

Design and Implementation of a Heterogeneous Sensor-based Embedded System for Flood Management

Thesis submitted in partial fulfilment of the requirements for the degree of
B. Tech. in Electronics and Communication Engineering

By

Subhajit Sahu (110EC0181)

Electronics and Communication Engineering (2010-14)

Under the guidance of

Prof. Debiprasad Priyabrata Acharya

Department of Electronics and Communication Engineering



National Institute of Technology, Rourkela

Rourkela - 769008, Odisha.



National Institute of Technology, Rourkela

Rourkela - 769008, Odisha.

Declaration

I hereby declare that the Project entitled “*Design and Implementation of a Heterogeneous Sensor-based Embedded System for Flood Management*” submitted in partial fulfillment of the requirements for the degree of Bachelor of Technology in Electronics and Communication Engineering at National Institute of Technology, Rourkela for the academic session 2013 - 2014 is a record of my original work done under Prof. D. P. Acharya, National Institute of Technology, Rourkela. Wherever contributions of others are involved, every endeavor has been made to acknowledge the same with due reference to literature.

Subhajit Sahu

(110EC0181)



National Institute of Technology, Rourkela

Rourkela - 769008, Odisha.

Certificate

This is to certify that the Project entitled “*Design and Implementation of a Heterogeneous Sensor-based Embedded System for Flood Management*” submitted by *Subhajit Sahu* in partial fulfillment of the requirements for the award of *Bachelor of Technology Degree* in *Electronics and Communication Engineering* (session 2010-2014) at *National Institute of Technology, Rourkela* is a credible and authentic work carried out by her under my supervision and guidance.

(Prof. D. P. Acharya)

Place: NIT Rourkela

Electronics and Communication Engineering

Date:

National Institute of Technology, Rourkela

Acknowledgement

I feel privileged to express my heartiest thanks and gratitude to Prof. D. P. Acharya, Professor, Department of Electronics and Communication Engineering for his inspiring and precise guidance, generous and valuable suggestions throughout this project. He contributed to a great extent in surmounting all hardships, and provided with requisite financial help through the institute. His tolerance in dealing with problems and finding quick solutions to them was a constant source of encouragement.

I am grateful to Prof S. K. Meher, Head of the Department, Electronics and Communication Engineering, for providing the necessary facilities in the department.

I am thankful to my parent, my brother, and my classmates, for providing essential information, keeping me up to date with current trends in engineering and technology, helping me reserve sufficient time for project work and acting as a constant source of inspiration and support.

Last but not the least; I would like to thank to all the staff of NIT, Rourkela; those who have directly or indirectly helped me to successfully complete this project.

Subhajit Sahu

(110EC0181)

Abstract

Floods are one of the most common forms of natural disasters in the world, and cause huge loss of life, and property. There is a critical requirement for development and installation of enhanced flood forecasting sites in various commonly flooding regions of the world. In this paper, we describe the design and implementation of a novel heterogeneous sensor-based embedded system for flood management.

This embedded system enables various types of electronic gages to be deployed at remote locations, wherever mobile network is available. Acquisition of hydrologic data occurs at user defined intervals of time, and is uploaded to a database, through the internet. Information acquired into the database can then be easily viewed from anywhere, used for analysis, and running flood forecasting simulation models.

The overall system architecture, module description, and results are described here.

Table of Contents

| | | |
|-----|----------------------------------|----|
| 1 | Introduction | 6 |
| 1-1 | Floods | 7 |
| 1-2 | Flood Warning..... | 8 |
| 1-3 | Floods in India | 9 |
| 1-4 | Motivation | 10 |
| 1-5 | Objective | 11 |
| 1-6 | Literature Review | 12 |
| 2 | System Design..... | 13 |
| 2-1 | Challenges | 14 |
| 2-2 | A Basic Idea | 14 |
| 2-3 | Sensor Module Design | 16 |
| 2-4 | Cloud-based Gateway Design | 20 |
| 2-5 | Local-Host Application..... | 22 |
| 3 | System Architecture | 24 |
| 3-1 | Hardware Architecture | 25 |
| 3-2 | Software Architecture | 27 |
| 4 | Results and Discussions | 36 |
| 4-1 | The Sensor Module | 37 |
| 4-2 | The Cloud-based Gateway..... | 39 |
| 4-3 | The Local-Host Application..... | 41 |
| 5 | Conclusion and Future Work | 43 |
| 5-1 | Conclusion | 44 |
| 5-2 | Future Work | 45 |

List of Figures

| | |
|--|----|
| Figure 1: Flash Flood in the south east of France | 7 |
| Figure 2: More than 5700 people presumed dead in the Uttrakhand Floods [8] | 10 |
| Figure 3: Example application of the system on the Mahanadi River | 15 |
| Figure 4: MPX5700 Pressure Sensor | 16 |
| Figure 5: Freeduino USB microcontroller development board | 17 |
| Figure 6: GSM/GPRS Module (wrongly displays SIM900, but it is actually FLY900) | 19 |
| Figure 7: AppHarbor website which provides PAAS service for .NET Apps | 21 |
| Figure 8: Overall hardware block diagram, of the sensor unit..... | 26 |
| Figure 9: Overall Data-Flow block diagram of the Embedded System | 28 |
| Figure 10: Block diagram of Data flow through the Sensor Module | 29 |
| Figure 11: Block Diagram of data-flow through the Cloud-based Gateway | 30 |
| Figure 12: Block Diagram of data-flow through Local-Host Application..... | 31 |
| Figure 13: Pressure Sensor connected to the Sensor Module..... | 37 |
| Figure 14: Microcontroller Side of the Sensor Module..... | 38 |
| Figure 15: Wireless Unit side of the Sensor Module..... | 39 |
| Figure 16: Cloud-based Gateway ASP.NET application uploaded onto the Server | 40 |
| Figure 17: Gateway displaying encoded data received from the Sensor Module | 40 |
| Figure 18: Local-Host Application fetching data from Gateway and storing it..... | 41 |
| Figure 19: Data stored in the SQLite database being looked up using SQLite Studio | 42 |
| Figure 20: Data loaded from database being displayed as a graph in Microsoft Excel | 42 |

