Proceeding ICMETA: Volume 1/2017, June 27th 2017 ISBN 978-602-397-058-2

Technical Implementation of Gas measurement system

Haryono
Balai Pengamatan Atmosfer dan Antariksa - LAPAN
Pasuruan, East Java, Indonesia

E-mail: haryono@lapan.go.id

ABSTRACT. To design gas measurement system needs knowledge about the electronics and the information technology broadly. Sometimes, it takes much more time or required iteration, sometimes, need to redesign because difficult to be implemented. This such problem will be consuming the time in some research. To help another researcher to develop measurement system project not only for the gas measurement: therefore this technical implementation is written. The aim is to add another option and overview in the design system and how to implement such technology. This design and implementation have been already proven successful on this project. The paper talks in systematically order and detail specific implementation, from the very beginning (how to acquire the data from the sensor) up to the end (how to display the data to the browser). This paper also talks about the technology being used. Gas Measurement System in this project is the system that can acquire many gas sensors and then send the data to the server. All the sensor can be placed in such location even far away from the database server, yet the data can be sent remotely. This paper can give a contribution to researcher or engineer insight of view how to design and implement successfully in the measurement system project..

Keywords: Gas Sensor, Arduino, Android, Bluetooth, Web server.

1. Introduction

Design a project or a system in the Gas Measurement System needs some knowledge. It is required to engineer or researcher to think about the design carefully so that the implementation will be possible to be implemented. If not, sometimes the design is facing bottleneck and could not be implemented. Hence the design needs to redesign again.

Other research that less or more related to this research such as like in [1] using air-collection system (dubbed HYDRA) to collects air samples from 18 different locations. The other research also has been conducted to gather the gas data in a portable emission measurement system for emissions of passenger rail locomotives [2].

Unlike above papers, in this paper will talk technically detail how to implement in collecting gas data. In this paper will give an overview how to design and implement the Gas Measurement System. It can give an overview of the technology being implemented. Not only for the Gas Measurement System but also another Measurement System that required to acquire the data remotely. The Gas Measurement System which has been developed can acquire the data remotely and automatically without a human being interruption. All data from sensors will be gathered and saved into the database server.

Proceeding ICMETA: Volume 1/2017, June 27th 2017 ISBN 978-602-397-058-2

This design has been implemented successfully. Some information why such technology is selected will be talked in this paper. The specific technical implementation that really much important has also been talked in this paper. Most will be talked about software programming related.

2. Method

The method to achieve the objective is:

1. Finding technology that can be possible to implement for this project
2. Observing by testing and implement the technology
3. Choose the best technology and fit with this Gas Measurement Project
4. Doing analyzation about the selected technology

3. System design

System design of the measurement system is shown in Figure 1. It showed the technology being used in each block. Programming Microcontroller can be done by many tools. For example using CodeVision AVR, ATMEL Studio, Arduino Studio or Visual Studio. The writer has already gone trough each technology, to see which technology to be suitable for this project and possible or easy to be implemented. After going a while the writer selects Visual Studio 2015 with **Visual Micro**. Visual Micro uses the configuration of the Arduino IDE means Everything that works on Arduino IDE will also work in Visual Studio [3].

Android Handphone can be programmed using Android Studio, Eclipse, and Visual Studio. The writer has also used the tree above but seems Visual Studio is really much easy to implement. Visual Studio work with Xamarin has been successfully implemented to handle the Bluetooth, Sending the Data to Server, and getting the Latitude/Longitude Location.

The web server can Linux or windows based. To take the advantages of the C# language, the ASP MVC was chosen. ASP MV is the framework from Microsoft that also been bundled with the WEB API. The WEB API can receive data from another device via URL. The device can select the Post or Get method to send the Data to the Server.

Another important aspect is the Database server. Many databases are available from free to paid. MySQL was selected because is relatively fast, no limitation data and free.

