dt);
                    dataGridView1.DataSource = dt;

                }

                else
                {
                    // show message
                    MessageBox.Show("You must Select search criteria");
                }
                // Close connection
                 con.Close();

                // Clear text boxes
                comboBox3.Text = "";
                SearchTxt.Text = "";
            }

            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using iTextSharp.text;
using iTextSharp.text.pdf;
using System.IO;



namespace Smart_manhole_cover
{

    public partial class Manhole_Details : Form
    {


        public Manhole_Details()
        {
            InitializeComponent();
        }
        // VARIABLE FOR OBJECT OF CALSS
        SqlDataAdapter sda;
        DataTable dt;


        private void Manhole_Details_Load(object sender, EventArgs e)
        {
            try
            {
                //SqlConnection con = new SqlConnection(ConString);
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);

                // SQL Querry to select data
                string CompQuery = @"SELECT Manhole_ID, Location, Type_of_manhole from Manhole ";

                // make object of class and assign parameters
                sda = new SqlDataAdapter(CompQuery, con);
                dt = new DataTable();

                // Fill datagridview
                sda.Fill(dt);
                dataGridView1.DataSource = dt;
            }

            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.   ↙
    Error);

            }

        }


        private void textBox1_TextChanged(object sender, EventArgs e)
        {

            try
            {
               // Connectionstring
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);

                //Open Connection
                con.Open();

                //If statement
                if (comboBox1.Text == "ManholeID")
                {
                    // SQL Querry to select data
                    string ManholeQuery = @"SELECT Manhole_ID, Location, Type_of_manhole FROM Manhole WHERE Manhole_ID LIKE '" + ManholeTxt.Text + "%'";

                    // make object of class and assign parameters
                    sda = new SqlDataAdapter(ManholeQuery, con);
                    dt = new DataTable();

                    // Fill datagridview
                    sda.Fill(dt);
                    dataGridView1.DataSource = dt;

                    // if statement
                }
                else if (comboBox1.Text == "Location")
                {
                    // SQL Querry to select data
                    string ManholeQuery = @"SELECT Manhole_ID, Location, Type_of_manhole FROM Manhole WHERE Location LIKE '" + ManholeTxt.Text + "%'";

                    // make object of class and assign parameters
                    sda = new SqlDataAdapter(ManholeQuery, con);
                    dt = new DataTable();

                    // Fill datagridview
                    sda.Fill(dt);
                    dataGridView1.DataSource = dt;
                }
                else if (comboBox1.Text == "Type of manhole")
                {
                    // SQL Querry to select data
                    string ManholeQuery = @"SELECT Manhole_ID, Location, Type_of_manhole FROM Manhole WHERE Type_of_manhole LIKE '" + ManholeTxt.Text + "%'";

                    // make object of class and assign parameters
                    sda = new SqlDataAdapter(ManholeQuery, con);
                    dt = new DataTable();

                    // Fill datagridview
                    sda.Fill(dt);
                    dataGridView1.DataSource = dt;
                }

                else
                {
                    // sHOW MESSAGE
                    MessageBox.Show("You must Select search criteria");
                }

            }   catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }


        }

        private void button4_Click(object sender, EventArgs e)
        {



            Document doc = new Document(iTextSharp.text.PageSize.LETTER, 10, 10, 42, 35);
            PdfWriter wri = PdfWriter.GetInstance(doc, new FileStream("Test.pdf", FileMode.Create));
            doc.Open(); //Open document to write

            //write some content
            Paragraph paragraph = new Paragraph("  .............................................................   Read, Control and Communication Device for Manholea  .............................................................. \n\n\n  ");
            doc.Add(paragraph);

            List list = new List(List.UNORDERED); // creating list in pdf file
            list.IndentationLeft = 30f; // list starts with the space of 30f
            list.Add(new ListItem("  .............................................................      ↙
    Manhole details Document ............................................................. \n\n\n"));

            doc.Add(list);




            PdfPTable table = new PdfPTable(dataGridView1.Columns.Count);

            for (int j = 0; j < dataGridView1.Columns.Count; j++)
            {
                table.AddCell(new Phrase(dataGridView1.Columns[j].HeaderText));

            }

            table.HeaderRows = 1;

            for (int i = 0; i < dataGridView1.Rows.Count; i++)
            {
                for (int k = 0; k < dataGridView1.Columns.Count; k++)
                {
                    if (dataGridView1[k, i].Value != null)
                    {
                        table.AddCell(new Phrase(dataGridView1[k, i].Value.ToString()));
                    }
                }
            }

            doc.Add(table);

            doc.Close(); //close document

            MessageBox.Show("Page is sucussfully saved as pdf in debug file"); // Show message


            PrintDialog pd = new PrintDialog();
            pd.ShowDialog();

        }


    }
}

```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Configuration;
using System.Data.SqlClient;
using iTextSharp.text;
using iTextSharp.text.pdf;
using System.IO;



namespace Smart_manhole_cover
{
    public partial class Alarm : Form
    {
        // Declaring variables
        SqlDataAdapter sda;
        DataTable dt;
        public Alarm()
        {
            InitializeComponent();

        }

        private void btnSøk_Click(object sender, EventArgs e)
        {



            try
            {
                // Connection String
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);

                //Open connection
                con.Open();

                //If sentence
                if (RBCover.Checked)
                {
                    // Select Query
                    string CompQuery = @"SELECT Alarm_Cover, Day, Month, Year, Hour, Minute, Manhole_ID FROM Situation_report WHERE Manhole_ID LIKE '" + comboBox1.Text + "%'";

                    // Declaration of object of classes
                    sda = new SqlDataAdapter(CompQuery, con);
                    dt = new DataTable();

                    // Assigning class dt as a parameter to method Fill in class sda
                    sda.Fill(dt);

                    // dt is assigned to method datasource in GridHostory
                    GridHistory.DataSource = dt;


                }
                else if (RBTemp.Checked)
                {

                    // Connection string
                    string CompQuery = @"SELECT Alarm_Temp, Day,Manhole_ID, Month, Year, Hour, Minute FROM Situation_report WHERE Manhole_ID LIKE '" + comboBox1.Text + "%'";

                    // Declaration of object of classes
                    sda = new SqlDataAdapter(CompQuery, con);
                    dt = new DataTable();

                    // Assigning class dt as a parameter to method Fill in class sda
                    sda.Fill(dt);

                    // dt is assigned to method datasource in GridHostory
                    GridHistory.DataSource = dt;
                }
              // if sentence
                else if (RBAll.Checked)
                {
                    // Sql Querry statement
                    string CompQuery = @"SELECT Alarm_Temp, Alarm_Water_level, Alarm_Cover, Manhole_ID, Alarm_Gas, Day, Month, Year, Hour, Minute FROM Situation_report WHERE Manhole_ID LIKE '" + comboBox1.Text + "%'";

                    // Declaration of object of classes
                    sda = new SqlDataAdapter(CompQuery, con);
                    dt = new DataTable();

                    // Assigning class dt as a parameter to method Fill in class sda
                    sda.Fill(dt);

                    // dt is assigned to method datasource in GridHostory
                    GridHistory.DataSource = dt;
                }
                    // Sentence sentence
                      else if (RBLevel.Checked)
                {
                          //Sql querry statement
                    string CompQuery = @"SELECT Alarm_Water_level, Day, Month, Year, Hour, Manhole_ID, Minute FROM Situation_report WHERE Manhole_ID LIKE '" + comboBox1.Text + "%'";

                    // Declaration of object of classes
                    sda = new SqlDataAdapter(CompQuery, con);
                    dt = new DataTable();

                    // Assigning class dt as a parameter to method Fill in class sda
                    sda.Fill(dt);

                    // dt is assigned to method datasource in GridHostory
                    GridHistory.DataSource = dt;

                            }

                else if (RBGas.Checked)
                {
                   // Querry select statement
                    string CompQuery = @"SELECT Alarm_Gas, Day, Month, Year, Hour, Manhole_ID, Minute FROM Situation_report WHERE Manhole_ID LIKE '" + comboBox1.Text + "%'";

                    // Declaration of object of classes
                    sda = new SqlDataAdapter(CompQuery, con);
                    dt = new DataTable();

                    // Declaration of object of classes
                   sda.Fill(dt);

                   // dt is assigned to method datasource in GridHostory
                    GridHistory.DataSource = dt;
                }
                else
                {
                    // show message
                    MessageBox.Show("You must Select search criteria");
                }
                //close connection
                con.Close();
            }

            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

        private void Alarm_Load(object sender, EventArgs e)
        {
            try
            {
                // Connection string
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);

                // Querry statement
                string CompQuery = @"SELECT Report_ID, Alarm_Cover, Alarm_Water_level, Alarm_Temp, Alarm_Gas, Day, Month, Manhole_ID, Year, Hour, Minute from Situation_report ";

                // Declaring Objects of classes
                sda = new SqlDataAdapter(CompQuery, con);
                dt = new DataTable();

                // Assigning class dt as a parameter to method Fill in class sda
                sda.Fill(dt);

                // dt is assigned to method datasource in GridHostory
                GridHistory.DataSource = dt;

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            // sql statement
            string Query = @"SELECT Manhole_ID from Situation_report ";

            try
            {
                // Connection string
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);


                // Declaring Objects of classes
                sda = new SqlDataAdapter(Query, con);
                dt = new DataTable();

                // Assigning class dt as a parameter to method Fill in class sda
                sda.Fill(dt);

                // dt is assigned to method datasource in GridHostory
                comboBox1.DataSource = dt;

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }


        }


        private void button4_Click(object sender, EventArgs e)
        {


            Document doc = new Document(iTextSharp.text.PageSize.LETTER, 10, 10, 42, 35);
            PdfWriter wri = PdfWriter.GetInstance(doc, new FileStream("Test.pdf", FileMode.Create));

            //Open document to write
            doc.Open();

            //write some content
            Paragraph paragraph = new Paragraph("  .............................................................   Read, Control and Communication Device for Manholea  .............................................................. \n\n\n  ");
            doc.Add(paragraph);
```

```

            // creating list in pdf file
            List list = new List(List.UNORDERED);

            // list starts with the space of 30f
            list.IndentationLeft = 30f;
            list.Add(new ListItem(".......................................................................↙
    .Manhole details Document............................................. \n\n\n"));

            doc.Add(list);

            PdfPTable table = new PdfPTable(GridHistory.Columns.Count);

            for (int j = 0; j < GridHistory.Columns.Count; j++)
            {
                table.AddCell(new Phrase (GridHistory.Columns[j].HeaderText));

            }

            table.HeaderRows = 1;

            for (int i = 0; i < GridHistory.Rows.Count; i++)
            {
                for (int k = 0; k < GridHistory.Columns.Count; k++)
                {
                    if (GridHistory[k, i].Value != null)
                    {
                        table.AddCell(new Phrase(GridHistory[k, i].Value.ToString()));
                    }
                }
            }

            doc.Add(table);

            //close document
            doc.Close();

            // Show message
            MessageBox.Show("Page is sucussfully saved as pdf in debug file");

            // Print dialog
            PrintDialog pd = new PrintDialog();
            pd.ShowDialog();


        }









    }
}

```

## COMPANY CODE

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Configuration;

namespace Smart_manhole_cover
{
    public partial class Company1 : Form
    {
        // Objects of classes
        SqlDataAdapter sda;
        DataTable ds;



        public Company1()
        {
            InitializeComponent();
        }

        private void Company1_Load(object sender, EventArgs e)
        {
            try
            {
                // Connection String
                SqlConnection Compcon = new SqlConnection(Properties.Settings.Default.DBCS);

            // Sql Querry statement
            string CompQuery = @"SELECT * from Company ";

                // aassinging parameters to class sda
            sda = new SqlDataAdapter(CompQuery, Compcon);
            ds = new DataTable();

                // Fill gridview
            sda.Fill(ds);
            GridView.DataSource = ds;

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }



        }



    }
}
```

# ADMIN LOGIN CODE

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Configuration;

namespace Smart_manhole_cover
{
    public partial class Admin_login : Form
    {
        public Admin_login()
        {
            InitializeComponent();
        }

        private void button2_Click(object sender, EventArgs e)
        {


            try
            {
                // Declaration of variables
                string AdminID = AdLogminTxt.Text;
                string AdminPassword = AdminPassTxt.Text;

                //Connection string
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);

                // Sql Querry and assigning of object of class Sqlcommand to sc
                SqlCommand sc = new SqlCommand("Select * from Administrator_Table where AdminID = '" + AdminID + "' and Adminpassword = '" + AdminPassword + "';", con);


                SqlDataReader myReader;
                //Open connection
                con.Open();

                // Assign method executeReader in class sc to myReader
                myReader = sc.ExecuteReader();

                //variable declaration
                int count = 0;

                // while sentence
                while (myReader.Read())
                {
                    // assigning value to variable count
                    count = count + 1;

                }
                // condition statement
                if (count == 1)
                {
                    // object of class admin page
                    Admin_page page6 = new Admin_page(AdminID);

                    //open page6 and lock current page
                    page6.Show();
                    this.Close();


                }
                else
                {
                    // show message
                    MessageBox.Show("Wrong UserName and Password ... Please try again");
                }
                //Close connection
                con.Close();

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }




        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }



        private void button3_Click(object sender, EventArgs e)
        {

            this.Close();
        }



        private void Admin_login_Load(object sender, EventArgs e)
        {

        }


    }
}

```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Smart_manhole_cover
{
    public partial class Register_Admin : Form
    {
        // variable for object of class
        SqlDataAdapter sda;
        DataTable dt;
        SqlCommandBuilder scb;
        public Register_Admin()
        {
            InitializeComponent();
        }

        private void Register_Admin_Load(object sender, EventArgs e)
        {

            try {
               // Connection string
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);


                //Query to select all the column in manhole table
                string CompQuery = @"SELECT AdminID, Adminpassword from Administrator_Table ";

                // object of class and assigning of parameters
                sda = new SqlDataAdapter(CompQuery, con);
                dt = new DataTable();

                // Fill data in datagridview
                sda.Fill(dt);
                dataGridView1.DataSource = dt;

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

        private void button1_Click(object sender, EventArgs e)
        {

            try
            {
                scb = new SqlCommandBuilder(sda); // operate object of a class
                sda.Update(dt);

              //Connectiion string
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);


                //Query to select all the column in manhole table
                string CompQuery = @"SELECT AdminID, Adminpassword from Administrator_Table ";

                // object of class and assigning of parameters
                sda = new SqlDataAdapter(CompQuery, con);
                dt = new DataTable();

                // Fill data in datagridview
                sda.Fill(dt);
                dataGridView1.DataSource = dt;

                // Show massage
                MessageBox.Show("Administrator is sucessfully Updated");

                dataGridView1.Show(); // Show dataGridview

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

    }
}

```

ADMIN PAGE CODE

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Smart_manhole_cover
{
    public partial class Admin_page : Form
    {
        public Admin_page(string AdminID)

        {
            InitializeComponent();

            // Assignment of text to label AdmLbl
            AdmLbl.Text = " You are logged in as" + " " +  AdminID;

        }


        private void button3_Click(object sender, EventArgs e)
        {
            // new object of class Edit manhole
            Edit_manhole tt = new Edit_manhole();

            //Open page tt
            tt.Show();

        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            // new object of class Edit manhole
            Edit_Company_Info dd = new Edit_Company_Info();

            //Open page dd
            dd.Show();

        }

        private void button4_Click(object sender, EventArgs e)
        {
            // new object of class Edit manhole
            Register_Admin hh = new Register_Admin();

            //Open page hh
            hh.Show();

        }

        private void button5_Click_1(object sender, EventArgs e)
        {
            // new object of class Edit manhole
            Homepage page8 = new Homepage();

            //Open page hh
            page8.Show();

            //Close current page
            this.Close();
        }


    }
}

```

ADD NEW MANHOLE CODE

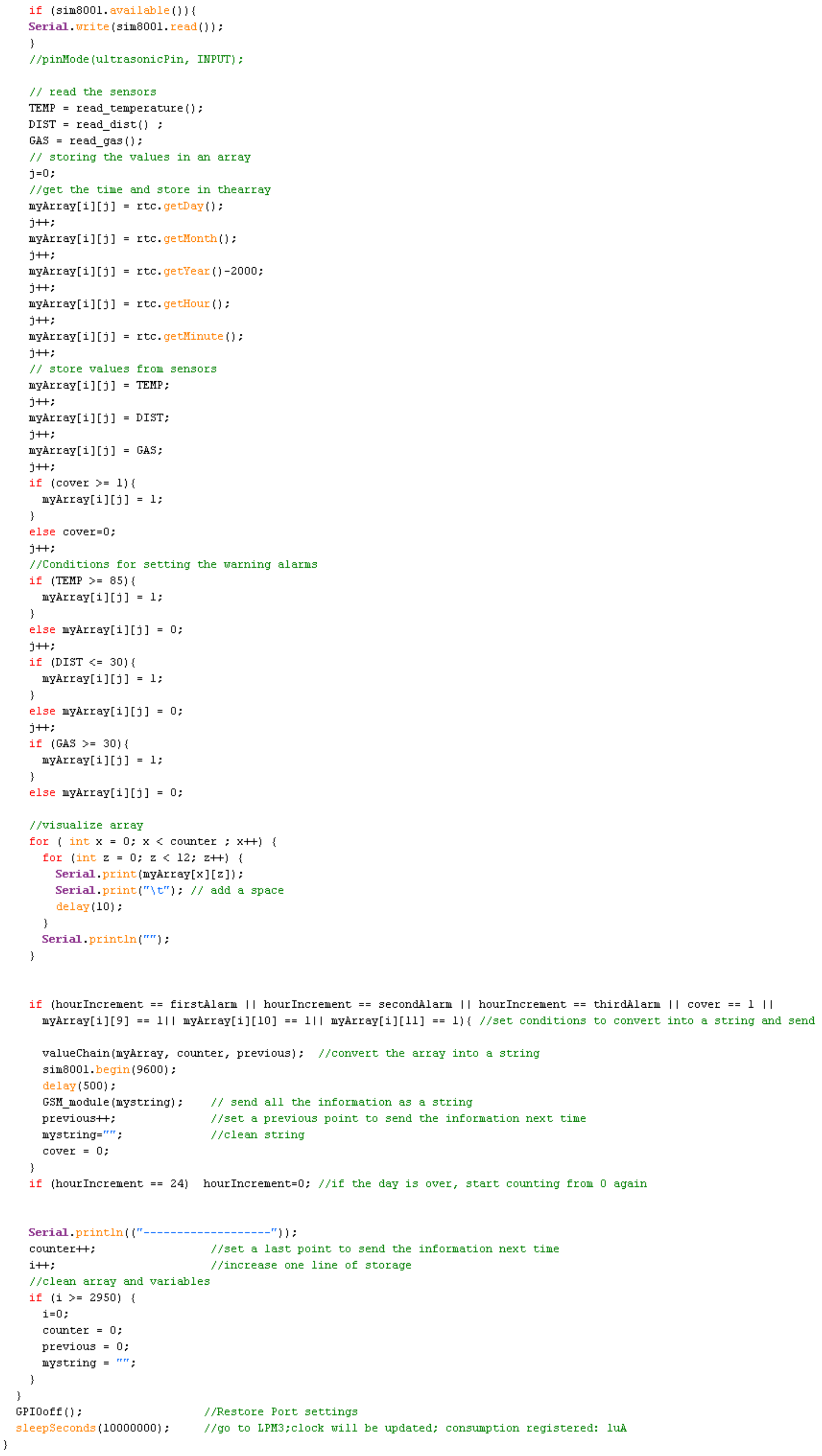
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;


namespace Smart_manhole_cover
{
    public partial class Add_new_manhole : Form
    {

        public Add_new_manhole()
        {
            InitializeComponent();

        }

        private void button1_Click(object sender, EventArgs e)
        {

            try
            {

                //Connection string
                string RegString = @"Data Source = BRUKER-PC\SQLEXPRESS; Initial Catalog = Read and control manhole cover; Integrated Security = True";

                // Object of sqlconnection class
            SqlConnection Regcon = new SqlConnection(RegString);

                //Inset Querry
            string RegQuery = @"Insert into Manhole( Manhole_ID, Location, Type_of_manhole, Company_name) Values(@ManholeID, @Location, @Type_of_manhole, @Company_name )";

            //Open connection
            Regcon.Open();


            SqlCommand Cmd = new SqlCommand(RegQuery, Regcon);

                // Assign data from textbox to database column
            Cmd.Parameters.AddWithValue("@ManholeID", ManholeTxt.Text);
            Cmd.Parameters.AddWithValue("@Location", LocationTxt.Text);
            Cmd.Parameters.AddWithValue("@Type_of_manhole", TypeofmanTxt.Text);
            Cmd.Parameters.AddWithValue("@Company_name", CompTxt.Text);

            Cmd.ExecuteNonQuery();

                //Display message
            MessageBox.Show(" New manhole added ");

               // Empty textboxes after program execution
            ManholeTxt.Text = "";
            LocationTxt.Text = "";
            TypeofmanTxt.Text = "";
            CompTxt.Text = "";

                //Close connection
            Regcon.Close();

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            //Close page
            this.Close();
            // Redirect current to Edit manhole page
            Edit_manhole ff = new Edit_manhole();
            ff.Show();
        }


    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;


namespace Smart_manhole_cover
{
    public partial class Edit_manhole : Form
    {
        // Declaration of variables for object of class
        SqlDataAdapter sda;
        DataTable dt;
        SqlCommandBuilder scb;

        public Edit_manhole()
        {
            InitializeComponent();
        }

        private void Edit_manhole_Load(object sender, EventArgs e)
        {
            // Clear combobox text
            comboBox1.Text = "";


            try
            {
               //Connectonstring
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);


                //Query to select all the column in manhole table
                string CompQuery = @"SELECT * from manhole ";

                // Assign parameters to class sda
                sda = new SqlDataAdapter(CompQuery, con);
                dt = new DataTable();

                // Fill datagridview
                sda.Fill(dt);
                dataGridView1.DataSource = dt;

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.   ↙
    Error);

            }

            try
            {
                // Connection string
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);

                //Query to select all the column in manhole table
                string CompQuery = @"SELECT Manhole_ID from manhole ";

                // Assign parameters to class sda
                sda = new SqlDataAdapter(CompQuery, con);
                dt = new DataTable();

                // Fill Combobox
                sda.Fill(dt);
                comboBox1.DataSource = dt;

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.  ↙
    Error);

            }



        }

        private void button4_Click(object sender, EventArgs e)
        {

            try
            {
                //Connection string
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);

                //Query to select all the column in manhole table
                string CompQuery = @"SELECT * from manhole ";

                // Assign parameters
                sda = new SqlDataAdapter(CompQuery, con);
                dt = new DataTable();

                // Fill datagridview
                sda.Fill(dt);
                dataGridView1.DataSource = dt;

                // Show messagebox
                MessageBox.Show("Manhole is Successful Updated");

                // Clear combobox text
                comboBox1.Text = "";

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.  ↙
    Error);

            }




        }

        private void button3_Click(object sender, EventArgs e)
        {

            try
            {
                // Connection string
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);

                // Open connection
                con.Open();

                // If statement
                if (comboBox1.Text != "" )
                {

                    // Querry statement
                    String DeletemanholeQuerry = "delete from manhole where Manhole_ID = '" + comboBox1. ↙
    Text + "'";


                    SqlCommand DeletemanholeCmd = new SqlCommand(DeletemanholeQuerry, con);
                    DeletemanholeCmd.ExecuteNonQuery();
                }

                else

                {
                    // show meassge
                    MessageBox.Show("You must Select manhole ID to delete");
                }
                // Close connection
                con.Close();

                // show message
                MessageBox.Show("ManholeID" + comboBox1.Text + "is sucessfully deleted");
                comboBox1.Text = "";
            }

            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }


        }

        private void button2_Click(object sender, EventArgs e)
        {
           // object of class
            Add_new_manhole rr = new Add_new_manhole();

            // open page rr
            rr.Show();

            //hide the current page
            this.Hide();

        }


    }
}
```

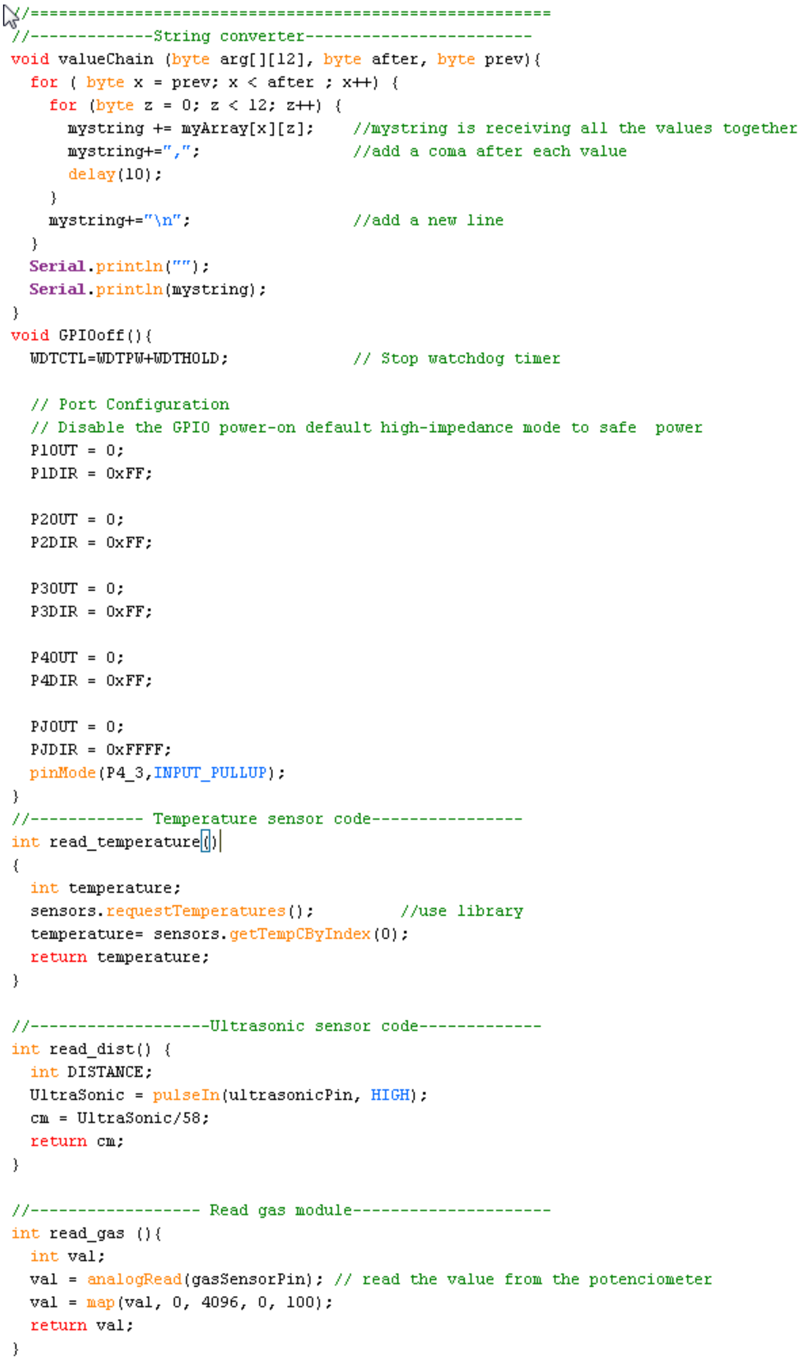
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Smart_manhole_cover
{
    public partial class Edit_Company_Info : Form
    {
        // variable for object of class
        SqlDataAdapter sda;
        DataTable dt;
        SqlCommandBuilder scb;
        public Edit_Company_Info()
        {
            InitializeComponent();


        }

        private void Edit_Company_Info_Load(object sender, EventArgs e)
        {

            try
            {
                // Connection string
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);


                //Query to select all the column in manhole table
                string CompQuery = @"SELECT * from Company ";


                // make new object of class and assign parameters
                sda = new SqlDataAdapter(CompQuery, con);
                dt = new DataTable();

                // fill datagridview
                sda.Fill(dt);
                dataGridView1.DataSource = dt;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.    ↙
    Error);

            }

        }

        private void button2_Click(object sender, EventArgs e)
        {
            scb = new SqlCommandBuilder(sda);
            sda.Update(dt);


            try
            {

              //connection string
                SqlConnection con = new SqlConnection(Properties.Settings.Default.DBCS);


                //Query to select all the column in manhole table
                string CompQuery = @"SELECT * from Company ";

                // make new object of class and assign parameters
                sda = new SqlDataAdapter(CompQuery, con);
                dt = new DataTable();

                // Fill Datagridview
                sda.Fill(dt);
                dataGridView1.DataSource = dt;

                // show meaasge
                MessageBox.Show("Company is Sucessful updated");

            }
            catch (Exception ex)
            {
                MessageBox.Show("Error\n" + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.    ↙
    Error);

            }

        }


    }


        }