List of Abbreviations

| | |
|---------|---|
| US | United States |
| CWC | Central Water Commission |
| NIDM | National Institute of Disaster Management (India) |
| WRIS | Water Resources Information System (India) |
| SFFWS | Susquehanna Flood Forecast and Warning System |
| GOES | Geostationary Orbiting Environmental Satellite |
| ADC | Analog to Digital Converter |
| IDE | Integrated Development Environment |
| USB | Universal Serial Bus |
| RF | Radio Frequency |
| WiFi | Wireless Fidelity |
| GSM | Global System for Mobile communications |
| GPRS | General Packet Radio Service |
| UART | Universal Asynchronous Receiver and Transmitter |
| SIM | Subscriber Identity Module |
| IP | Internet Protocol |
| TCP | Transmission Control Protocol |
| HTTP | HyperText Transfer Protocol |
| PAAS | Platform As A Service |
| ASP.NET | Active Server Pages using .NET Framework |

| | |
|------|-------------------------------|
| FIFO | First In First Out |
| URL | Uniform Resource Locator |
| SQL | Structured Query Language |
| ID | Identifier |
| TTL | Transistor - Transistor Logic |
| SMA | SubMiniature version A |
| DC | Direct Current |
| HTML | HyperText Markup Language |
| FTP | File Transfer Protocol |
| AT | ATtention |

1 Introduction

1-1 Floods

Flooding affects people in many different ways. Property is damaged, and lives of humans and animals are endangered. Rapid water runoff causes soil erosion and concomitant sediment deposition elsewhere (such as further downstream or down a coast). The spawning grounds for fish and other wildlife habitats can become polluted or completely destroyed. Some prolonged high floods can delay traffic in areas which lack elevated roadways. **Figure 1** shows the destruction caused by a Flash Flood in the south east of France that resulted several fatalities and severe damage [1].



Figure 1: Flash Flood in the south east of France



Floods can interfere with drainage and economic use of lands, such as interfering with farming. Structural damage can occur in bridge abutments, bank lines, sewer lines, and other structures within flood-ways. Waterway navigation and hydroelectric power are often impaired. Financial losses due to floods are typically millions of dollars each year, with the worst floods in recent US history having cost billions of dollars [2].

1-2 Flood Warning

Flood warning is the provision of advance warning of conditions that are likely to cause flooding to property and a potential risk to life. The main purpose of flood warning is to save life by allowing people, support and emergency services time to prepare for flooding [3].

The secondary purpose is to reduce the effects and damage of flooding (Defra, 2004). The benefits associated with flood forecasting and warning are inextricably linked with the effectiveness of the warning dissemination programs and the activities of the public and supporting agencies (both voluntary and official) in their response [3].

The total benefits can be defined as ‘the reduction in losses (tangible and intangible) resulting from the provision of a warning when compared to the situation prior to the operation of the warning system’ [3].

1-3 Floods in India

Floods are the most common form of natural disaster in India, and cause huge loss of life, and property [4]. **Table 1** shows the damage caused by floods in India. Central Water Commission (CWC), India has installed around 175 flood forecasting stations in India, with only 3 forecasting sites in the Uttarakhand state [5].

| Category | Average | Maximum (Year) |
|---|-----------|------------------|
| Area Affected (Million Hectare) | 7.55 | 17.50 (1978) |
| Crop Area Affected (Million Hectare) | 3.54 | 10.15 (1988) |
| Population Affected (crore) | 3.286 | 7.045 (1978) |
| Human Lives Lost (Nos.) | 1,589 | 11,316 (1977) |
| Cattle Lost (Nos.) | 94,839 | 6,18,248 (1979) |
| Houses Damaged (Nos.) | 12,17,918 | 35,07,542 (1978) |
| Value of damage to crops (crore) | 710.63 | 4246.62 (2000) |
| Value of damage to house (crore) | 270.59 | 1307.89 (1995) |
| Value of damage to public utilities (crore) | 820.75 | 5604.46 (2001) |
| Value of damage to crops, houses & public utilities (crore) | 1805.18 | 8864.54 (2000) |

Table 1: Flood damage in India during 1953 to 2005 [6]

Due to this insufficiency of flood forecasting stations, CWC was only able to forecast a medium-level flood at Rishikesh and Haridwar on June 18, 2013 [7]. This led to a huge loss of life [8], and expenditure of more than ₹1000 crore [9].



Figure 2: More than 5700 people presumed dead in the Uttarakhand Floods [8]

1-4 Motivation

Today, there exist quite a number of flood forecasting algorithms, developed by researchers. However, the availability of indigenous, state of the art, infrastructure required for flood forecasting is not there.



Most of the flood forecasting systems currently installed at various sites in India are imported from foreign suppliers and hence the cost per monitoring unit, and the overall system is usually high. Hence, only a small number of flood monitoring sites are currently in use in our country. Apart from that, it becomes difficult to fix any issue that comes up during the operation of the system, and would normally take a lot of time or simply get neglected. Thus, the overall effective flood prediction system becomes difficult to use, maintain, and is not capable of providing sufficient data for accurate modeling and forecasting.

There is no available system that allows people to know about the flood danger level of a location in real-time (such a system is available for trains). Work is required on the design of an automated flood forecasting system that is capable of collecting data from installed sites continuously, and making it readily available to the experts for flood forecasting.

1-5 Objective

The objective of this project is to design and implement a modern, internet-based flood monitoring system, which is simple, cost effective, easy to deploy and use.



1-6 Literature Review

Researchers and engineers in the world have taken various approaches to the design of a flood management system. The Susquehanna Flood Forecast and Warning System (SFFWS), which is one of U.S premier flood warning systems, provides advanced flood/flash flood warning for residents of the 27,500-square-mile Susquehanna River Basin. Its foundation is a network of more than 60 stream gages and 70 rain gages that read, record, and transmit critical hydrologic data, which is transmitted from the field by way of the GOES (Geostationary Orbiting Environmental Satellite) satellite network, for incorporation into hydrologic models that provide river forecasts at stream gages [10]. On the other hand, we use wireless electronic gages, with a wireless communication module capable of transmitting acquired hydrological data to our flood management dedicated server through a custom data queue service in cloud servers. This not only helps reduce the cost of implementation, but also enhances maintainability due to the low power requirement of on-field sensor modules.