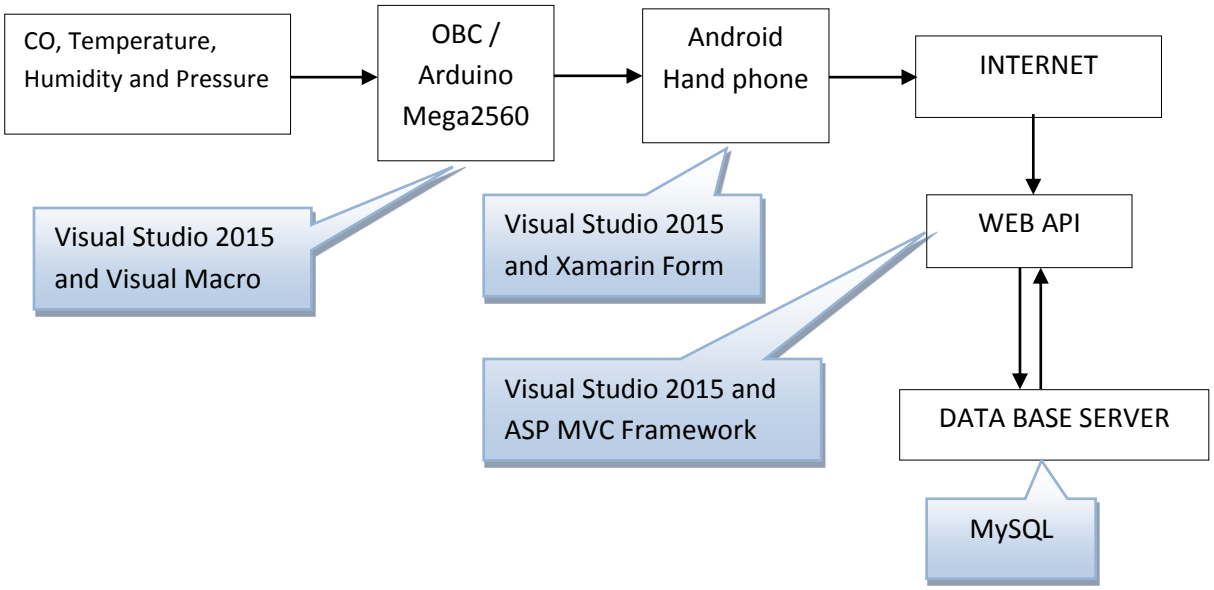


Figure 1. System Design

Proceeding ICMETA: Volume 1/2017, June 27th 2017 ISBN 978-602-397-058-2

4. Technical implementation

4.1. Acquiring the co, temperature, humidity and pressure sensor

4.11. Carbon Monoxide (CO) Sensor

The CO sensor that is used is Electrochemical CO Module ZE07-CO. The ZE07-CO is selected because in the Electrochemical CO Module ZE07-CO is built temperature sensor that can do temperature compensation [4]. Another reason is very easy to get the data CO in ppm. To get CO data below is the formula which is available in the datasheet.

Table 1. Data output of ZE07-CO.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Start Byte	Gas Type	Unit	No. of decimal	Concentration (High Byte)	Concentration (Low Byte)
0xFF	CO=0x04	ppm=0x03	1= 0x01	0x00	0x25

Gas concentration value is writen in (1).

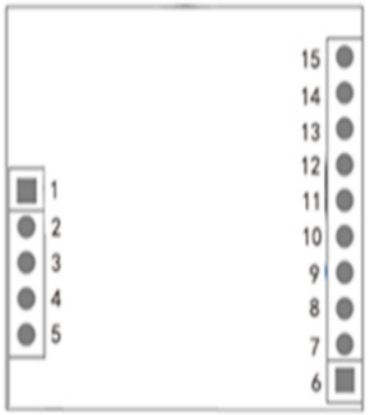
CO ppm = (High Byte*256+Low Byte) x 0.1 [4] **(1)**

High Byte and Low Byte is in Hexadecimal. So need to convert to decimal first. If High Byte **0A** and Low Byte **BB** then the formula is shown in (2).

PPM = 10 * 256 + 187 **(2)**

The configuration of hardware relatively simple, figure 2 is the connectivity of the component. Only 4 cables that are required to get the sensor is working. They are 2 power negative and positive, and 2 for Tx (Transmit) and Rx (Recieve).