```

# Attachment I

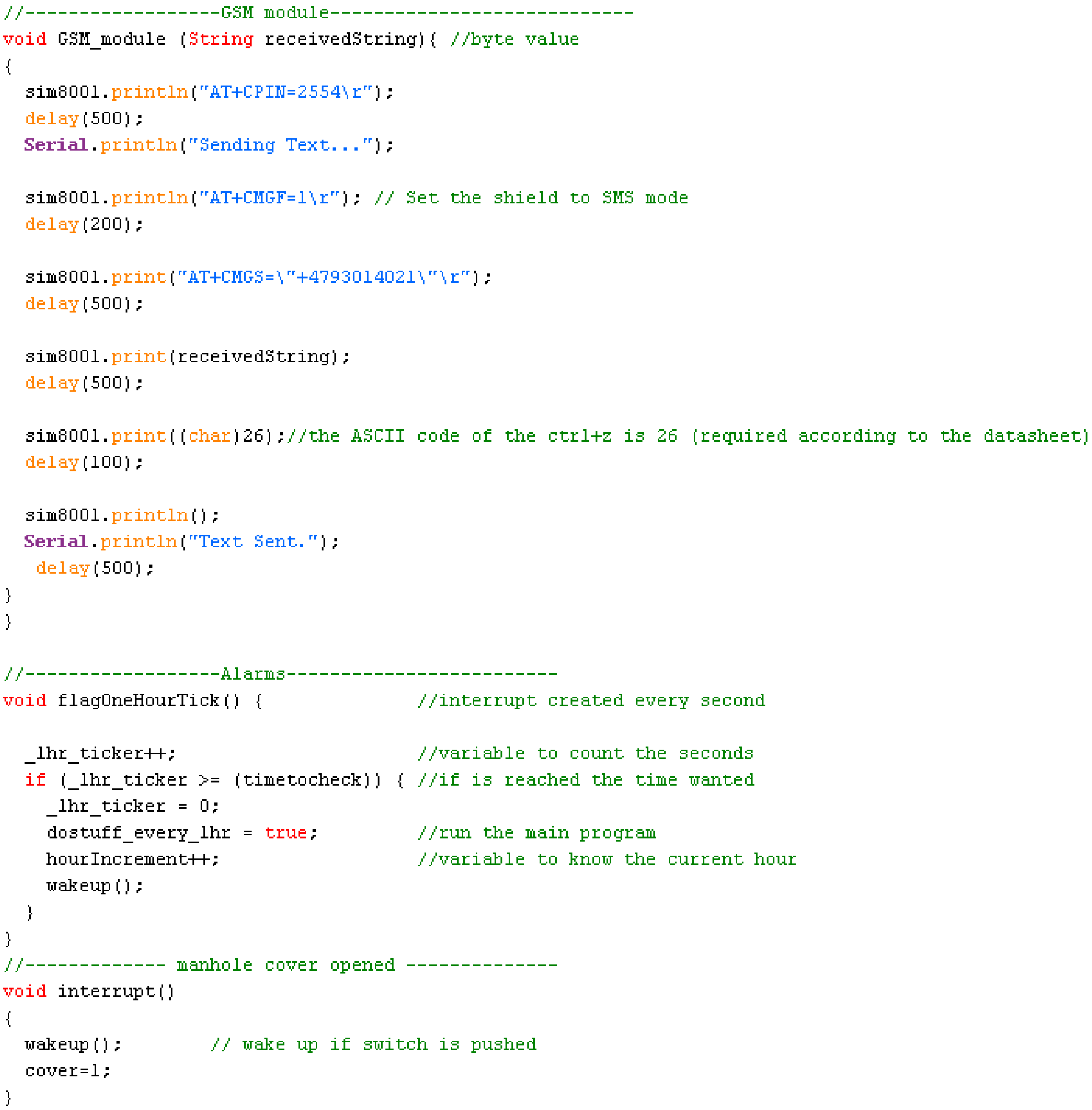
```
//------library for time------
#include <RTC_B.h>  //library to set the RTC_B clock mode

byte firstAlarm = 8;      //8:00
byte secondAlarm = 12;    //12:00
byte thirdAlarm = 15;     //15:00
int timetocheck = 3600; //check every 1 hour

byte hourIncrement;
volatile uint16_t _1hr_ticker = 0;
volatile boolean dostuff_every_1hr = false;

//------libraries for temperature sensor------------
#include <OneWire.h>
#include <DallasTemperature.h>
#define tempSensorPin 12   //pin for temperature sensor
OneWire oneWire(tempSensorPin);
DallasTemperature sensors(&oneWire);
int TEMP;
//------ libraries for SIM800l-----------------------
#include <SoftwareSerial.h>
#include <SoftwareSerial.h>
#include <String.h>
SoftwareSerial sim800l(2, 3); // RX, TX

//-------------pins for gas sensor---------------------
#define gasSensorPin 18 // select input pin for gasSensorPin
int GAS = 0;
//-------------pins for Distance sensor----------------
#define ultrasonicPin 19    // ultrasonic pin number
int DIST;        // initialize variables
long UltraSonic, cm;
//-------------pins for cover switch ------------------
#define SwitchPin 11

//--------------some variables-------------------------
//creating the array in the FRAM position
#define PERSIST __attribute__((section(".text")))
uint8_t myArray[2700][12] PERSIST;
String mystring="";

byte  i, j, counter, previous, cover = 0;

//===================== setup ======================
void setup()
{
  Serial.begin(9600);
  //
  GPIOoff(); //set all the GPIO ports to output low

  //time inicialization
  rtc.begin();    // Start RTC calendar mode
  rtc.begin( MONDAY, 4, 10, 2016, 8, 29, 0);
  rtc.attachPeriodicInterrupt(1, flagOneHourTick); //alarm every one hour

  pinMode(SwitchPin, INPUT_PULLUP);              // Make push button input
 attachInterrupt(SwitchPin, interrupt, HIGH);// Attach ISR to PUSH1
  pinMode(18, INPUT);
   pinMode(ultrasonicPin, INPUT);
}
//======================= Loop ============================
void loop()
{

  if (dostuff_every_1hr)
  {
    //set sensors and mosfet pins
      //mosfets
    pinMode(P2_4, HIGH); //feeds the sensors
    pinMode(P1_5, HIGH); //feeds the regulator and gas sensor
    pinMode(P3_6, HIGH); //feeds de GSM module

      //sensor pins
    pinMode(gasSensorPin, INPUT);  //Set P3.0 SEL as Input- gas
    pinMode(tempSensorPin, INPUT); //Set P3.0 SEL as Input- temp
    pinMode(ultrasonicPin, INPUT); //Set P1.2 SEL as Input- ultrasonic
```



```
    if (sim800l.available()){
    Serial.write(sim800l.read());
    }
    //pinMode(ultrasonicPin, INPUT);

    // read the sensors
    TEMP = read_temperature();
    DIST = read_dist() ;
    GAS = read_gas();
    // storing the values in an array
    j=0;
    //get the time and store in thearray
    myArray[i][j] = rtc.getDay();
    j++;
    myArray[i][j] = rtc.getMonth();
    j++;
    myArray[i][j] = rtc.getYear()-2000;
    j++;
    myArray[i][j] = rtc.getHour();
    j++;
    myArray[i][j] = rtc.getMinute();
    j++;
    // store values from sensors
    myArray[i][j] = TEMP;
    j++;
    myArray[i][j] = DIST;
    j++;
    myArray[i][j] = GAS;
    j++;
    if (cover >= 1){
      myArray[i][j] = 1;
    }
    else cover=0;
    j++;
    //Conditions for setting the warning alarms
    if (TEMP >= 85){
      myArray[i][j] = 1;
    }
    else myArray[i][j] = 0;
    j++;
    if (DIST <= 30){
      myArray[i][j] = 1;
    }
    else myArray[i][j] = 0;
    j++;
    if (GAS >= 30){
      myArray[i][j] = 1;
    }
    else myArray[i][j] = 0;

    //visualize array
    for ( int x = 0; x < counter ; x++) {
      for (int z = 0; z < 12; z++) {
        Serial.print(myArray[x][z]);
        Serial.print("\t"); // add a space
        delay(10);
      }
      Serial.println("");
    }


    if (hourIncrement == firstAlarm || hourIncrement == secondAlarm || hourIncrement == thirdAlarm || cover == 1 ||
      myArray[i][9] == 1|| myArray[i][10] == 1|| myArray[i][11] == 1){ //set conditions to convert into a string and send

      valueChain(myArray, counter, previous);  //convert the array into a string
      sim800l.begin(9600);
      delay(500);
      GSM_module(mystring);    // send all the information as a string
      previous++;              //set a previous point to send the information next time
      mystring="";             //clean string
      cover = 0;
    }
    if (hourIncrement == 24)  hourIncrement=0; //if the day is over, start counting from 0 again


    Serial.println(("-------------------"));
    counter++;                 //set a last point to send the information next time
    i++;                       //increase one line of storage
    //clean array and variables
    if (i >= 2950) {
      i=0;
      counter = 0;
      previous = 0;
      mystring = "";
    }
  }
  GPIOoff();                   //Restore Port settings
  sleepSeconds(10000000);      //go to LPM3;clock will be updated; consumption registered: 1uA
}
```



```
//=====================================================
//-------------String converter-----------------------
void valueChain (byte arg[][12], byte after, byte prev){
  for ( byte x = prev; x < after ; x++) {
    for (byte z = 0; z < 12; z++) {
      mystring += myArray[x][z];    //mystring is receiving all the values together
      mystring+=",";                //add a coma after each value
      delay(10);
    }
    mystring+="\n";                 //add a new line
  }
  Serial.println("");
  Serial.println(mystring);
}
void GPIOoff(){
  WDTCTL=WDTPW+WDTHOLD;             // Stop watchdog timer

  // Port Configuration
  // Disable the GPIO power-on default high-impedance mode to safe  power
  P1OUT = 0;
  P1DIR = 0xFF;

  P2OUT = 0;
  P2DIR = 0xFF;

  P3OUT = 0;
  P3DIR = 0xFF;

  P4OUT = 0;
  P4DIR = 0xFF;

  PJOUT = 0;
  PJDIR = 0xFFFF;
  pinMode(P4_3,INPUT_PULLUP);
}
//------------ Temperature sensor code----------------
int read_temperature()
{
  int temperature;
  sensors.requestTemperatures();         //use library
  temperature= sensors.getTempCByIndex(0);
  return temperature;
}

//-------------------Ultrasonic sensor code-------------
int read_dist() {
  int DISTANCE;
  UltraSonic = pulseIn(ultrasonicPin, HIGH);
  cm = UltraSonic/58;
  return cm;
}

//------------------ Read gas module---------------------
int read_gas (){
  int val;
  val = analogRead(gasSensorPin); // read the value from the potenciometer
  val = map(val, 0, 4096, 0, 100);
  return val;
}
```



```
//------------------GSM module----------------------------
void GSM_module (String receivedString){ //byte value
{
  sim800l.println("AT+CPIN=2554\r");
  delay(500);
  Serial.println("Sending Text...");

  sim800l.println("AT+CMGF=1\r"); // Set the shield to SMS mode
  delay(200);

  sim800l.print("AT+CMGS=\"+4793014021\"\r");
  delay(500);

  sim800l.print(receivedString);
  delay(500);

  sim800l.print((char)26);//the ASCII code of the ctrl+z is 26 (required according to the datasheet)
  delay(100);

  sim800l.println();
  Serial.println("Text Sent.");
   delay(500);
}
}

//------------------Alarms-------------------------
void flagOneHourTick() {              //interrupt created every second

  _1hr_ticker++;                      //variable to count the seconds
  if (_1hr_ticker >= (timetocheck)) { //if is reached the time wanted
    _1hr_ticker = 0;
    dostuff_every_1hr = true;         //run the main program
    hourIncrement++;                  //variable to know the current hour
    wakeup();
  }
}
//------------- manhole cover opened --------------
void interrupt()
{
  wakeup();        // wake up if switch is pushed
  cover=1;
}
```

Attachment J

| List of variables | Type | Explanation |
|---|---|---|
| firstAlarm | byte | Alarms for converting and sendind the data |
| secondAlarm | byte | |
| thirdAlarm | byte | |
| timetocheck | int | Check the sensors x seconds |
| hourIncrement | byte | Times system had checked sensors |
| _1hr_ticker | uint16_t | Counter |
| dostuff_every_1hr | boolean | |
| TEMP | int | Temperature sensor storage |
| temperature | int | Temperature sensor storage in function |
| GAS | int | Gas sensor storage |
| val | int | Gas sensor storage in function |
| DIST | int | Distance sensor storage |
| UltraSonic | long | Distance sensor output storage in function |
| cm | long | Distance sensor storage in function |
| myArray[2700][12] | uint8_t | Array for all the data |
| mystring | String | String to convert the array |
| cover | byte | Cover opened or closed, 0 or 1 |
| i | byte | Index for the array |
| j | byte | |
| x | byte | |
| z | byte | |
| counter | byte | Count every time system has checked the values |
| previous | byte | Count every time system has send the string |

| List of libraries used | Explanation |
|---|---|
| <RTC_B.h> | Used for declare RTC_B clock mode |
| <OneWire.h> | Used for temperature sensor |
| <DallasTemperature.h> | |
| <SoftwareSerial.h> | Used for SIM800l |
| <SoftwareSerial.h> | |
| <String.h> | |

University College of Southeast Norway
Faculty of technology
Bachelor of Science

**REPORT FROM 6TH SEMESTER PROJECT SPRING 2016**

PRH612 Bachelor thesis

IA6-5-16

# Attachment K
# Test plan document

# TABLE OF CONTENT

# 1 INTRODUCTION

This document contains the strategies, processes and methodologies that implemented to test the functionality of the system “Read, control and communication unit for manholes”.

## 1.1 Scope

The scope of the testing includes testing of all functional requirements listed in the Software Requirements Specification and Software Design Document, as well as testing by performing the use cases defined in the same document.

## 1.2 Objective

The object of testing the software program is to ensure that it meets the system requirements and that all use-case scenarios are satisfied.

# 2 TEST METHODOLOGY

## 2.1 Unit testing

This is performed during the development of the system to ensure that the different parts/unit of the system work as intended. This will be documented in different test reports.

## 2.2 System and integration testing

This is carried out during development stage to verify that the different units of the system work together.

## 2.3 Functionality testing

This testing is executed at the end of the programming stage to ensure that the program meets the functionality requirement of the system. This testing will continue until the delivery of the system.

Functionality testing includes the scenario described in the following sub-chapter.

### 2.3.1 Data from manhole

Testing of “Data from manhole” functions should include the following:

- Display “Data from manhole”
- Search data from “Data from manhole”

### 2.3.2 Alarms

Testing of alarms functions should include the following
Display alarms in colors.

- “Green” should indicate normal situation
- “Red” should indicates abnormal situation that need urgent attention.

### 2.3.3 Alarm History

Testing of **Alarm history** functions should include the following:

- Display alarms notifications with date and time
- Search for data in **Alarm history**

### 2.3.4 Manhole Details

Testing of **Manhole details** functions should include the following:

- Display the details of the manhole listed in the system
- Search for data in **Manhole details**

### 2.3.5 Login as administrator

Testing of Login as administrator functions should include the following:

- An administrator should be able to login into the admin page with adminID and adminPassword.

### 2.3.6 Edit manhole details

Testing of **Edit manhole details** functions should include the following:

- Editing/change of information about any given manhole by the administrator.
- Function to add new manhole to the system by the administrator
- Function to delete an existing manhole from the system by the administrator

### 2.3.7 Edit Company information

Testing of **Edit company information** functions should include the following:

- Editing/change of information about the company by the administrator.

### 2.3.8 Edit Administrator

Testing of **Edit administrator** functions should include the following

- Editing/change of information about an administrator by the administrator.
- Ability to add new administrator: fill in admin ID and password

## 2.4 Final testing

Developers will perform the final testing. This should test the communication between the software and the unit, how the software receives data from the unit. The usability and the functionality of the program will also be tested.

# 3 RESOURCES AND ENVIRONMENTAL NEEDS

## 3.1 Testing environment

Testing will be performed on computers running windows 10. The tests require the following software installed:

- SQL server 2012/2014
- VMvare player

Attachment L

# GSM communication mobile test

| | |
|---|---|
| Requested by | IA6-5-16 group |
| Entity | University College of Southeast Norway |
| Date | 08-03-2016 |

**Object:** The object of this test was based in the possibility of being able to communicate with a GSM signal through the manhole cover (made of steel) and receive an SMS and a call.

**Test Purpose:** The purpose of the test was to verify that the GSM Module of the smartphone was able to receive an SMS and a call from the outside of the manhole cover. It meant that the emitter had enough signal strength to penetrate the steel/asphalt of the manhole cover.

## 1. Introduction

It was used an old smartphone, Samsung galaxy S2, as a receiver of the signal. It worked and was able to send and receive SMS by using the 2G network. Also it could use 3G, but it was not the purpose of the test.

| | |
|---|---|
| | |
| Front of Galaxy S2 | Back of the Galaxy S2 |

## 2. Test conditions

The test consisted of two main parts:

- Call from the outside.
- Send and receive SMS from the outside.

## 3. Setup

To prepare the test it was needed a plastic bag to cover the smartphone, otherwise, if it fell down to the water, it would be spoiled. Then, it was placed on a screw in the manhole.

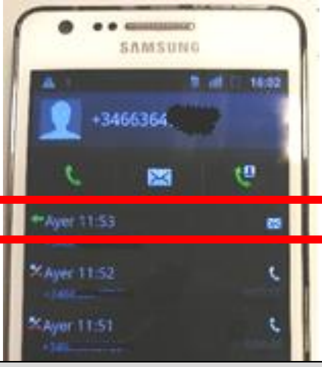
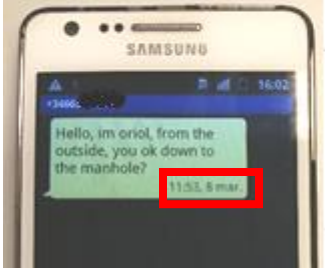
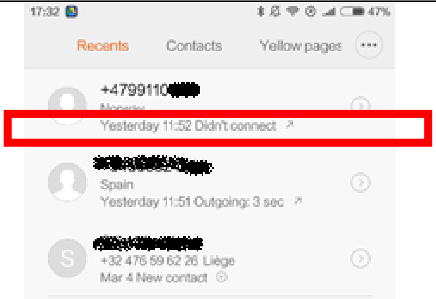
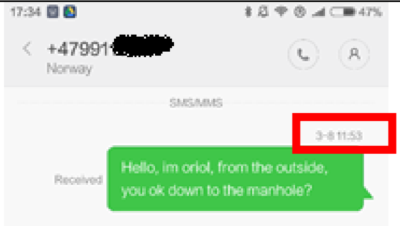
| | | | |
|---|---|---|---|
| Smartphone covered | Manhole cover used | Smartphone placed inside | Everything ready |

**Note:** The smartphone was placed at 1 meter under the cover.

## 4. Conclusion

When everything was ready, it was proceed to make the call. To know the answer, the smartphone was with the volume activated, and also, it is possible to see the register of calls. The register is shown below.

Afterwards, the next step was sending an SMS, for checking if it was successful send, it is needed to see the register either.

| <br>16:02<br>+3466364[redacted]<br>Ayer 11:53<br>Ayer 11:52<br>Ayer 11:51 | <br>16:02<br>+3466[redacted]<br>Hello, im oriol, from the outside, you ok down to the manhole?<br>11:53, 8 mar. |
|---|---|
| Register of the calls | Register of the SMS |
| <br>17:32 47%<br>Recents Contacts Yellow pages<br>+4799110[redacted]<br>Norway<br>Yesterday 11:52 Didn't connect<br>[redacted]<br>Spain<br>Yesterday 11:51 Outgoing: 3 sec<br>[redacted]<br>+32 476 59 62 26 Liège<br>Mar 4 New contact | <br>17:34 47%<br>+47991[redacted]<br>Norway<br>SMS/MMS<br>3-8 11:53<br>Received<br>Hello, im oriol, from the outside, you ok down to the manhole? |
| Evidence of the call | Evidence of the SMS |

## 5. Results

| Test | Result |
|---|---|
| Call from the outside. | OK |
| Send and SMS from the outside. | OK |

Attachment M

# Water Level Sensor Test

| | |
|---|---|
| Requested by | IA6-5-16 group |
| Entity | University College of Southeast Norway |
| Date | 04-05-2016 |

| |
|---|
| **Object:** The object of the test was to register if the level sensor worked in different conditions. |
| **Test Purpose:** The purpose of this test was to see if the program worked and sent status, and the detection range of the sensor. |

## 1. Introduction

In this test it was used an ultrasonic sensor MB7052 from MaxBotix. The sensor was connected to a microcontroller, MSP430FR5969. The tools are shown in Figure 1.

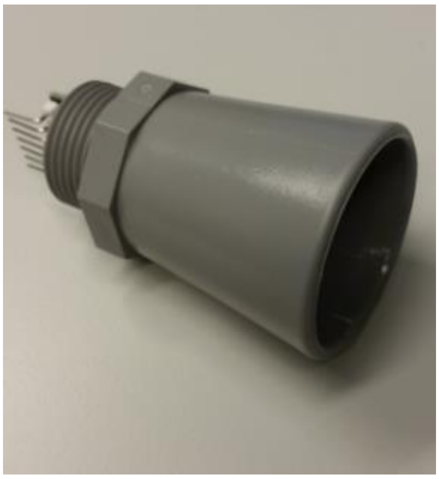
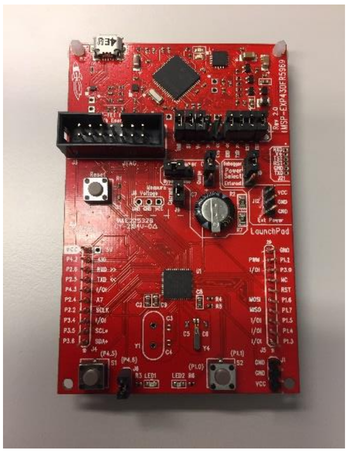
| | |
|---|---|
|  |  |
| Ultrasonic sensor | MSP430FR5969 |

*Figure 1: Ultrasonic sensor - Microcontroller*

## 2. Test conditions

To test the functionality of the level sensor, there were made different tests:

- The program works and sends status
- The minimum and maximum detection distance of sensor
- The sensor detects different materials

## 3. Setup

To prepare the test it is needed to program the sensor and connect it to the microcontroller. The PW to prepare the test it was needed to program the sensor and connect it to the microcontroller. The PW pin on the sensor was connected to the pin P1_2, which is the only one that can generate and detect a pulse width modulation. Figure 1 shows how the sensor was connected, the schematic is shown in Figure 2. How the program was written, is shown in Figure 3.