In [11], the authors have developed a multi-tiered architecture for a Web-based flood forecasting system in the Shuangpai region of China. Though, this approach allows the sensors to be deployed at any desired location, the system developed in this project has low power requirements, is more maintainable, and is extremely low-cost. Our system does not follow a multi-tiered approach which should lead to lower latency, and requires lesser resources.

2 System Design



2-1 Challenges

Flood forecasting and management activities require huge volumes of hydrologic data to be readily available for incorporation into various forecasting models [12]. There, however, exist hurdles which need to be crossed before any information can be made available at the desired location, where it can be processed. These include: on-field acquisition of different types of hydrologic data from a large number of monitoring sites; transmission of collected information to a dedicated server; storage of the hydrologic data in a form which ensures that it can be properly assimilated by the flood management officers. The system developed in this project is a possible solution.

2-2 A Basic Idea

Before one goes on to the implementation a system for any purpose, it is essential to have a proper system design as it not only enhances clarity, but also allows people to cooperate on a single design. Here, we start with a basic idea first and then go on to specify the design of the entire system. **Figure 3** shows an example application.

The following points describe the basic idea of how we plan to develop the final embedded system for Flood Monitoring and Management:

- The system should consist of a number of sensor units that measure various parameters for flood monitoring, and should have a centralized storage computer that stores all the acquired data. It should be possible to easily extract data as per the desired search criteria and obtain the data, which can be used for flood forecasting. The system to be developed should support a large number of sensor units to be deployed in various flood forecasting sites.
- It should also be able to distinguish between the data captured from sensor.
- Each sensor unit should be wireless so as to allow for easier deployment and maintenance. The sensor units should be able to capture and transmit data at user-defined intervals of time.

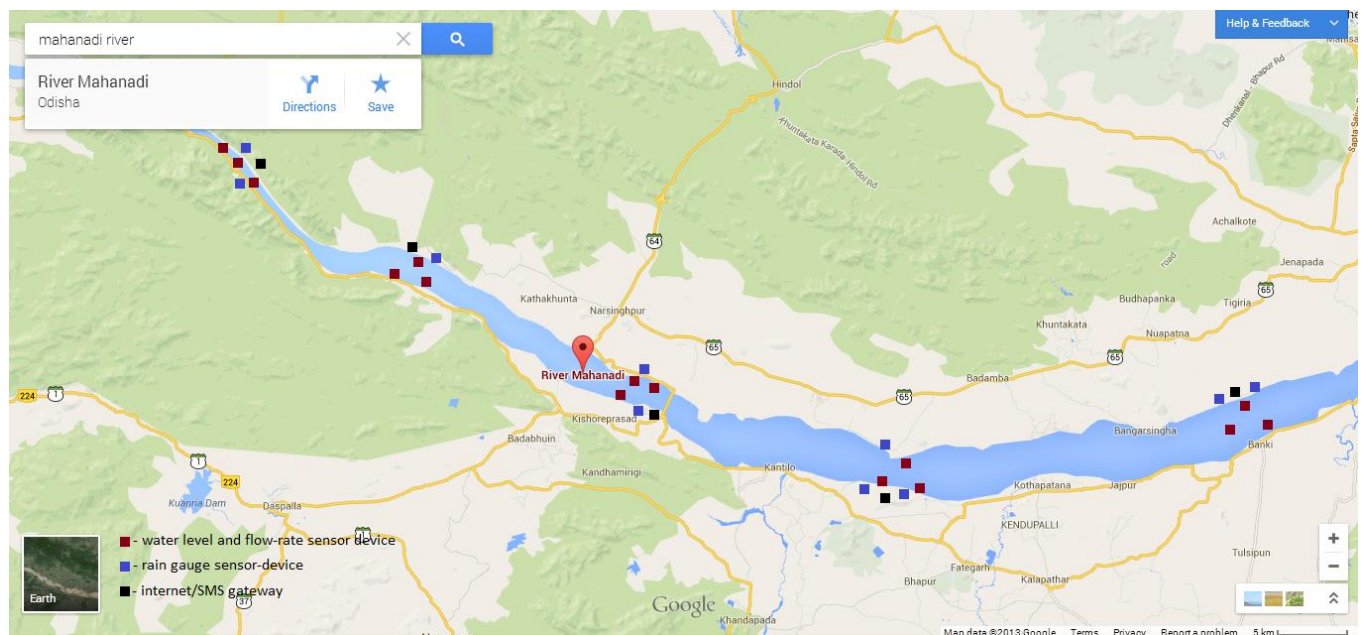


Figure 3: Example application of the system on the Mahanadi River

2-3 Sensor Module Design

For this project we consider to monitor the water level, only a particular type of parameter. However, this single parameter is enough to demonstrate that our system works properly and could possibly be used in the actual scenario. Water level can be sensed by using various techniques, and one of the simplest of them is by measurement of water pressure.

Various kinds of pressure sensors are available in the market, but the one we need should be a liquid pressure sensor with the ability to measure a maximum of around 50 - 60 meters height of water. This leads us to a maximum pressure requirement of 600 - 700 kPa. Piezo-resistive liquid pressure sensors are available in this range and are cost effective as well, hence, good enough for our purpose. Hence we choose to go with the MPX5700 by Freescale Semiconductor [13], shown in **Figure 4**.