PIN15	Vin（Voltage input 5V-12V）
PIN5、PIN14	GND
PIN1	VOUT(Voltage output 3.0V)
PIN3	Reserved
PIN4	Reserved
PIN7	UART (RXD) 0~3.0V Data input
PIN8	UART(TXD) 0~3.0V Data output
PIN9	Sensor analog signal
PIN10	DAC 0.4V-2V (0 - full range)
PIN2/ PIN6/ PIN11/ PIN12/ PIN13	NC



Proceeding ICMETA: Volume 1/2017, June 27th 2017 ISBN 978-602-397-058-2

Figure 2. Pin Description of ZE07-CO

Below is how to get the CO sensor in **C Code**. The result of the data is already in the PPM.

```
int highByteCO = dataCO[4];
int lowByteCO = dataCO[5];
float coPPM = (highByteCO*256+lowByteCO)* 0.1;
```

4.1.2. Temperature and Pressure Sensor

The sensor which is used for the Temperature and Pressure Sensor is BMP180. BMP180 is relatively simple. From the temperature and pressure sensor is relatively stable. This sensor is made by Bosch Sensortec. Weather forecast application can be achieved by this sensor [5]. Bosch BMP180 pressure sensor library for the Arduino microcontroller is already available in [6]. As said in the datasheet this sensor is fully calibrated [7].

The configuration of hardware is shown in figure 3 is the connectivity of the component. Only 4 cables that are required to get the sensor is working. They are 2 power negative and positive, and 2 for SCL and SDA.

4.1.3. Humidity Sensor

The sensor which is used for the Humidity is DHT11. The library has also available to get the humidity data, it is available in [8]. This sensor also can get the temperature but the result is not good to compare to the BMP180, the result is fluctuating. In the next project will try to figure out what is the actual problem for this sensor. The output of the DHT11 is analog, so that need to be connected to the analog pin of the microcontroller. The code below shown, A10 is defined as analog input from the DHT11.

```
#define DHT11_PIN A0
```

In Figure 4 showed the connection between the microcontroller and the DHT11. Only 3 pins that can be connected. Two pins are power and one pin is for the analog data to be connected to the Microcontroller.

Figure 3 BMP180 Module Pin

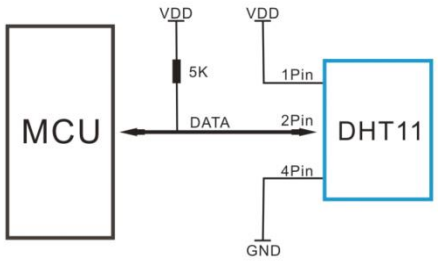


Figure 4 Typical Application DHT11

4.2. Programming in on board computer / arduino.

The Arduino Mega2560 is used. It has many UART port so that can fulfill the requirement and many input or output pins are available so that in the future there is no problem when many sensors are added. To program the Arduino the Visual Studio Community 2015 plus **Visual Micro** is selected. This tool is free for non-commercial used. The most interesting to use this tool for the writer is very easy to debug because there is serial port integrated into the IDE, can be docked in anywhere inside the IDE. In the **visual micro pro,** it can be debugged in code line by putting the breakpoint. Another think is because the writer is having experience in using Visual Studio. Most of the features in Visual

Proceeding ICMETA: Volume 1/2017, June 27th 2017 ISBN 978-602-397-058-2

Studio can be used, such as refactoring, highlighting, intelisence, explorer etc. All data from the sensor is handled by the string variable. Below how to package all data to be sent to Android Device:

```
double temperature;
double pressure = GetPresureAndTemperature(temperature);
String dataSendTo = "#";
dataSendTo += "|";
dataSendTo += GetCoInPPM();
dataSendTo += "|";
dataSendTo += GetHumidity();
dataSendTo += "|";
dataSendTo += temperature;
dataSendTo += "|";
dataSendTo += pressure;
dataSendTo += "|";
dataSendTo += "\n";
Serial3.print(dataSendTo);
```

All data is put in the **dataSendTo** variable. To make the code is clean and easy to be read, each code to read the sensor are separated in each method. The result of the reading of sensor is combined together in one string by separated "|" delimiter. So that when reaching in Android Device just separated by that delimiter.

4.3. Bluetooth module to connect android device

To send the data from a microcontroller to Android Handphone requires **Bluetooth module**. HC-06 Bluetooth is used. Figure 5 is the HC-06 module with its pins. In order to send the data from the microcontroller to the Bluetooth, Tx and Rx pin are connected to the UART port of the microcontroller. In microcontroller just programmed like usual programming the UART when sending the data to UART. There is no different.