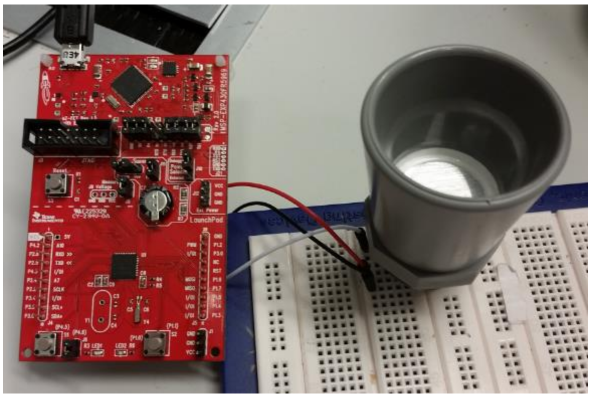


*Figure 2: Level sensor connected to the MSP430FR5969*

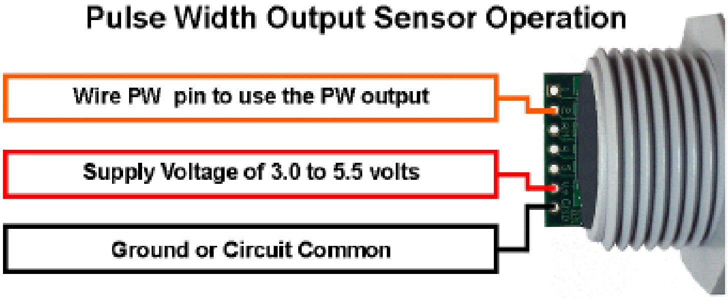
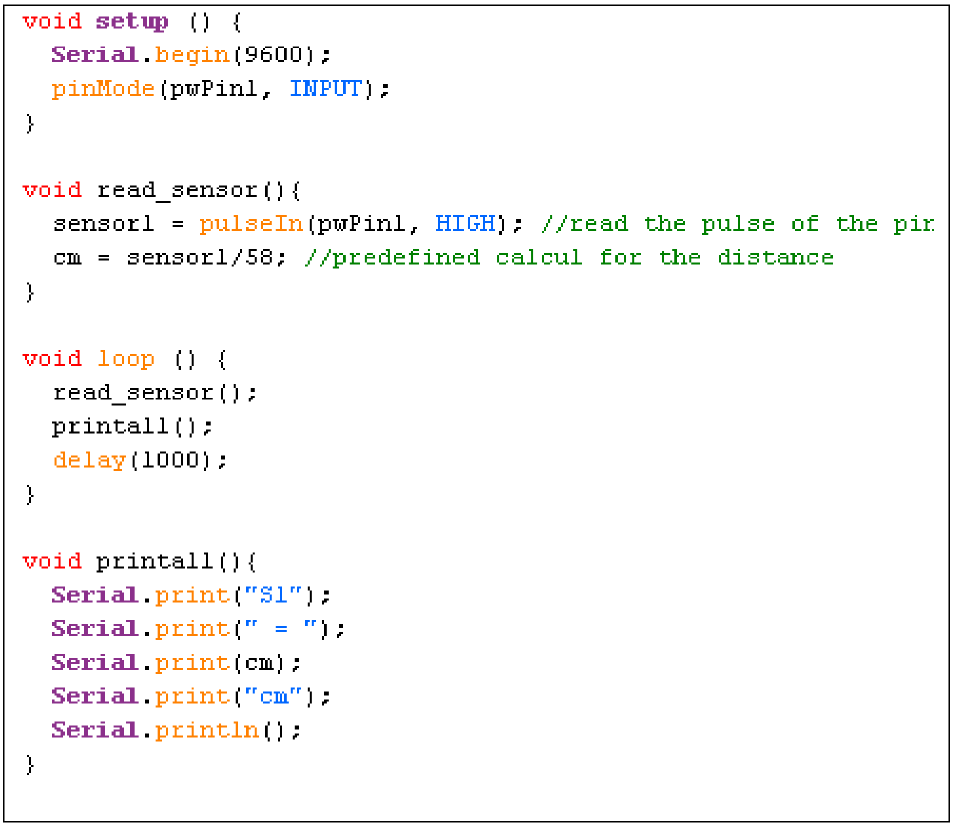

Pulse Width Output Sensor Operation
Wire PW pin to use the PW output
Supply Voltage of 3.0 to 5.5 volts
Ground or Circuit Common

*Figure 3: Schematic for level sensor*



```
void setup () {
  Serial.begin(9600);
  pinMode(pwPin1, INPUT);
}

void read_sensor(){
  sensor1 = pulseIn(pwPin1, HIGH); //read the pulse of the pir
  cm = sensor1/58; //predefined calcul for the distance
}

void loop () {
  read_sensor();
  printall();
  delay(1000);
}

void printall(){
  Serial.print("S1");
  Serial.print(" = ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
}
```

*Figure 4: Program-code for the level sensor*

Microcontroller read the pulse width signal as an input from the sensor, calculated it and sent a distance output in centimeters. The measurement was shown in Serial Monitor communication interface of Energia.

1. Test if the program works and send status

The test was performed several time as described in the datasheet, shown in Figure 1. The result did not show correct distances. Therefore it was tried to measure from the head of the horn, like Figure 2 shows. The result seemed like it was correct.

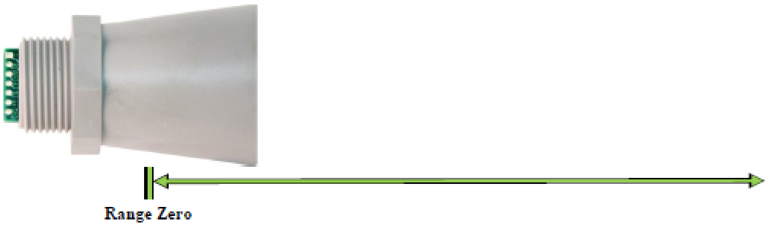


*Figure 5: The range is measured from the front of the transducer to the target*

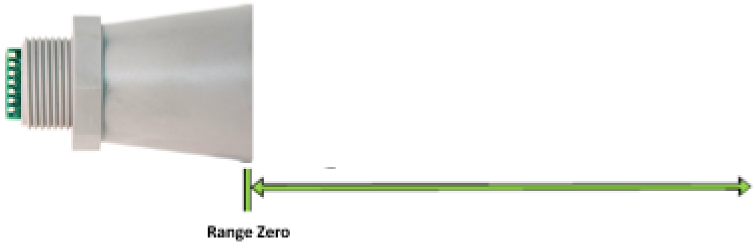


*Figure 6: The range is measured from the head of horn*

The Figure 3 shows how to perform the test.  All measurements were plotted in excel as a graph, shown in Figure 4. This part of the test consisted in making a sweep from 30 cm until the minimum value, and then see the reaction of the sensor.

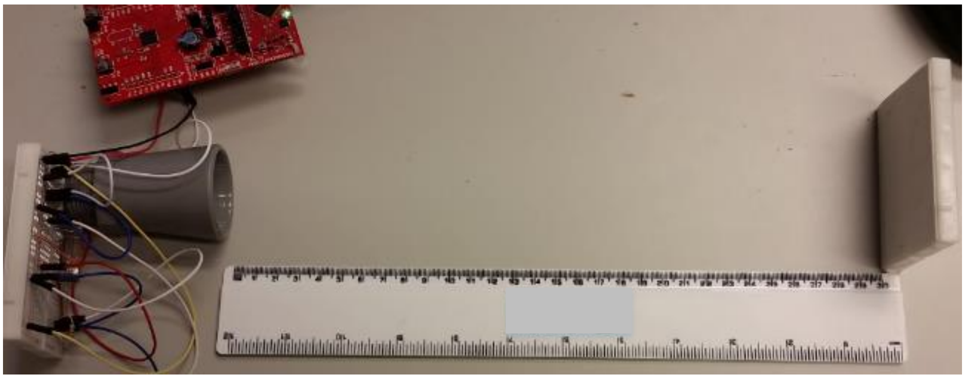


*Figure 7: Sensor detects distance in 30 cm*

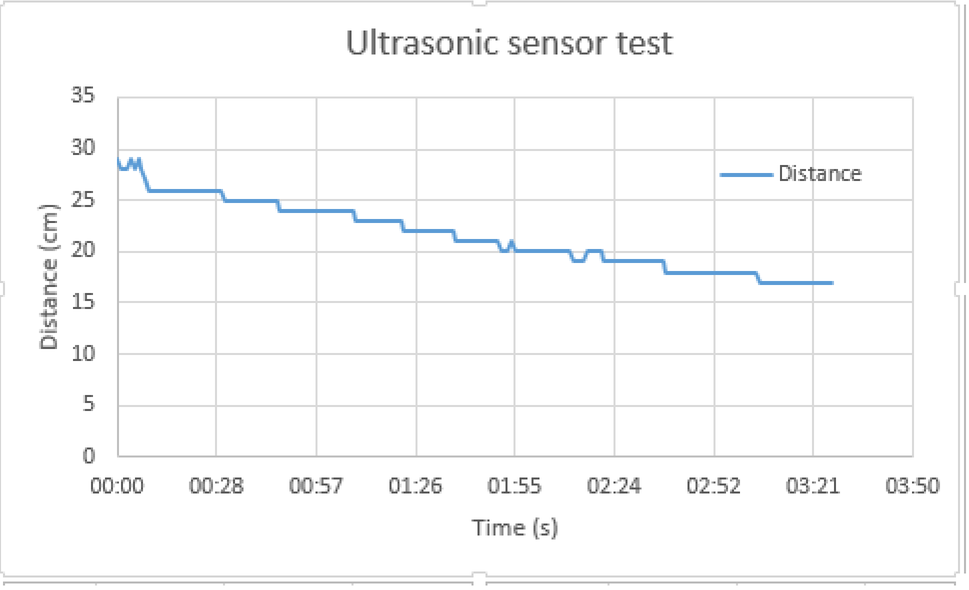


*Figure 8: Graph*

As it can be seen, the output of the sensor was stable all the time without peaks or random values.

2. The minimum, maximum distance test

To perform the test for minimum distance, the target was placed at different distances less than 20 centimeter. The Figure 8 shows where distance is calculated from and how to perform the test for the minimum distance.

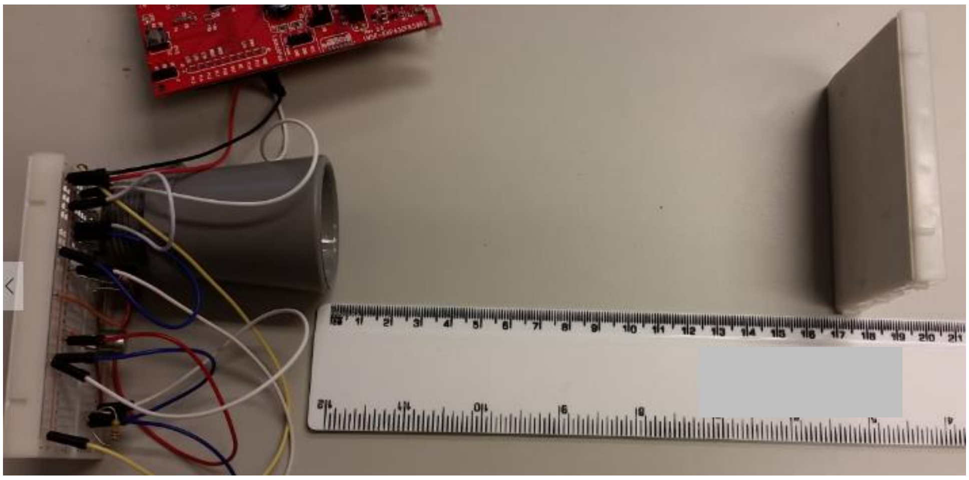


*Figure 9: Sensor detects distance in 17cm*

3. Test for different materials

The test was performed several materials, the main reason was to check if it could detect transparent water, or others, in example detecting glass. After trying with metal (aluminum), wood, glass and finally water, the results were successful in each measurement.

## 4. Conclusion

- The sensor works and the measurement could be monitored using the interface of Energia.
- The minimum distance sensor can detect is 17 cm. If the distance is less than 17 cm, the sensor gives wrong values of the target.
- The maximum distance is 765cm.
- The sensor can detect all kind of materials, including water and glass.

## 5. Results

| Test | Result |
|---|---|
| Sensor works, send status | OK |
| Maximum distance | OK |
| Minimum distance | OK |
| Sensor detect difference materials | OK |

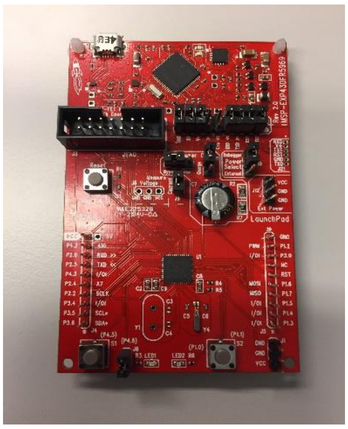
# Cover Sensor Test

| | |
|---|---|
| Requested by | IA6-5-16 group |
| Entity | University College of Southeast Norway |
| Date | 10-05-2016 |

| |
|---|
| **Object:** The object of the test was to register if the cover is open. |
| **Test Purpose:** The purpose of this test was to see if the program works and send status of cover if the cover if it is opened. |

## 1. Introduction

In this test it was used a cover sensor SSCEB31C from Honeywell. The sensor was connected to a microcontroller, MSP430FR5969.

| | |
|---|---|
| |  |
| Cover sensor | MSP430FR5969 |

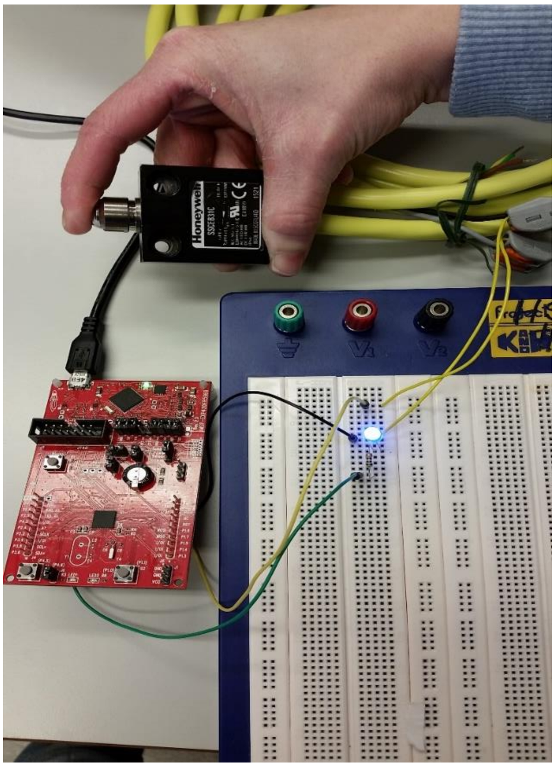
## 2. Test conditions

To test if the program sends the status of cover when interrupted, the cover sensor must be pressed down to create the interruption.  Then a led has to be set, and a message is sent to the serial monitor communication interface of Energia.

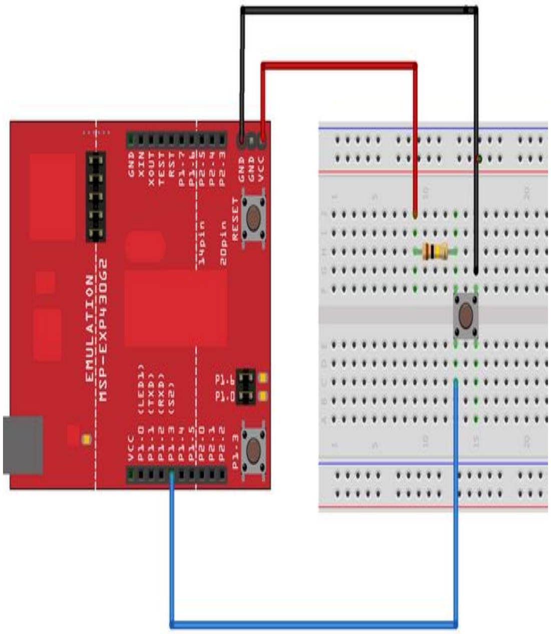
The system will remain asleep into the LPM3 (Low Power Mode 3) until an interruption is received from the cover sensor.

## 3. Setup

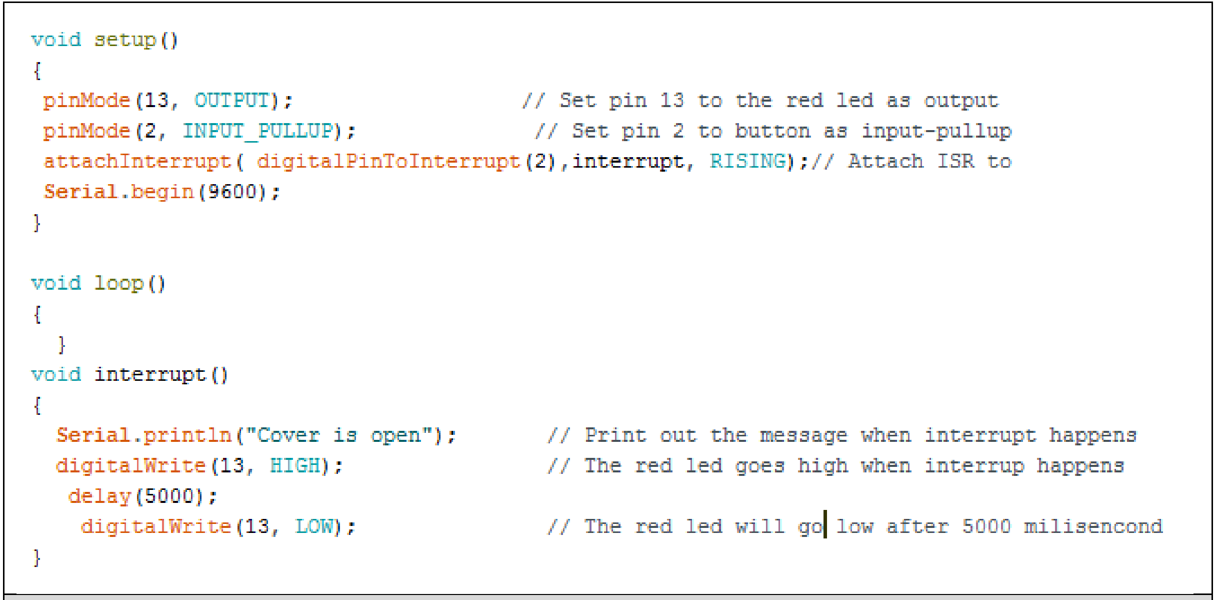
To prepare the test was needed to program the sensor and connect it to the microcontroller. For showing how the interruption works, it was connected a led to the unit with a breadboard. When the cover sensor was pressed, the LED was going to turn on, that meant the interruption was created. The picture below shows how the cover sensor was connected and how the program was written.



Cover sensor is connected to microcontroller



Schematic how to connect sensor



```
void setup()
{
 pinMode(13, OUTPUT);                    // Set pin 13 to the red led as output
 pinMode(2, INPUT_PULLUP);                // Set pin 2 to button as input-pullup
 attachInterrupt( digitalPinToInterrupt(2),interrupt, RISING);// Attach ISR to
 Serial.begin(9600);
}

void loop()
{
  }
void interrupt()
{
  Serial.println("Cover is open");        // Print out the message when interrupt happens
  digitalWrite(13, HIGH);                 // The red led goes high when interrup happens
   delay(5000);
    digitalWrite(13, LOW);                // The red led will go low after 5000 milisencond
}
```

Program-code for the cover sensor

First all the pins was decklared and the attachInterrupt was defined for an interrupt function. Secondly the microcontroller read the status of sensor, and sent it out to the monitor communication interface of Energia.

## 4. Conclusion

When everything was ready, the microcontroller was connected to the PC with a USB cable. It was proceeded to start the measurement. The output message could be read on the screen and the LED wasturned on for about 5 seconds, then it was turned off again until the next interruption. The output was responding as expected and the test was successful.

## 5. Results

| Test | Result |
|---|---|
| Code work | OK |
| Receiving of interruption | OK |

Attachment O

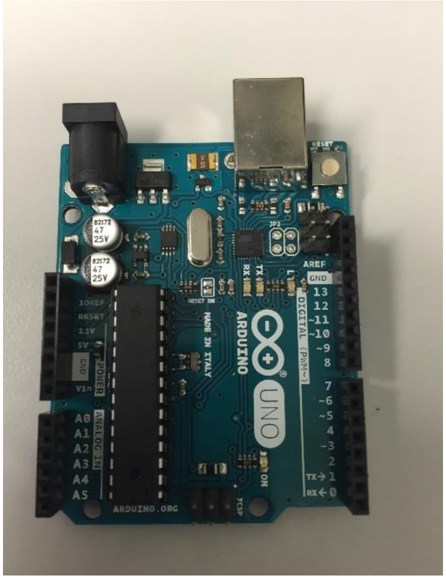
# Gas Sensor Test

| | |
|---|---|
| Requested by | IA6-5-16 group |
| Entity | University College of Southeast Norway |
| Date | 10-05-2016 |

| |
|---|
| **Object:** The object of the test was to register $H_2S$ (hydrogen sulfide) gas content. |
| **Test Purpose:** The purpose of this test was to see if the program worked and if the sensor approximately measured the right content of $H_2S$ gas. |

## 1. Introduction

In this test it was used a gas sensor from Figaro, TGS2602. The sensor was directly connected to an Arduino UNO microcontroller. It was used an Arduino instead of MSP430FRD5969 because of the 5V pin needed that the MSP microcontroller did not have. It was desirable that the sensor was able to measure the correct $H_2S$ gas content.

| | |
|---|---|
| |  |
| Gas sensor | Arduino UNO |

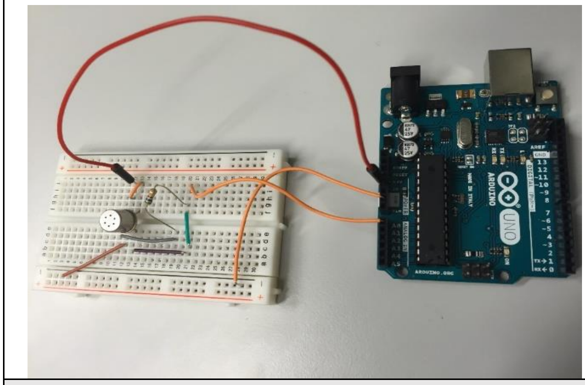
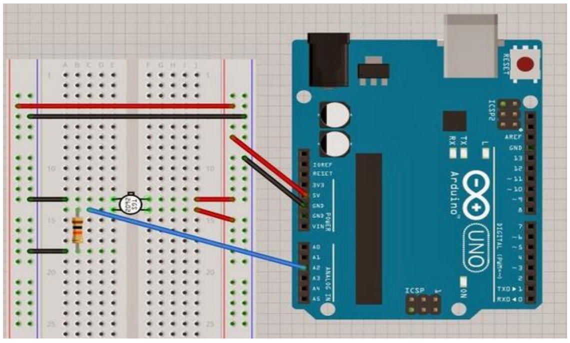
## 2. Test conditions

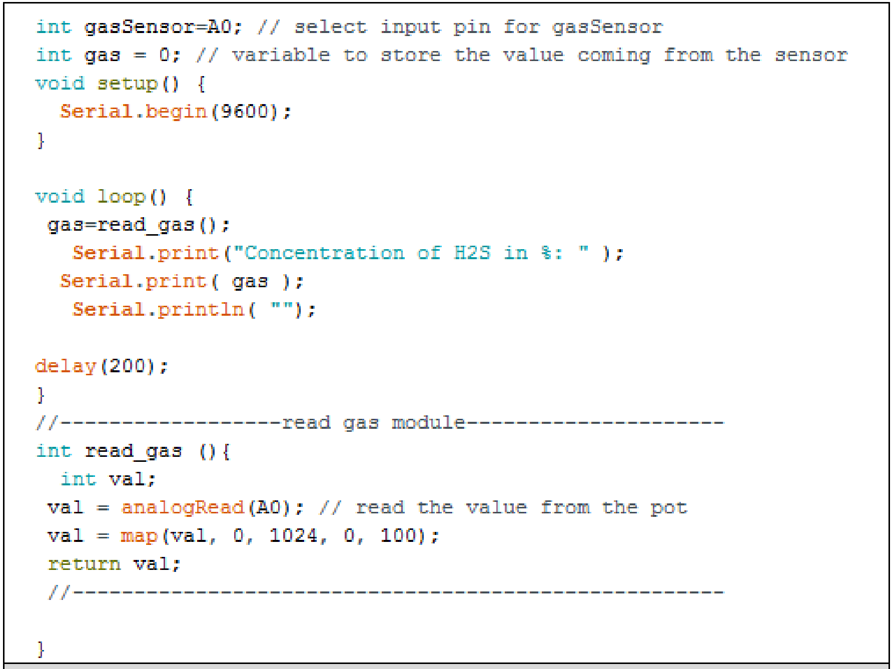
$H_2S$ is a colorless, toxic and flammable gas at room temperature, it was therefore important to take precautions to perform this in a responsible and safe way. To test if the gas sensor was working correctly, the sensor was exposed to $H_2S$ that was created by mixing $Na_2S$ (Sodium sulfide) and HCl (Hydrogen chloride) in a test chamber at laboratory B-172 at University College of Southeast Norway. The initial conditions for the test were:

- Standard atmospheric pressure, 101,325 kPa
- Atmospheric typical gases: 78% nitrogen and 21% oxygen
- 25 °C room temperature.

## 3. Setup

To program the microcontroller, it was needed to learn how the sensor worked, and therefore, understand the schematic of it. It was programmed in Arduino and the program by itself was the same as in Energia, however, the connections pins were changed from Energia to adapt them to Arduino. The sensor was connected to a breadboard and wired up to the microcontroller. The pictures below shows how it was connected and how the program was written.

| | |
|---|---|
|  |  |
| Gas sensor connected to Arduino | Schematic |



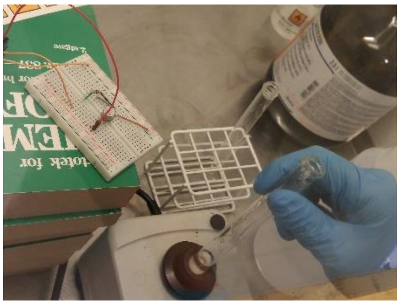
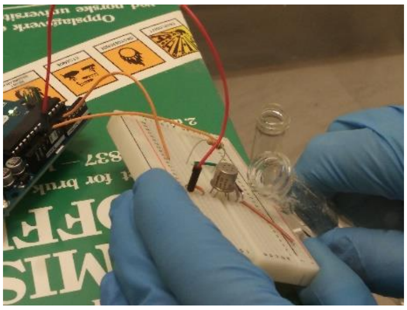
```
int gasSensor=A0; // select input pin for gasSensor
int gas = 0; // variable to store the value coming from the sensor
void setup() {
  Serial.begin(9600);
}

void loop() {
 gas=read_gas();
   Serial.print("Concentration of H2S in %: " );
  Serial.print( gas );
   Serial.println( "");

delay(200);
}
//-------------------read gas module---------------------
int read_gas (){
  int val;
 val = analogRead(A0); // read the value from the pot
 val = map(val, 0, 1024, 0, 100);
 return val;
 //-----------------------------------------------------

}
```

Program-code for the gas sensor

The program read the analog values from pin A0 on the microcontroller. The analog pin had 10 bits resolution, meaning 1024 steps of accuracy. Then, it was needed to know the measurement in tan [tan?] per cent (%). It was needed to use the command “map” (mapping), which can adapt from 1024 steps to 0 to 100. As it could be seen in the loop, the program measured the content of $H_2S$ every 0.2 second, and thereafter the program printed the values in present.

When everything was ready, the Arduino was connected to a computer with a USB cable. It was proceeded to start the measurement.