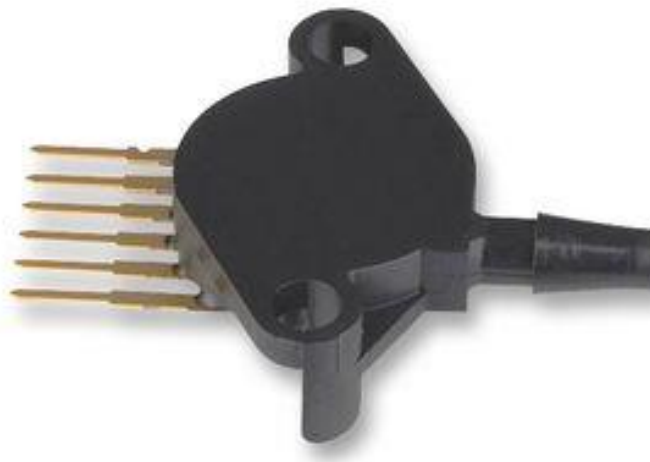


Figure 4: MPX5700 Pressure Sensor

Now since the output generated by the pressure sensor is an analog voltage, we need an ADC so as to be able to convert the analog voltage to a digital value. Microcontrollers are available in the market with internal ADCs with a sufficient bit-width (10 - 16 bits) and more than sufficient sampling speeds (15 - 2000 ksps) [14]. Arduino microcontrollers and their clones are a lot easier to use compared to standard microcontroller development boards because of the very simple and easy to use Arduino Library and IDE. Since Arduino clones are more cost effective compared to the original Arduino, we go with an Arduino clone namely Freeduino USB shown in **Figure 5**.

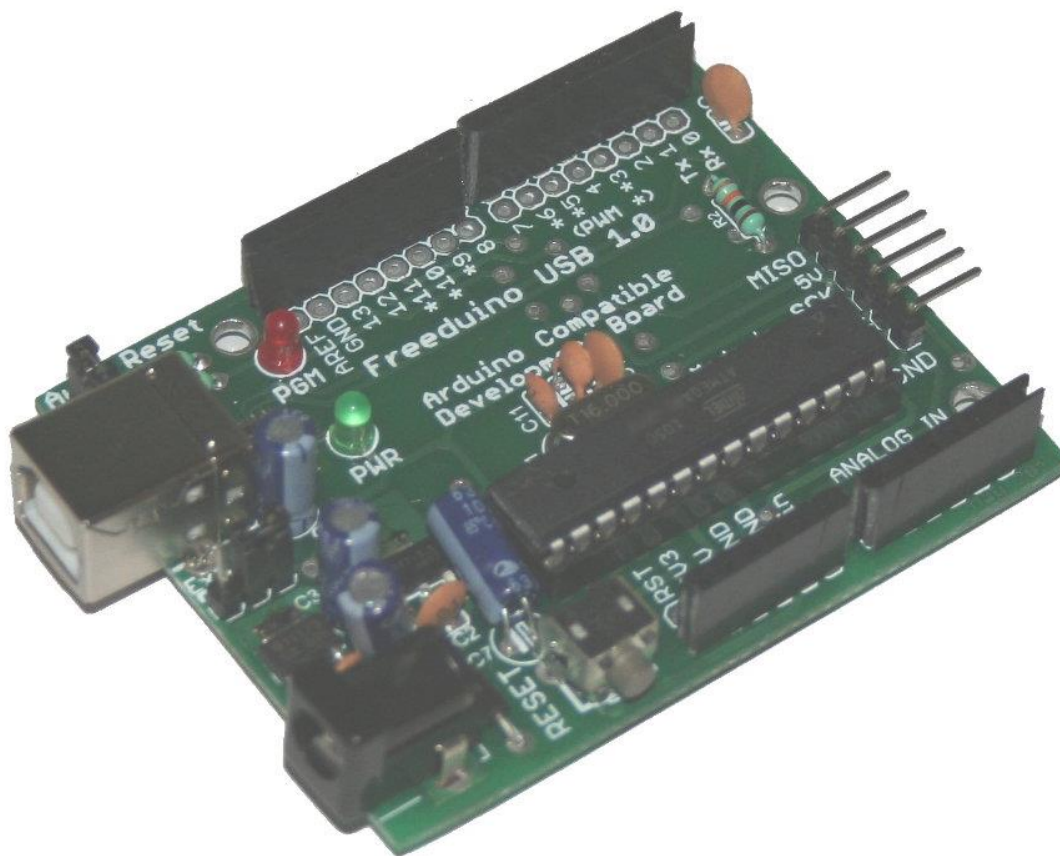


Figure 5: Freeduino USB microcontroller development board



Finally coming to the wireless transmission side, we have to choose between a number of technologies namely RF (433 MHz, 2.4GHz), WiFi, GSM/GPRS and others. Using RF would mean that we would have to implement a large RF network from the monitoring location to the receiving computer, which renders this method very difficult to implement and costly. We could use WiFi but it requires that there be nearby WiFi routers for the wireless connection which is nearly impossible in many remote locations.

Wireless communication through GSM/GPRS offers a huge benefit. It not only allows the device to be located at any desired location, but also allows the device to be mobile (moving). Though we do not need the mobility feature, since our sensor module is going to be stable, it is great to have it for future upgrades. Using GSM/GPRS requires a GSM/GPRS Modem which is indeed available in the market. Some of these modems come in the form of a board and allow direct interfacing with a microcontroller through the UART port. One of the most popular is the SIM900 based GSM/GPRS board, but since it is quite costly, we choose to go with its cost effective clone, namely FLY900 GSM/GPRS module [15] shown in **Figure 6**.

These GSM/GPRS modules require a SIM card in order to operate. Additionally, if one desires to access the internet through GPRS, it is require to have proper GPRS Proxy settings. This is because internet access through GPRS occurs through a Proxy Server on the GPRS network and each GPRS device is assigned a private IP address. These configuration settings need to be provided to the GSM/GPRS module every time a

connection is made. Note that, because of the presence of a proxy server in between it is not possible to use the TCP protocol directly, but instead we have to go with the HTTP protocol in proxy format.