4.4. Programming android handphone

To program android, visual studio, Xamarin and Prism are used. The advantages of using those tools are

1. Can create the form using XAML,
2. Able to binding the data easily,
3. Enable MVVM framework,
4. Enable to use the power of C#,
5. Easy navigation,
6. Multiplatform.

Those advantages have been utilized by the writer on this project successfully.

XAML, Binding, MVVM, Navigation, C# features

Model–view–view-model (MVVM) is a software architectural pattern. It is Model View and View Model. The code is separated between them so that easily maintained and testable. By using MVVM the View, View Model, and Model are residing in different place. Because of this, the code is easy to maintain and reusable. Besides, it can be easier to create unit testing. From the experiment, the writer found that Xamarin and visual studio work together with the Prims can have very a good environment to program the Android Handphone. Prism 6 is a fully open source version of the Prism guidance originally produced by Microsoft patterns & practices [9].

Proceeding ICMETA: Volume 1/2017, June 27th 2017 ISBN 978-602-397-058-2

Figure 6 is showed the connection to the LG20 handphone. Figure 7. The structure file environment using MVVM pattern using **Prism**.

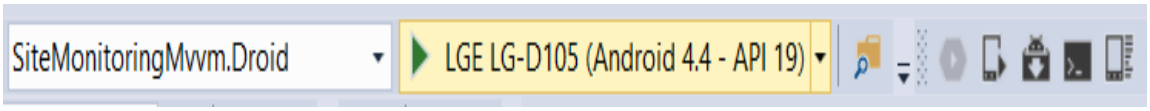


Figure 6. The LG 20 handphone is detected in Visual studio 2015

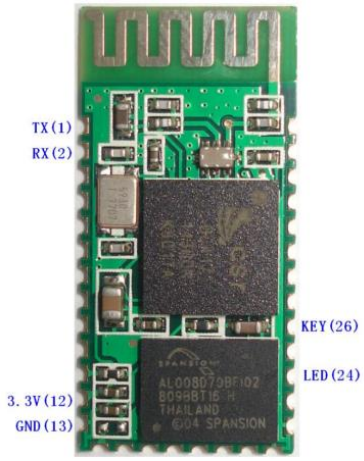


Figure 5 the HC-06 module

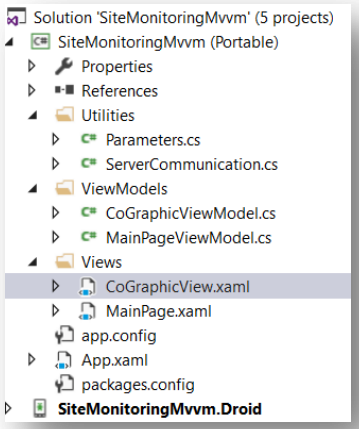


Figure 7. The structure file environment

The most interesting part to use this tool is because can implement the Binding data easily. XAML is Extensible Application Markup Language. It is used for creating the View of the application. Binding is a way to synchronize the data from View Model and show to the View. Below is XAML code snippet that was used:

```
<Label Text="Data Row:"/>
<Label Text="{Binding DataFromBluetooth}" />
<Label Text="CO:"/>
<Label Text="{Binding CO}" />
<Label Text="Humidity:"/>
<Label Text="{Binding Humidity}" />
<Label Text="Temperature:"/>
<Label Text="{Binding Temperature}" />
<Label Text="Pressure:"/>
<Label Text="{Binding Pressure}" />
```

For example {Binding **Presure**} is acting to bind the Pressure property from the View Model. Each time the Pressure property data is changed the View will automatically change. In order to be able changed automatically need to implement the **InotifyPropertyChanged**. So that in the View Model code will be:

```
public string Pressure
{
  get { return _presure; }
  set { SetProperty(ref _presure, value); }
}
```