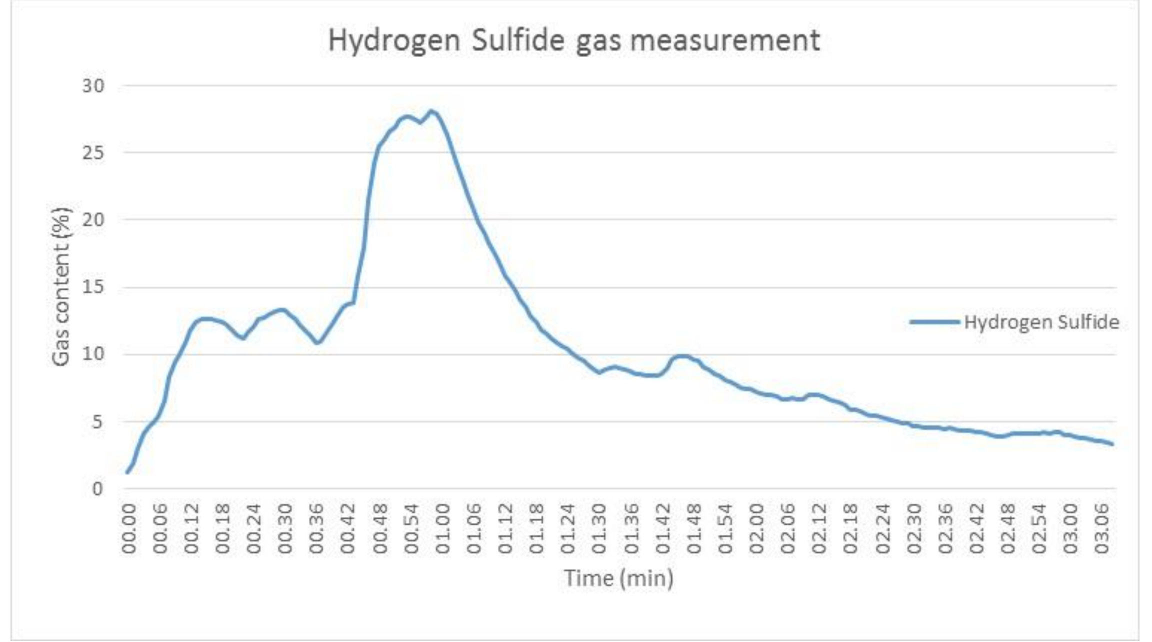
The pictures below shows how the test was performed. First, the sensor and the microcontroller were placed in the test chamber. Then the $H_2S$ gas was prepared. Finally, the sensor were exposed to $H_2S$ gas.

| | | |
|---|---|---|
| |  |  |
| Gas sensor placed in test chamber | Preparing the $H_2S$ gas | Gas sensor exposed to $H_2S$ gas |

## 4. Conclusion

The output of the measurements that could be read on the computer showed that the sensor could measure the $H_2S$ gas and the test was successful.

The graph shows that the content of $H_2S$ gas increases when the sensor is exposed for $H_2S$, also it can be seen that the graph reaches a peak when the $H_2S$ production increase suddenly and thereafter decrease when the $H_2S$ content reduces.



Graph showing the content of $H_2S$ gas

## 5. Results

| Test | Result |
|---|---|
| Code works | OK |
| Hardware works | OK |
| Measure $H_2S$ | OK |

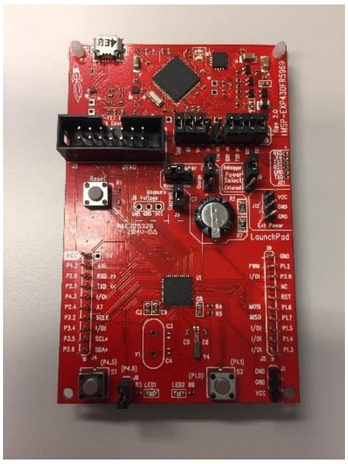
# Temperature Sensor Test

| | |
|---|---|
| Requested by | IA6-5-16 group |
| Entity | University College of Southeast Norway |
| Date | 04-05-2016 |

| |
|---|
| **Object:** The object of the test was to register the temperature. |
| **Test Purpose:** The purpose of this test was to see if the program worked and measured the temperature |

## 1. Introduction

In this test it was used a temperature sensor from Dallas, DS18B20. The sensor was connected to a microcontroller, MSP430FR5969.

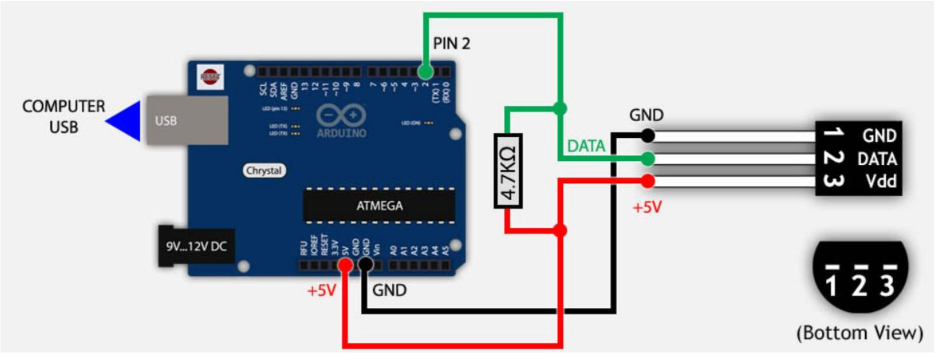
| | |
|---|---|
| |  |
| Temperature sensor | MSP430FR5969 |

## 2. Test conditions

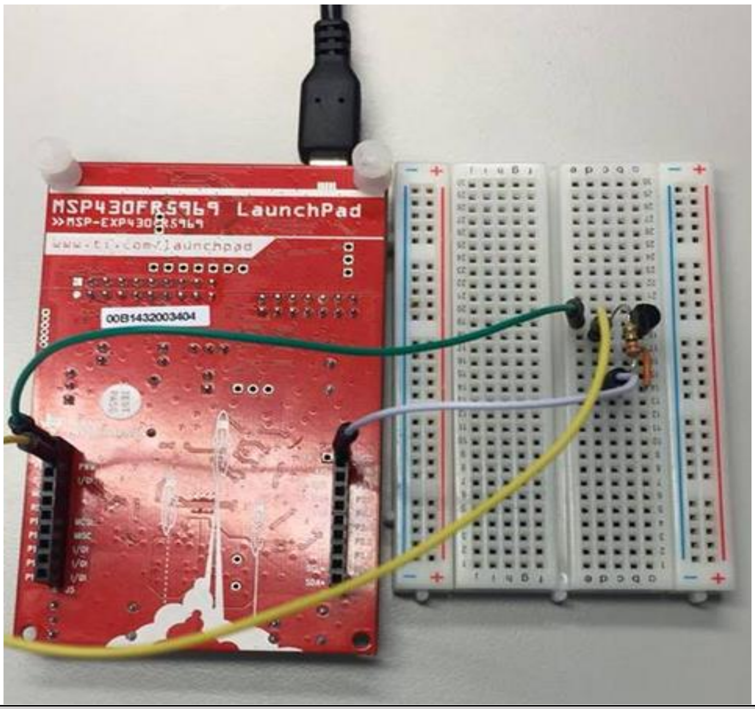
To test if the temperature was working correctly the sensor was placed in C-230d, the group room for the project group at University College of Southeast Norway, to measure the room temperature. It was used a multimeter with a thermometer incorporated, Tektronix DMM916 True RMS, it checked and contrasted the results given by the sensor.

## 3. Setup

To prepare the test it was needed to program the sensor and connect it to the microcontroller. It was programmed in Energia. The sensor and a 4.7 kohm resistor, was connected to a breadboard with wires to connect to the microcontroller. The pictures below shows how it was connected and how the program was written.

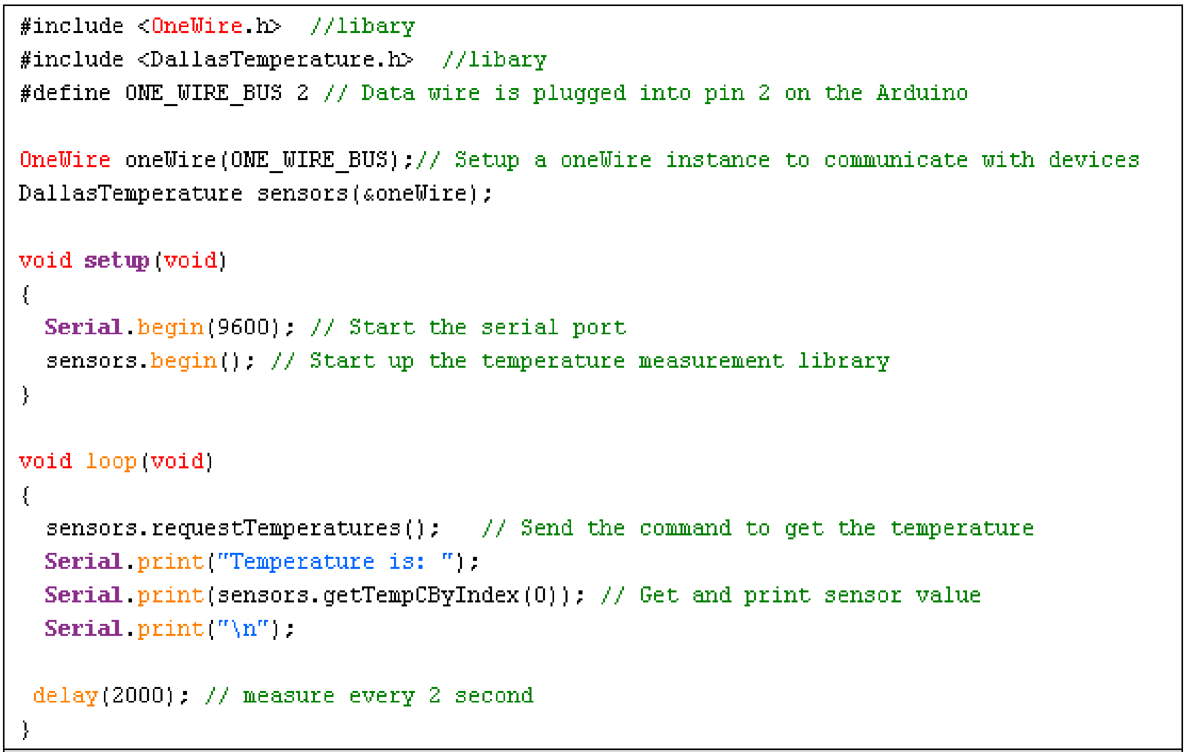


Schematic of Temperature sensor



Temperature sensor connected to the MSP430FR5969

The sensor was connected to digital pin 2, GND and 5V on the microcontroller, however the sensor can work at 3.6V.



```
#include <OneWire.h>  //libary
#include <DallasTemperature.h>  //libary
#define ONE_WIRE_BUS 2 // Data wire is plugged into pin 2 on the Arduino

OneWire oneWire(ONE_WIRE_BUS);// Setup a oneWire instance to communicate with devices
DallasTemperature sensors(&oneWire);

void setup(void)
{
  Serial.begin(9600); // Start the serial port
  sensors.begin(); // Start up the temperature measurement library
}

void loop(void)
{
  sensors.requestTemperatures();   // Send the command to get the temperature
  Serial.print("Temperature is: ");
  Serial.print(sensors.getTempCByIndex(0)); // Get and print sensor value
  Serial.print("\n");

 delay(2000); // measure every 2 second
}
```

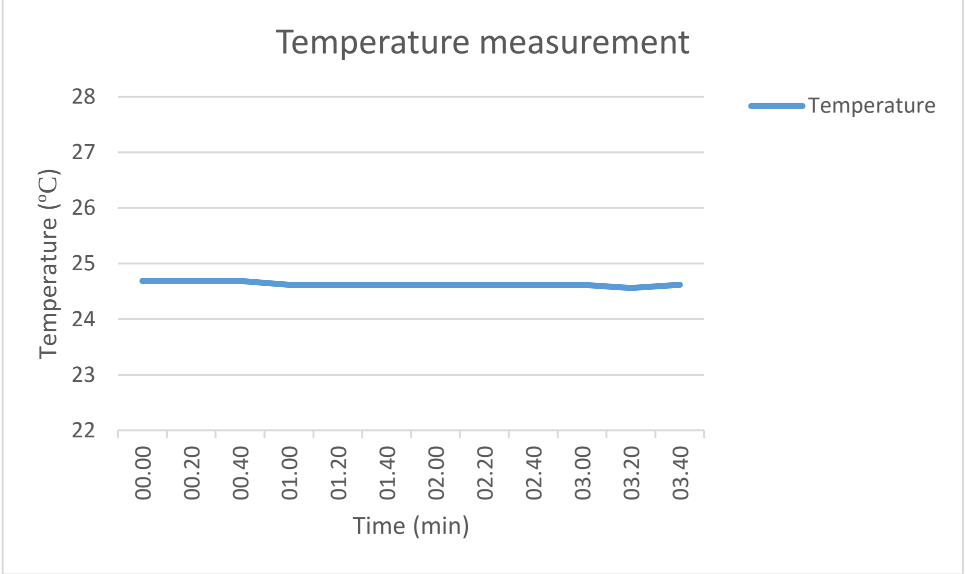
Program-code for the temperature sensor

The program read the digital values from pin 2 on the Arduino. Using “getTempCByIndex”, the value of the sensor is being converted into Celsius degrees. As it can be seen in the loop, the program measure the temperature every 2 second, and thereafter the program print the value in degrees.

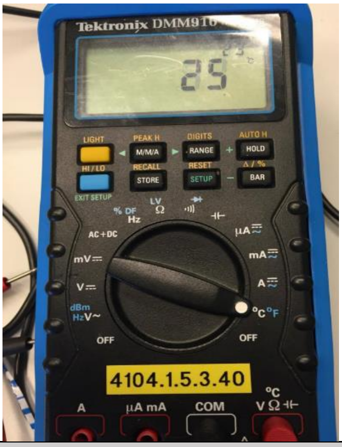
## 4. Conclusion

When everything was ready, the microcontroller, MSP430FR5969, was connected to the computer with a USB cable. It was proceeded to start the measurement. The output of the measurements could be read on the computer and showed that the temperature became stable and the test was successful.

The graph and the pictures below shows the results of the measurements of the temperature sensor. It shows that the temperature stabilized and worked as expected, also the temperature from the multimeter showed almost the same values.



Result room temperature



Multimeter with screen result of room temperature

## 5. Results

| Test | Result |
|---|---|
| Code works | OK |
| Hardware works | OK |
| Measure temperature | OK |

Attachment Q

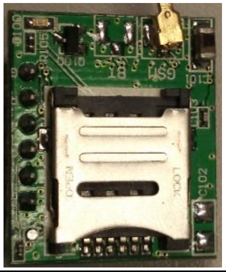
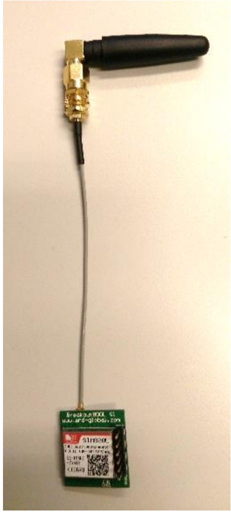
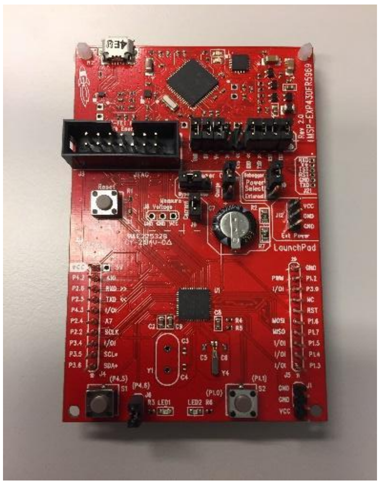
# GSM/Antenna Test

| | |
|---|---|
| Requested by | IA6-5-16 group |
| Entity | University College of Southeast Norway |
| Date | 10-05-2016 |

| |
|---|
| **Object:** The object of the test is to register that it is possible to send information. |
| **Test Purpose:** The purpose of this test is to see that the microcontroller can send measured values as a message to the mobile phone, and see how the message looks like. |

## 1. Introduction

In this test it is used a Quad-band Cellular GSM antenna SMA 2Dbi, a GSM SIM module and a mobile phone. The SIM card is placed in the SIM module and connected to the microcontroller, MSP430FR5969. Quad-band Cellular GSM antenna SMA 2Dbi was also connected to the microcontroller to test whether it works.

| | | | |
|---|---|---|---|
| <br> | |  |  |
| GSM SIM800L module | Quad-band Cellular GSM antenna SMA 2Dbi | GSM and antenna assembling | MSP430FR5969 uC |

## 2. Test conditions

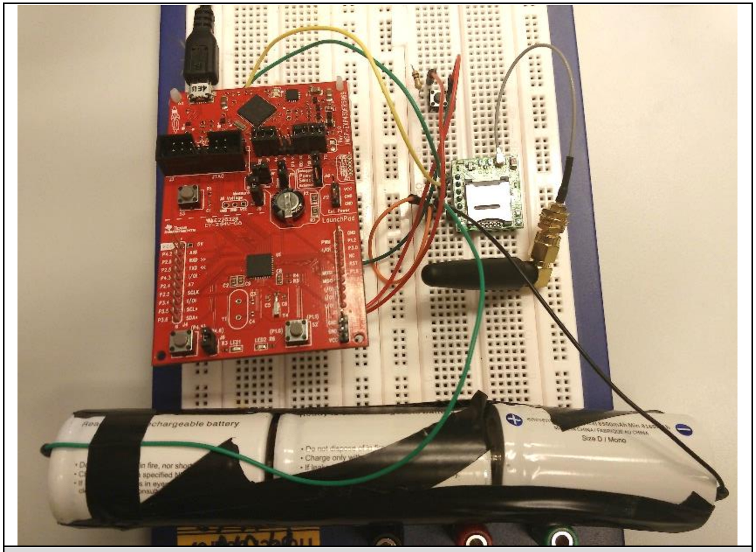
The test consist in being able to transmit a “string” through the GSM module and, therefore, receive this “string” as a SMS in the mobile phone.

## 3. Setup

The GSM module Rx and Tx pins (receiving and transmitting pins) will be connected to the uC, however, the power supply of the module has to be connected to an external battery. That is because during the transmission, the GSM module has to be fed with 2 Amps as a peak, which is too much for the microcontroller, thus it can be burned.

To test if the system can send the status of all the sensors values and alarms, the current program needs, obviously, the GSM plus the antenna plus the uC, but also it is added a switch to notice when it has to send the information.

Also it is used a personal SIM card to being able to send the information, it is placed in the right position in the GSM module.

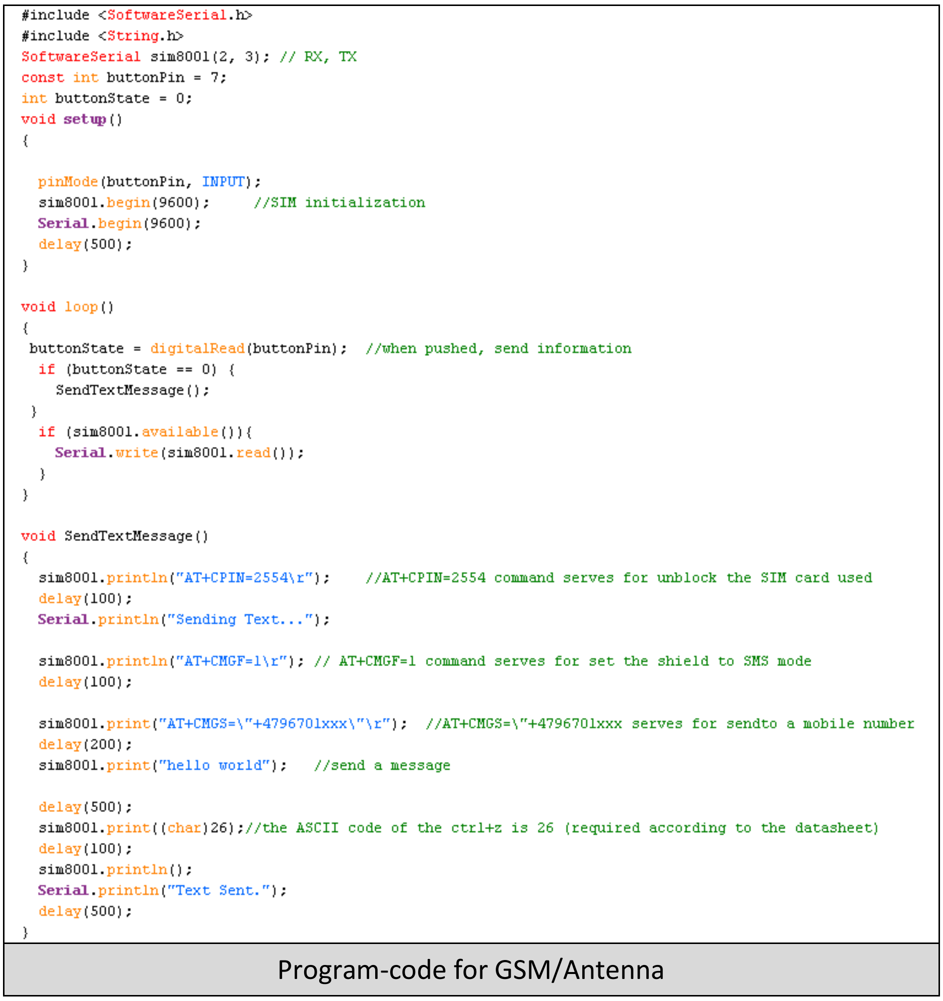


Setup of the GSM plus the antenna connected to the MSP430FR5969 and the battery

SIM card placed

Button as a pull up resistor

As it can be seen in the setup, there is the battery feeding the GSM module. Each battery has 1.2 V, as they three are placed in series, the voltage adds until 3.6 V, enough for the GSM.



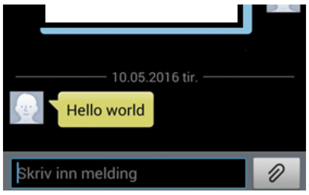
```
#include <SoftwareSerial.h>
#include <String.h>
SoftwareSerial sim8001(2, 3); // RX, TX
const int buttonPin = 7;
int buttonState = 0;
void setup()
{

  pinMode(buttonPin, INPUT);
  sim8001.begin(9600);     //SIM initialization
  Serial.begin(9600);
  delay(500);
}

void loop()
{
 buttonState = digitalRead(buttonPin);  //when pushed, send information
  if (buttonState == 0) {
    SendTextMessage();
 }
  if (sim8001.available()){
    Serial.write(sim8001.read());
  }
}

void SendTextMessage()
{
  sim8001.println("AT+CPIN=2554\r");    //AT+CPIN=2554 command serves for unblock the SIM card used
  delay(100);
  Serial.println("Sending Text...");

  sim8001.println("AT+CMGF=1\r"); // AT+CMGF=1 command serves for set the shield to SMS mode
  delay(100);

  sim8001.print("AT+CMGS=\"+4796701xxx\"\r");  //AT+CMGS=\"+4796701xxx serves for sendto a mobile number
  delay(200);
  sim8001.print("hello world");   //send a message

  delay(500);
  sim8001.print((char)26);//the ASCII code of the ctrl+z is 26 (required according to the datasheet)
  delay(100);
  sim8001.println();
  Serial.println("Text Sent.");
  delay(500);
}
```

Program-code for GSM/Antenna

This code is the one used for testing the GSM module. As it can be seen in the loop, always is checking if the button is pushed and if the GSM is available. Then, if the button is pushed, the program runs into “sendTextMessage()”, where it unblock the sim card, set it as SMS mode, add the mobile phone and  finally send the “string”.

## 4. Conclusion

After several tries, the GSM module send in the correct way the message wanted. The next screenshot shows the accomplishment of the goal.



10.05.2016 tir.

Hello world

Message received demonstration.

## 5. Results

| Test | Result |
|---|---|
| Code works | OK |
| Hardware works | OK |
| Receive a message | OK |

Attachment R

# SD-card write/read-speed test

| | |
|---|---|
| Requested by | IA6-5-16 group |
| Entity | University College of Southeast Norway |
| Date | 17-03-2016 |

**Object:** The object of this test was to observe the speed of writing and reading in the SD-card samples available.

**Test-purpose:** The purpose of the test was to observe the speed of the samples available and see which brand and size had the highest write/read ratio.

## 1. Introduction

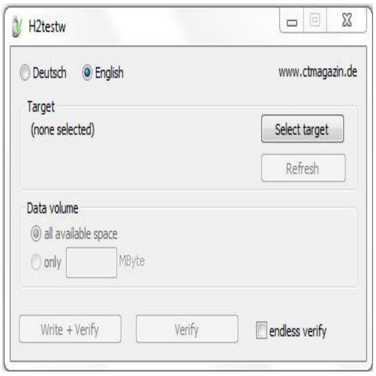
It was used SD-cards that the group got from friends and family. The brands and sizes are:

| **Brand** | **Size** |
|---|---|
| Canon | 8MB |
| Canon | 32MB |
| PNY Technologies | 256MB |
| Integral | 2GB |
| Transcend | 2GB |

For the test a SD-card reader (programmed, not a test of microcontroller) was necessary. This is a common feature in laptops.

## 2. Setup

The SD-card was connected to the laptop, with software for testing accessible online without installation. In the software it was possible to allocate 0MB to SD-card max to test with. The software did the rest of the job. (A better screenshot of software will be available in the conclusion). SD-cards were formatted (formatted to FAT32 for better ram-usage, support for bigger code and support for some extended libraries, which are not accessible on FAT-format), and allocated test-size was 300MB, for the cards that had greater size. Allocated test size for the smaller cards was max.

| | | |
|---|---|---|
| <br>H2testw<br>○ Deutsch ● English<br>www.ctmagazin.de<br>Target<br>(none selected)<br>Select target<br>Refresh<br>Data volume<br>● all available space<br>○ only MByte<br>Write + Verify<br>Verify<br>□ endless verify | | |
| Software screenshot | Connector in laptop | Connector in laptop |

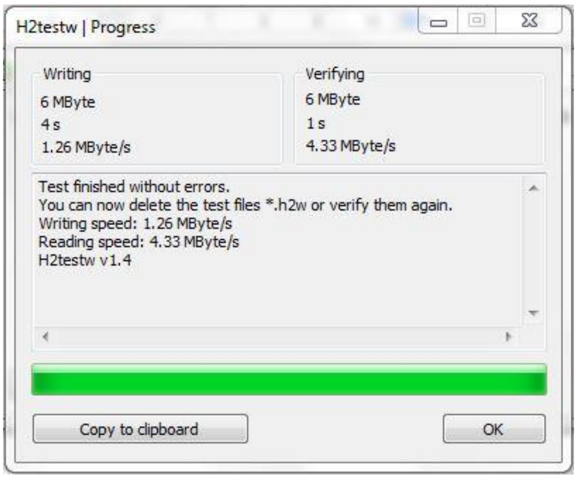
### 3. Conclusion

When the software was ready and the SD-cards are formatted, the SD-cards were tested. Results will be presented in a table.

| Brand | Write | Read |
|---|---|---|
| Canon 8MB | 1.26 MB/s | 4.33 MB/s |
| Canon 32MB | 1.18 MB/s | 3.89 MB/s |
| PNY Technologies 256MB | 6.29 MB/s | 7.54 MB/s |
| Integral 2GB | 6.14 MB/s | 16.2 MB/s |
| Transcend 2GB | 5.36 MB/s | 9.55 MB/s |

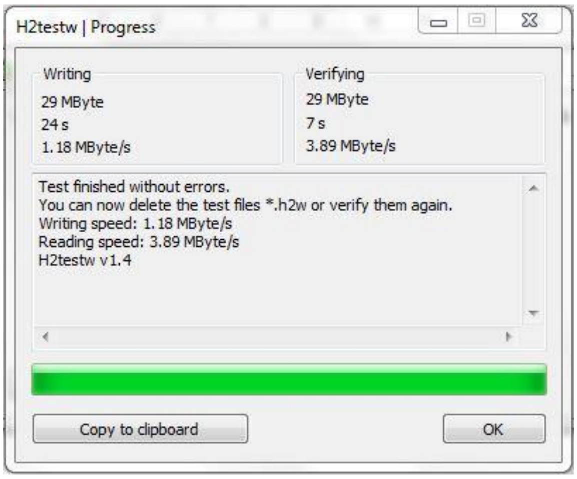
As presented in the table, the two SD-cards with the highest, similar writing speed were the PNY and the Integral SD-cards, although the Integral SD-card surpassed the reading speed of the PNY greatly.