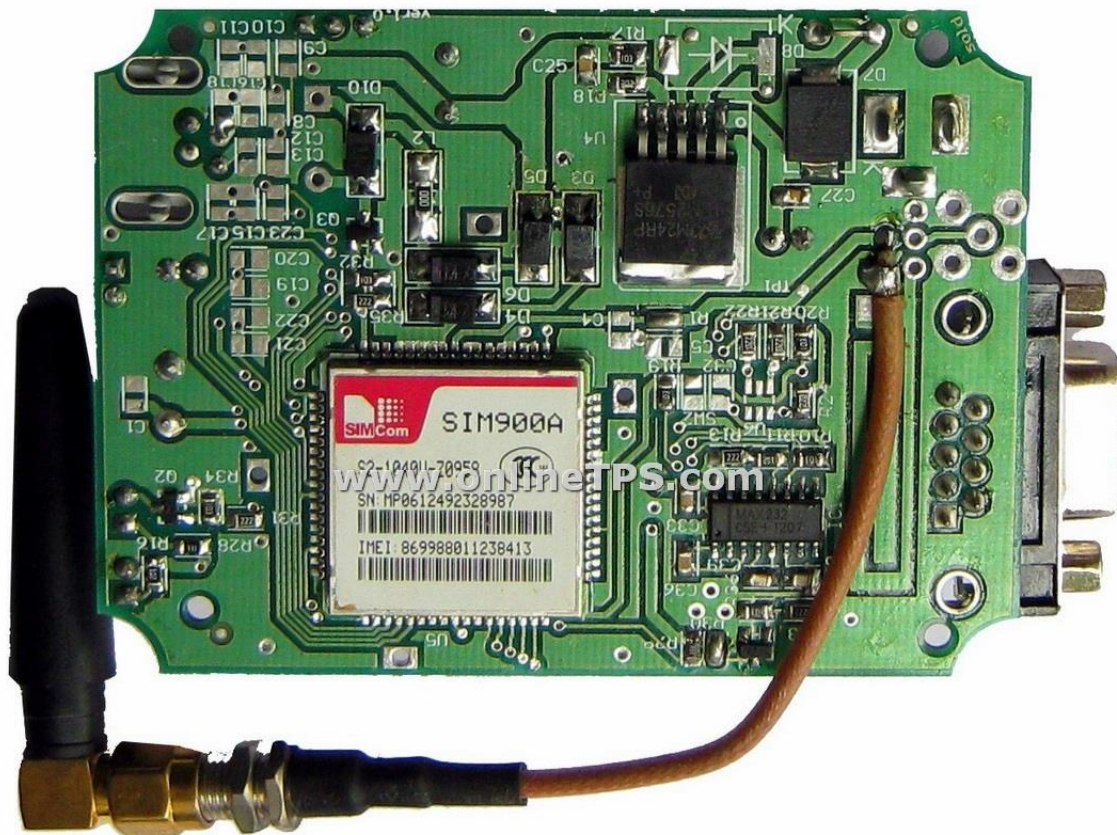


Figure 6: GSM/GPRS Module (wrongly displays SIM900, but it is actually FLY900)

GSM/GPRS modules make use of text based AT commands and terminate a message using the Ctrl-Z character (26 in decimal) [16]. This means that we cannot use binary format and hence we choose to encode our data into hexadecimal format. This will ensure that our data itself does not contain the Ctrl-Z character. Coming back to the



data transmission through the HTTP protocol, we could use the redundant “User-Agent” field of the HTTP header to transmit our data.

2-4 Cloud-based Gateway Design

It might seem weird, but it is not possible for any two computers connected to the internet to directly communicate with each other. However, they can communicate with each other through an intermediate Public Server. The problem lies in the fact that most of us connect to the internet through a private IP address. Private IP addresses are used because they allow for reuse of the same range of IP address in many different places, and IP addresses are limited.

This also means that the sensor modules we are going to design will not be able to transmit data directly to user computer for data storage. So, we need an intermediate Public Server that can act as an intermediate gateway in between, and pass on our data from the sensor modules to the user computer. Luckily, many companies have started to provide PAAS for Web Application onto their servers. Each of them supports a different set of application platforms and have a different pricing plan.

ASP.NET is one of the most popular web application development technologies developed by Microsoft Corporation. It is quite easy to use and is well integrated with

Visual C#. We choose to go with ASP.NET, Visual C# for the application development platform, and look for a company which also provides a free plan for low server usage. Accordingly, we choose a service provider called AppHarbor who also provides direct uploading of server code through a GIT repository, which makes the process of uploading the code to the server for execution quite simple.