Proceeding ICMETA: Volume 1/2017, June 27th 2017 ISBN 978-602-397-058-2

SetProperty is the method that handles in updating the View behind the scene.To navigate between form is very simple. Using Prism the navigation is easier. There is InavigationAware from Prism. This interface will handle the navigation: Navigated From and Navigated To. Sending data from old View to new View is also done using InavigationService.
To make the processing task is running well, because android handphone is not like a desktop that has many resources. So that need to program some task in the background. Below is the code to listen to the data which is coming from Bluetooth:

```
Task.Run(() => { Listener(); });
public void Listener()
{
    byte[] read = new byte[1];
    while (true)
    {
      con.thisSocket.InputStream.Read(read, 0, 1);
    }
}
```

To get the location of the android handphone, **Geolocator** library is used [10]. The think that needs to consider is the background process. This process to get the data should be run in the background process. The writer is able to get the location without hang by using "Xamarin.Forms.Device. StartTimer". Below is the code that how to achieve:

```
Xamarin.Forms.Device.StartTimer(
TimeSpan.FromSeconds(10), () => {
     Task.Run(() => { GetLocation(); });
});
```

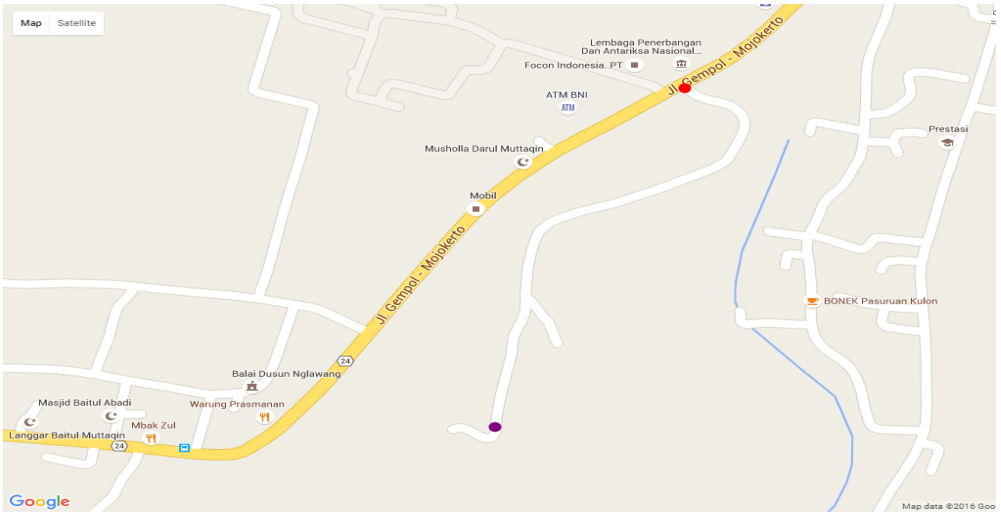
4.5. Programming web server and web api

To program the Web Server and Web API, ASP MVC is implemented. ASP MVC is a framework to program the Web Server more easily. By using ASP MVC server task can be programmed using C# language, this can make programming easier. MVC is Model View and Controller. By separating the code can make a code easy to be maintained, testable and clean. Web API is standby to receive the data from android. **ApiController** class is used to receive the data. When receiving data from android the server will also classify the location based on district automatically.
In the android will access the url, for example **..\api\time?12/12/2016**

```
NameValueCollection valueCollection = HttpUtility.ParseQueryString(Request.RequestUri.Query);
string timeStr = nameValueCollection.Get(0);
```

By using above code the timeStr will be **12/12/2016.** After all, data have been extracted, they can be saved to the database. The accessing database is using EntityFramework, unlike SQL language, the EntityFramework makes the code clean and avoids error in the typing SQL script.
Showing the data to the View is done by showing the list of the row data and to the google map. Showing the data into google map can be done using Google Apis. In the Figure 8 showed the system that has been made, it can show the data CO on the Map based on location and co concentration.

Proceeding ICMETA: Volume 1/2017, June 27th 2017 ISBN 978-602-397-058-2



CO Level	Color
< 0	
1 – 9	
10 – 24	
25	
26 – 50	
51 - 12800	

Figure 8 CO level in Map

5. Analysis result

The data that have been gotten from sensor can be analyzed below:

Sensor CO: the data from sensor CO is correct and can increase smoothly when Carbon Monoxide is generated. In the normal room, it is got 0.05 PPM this data is correct according to the [11].