The best (current sample) SD-card for the system was the Integral SD-card with similar writing speed to the second best, but higher reading speed. Also a factor was the storage space, with the Integral card having 2GB space, and PNY just 256 MB. This meant a longer time for the card to be written to, without formatting. This was important, where the SD-card lifetime was based on number of times a SD-card could written to, with 100,000 write-cycles was the average. “*The thinking behind this is simple: with a limit on the number of times data can be written to SD cards, and the fact that data written to the device should be spread out into untouched areas before going back to the beginning, there is less chance of writing to the same area of the card. Choosing 16GB* *over 8GB will cut by half the number of rewrites. In theory this will double the life expectancy of your storage.*" [1] For the SD-card samples, and among the two most suitable there was an increase from 256MB to 2GB, this meant in theory, a 8 time longer life-expectancy for the 2GB SD-card compared to the 256MB card.



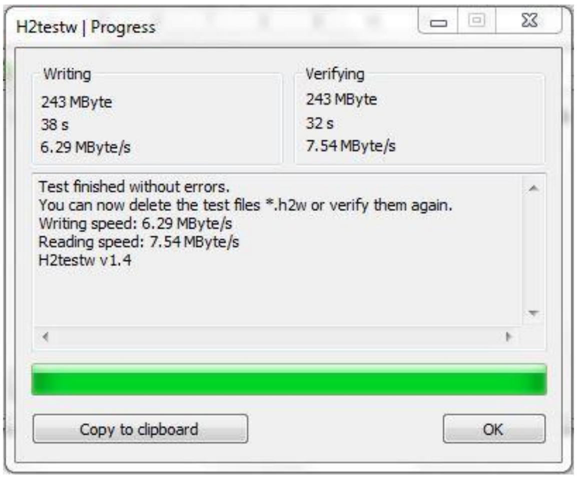
```
H2testw | Progress

Writing              Verifying
6 MByte              6 MByte
4 s                  1 s
1.26 MByte/s         4.33 MByte/s

Test finished without errors.
You can now delete the test files *.h2w or verify them again.
Writing speed: 1.26 MByte/s
Reading speed: 4.33 MByte/s
H2testw v1.4
```

Canon 8MB test



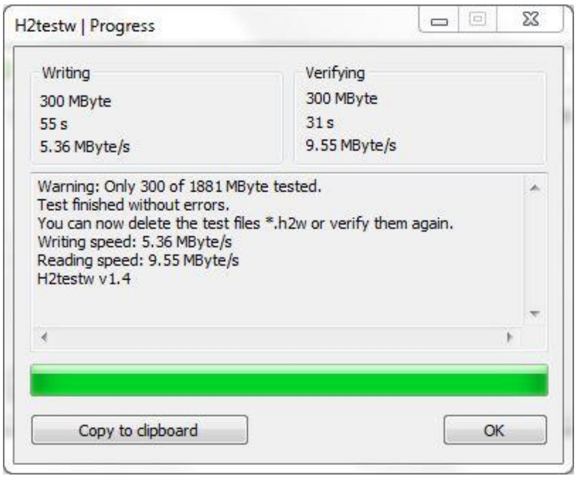
```
H2testw | Progress

Writing              Verifying
29 MByte             29 MByte
24 s                 7 s
1.18 MByte/s         3.89 MByte/s

Test finished without errors.
You can now delete the test files *.h2w or verify them again.
Writing speed: 1.18 MByte/s
Reading speed: 3.89 MByte/s
H2testw v1.4
```

Canon 32MB test



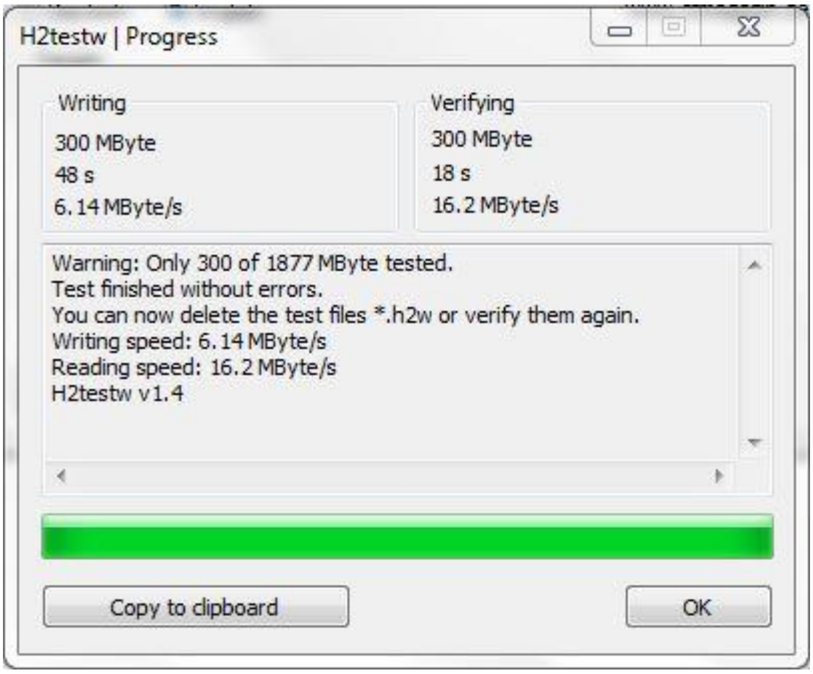
```
H2testw | Progress

Writing              Verifying
243 MByte            243 MByte
38 s                 32 s
6.29 MByte/s         7.54 MByte/s

Test finished without errors.
You can now delete the test files *.h2w or verify them again.
Writing speed: 6.29 MByte/s
Reading speed: 7.54 MByte/s
H2testw v1.4
```

PNY Tech. 256MB test



```
H2testw | Progress

Writing              Verifying
300 MByte            300 MByte
55 s                 31 s
5.36 MByte/s         9.55 MByte/s

Warning: Only 300 of 1881 MByte tested.
Test finished without errors.
You can now delete the test files *.h2w or verify them again.
Writing speed: 5.36 MByte/s
Reading speed: 9.55 MByte/s
H2testw v1.4
```

Transcend 2GB test



H2testw | Progress

Writing
300 MByte
48 s
6.14 MByte/s

Verifying
300 MByte
18 s
16.2 MByte/s

```
Warning: Only 300 of 1877 MByte tested.
Test finished without errors.
You can now delete the test files *.h2w or verify them again.
Writing speed: 6.14 MByte/s
Reading speed: 16.2 MByte/s
H2testw v1.4
```

Integral 2GB test (**best write/read speed**)

### 4. References:

[1] (http://www.makeuseof.com/tag/extend-life-raspberry-pis-sd-card/)

Attachment S

# Power Consumption Test

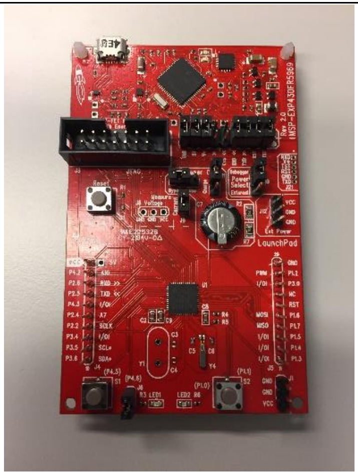
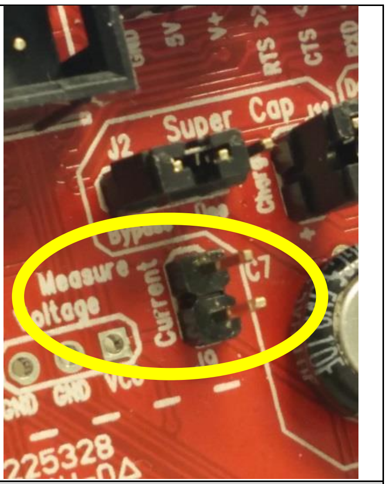
| | |
|---|---|
| Requested by | IA6-5-16 group |
| Entity | University College of Southeast Norway |
| Date | 20-04-2016 |

**Object:** The object of the test wasto see if the microcontroller could reach a low power mode.

**Test Purpose:** The purpose of this test was to see how much power microcontroller used in sleep mode.

## 1. Introduction

In this test it was used a Tektronix DMM916 True RMS Multimeter and microcontroller MSP430FR5969. The microcontroller had 2 external pins that served to measure the current. The probes were going to be connected to these pins and register the power consumption.

| | | |
|---|---|---|
| |  |  Super Cap<br>J2<br>Measure Voltage<br>Current<br>C7<br>GND<br>GND<br>VCC<br>225328 |
| Tektronix DMM916 true RMS | MSP430FR5969 | Current measurement pins |

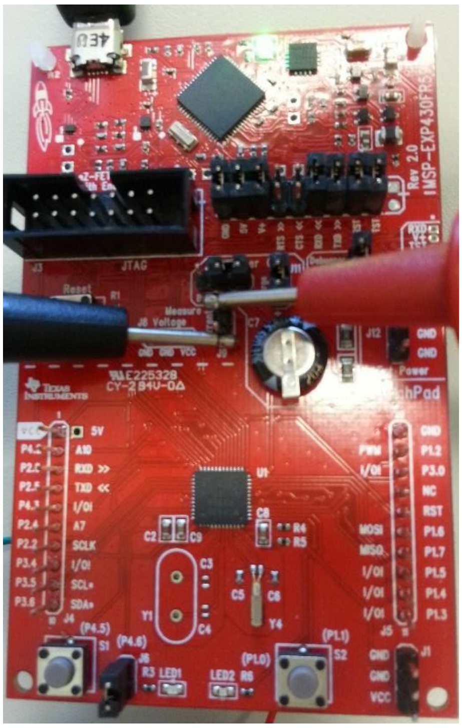
## 2. Test conditions

To test how much power consumption the microcontroller had, it is made two test parts.

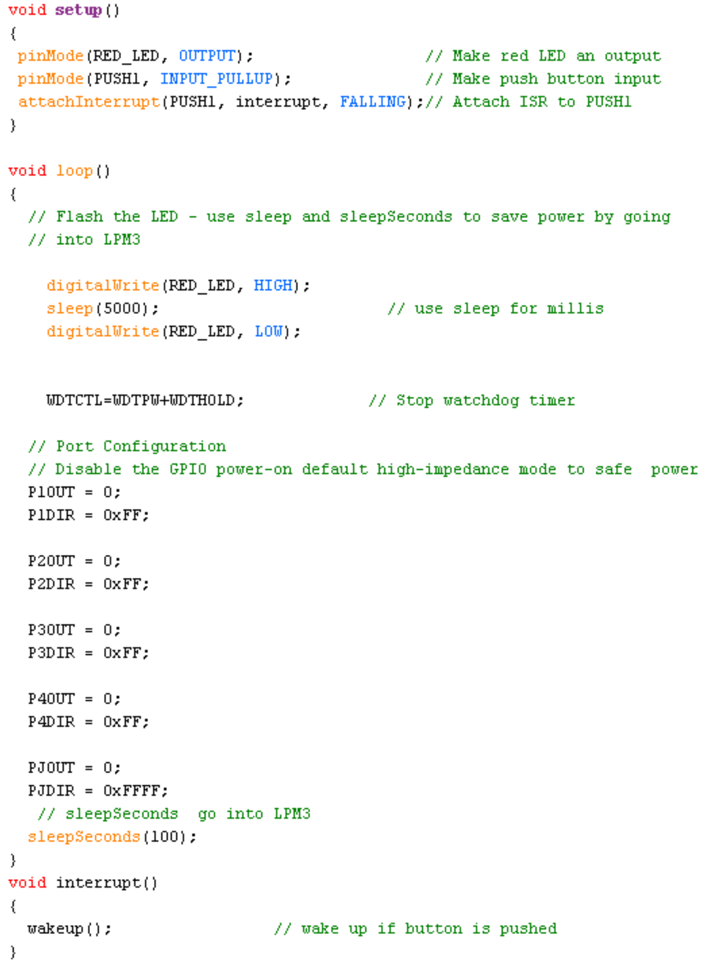
- Reach LPM3.
- Register the lowest power consumption.

## 3. Setup

To prepare the test it was needed to connect 2 probes to the microcontroller current pins, for measuring. The picture below shows how it was connected.



Probes connected to the MSP430FR5969



```
void setup()
{
 pinMode(RED_LED, OUTPUT);                  // Make red LED an output
 pinMode(PUSH1, INPUT_PULLUP);              // Make push button input
 attachInterrupt(PUSH1, interrupt, FALLING);// Attach ISR to PUSH1
}

void loop()
{
  // Flash the LED - use sleep and sleepSeconds to save power by going
  // into LPM3

    digitalWrite(RED_LED, HIGH);
    sleep(5000);                        // use sleep for millis
    digitalWrite(RED_LED, LOW);


    WDTCTL=WDTPW+WDTHOLD;             // Stop watchdog timer

  // Port Configuration
  // Disable the GPIO power-on default high-impedance mode to safe  power
  P1OUT = 0;
  P1DIR = 0xFF;

  P2OUT = 0;
  P2DIR = 0xFF;

  P3OUT = 0;
  P3DIR = 0xFF;

  P4OUT = 0;
  P4DIR = 0xFF;

  PJOUT = 0;
  PJDIR = 0xFFFF;
   // sleepSeconds  go into LPM3
  sleepSeconds(100);
}
void interrupt()
{
  wakeup();                 // wake up if button is pushed
}
```

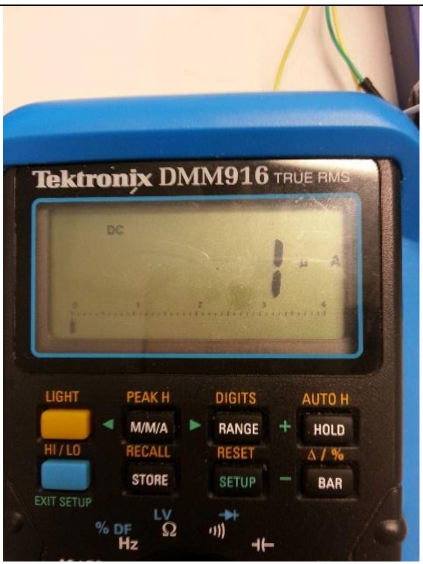
Program-code for the cover sensor

Using the previous code, it was possible to go to sleep consuming less than 1 µA. While sleeping, by pressing a button, the microcontroller woke up and turned on the red led. After 5 seconds, it turned the led off, disabled the GPIO ports, and went to sleep.

## 4. Conclusion

After doing the test, it was possible to check in the screen of the multimeter that the power consumption was less than 1 µA. It was not possible to determinate with high accuracy, because the multimeter needed more resolution. It was possible to see that the value was oscillating between 0 and 1 µA, therefore, the power consumption should be 0.4 µA as specified into the datasheet for the LMP3.

Also the power consumption was checked when in active mode, which can be seen in the following pictures.



Sleep mode power consumption

## 5. Results

| Test | Result |
|---|---|
| Reach LPM3. | OK |
| Register the Sleep mode power consumption | OK |

University College of Southeast Norway
Faculty of technology
Bachelor of Science

**REPORT FROM 6TH SEMESTER PROJECT SPRING 2016**

PRH612 Bachelor thesis

IA6-5-16

# Attachment T
# SOFTWARE TEST DOCUMENT

# TABLE OF CONTENT

# 1 INTRODUCTION

This test document contains information about testing of Software for **Read, Control and Communication unit for manholes**. It explains the various steps taken to ascertain if the program meet up to the requirements stipulated in the test plan.

# 2 TEST RESULT

The program was tested in different scenarios to determine if the requirements and functionalities were satisfied. Functions tested include:

- Display of Alarms
- Display of “Data from Manhole” from the database
- Search of data in “Data from Manhole”
- Display of “Manhole details” from the database
- Search of data in “Manhole details”
- Display of “Alarm list” from the database
- Search of data in “Alarm List”
- Display of “Company’s information”
- Login for administrator
- Editing of manhole details
- Edit administrators information
- Editing of company’s information
- Save and print of “manhole details” page
- Save and print of “data from manhole” page

## 2.1 Alarm Display

**Tested:**

1. Open Homepage window – ok

2. Colored boxes to signify situation of events – ok

**Test results**

1. Green color for normal situation

2. Red color for abnormal situation

## 2.2 Display “Data from manhole”

**Tested:**

Click “Data from manhole” on homepage – ok

**Test result:**

Display data from manhole – ok

## 2.3 Search “Data from manhole”

**Tested**:

1. Select criteria for search

2. Type in value

**Test result**:

1. Search for data - ok

## 2.4 Save/print data as pdf file from manhole

**Tested**:

1. Click “Save pdf file/print”

**Test result:**

1. File saved as pdf - ok

## 2.5 Display “Manhole details”

**Tested:**

Click “Data from manhole” on homepage – ok

**Test result:**

Display data from manhole – ok

## 2.6 Search “Manhole details”

**Tested**:

1. Select search criteria
2. Type in value

**Test result:**

Search for data - ok

## 2.7 Save/print data as pdf file from “Manhole details”

**Tested**:

Click “Save pdf file/print”

**Test result:**

File saved as pdf - ok

## 2.8 Display “Company’s Information”

**Tested:**

Click “Company information” on homepage

**Test result**:

Displaying of company information.  – Ok

## 2.9 Display “Alarm history”

**Tested:**

Click “Alarm history” on homepage – ok

**Test result:**

Display data from “Alarm history” – ok

## 2.10 Search in “Alarm history”

**Tested**:

1. Select search criteria – ok
2. Select manhole ID – ok

**Test result:**

Searched data is displayed – ok

## 2.11 Save data as pdf and print

**Tested:**

Click “Save data as pdf/print” – ok

**Test result:**

1. Data is saved as pdf - ok
2. Page print – ok

## 2.12 Login as administrator

**Tested:**

1. Type in AdminID – ok
2. Type in Password –ok

**Test result:**

1. Login successful – ok
2. Admin page is displayed – ok

## 2.13 Update Manhole details

**Tested:**

1. Select data to edit – ok

2. Click “Update” -ok

**Test result:**

1. Manhole data is updated in the database – ok
2. Message to indicate that data is successfully saved – ok

## 2.14 Delete Manhole details

**Tested:**

1. Select manholeID– ok
2. Click “Delete” - ok

**Test result:**

1. Manhole data is Deleted from the database – ok
2. Message to indicate that data is successfully deleted – ok

## 2.15 Add Manhole details

**Tested:**

1. Click “Add manhole” – ok
2. Type in ManholeID - ok
3. Type in Location - ok
4. Type in type of manhole – ok
5. Click “Add manhole”

**Test result:**

1. New manhole is added to the database – ok
2. Message to indicate that manhole is successfully added – ok

## 2.16 Edit Admin

**Tested:**

1. Select info to edit – ok
2. Click “Update Admin” - ok

**Test result:**

1. Administrator is successfully updated in the database – ok
2. Message to certify that admin info is successfully updated – ok

## 2.17 Edit Company’s info

**Tested:**

1. Select company info to edit – ok
2. Click “Edit/change”

**Tested result:**

1. Company information is updated in the database – ok
2. Message to certify that company info is successfully updated – ok

**University College of Southeast Norway**

Faculty of technology

Bachelor of Science

**REPORT FROM 6TH SEMESTER PROJECT SPRING 2016**

PRH612 Bachelor thesis

IA6-5-16

# Attachment U
# SOFTWARE DOCUMENT

# TABLE OF CONTENT

# 1 INTRODUCTION

This document tells how this application works and how the technical requirements has met the acceptable criteria. It also inform how the various task and modules in the application are interconnected.

# 2 PURPOSE AND BENEFITS

The software is to enable users to interact with data from Read, Control and Communication unit for manholes in an easy and friendly manner. It is to inform users about normal and abnormal situations in the manholes and display useful data from manholes for planning, safety and maintenance.

# 3 TECHNICAL REQUIREMENTS

This aspect sets out and describe the systems functions and services. It also describes how the system is expected to react to a particular input.

## 3.1 Functions

This is how the software is required to react to inputs from the users. The program shall have two type of users, ordinary users and administrators. It should be able to display data from the manholes and inform users about situations in the manhole. Users should be able to search for relevant data through a given search criteria. There should be possibility to save and print data. ‘

Administrators should be able to register and edit information about manholes. They should also have the capability to change administrator’s password and company information.

Normal and abnormal situations shall be displayed continuously on the home page as alarms. Both manhole details, Alarm history, Data from manholes and company’s information shall be displayed in less than a second when users navigate to their various pages from the home page. Admin page should be displayed in less than three seconds after login. Changes made by an administrator should be saved in the database immediately.

## 3.2 User Interface

This is the users interacting section (page) with the program.

### 3.2.1 Home Page

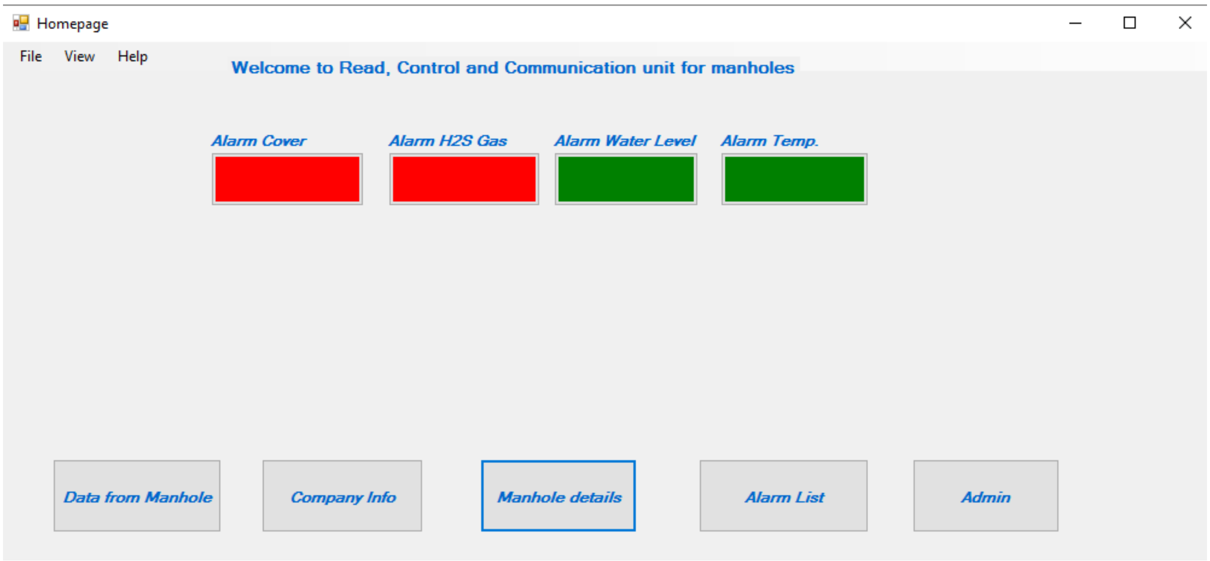
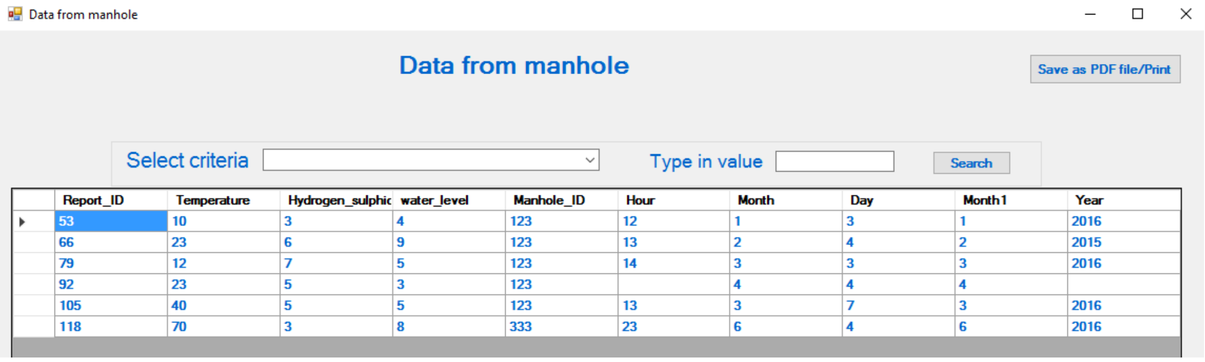