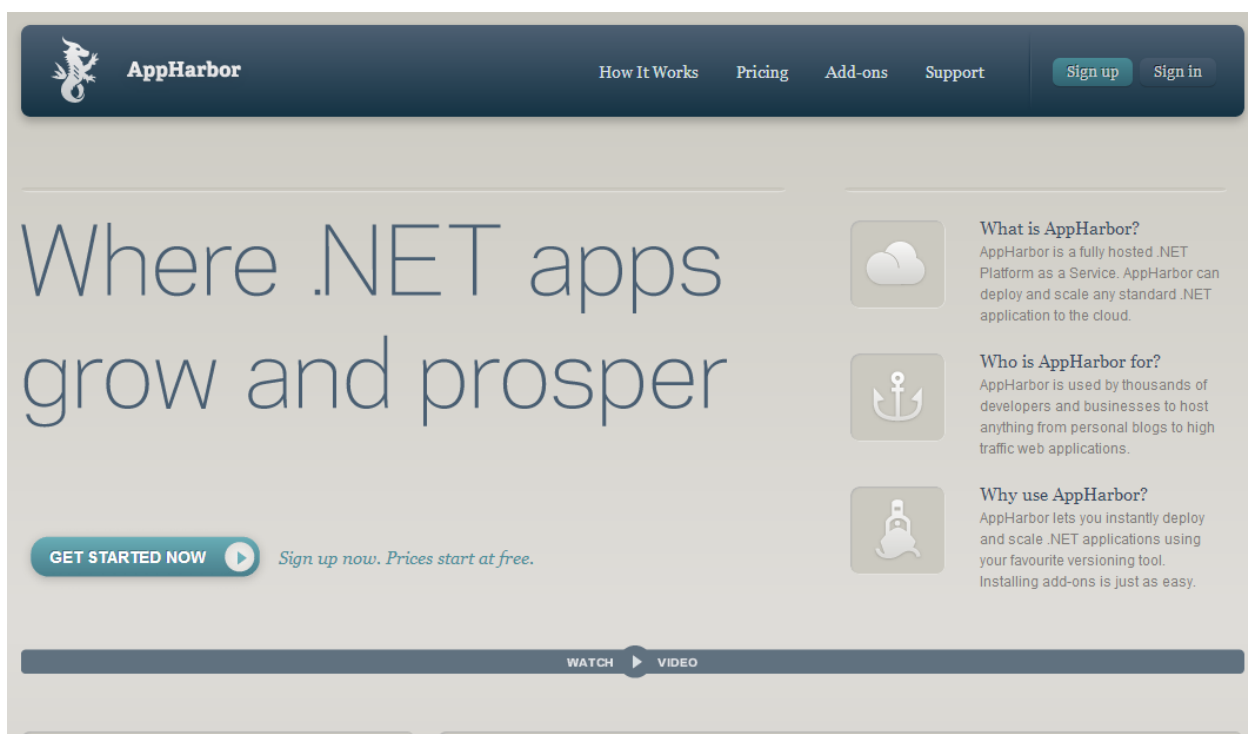


Figure 7: AppHarbor website which provides PAAS service for .NET Apps

Now, the Cloud-based Gateway receives the data from the Sensor Module as an HTTP GET request with “User-Agent” field value set to hexadecimal encoded value of the actual binary transmitted data. However, the HTTP request sent from a Web Browser has the “User-Agent” field set to a value representing the name of the Web Browser.



This difference allow the Cloud-based Gateway to distinguish between the data sent from the Sensor Module, and the request sent from a Web Browser.

When the Cloud-based Gateway receives data from the Sensor Module, it should store it temporarily in a FIFO manner, such as in a queue, and hand it over to the user computer upon request from it (Web Browser or a custom application that we are going to make). The Cloud-based Gateway is assigned a URL and bot the Sensor Module and user computer application must be configured to access that URL.

2-5 Local-Host Application

The Local Host Application, which would be executed on the user computer, can be written in almost any programming language. However, since Visual C# is one of the easiest high-level programming language, and has inbuilt classes for Web Client, we decide to go with it. Again, Console Application is much easier to code that a Windows Forms Application, and hence we decide to go with it.

There exist many database technologies but SQL is still the most popular, and hence well documented. Unlike many SQL database engines SQLite is an in-application database engine, which means there is no need of any additional SQL Server setup



necessary. Hence, we choose to go ahead with SQLite in Visual C# using the “System.Data.SQLite” library.

The Local-Host Application is supposed to periodically check the Cloud-based Gateway for any new received information and retrieve from it. Finally it should decode the obtained data into binary (actual) format and then store it into the database in an appropriate format.

We have not yet discussed about the data that should actually be sent from the Sensor Module. We could simply send the value (of water level), but this means that we will not be able to distinguish between data sent from different sensors. Hence we need an extra Sensor ID field. Again, there might be a noticeable delay between the Sensor Module capturing the value, and the Local-Host Application storing the data into the database. Hence, it is always better to add a Time field along with the value. This altogether is a good enough data format.

3 System Architecture

3-1 Hardware Architecture

The hardware part of this embedded system consists of one or more Sensor Modules deployed, per monitoring site. Such Sensor Modules can be easily used in multiple sites without any added cost, except the cost of the Sensor Modules themselves.

The Sensing Module was designed such that multiple sensors can be easily connected to a single unit. Each Sensor Module consists of a low-power, low-cost microcontroller, a wireless communication module and includes all the sensors connected to it. The sensor unit in itself is portable due to the fact that it is about the size of a pencil box, is battery-powered, and fully wireless (it does not require any kind of wireless support device for data transmission via the internet).

The microcontroller used is Freeduino USB microcontroller board, which is an Arduino Uno clone. Arduino is an open-source board based on the ATmega328 microcontroller by Atmel Corporation. It has 14 digital input / outputs, 6 analog inputs, a TTL UART interface, and can be easily battery powered [17]. Though ATmega328 microcontroller internally contains only one ADC, which is multiplexed to provide 6 analog inputs, it can be easily used to connect 6 different analog sensors without any issue.

The wireless communication module used is a FLY900 based GSM / GPRS modem with a 3-pin male header TTL UART interface which connects the module to the microcontroller. A GSM / GPRS antenna is connected to the module via an SMA connector. The modem can be powered with a power supply of 5V DC. It contains a 3V / 1.8V SIM interface with a SIM slot, where a SIM card belonging to a mobile service provider is attached [18].

The sensor unit also contains a power-supply module, which is part of the Freeduino USB microcontroller board. A Lithium-ion rechargeable battery can be used to power the sensor unit, however in this project we instead make use of a 9V DC adapter. The power supply unit provides each module of the sensor module the voltage supplies they need. The overall hardware block diagram is shown in **Figure 8**.