BMP180: The data from this sensor is stable. The pressure data and the temperature have been compared to the other calibrated sensor. It got the same result.

DHT11: this sensor actually has temperature and humidity sensor but not stable, the fluctuation occurs frequently.

The different between programming in the desktop compare to the android programming is about the background task. From experience the process is not working well when some task needs long processing, for example sending the data to the server and connecting Bluetooth. Those such processes need to be done in the background, otherwise, the android will frequently hang. Programming android using C# language is now possible and can be done successfully. During writing the code there is not so much problem in using **Xamarin.Form**. The function to handle the **Bluetooth**, **Getting Location**, **Sending to the server** are done in the Xamarin and Visual Studio environment smoothly.

Analyzing the ASP MVC for this project that is gotten: The data from android can be received safely using the **Get** method. The advantage of using android, it can be tested in the local area network, no need to deploy in the online internet server. Unlike GPRS module, it is needed to deploy in the internet server as they don't have local area network.

Proceeding ICMETA: Volume 1/2017, June 27th 2017 ISBN 978-602-397-058-2

6. Conclusion and future work

The technical implementation of this project has been talked in systematical order and the important aspect about how to get implementation successfully has also been published in this paper. The gas measurement system has been implemented successfully, all data from gas sensor have been saved in the database server and can be accessed from the browser. Future works would be selecting the best gas sensor and validated by the proven gas laboratory institution.

Acknowledgments

This project is supported by Balai Pengamatan Atmosfer dan Antariksa - LAPAN, Mr. Dian Yudha Risdianto, ST., M.T as Head of Balai Pengamatan Atmosfer dan Antariksa LAPAN, and my colleges Mr. Noi, Mr. Imron and Mr. Joyo. We would like to acknowledge for their support in this project.

References

[1] S. P. Burns *et al.*, "An Evaluation of Calibration Techniques for In Situ Carbon Dioxide Measurements Using a Programmable Portable Trace-Gas Measuring System," *http://dx.doi.org/10.1175/2008JTECHA1080.1*, 01-Feb-2009. [Online]. Available: http://journals.ametsoc.org/doi/abs/10.1175/2008JTECHA1080.1. [Accessed: 03-Nov-2016].

[2] H. Frey, H.-W. Choi, and K. Kim, "Portable Emission Measurement System for Emissions of Passenger Rail Locomotives," *Transportation Research Record: Journal of the Transportation Research Board*, Nov. 2012.

[3] "Visual Studio or Atmel Studio? - Arduino IDE for Visual Studio and Atmel Studio." [Online]. Available: http://www.visualmicro.com/page/User-Guide.aspx?doc=Getting-started-which-IDE.html. [Accessed: 11-Oct-2016].

[4] "Electrochemical CO Module, carbon monoxide sensor module, CO gas sensor-Winsen Electronics." [Online]. Available: http://www.winsen-sensor.com/products/co-module/ze07-co.html. [Accessed: 19-Oct-2016].

[5] "BMP180." [Online]. Available: https://www.bosch-sensortec.com/bst/products/all_products/bmp180. [Accessed: 19-Oct-2016].

[6] "LowPowerLab/SFE_BMP180," *GitHub*. [Online]. Available: https://github.com/LowPowerLab/SFE_BMP180. [Accessed: 25-Oct-2016].

[7] B. Sensortec, "Data sheet BMP180 Digital pressure sensor." [Online]. Available: https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf. [Accessed: 25-Oct-2016].

[8] pert, "Class for DHTxx sensors." [Online]. Available: http://www.arduino.cc/playground/Main/DHTLib. [Accessed: 25-Oct-2016].

[9] "PrismLibrary/Prism," *GitHub*. [Online]. Available: https://github.com/PrismLibrary/Prism. [Accessed: 11-Oct-2016].

[10] "jamesmontemagno/GeolocatorPlugin," *GitHub*. [Online]. Available: https://github.com/jamesmontemagno/GeolocatorPlugin. [Accessed: 27-Oct-2016].

[11] "Indoor Air Quality (IAQ) | US EPA." [Online]. Available: https://www.epa.gov/indoor-air-quality-iaq. [Accessed: 27-Oct-2016].