Figure 3.1 Hope page

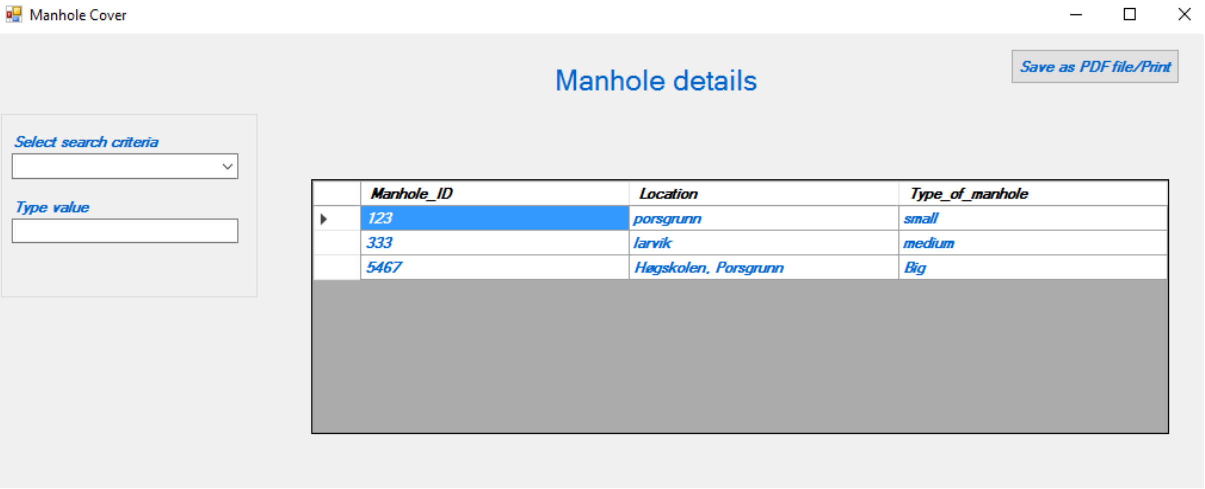
### 3.2.2 Data from Manholes



| Report_ID | Temperature | Hydrogen_sulphic | water_level | Manhole_ID | Hour | Month | Day | Month1 | Year |
|---|---|---|---|---|---|---|---|---|---|
| 53 | 10 | 3 | 4 | 123 | 12 | 1 | 3 | 1 | 2016 |
| 66 | 23 | 6 | 9 | 123 | 13 | 2 | 4 | 2 | 2015 |
| 79 | 12 | 7 | 5 | 123 | 14 | 3 | 3 | 3 | 2016 |
| 92 | 23 | 5 | 3 | 123 |  | 4 | 4 | 4 |  |
| 105 | 40 | 5 | 5 | 123 | 13 | 3 | 7 | 3 | 2016 |
| 118 | 70 | 3 | 8 | 333 | 23 | 6 | 4 | 6 | 2016 |

Figure 3.2 Data from manholes

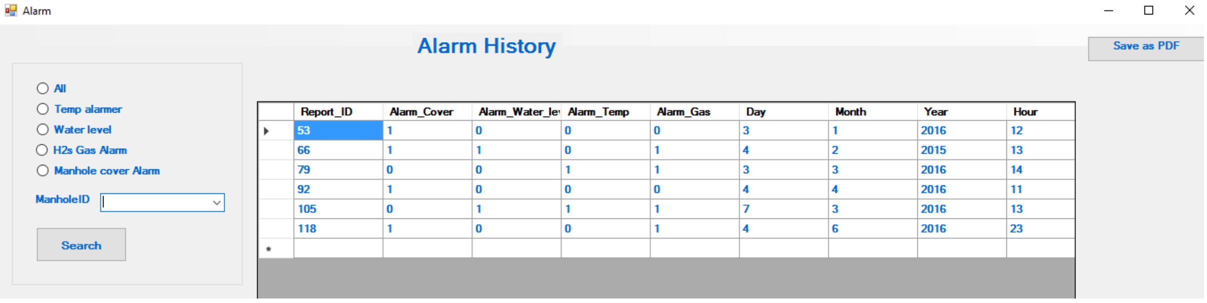
### 3.2.3 Manhole details



| Manhole_ID | Location | Type_of_manhole |
|---|---|---|
| 123 | porsgrunn | small |
| 333 | larvik | medium |
| 5467 | Høgskolen, Porsgrunn | Big |

Figure 3.3 Manhole details Page

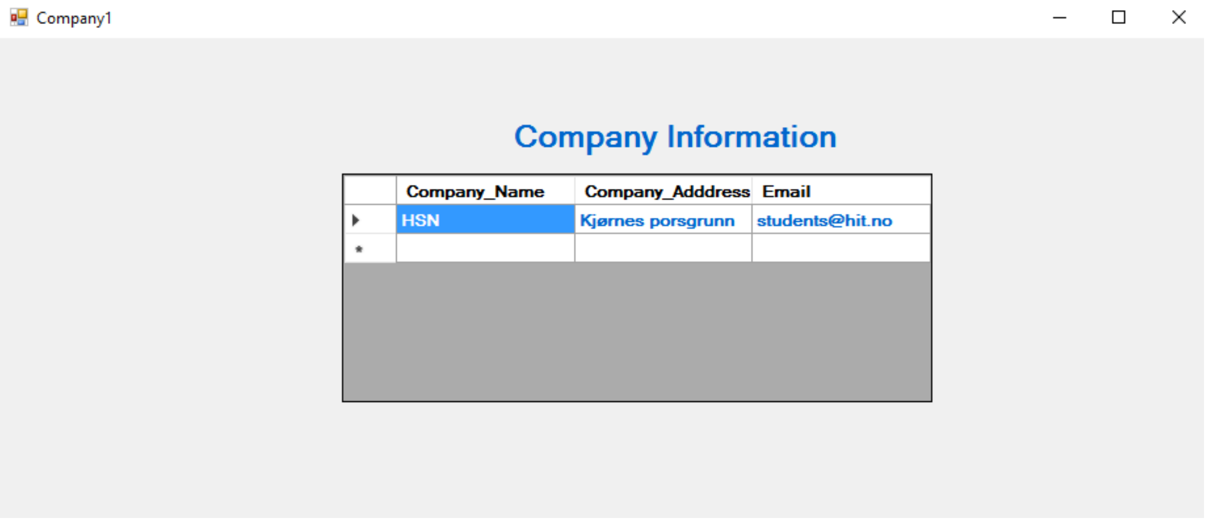
### 3.2.4 Alarm history



| Report_ID | Alarm_Cover | Alarm_Water_le | Alarm_Temp | Alarm_Gas | Day | Month | Year | Hour |
|---|---|---|---|---|---|---|---|---|
| 53 | 1 | 0 | 0 | 0 | 3 | 1 | 2016 | 12 |
| 66 | 1 | 1 | 0 | 1 | 4 | 2 | 2015 | 13 |
| 79 | 0 | 0 | 1 | 1 | 3 | 3 | 2016 | 14 |
| 92 | 1 | 0 | 0 | 0 | 4 | 4 | 2016 | 11 |
| 105 | 0 | 1 | 1 | 1 | 7 | 3 | 2016 | 13 |
| 118 | 1 | 0 | 0 | 1 | 4 | 6 | 2016 | 23 |

Figure 3.4 Alarm history page

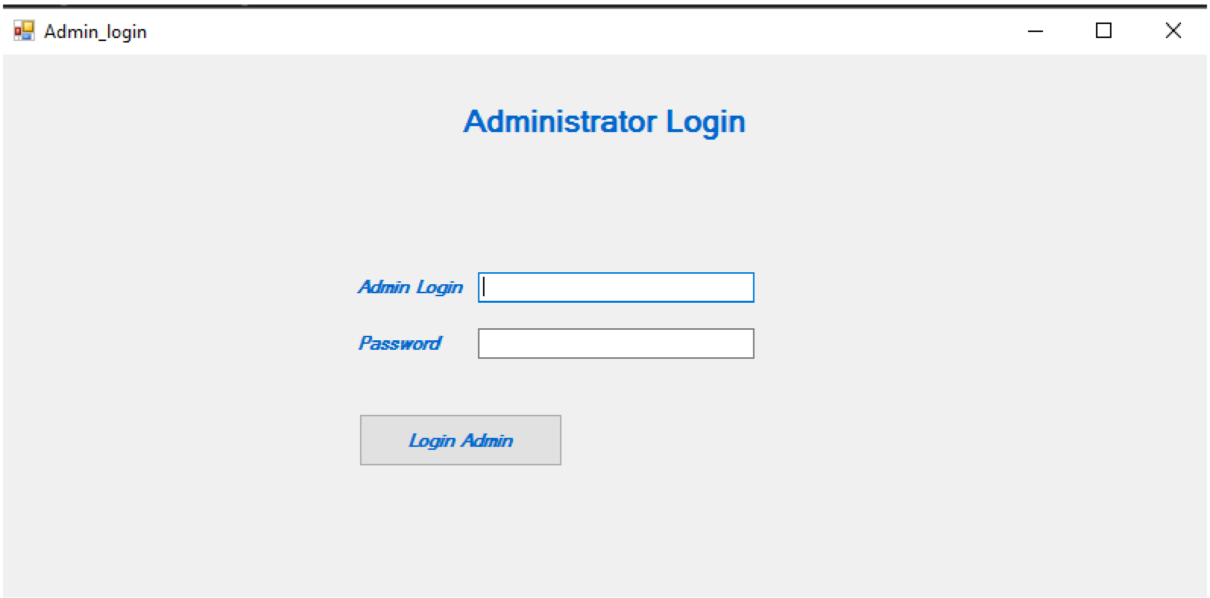
### 3.2.5 Company information



| Company_Name | Company_Adddress | Email |
|---|---|---|
| HSN | Kjørnes porsgrunn | students@hit.no |

Figure 3.5 Company's information page

### 3.2.6 Administrator's login



Admin_login

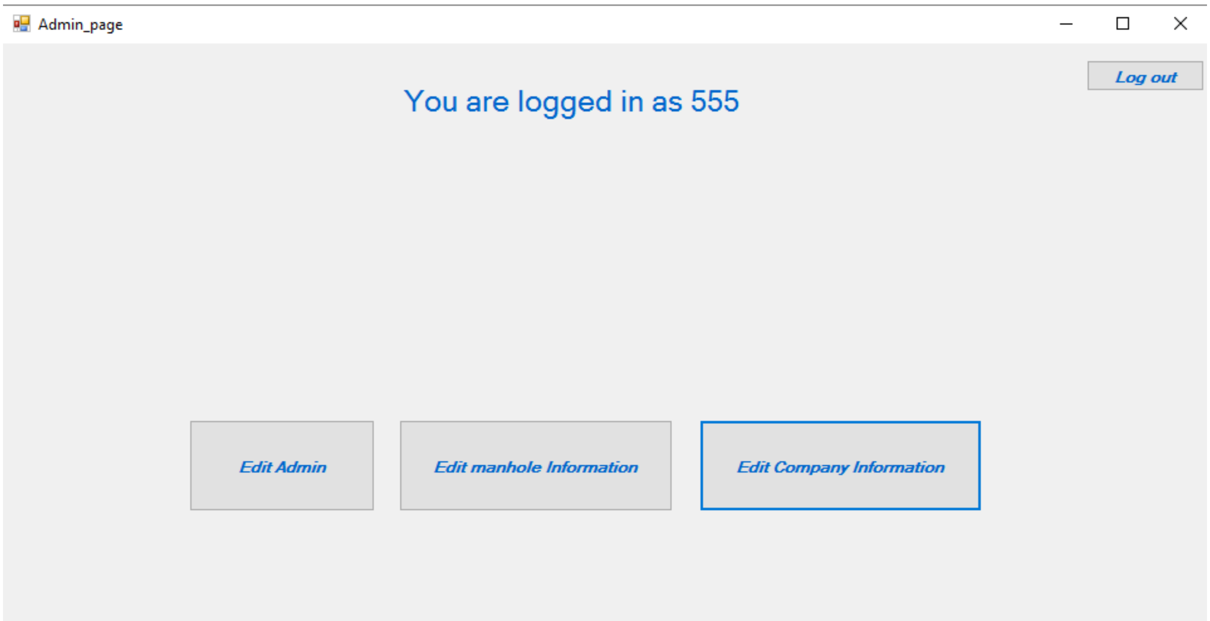
Administrator Login

Admin Login

Password

Figure 3.6 Admin Login page

### 3.2.7 Administrator's Page

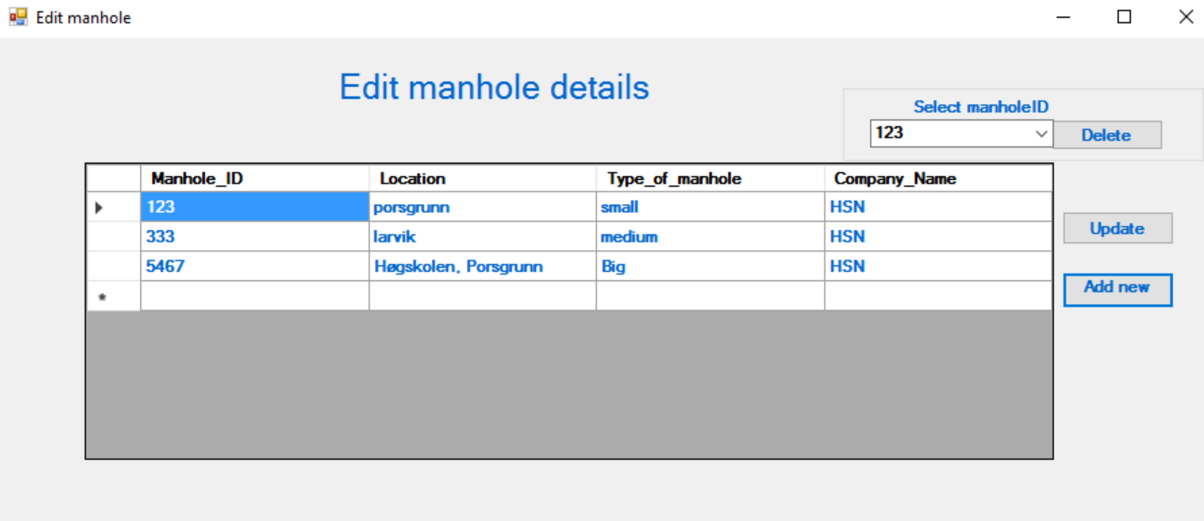


Admin_page

You are logged in as 555

Figure 3.7 Admin page

### 3.2.8 Edit manhole details



| Manhole_ID | Location | Type_of_manhole | Company_Name |
|---|---|---|---|
| 123 | porsgrunn | small | HSN |
| 333 | larvik | medium | HSN |
| 5467 | Høgskolen, Porsgrunn | Big | HSN |

Figure 3.8 Edit manhole details page

### 3.2.9 Add new manhole

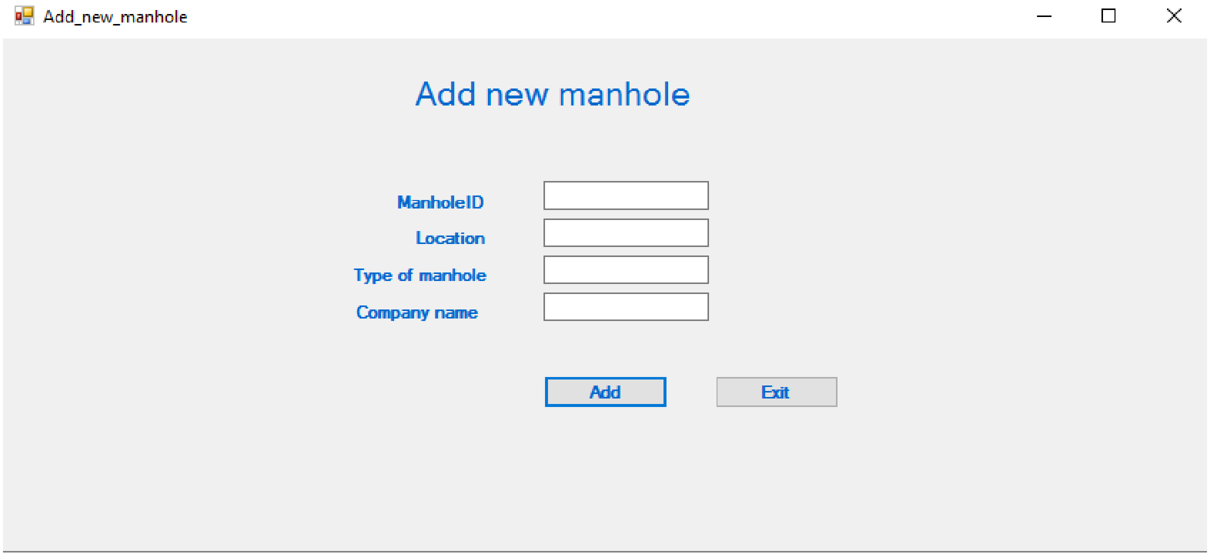
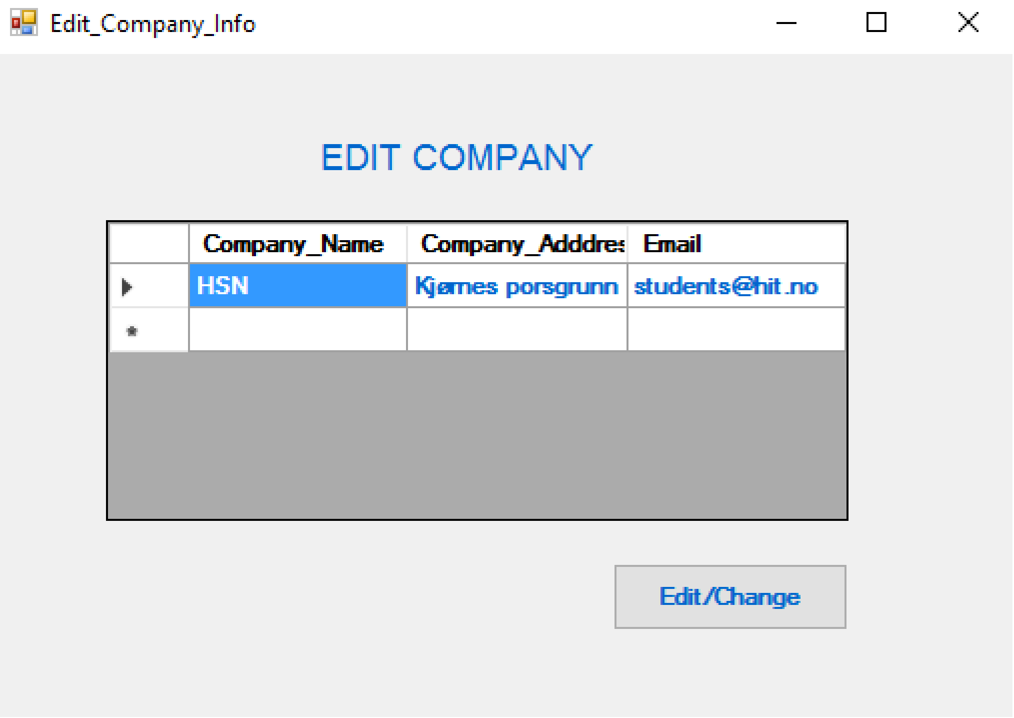


Figure 3.9 Add new manhole page

### 3.2.10 Edit Company



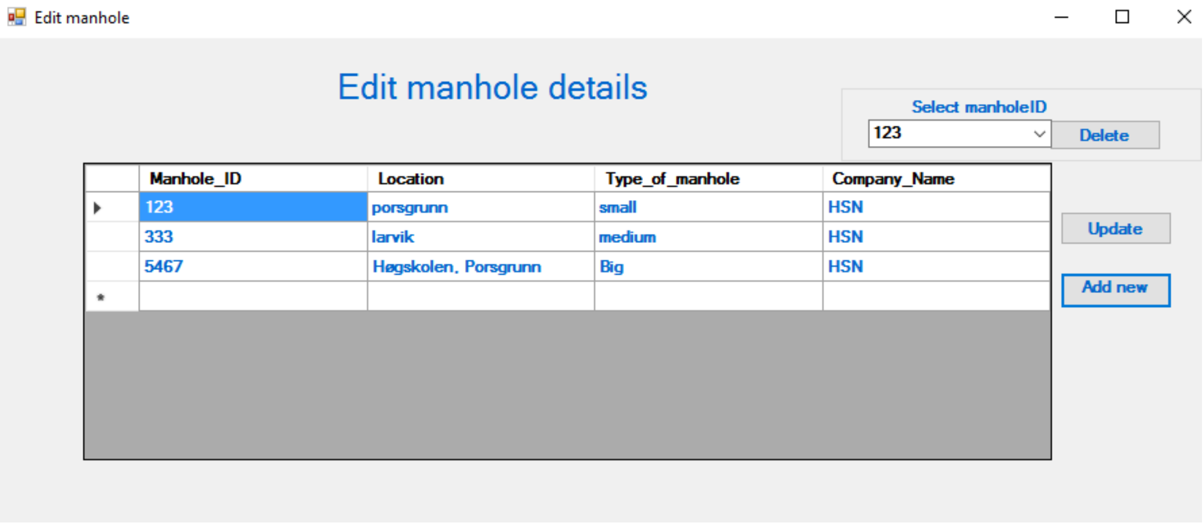
Edit_Company_Info

EDIT COMPANY

| | Company_Name | Company_Adddres | Email |
|---|---|---|---|
| ▶ | HSN | Kjørnes porsgrunn | students@hit.no |
| * | | | |

Figure 3.10 Edit Company

### 3.2.11 Edit manhole details



Edit manhole

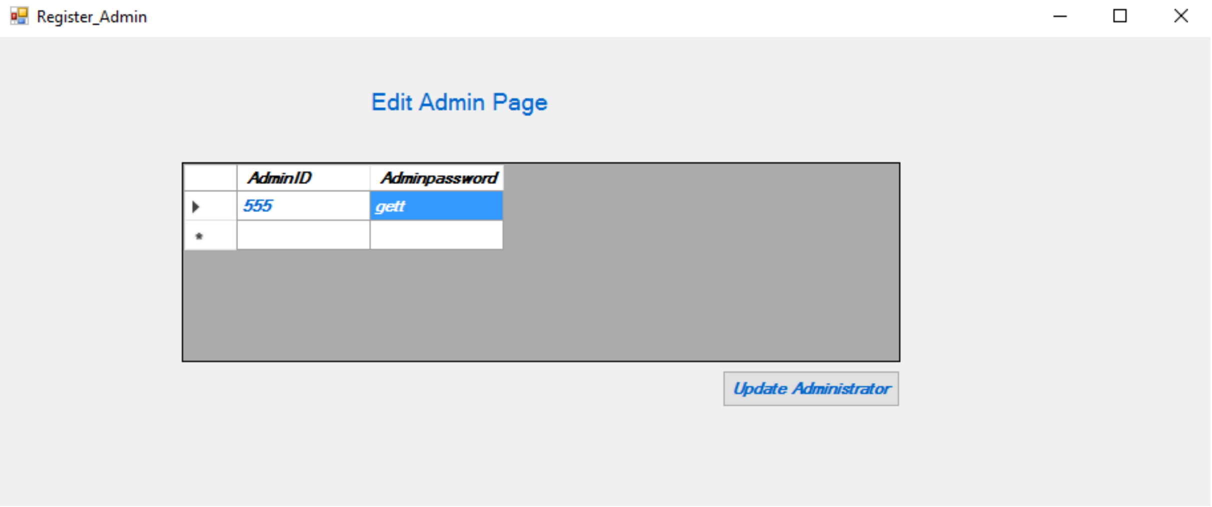
Edit manhole details

Select manholeID: 123

| | Manhole_ID | Location | Type_of_manhole | Company_Name |
|---|---|---|---|---|
| ▶ | 123 | porsgrunn | small | HSN |
| | 333 | larvik | medium | HSN |
| | 5467 | Høgskolen, Porsgrunn | Big | HSN |
| * | | | | |

Figure 3.11 Edit manhole details

### 3.2.12 Edit Administrator



| AdminID | Adminpassword |
|---|---|
| 555 | gett |

Figure 3.12 Edit Admin page

## 3.3 User Task Flow

This shows how users navigate through the program templates to view data from the Database and register/ edit data about manholes and administrators

Flow chart for software Read, Control and Communication

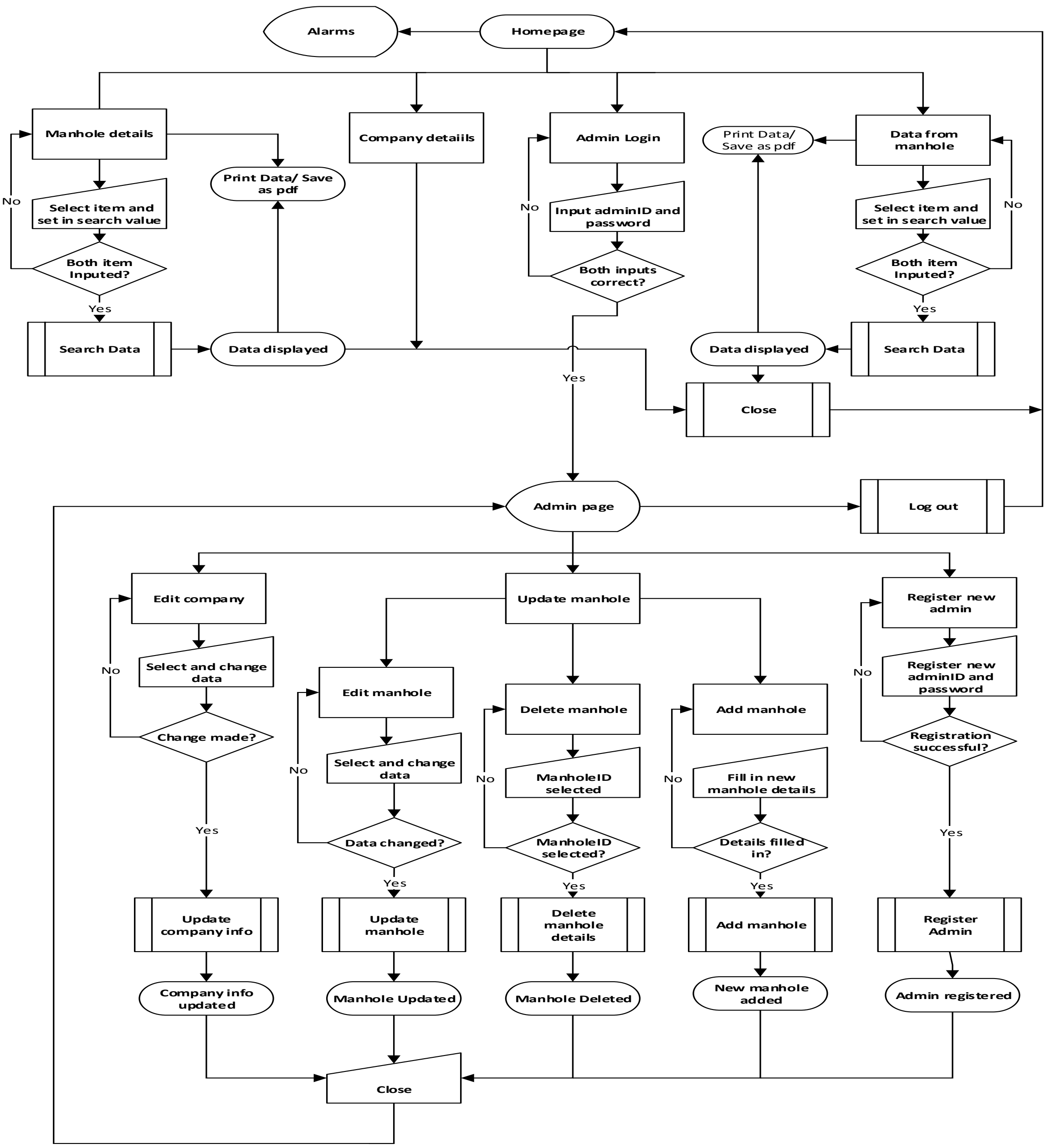


Figure 3.13 Flow chart

## 3.4 Module

The application has two main module, the visual studio application and the Database. Visual studio has been used to develop the Graphical Users Interface (GUI) as well as the codes. While the Database helps to save both data from the manholes and manhole's information.