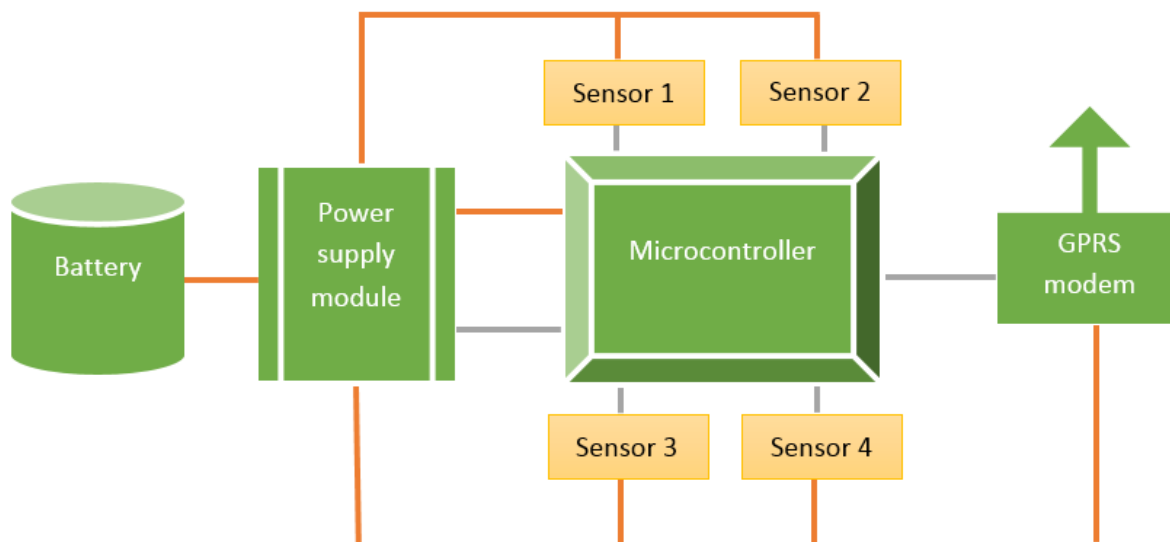


Figure 8: Overall hardware block diagram, of the sensor unit

3-2 Software Architecture

The simplified software architecture of this embedded system is the same as depicted in **Figure 9**. Each software module operates independently and communicates with the next, and / or previous module in a specific data exchange format. The dashed arrows shown in **Figure 9** depicts a long-range communication path, i.e., the internet. Each sensor in a sensor unit is assigned a 32-bit unique identifier which uniquely identifies it through the system. Each hydrologic data that is captured and transmitted through this embedded system, is accompanied by the sensor identifier of the sensor which captured it, and the time of capture.

The sensing module does the task of acquisition of hydrologic data, along with sensor identifier and capture time, and forwards it to the encoding module. The algorithm it runs is shown in **Algorithm 1**. The encoding module receives values from the sensing module, puts it in a packet format, encodes the packet data into a form which can be easily transmitted, and then forwards it to the wireless communication module. The algorithm it runs is shown in **Algorithm 2**. The wireless communication modem controls the GPRS modem, packs the encoded data from the encoding module into an HTTP packet, suitable for transmission through the internet and sends it to the cloud data queue module over the internet. The algorithm it runs is shown in **Algorithm 3**.

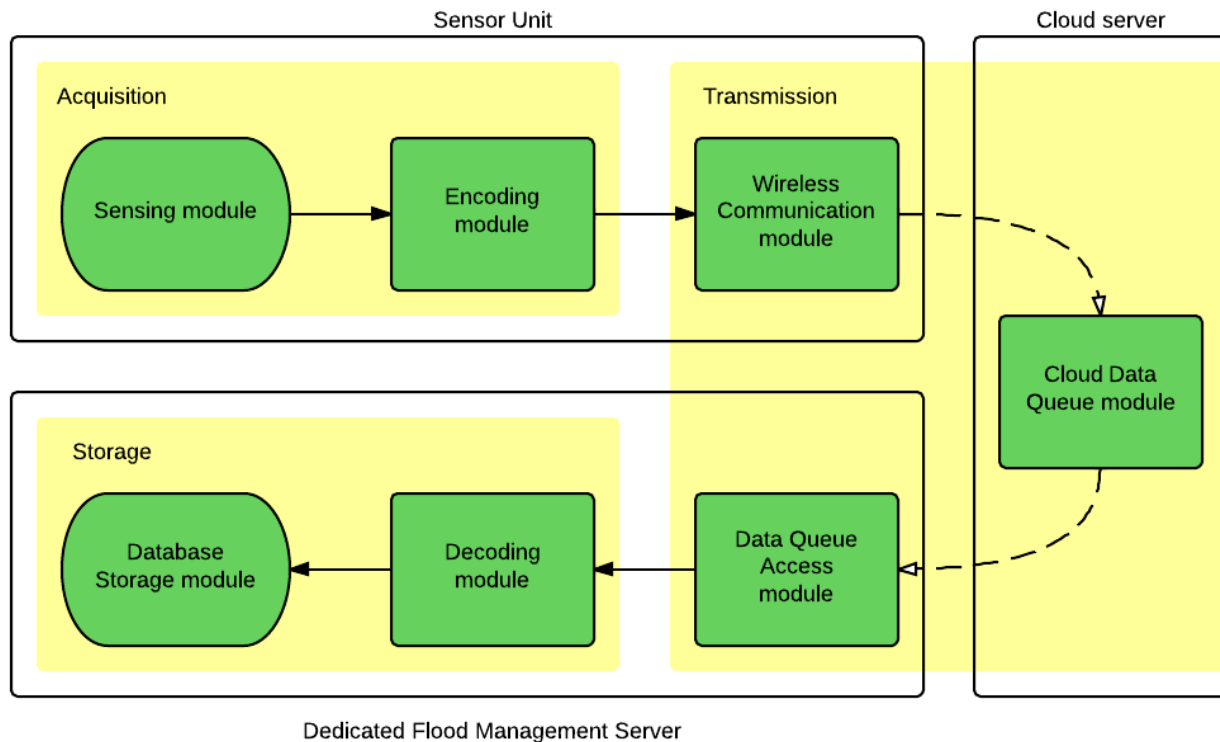


Figure 9: Overall Data-Flow block diagram of the Embedded System

All the code relating to the sensor unit is written using the Arduino IDE, which is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier. The overall block diagram of data-flow through the Sensor Module is shown in **Figure 10**.

The cloud data queue module receives HTTP packet from the wireless communication module, takes in only the encoded packet contained in it, and stores it in its temporary store. Whenever the data queue access module queries this server of any received

packet, it sends all the packets received by it and then removes them from its temporary store. The algorithm run by it is shown in **Algorithm 4**. The block diagram for data-flow through the Cloud-based Gateway module is shown in **Figure 11**.