Data from the manhole are sent from the system unit via GSM/GPRS to the Database. These data are displayed on the GUI with the help of the programming codes for users view. Data are also sent from the GUI to the database by an administrator. Administrator send data to the database both when new manholes are registered or when an editing task is performed. In other words, both GUI and the Database work hand-in-hand to execute the software task with the help of the programming codes.

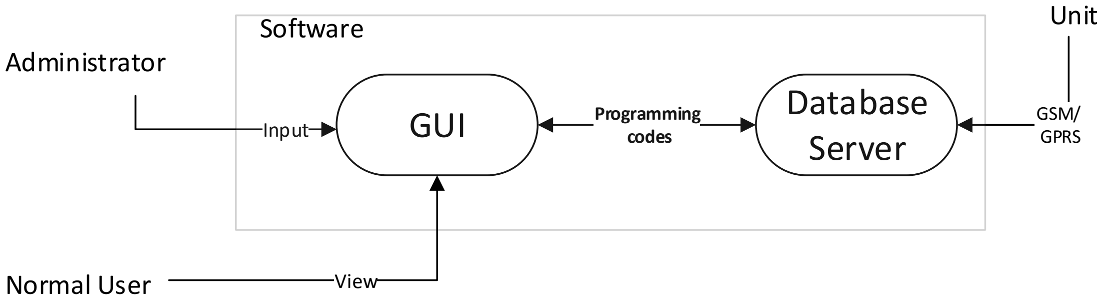


Figure 3.14:Sofrware Module

## 3.5 ER-diagram

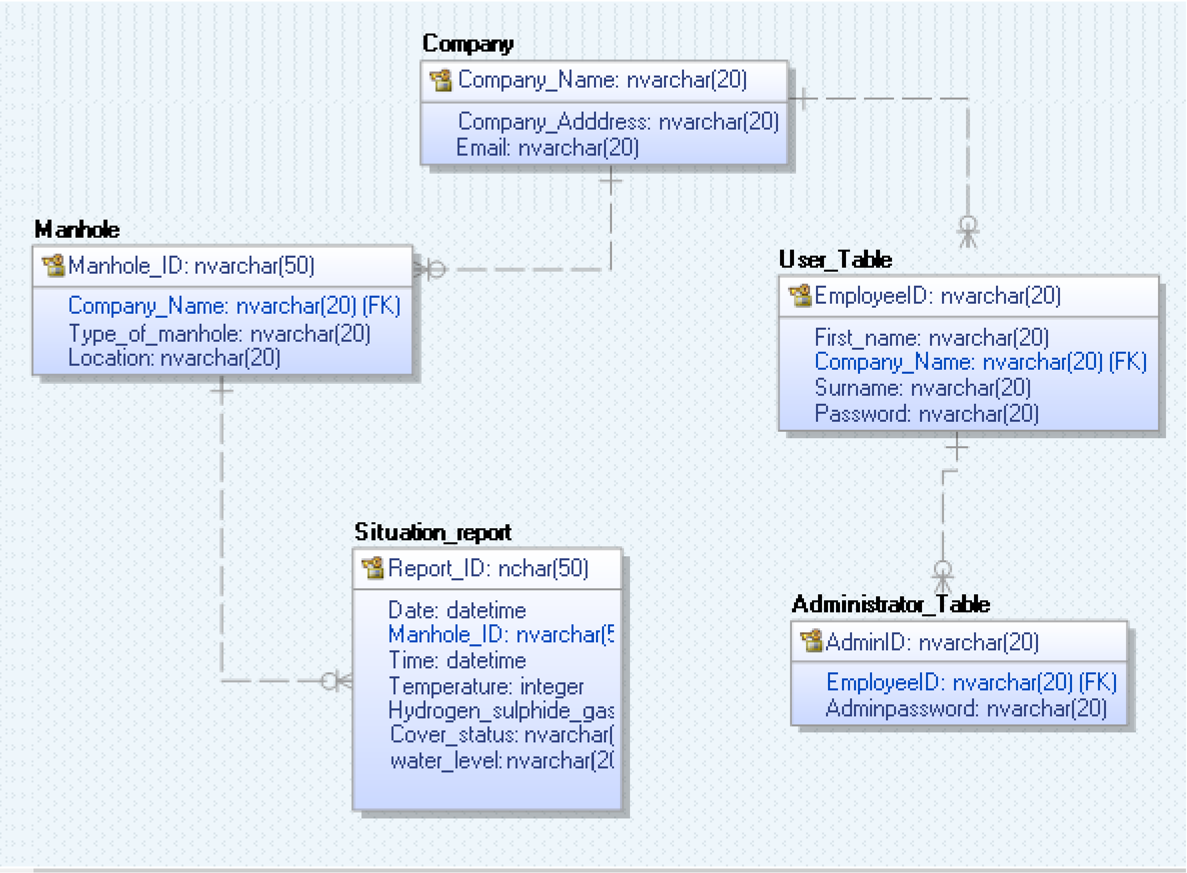


Figure 3.15 ER- Diagram

## 3.6 Use Case Diagram

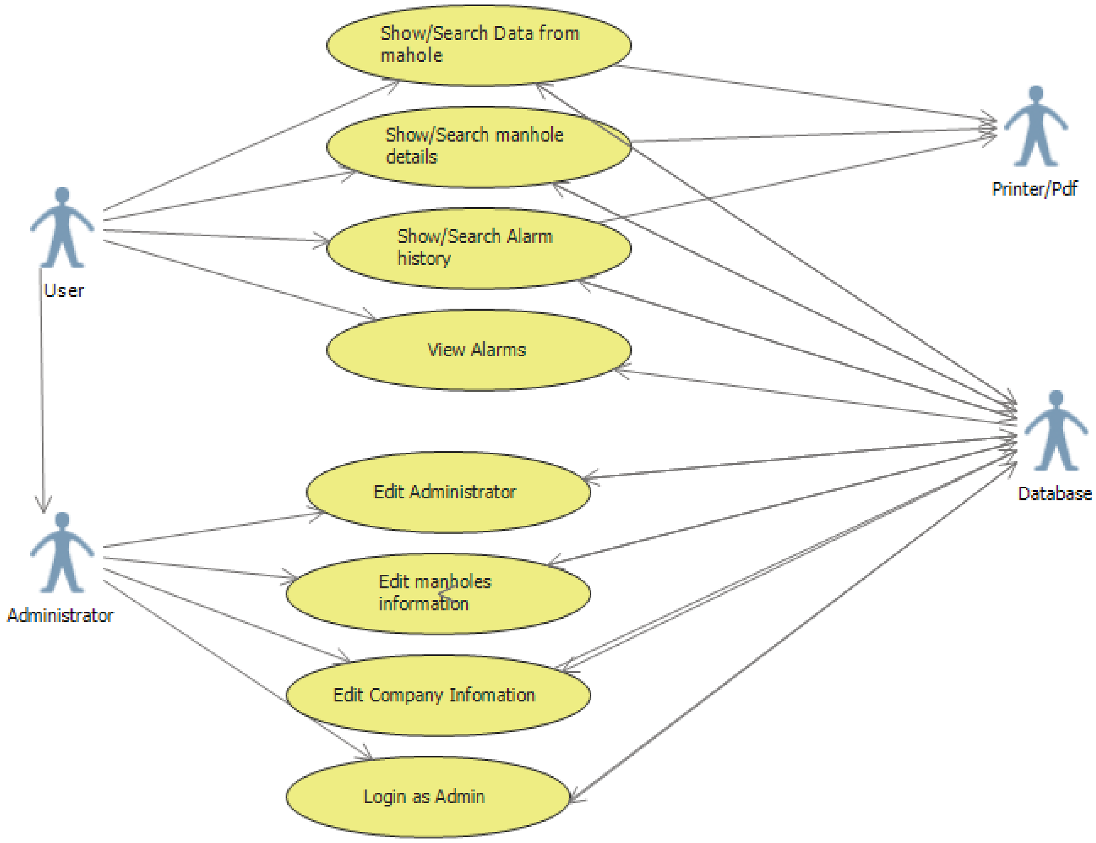

Show/Search Data from mahole
Show/Search manhole details
Show/Search Alarm history
View Alarms
Edit Administrator
Edit manholes information
Edit Company Infomation
Login as Admin
User
Administrator
Printer/Pdf
Database

Figure 3.16 Use Case Diagram

## 3.7 Class Diagram

Figure 3.17 Class Diagram

# 4 ACCEPTANCE CRITERIA

The application is now working properly without any form of error and it is easy to use by both ordinary users and administrator.

# 5 VERIFICATION

The application has been tested in Microsoft Windows 10 operating Computer.

**University College of Southeast Norway**

Faculty of technology

Bachelor of Science

**REPORT FROM 6$^{TH}$ SEMESTER PROJECT SPRING 2016**

PRH612 Bachelor thesis

IA6-5-16

# Attachment V
# User Manual

**Address: Kjølnes ring 56, N-3918 Porsgrunn, Tel: +47 31 00 80 00, www.usn.no**

Bachelor's programmes – Master's programmes – Ph.D. programmes

# TABLE OF CONTENT

# 1 ABOUT THE PROGRAM

Read, control and communication unit for manholes is a software program for monitoring all measurement from manholes. The software is for two types of users: regular user and administrator user. Both types can monitor and control all measurements, in addition the administrator can also edit administrator-, manholes- and company information.

# 2 USER MANUAL

This user manual describes the normal use of the software.

## 2.1 Homepage

The homepage consists of five buttons: **Data from Manhole**, **Company Info, Manhole details, Alarm List, Admin** and alarm status for four measurements: **Alarm Cover, Alarm H2S Gas, Alarm Water Level, Alarm Temp.** Red color means something wrong with measurement, green color means everything is ok.

There are two ways to navigate to different windows, the user can click on different buttons or click on **View** tab on top left corner in the window.

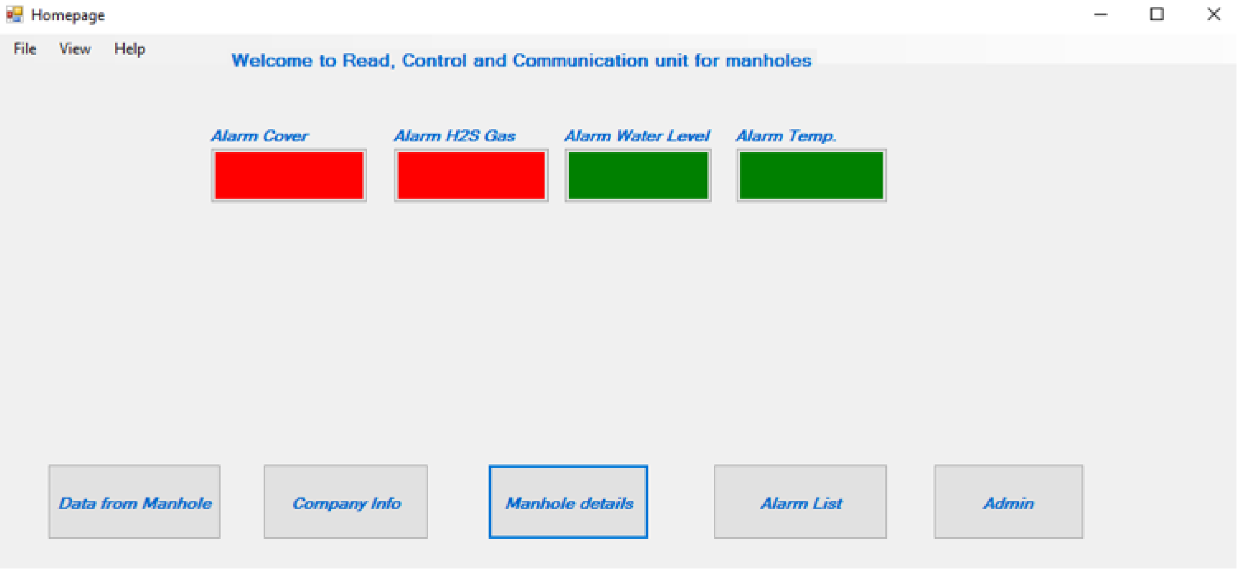


Figure 2.1-1: Homepage

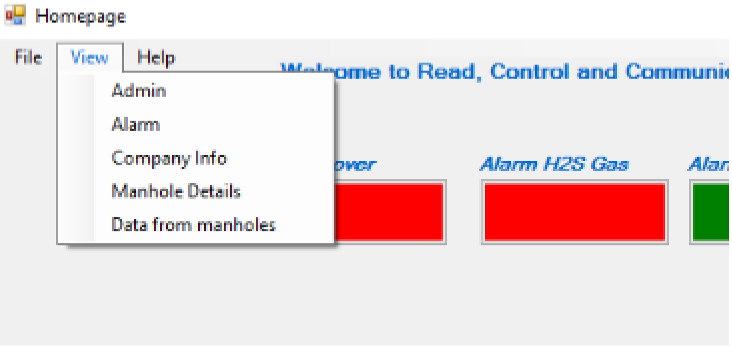
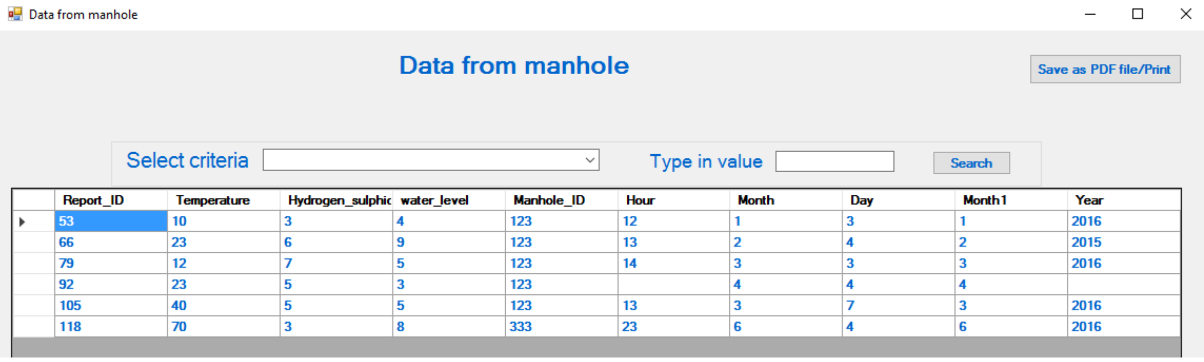


Figure 2.1-2: View field

## 2.2 Data from manhole

To view data from manhole, click on the **Data from Manhole** button in **Homepage** or in **View**.

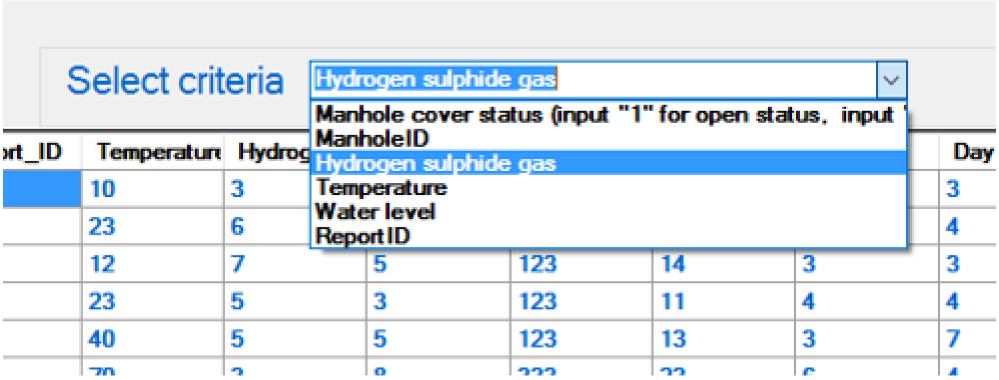


| | Report_ID | Temperature | Hydrogen_sulphic | water_level | Manhole_ID | Hour | Month | Day | Month1 | Year |
|---|---|---|---|---|---|---|---|---|---|---|
| ▸ | 53 | 10 | 3 | 4 | 123 | 12 | 1 | 3 | 1 | 2016 |
| | 66 | 23 | 6 | 9 | 123 | 13 | 2 | 4 | 2 | 2015 |
| | 79 | 12 | 7 | 5 | 123 | 14 | 3 | 3 | 3 | 2016 |
| | 92 | 23 | 5 | 3 | 123 | | 4 | 4 | 4 | |
| | 105 | 40 | 5 | 5 | 123 | 13 | 3 | 7 | 3 | 2016 |
| | 118 | 70 | 3 | 8 | 333 | 23 | 6 | 4 | 6 | 2016 |

Figure 2.2-1: Data from manhole

## 2.2.1 Specific search

Select criteria by clicking on the **Select search criteria** drop-down list, enter the manhole ID in **Type in value** field, then click on the **Search** button.



| rt_ID | Temperatur | Hydrog | | | | | Day |
|---|---|---|---|---|---|---|---|
| | 10 | 3 | | | | | 3 |
| | 23 | 6 | | | | | 4 |
| | 12 | 7 | 5 | 123 | 14 | 3 | 3 |
| | 23 | 5 | 3 | 123 | 11 | 4 | 4 |
| | 40 | 5 | 5 | 123 | 13 | 3 | 7 |

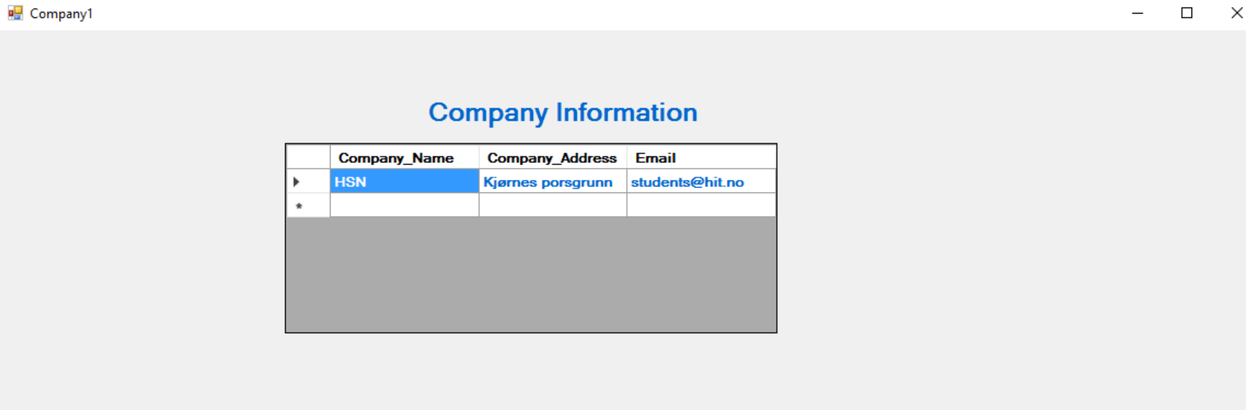
Figure 2.2-2: Select criteria drop-down list

## 2.2.2 Save as PDF/Print

Save the manhole information as a PDF file to the computer, click on the **Save as PDF file/print** button.

## 2.3 Company Information

To access company information, click on the **Company information** button from **Homepage** or in **View**.



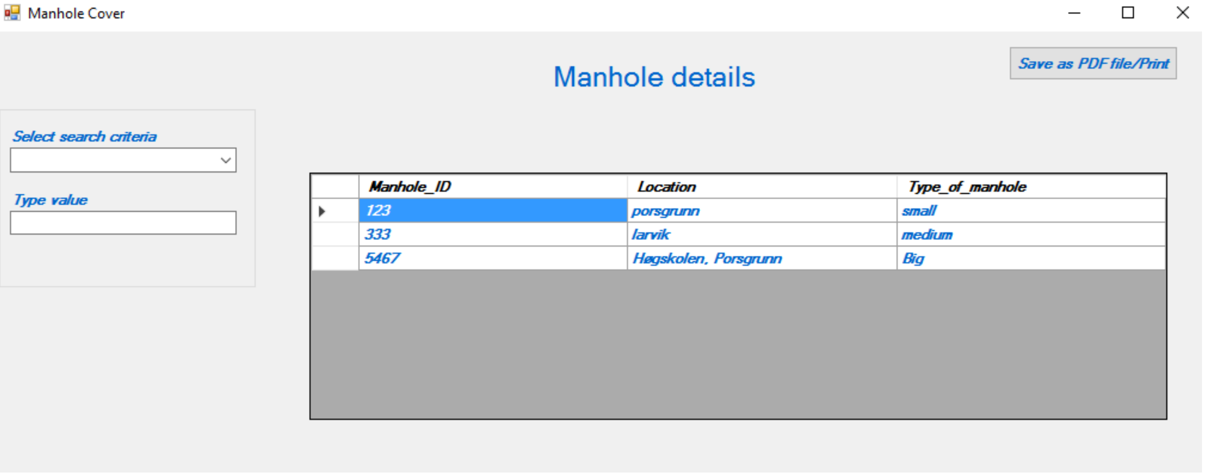
Company Information

| Company_Name | Company_Address | Email |
|---|---|---|
| HSN | Kjørnes porsgrunn | students@hit.no |
| | | |

Figure 2.3-1: Company Information

## 2.4 Manhole details

To view manhole detail, click on the **Manhole details** button in **Homepage** or in **View**.



Manhole details

| Manhole_ID | Location | Type_of_manhole |
|---|---|---|
| 123 | porsgrunn | small |
| 333 | larvik | medium |
| 5467 | Høgskolen, Porsgrunn | Big |

Figure 2.4-1: Manhole details

### 2.4.1 Specific search

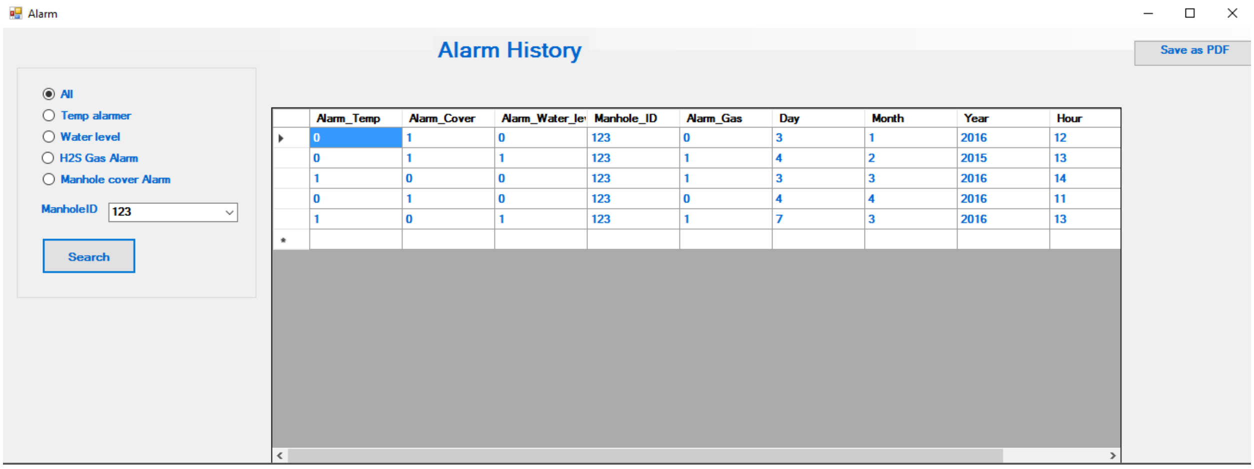
Select criteria by clicking on the **Select search criteria** drop-down list and enter the manhole ID in **Type in value** field, the desired data will show automatically in data grid view.

### 2.4.2 Save as PDF

Same as 2.2.2

## 2.5 Alarm History

To access alarm list, click on the **Alarm History** button from **Homepage** or in **View.** Number **1** stands for active alarm and **0** stands for non-active alarm.



● All
○ Temp alarmer
○ Water level
○ H2S Gas Alarm
○ Manhole cover Alarm

ManholeID 123

| Alarm_Temp | Alarm_Cover | Alarm_Water_le | Manhole_ID | Alarm_Gas | Day | Month | Year | Hour |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 123 | 0 | 3 | 1 | 2016 | 12 |
| 0 | 1 | 1 | 123 | 1 | 4 | 2 | 2015 | 13 |
| 1 | 0 | 0 | 123 | 1 | 3 | 3 | 2016 | 14 |
| 0 | 1 | 0 | 123 | 0 | 4 | 4 | 2016 | 11 |
| 1 | 0 | 1 | 123 | 1 | 7 | 3 | 2016 | 13 |

Figure 2.5-1: Alarm History

### 2.5.1 Specific search

Select alarm type, choose the manhole ID from the drop-down list, then click so on the **Search** button.

### 2.5.2 Save as PDF

Same as 2.2.2

## 2.6 Admin Login

Log in as administrator by clicking on the **Admin** button from **Homepage** or in **View.**

Enter AdminID and password then click on the **Login Admin** button.

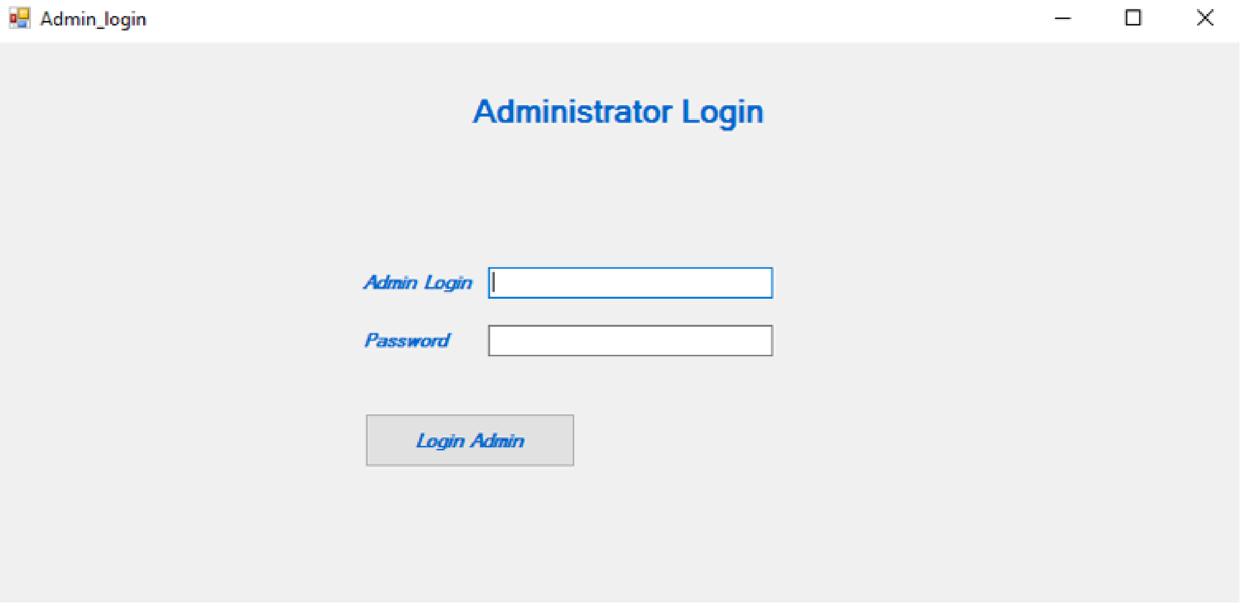
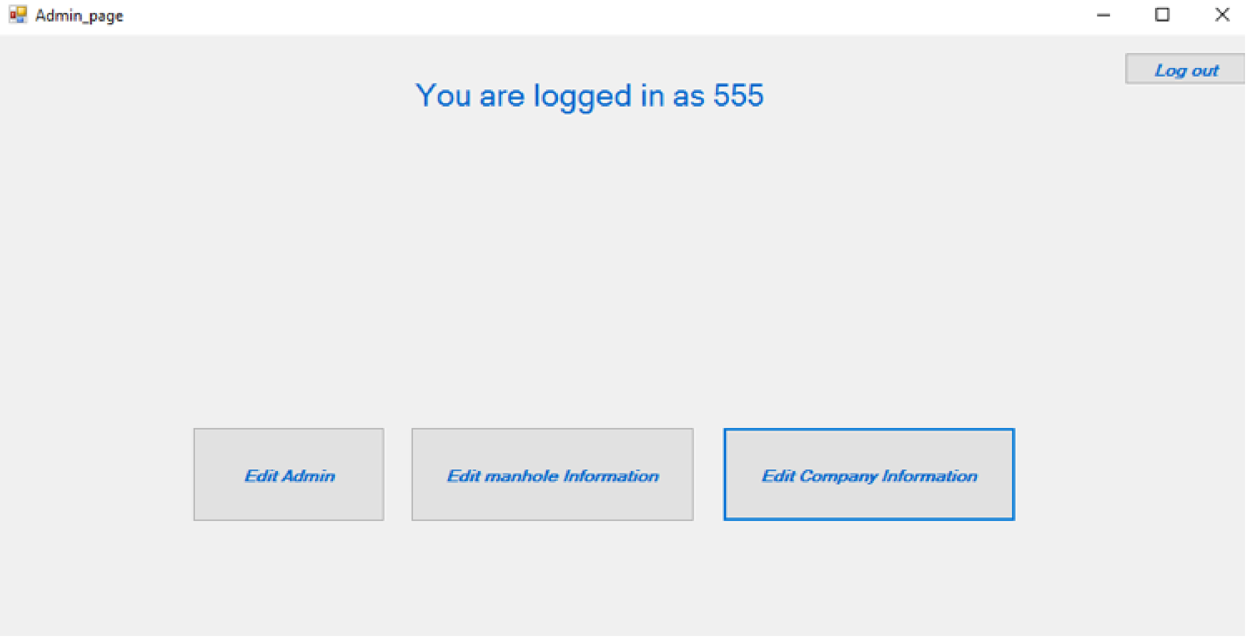


Figure 2.6-1: Administrator Login

## 2.7 Administrator’s page

If the login is successful, the administrator is now allowed to perform the following tasks: Edit Admin, Edit manhole information, Edit Company Information by clicking on the various buttons. The administrator can during any point exit the admin window by clicking **Log out** button in top right corner.

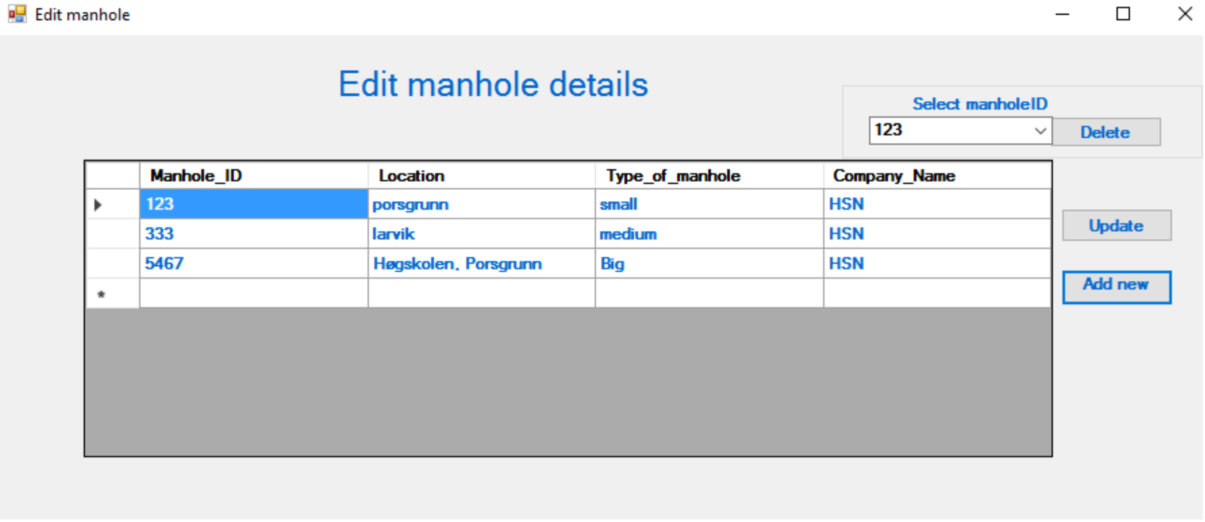


You are logged in as 555

Figure 2.7-1: Administrator’s page

## 2.8 Edit manhole

Click on the **Edit manhole Information** button from Administrator page



Edit manhole details

Select manholeID: 123

| | Manhole_ID | Location | Type_of_manhole | Company_Name |
|---|---|---|---|---|
| ▸ | 123 | porsgrunn | small | HSN |
| | 333 | larvik | medium | HSN |
| | 5467 | Høgskolen, Porsgrunn | Big | HSN |
| * | | | | |

Figure 2.8-1: Edit manhole

### 2.8.1 Update manhole details

Move the cursor to the field that needs to be edited, type in the new information, then click on the **Update** button to save the change.

### 2.8.2 Delete manhol detail

Select the manhole ID from drop-down field, then click on the **Delete** button.

### 2.8.3 Add new manhole

Click on the **Add new** button in the **Edit manhole details** window.

Type in manhole ID (which must be a number), Location, Type of manhole and Company name in the empty fields.

Click on the **Add** button to add the manhole to the database. Click on **Exit** button to navigate back to the **Edit manhole details** window.

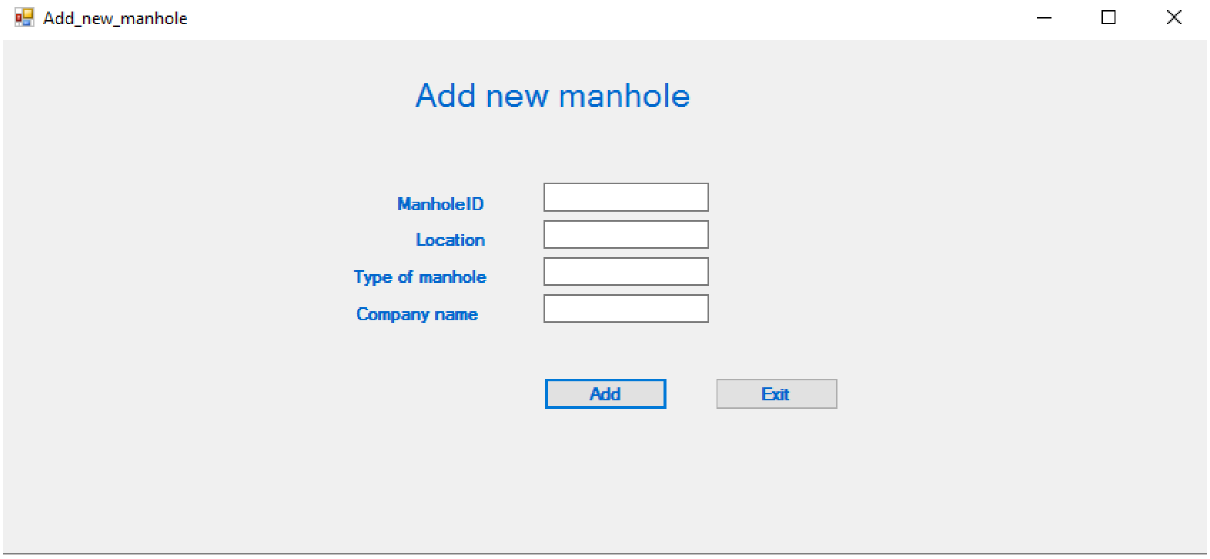
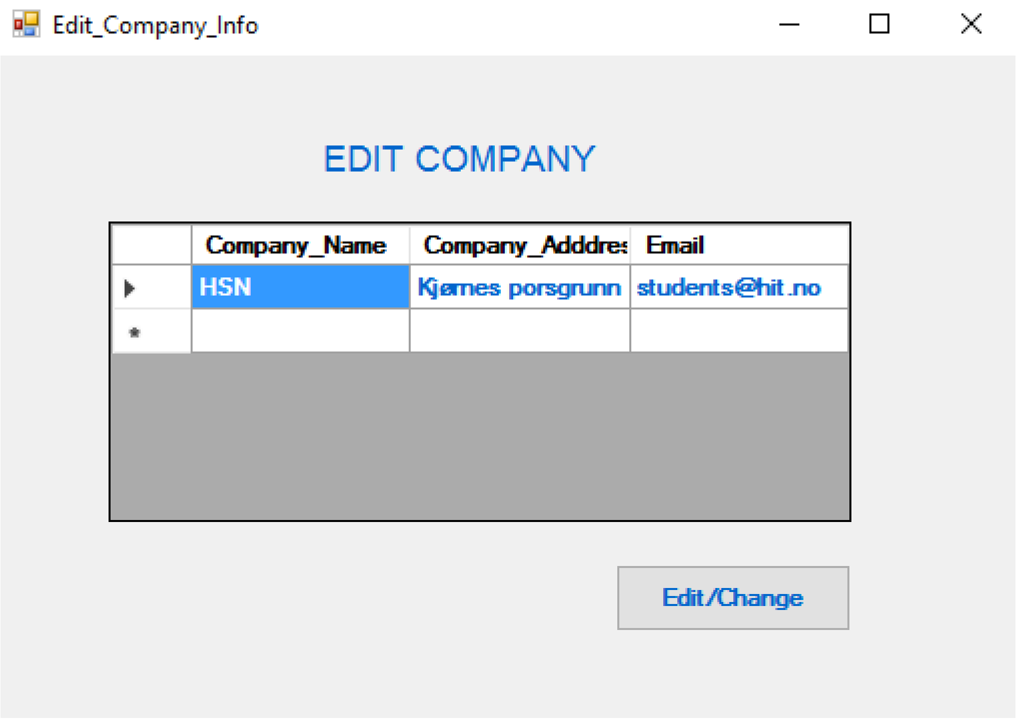


Figure 2.8-2: Add new manhole

## 2.9 Edit company

Click on the **Edit Company Information** button in the **Administrator's page.**

Type directly on the field that needs to be edited. Click on the **Edit/Change** button to save the changes.



Edit_Company_Info

EDIT COMPANY

| | Company_Name | Company_Adddres | Email |
|---|---|---|---|
| ▸ | HSN | Kjønes porsgrunn | students@hit.no |
| * | | | |

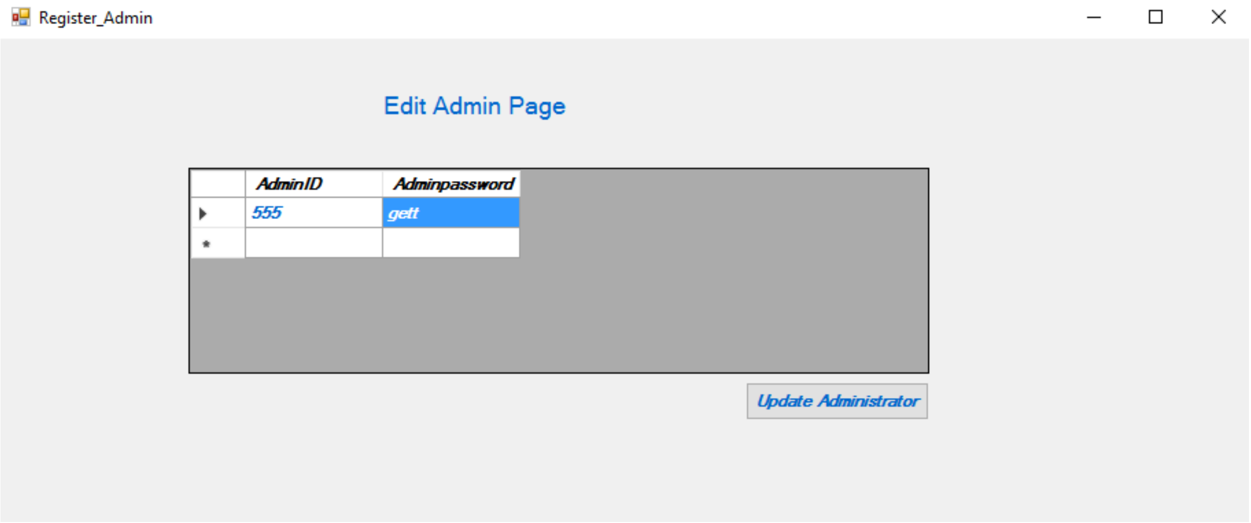
Figure 2.9-1: Edit Company

## 2.9.1 Edit/change administrator

Click on the **Edit Admin** button from Administrator page.

Move the cursor to the field that need to be edited, type in the new information.

Click on the **Update Administrator** button.



Register_Admin

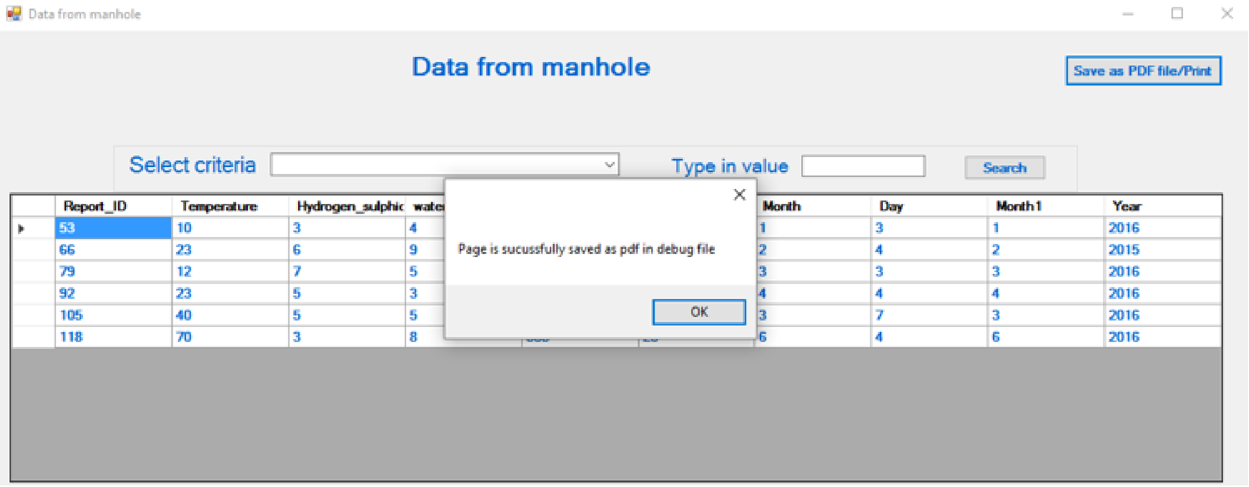
Edit Admin Page

| | AdminID | Adminpassword |
|---|---|---|
| ▸ | 555 | gett |
| * | | |

Figure 2.9-2: Edit Admin Page

## 2.10 Open page as PDF file

Click on **Save as PDF file/print** button. Locate **System program** on the computer, double click on the program to open the manhole prosject. Then click on the **Open Test File.**

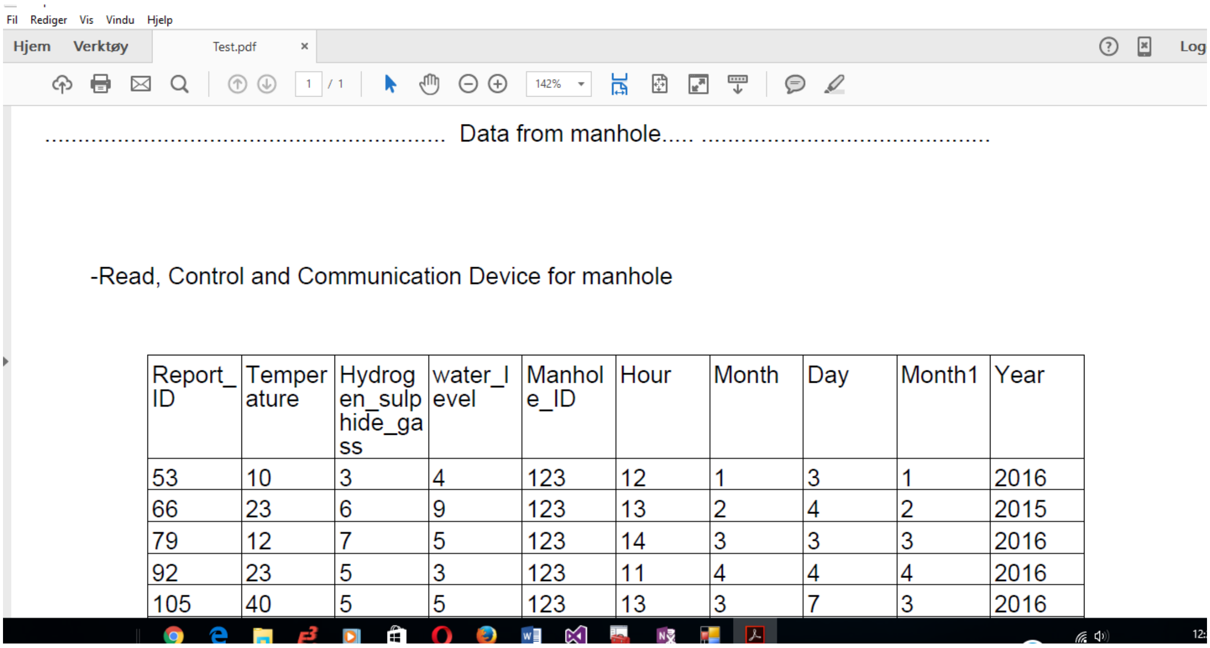


Page is sucussfully saved as pdf in debug file

| Report_ID | Temperature | Hydrogen_sulphic | wate | | | Month | Day | Month1 | Year |
|---|---|---|---|---|---|---|---|---|---|
| 53 | 10 | 3 | 4 | | | 1 | 3 | 1 | 2016 |
| 66 | 23 | 6 | 9 | | | 2 | 4 | 2 | 2015 |
| 79 | 12 | 7 | 5 | | | 3 | 3 | 3 | 2016 |
| 92 | 23 | 5 | 3 | | | 4 | 4 | 4 | 2016 |
| 105 | 40 | 5 | 5 | | | 3 | 7 | 3 | 2016 |
| 118 | 70 | 3 | 8 | | | 6 | 4 | 6 | 2016 |

Figure 2.10-1: Save as PDF file

An example of a PDF file is shown in Figure 2.10-2



............................................................. Data from manhole..... ............................................

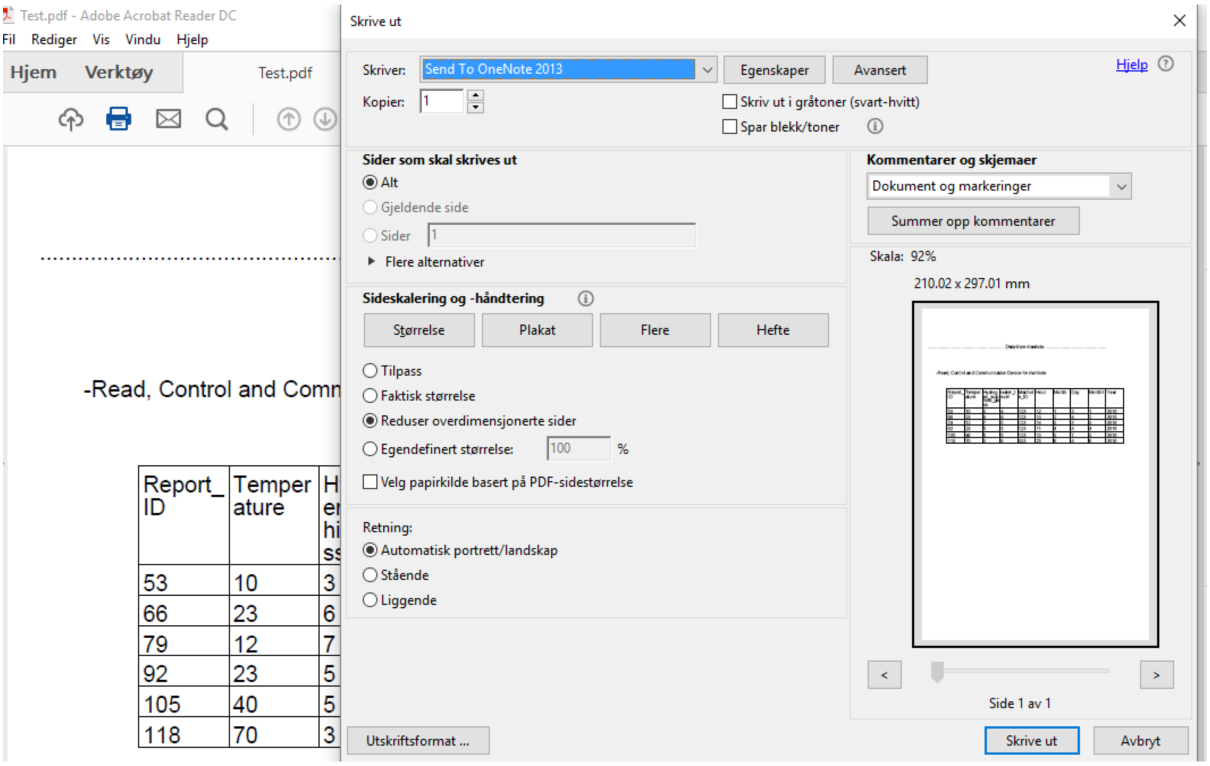
-Read, Control and Communication Device for manhole

| Report_ID | Temperature | Hydrogen_sulphide_gass | water_level | Manhole_ID | Hour | Month | Day | Month1 | Year |
|---|---|---|---|---|---|---|---|---|---|
| 53 | 10 | 3 | 4 | 123 | 12 | 1 | 3 | 1 | 2016 |
| 66 | 23 | 6 | 9 | 123 | 13 | 2 | 4 | 2 | 2015 |
| 79 | 12 | 7 | 5 | 123 | 14 | 3 | 3 | 3 | 2016 |
| 92 | 23 | 5 | 3 | 123 | 11 | 4 | 4 | 4 | 2016 |
| 105 | 40 | 5 | 5 | 123 | 13 | 3 | 7 | 3 | 2016 |

Figure 2.10-2: Example PDF file

### 2.10.1 Print page

Click on the printer icon on the PDF file. A new window will open, click **Print/Skriv ut.**



-Read, Control and Comm

| Report_ID | Temperature | H<br>er<br>hi<br>ss |
|---|---|---|
| 53 | 10 | 3 |
| 66 | 23 | 6 |
| 79 | 12 | 7 |
| 92 | 23 | 5 |
| 105 | 40 | 5 |
| 118 | 70 | 3 |

Skriver: Send To OneNote 2013

Kopier: 1

□ Skriv ut i gråtoner (svart-hvitt)

□ Spar blekk/toner

**Sider som skal skrives ut**

● Alt

○ Gjeldende side

○ Sider 1

▸ Flere alternativer

**Sideskalering og -håndtering**

○ Tilpass

○ Faktisk størrelse

● Reduser overdimensjonerte sider

○ Egendefinert størrelse: 100 %

□ Velg papirkilde basert på PDF-sidestørrelse

Retning:

● Automatisk portrett/landskap

○ Stående

○ Liggende

**Kommentarer og skjemaer**

Dokument og markeringer

Skala: 92%

210.02 x 297.01 mm

Side 1 av 1

Figure 2.10-3: Print PDF file

**Attachment W**

# Installation guide for software read, control and communication unit for manhole

## 1 System requirements

- ✓ Windows 10 (32-bit or 64-bit)
- ✓ Intel Pentium 233MHz processor or faster
- ✓ 6.58 MB of free space on your hard drive
- ✓ DVD-R drive or 6.59MB removable USB drive

## 2 Create new database

- ✓ Download and Open **SQL Server Management studio with tools**
- ✓ Right click on **Database**
- ✓ Click **new Database**
- ✓ Register new Database
- ✓ In the “Sql server management studio”, Click on **New query**
- ✓ Copy **Table Script** and paste on the page
- ✓ Select the newly created database from **available database** dropdown list.
- ✓ Click **Execute.**

  The necessary database tables and columns needed for the application is now available in the new database

### Configure “ReportID” column to generate numbers automatically

- ✓ Expand the **Database,** and **Tables.**
- ✓ Right click on table **Situation Report**
- ✓ Click **Design**
- ✓ Click row **ReportID**
- ✓ Under **Column properties**, expand **Identity Specification** and change **Is Identity** to **yes**
- ✓ Set **Identity Increment** to desired tall

### Set in initial Admin ID and Password

- ✓ Expand the **Database, Tables**
- ✓ Right click on **Administrator Table**
- ✓ Click **Edit Top 200 Rows**
- ✓ Register **Admin ID** and **Adminpassword**

# 3 Download software application from usb/cd

- ✓ Connect USB / CD to the computer
- ✓ Click windows – file settings – locate and open the USB or CD – Device for manhole
- ✓ Click **Download** then **Run.**
- ✓ Follow the steps in the setup dialogs. You will have the option to specify where to install the manhole device software on your computer.
- ✓ You must be an administrator on the computer on which you are installing the software application. It requires the Microsoft .NET Framework version 2.0 or higher.

# 4 Run the program

- ✓ Right Click on the program – open
- ✓ Open Database file
- ✓ Run “manhole Cover” Application