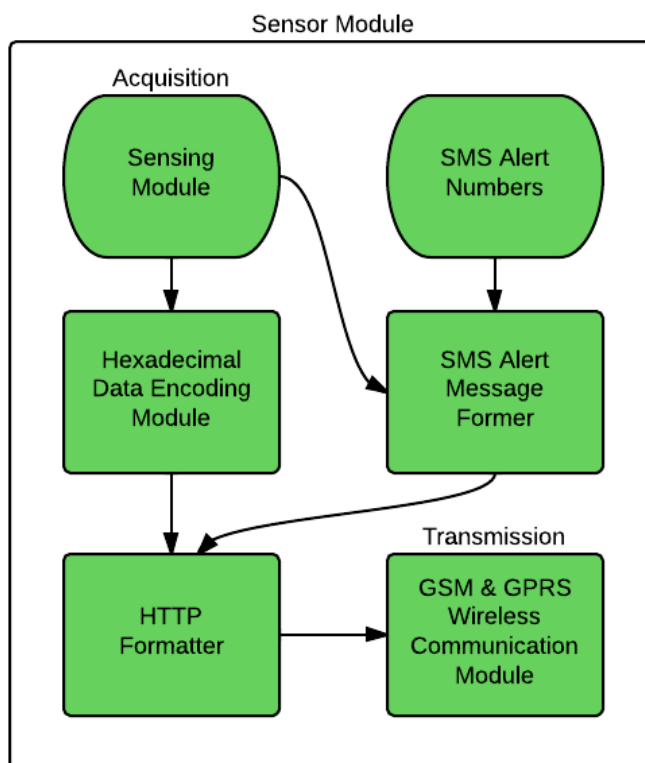


Figure 10: Block diagram of Data flow through the Sensor Module

All the code relating to the cloud server is written in ASP.NET with Visual C# as the code language, and Visual Studio as the IDE. ASP.NET is a server-side Web application framework designed for Web development to produce dynamic Web pages. It was developed by Microsoft to allow programmers to build dynamic web sites, web applications and web services. It was first released in January 2002 with version 1.0 of the .NET Framework, and is the successor to Microsoft's Active Server Pages (ASP) technology.

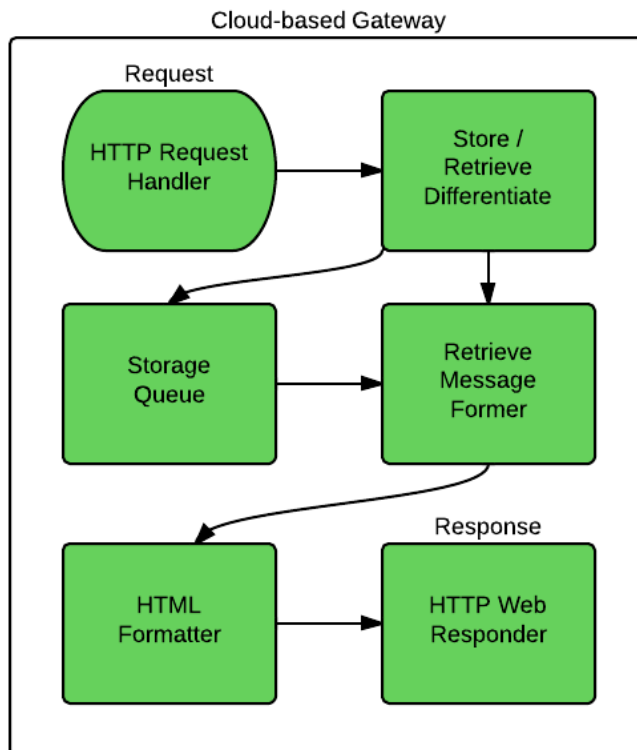


Figure 11: Block Diagram of data-flow through the Cloud-based Gateway

The data access queue module retrieves acquired data from the cloud data queue module, every user-defined time intervals, obtains the encoded packets, and sends them individually to the decoding module. The algorithm it runs is shown in **Algorithm 5**. The decoding module does the reverse of what the encoding module does, i.e., decodes the encoded packets, and forwards values to the database storage module. The algorithm it runs is shown in **Algorithm 6**. The database storage module obtains the values from the decoding module and finally stores them in the database in a form, such that it is easily accessible. The algorithm it runs is shown in **Algorithm 7**.

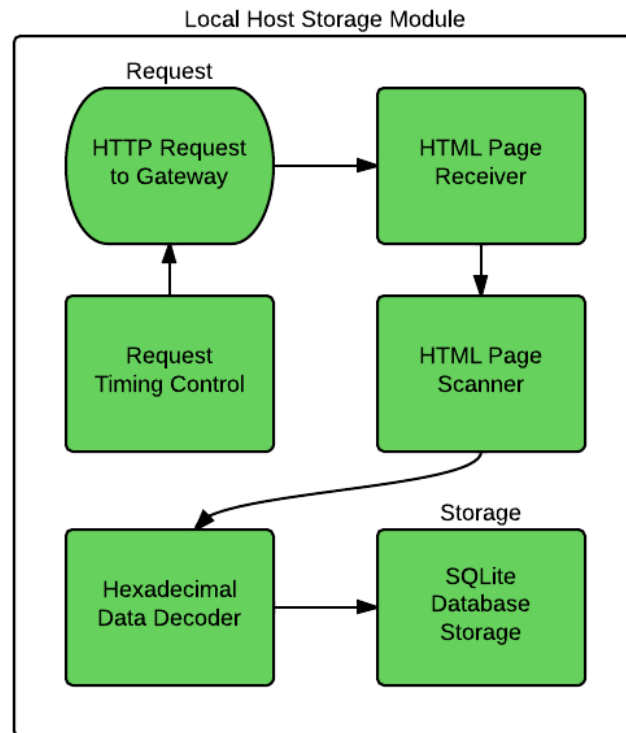


Figure 12: Block Diagram of data-flow through Local-Host Application

Code relating to the dedicated flood management server was written as a Console Application in Visual C#. SQLite was used as database. Visual C# is new, high-level, general-purpose programming language for building apps using Visual Studio and the .NET Framework. C# is designed to be simple, powerful, type-safe, and object-oriented.

Algorithm 1: Algorithm for the sensing module

1: Set [pin] = first used analog pin.

- 2: Wait for user-defined time.
 - 3: Select [pin] for A/D conversion.
 - 4: Set [mag] = 10-bit digital value after A/D conversion.
 - 5: Set [value] = 32-bit float value of [mag] after mapping.
 - 6: Set [time] = current time (32-bit UNIX date-time).
 - 7: Set [id] = 32-bit sensor id of the sensor being used.
 - 8: Send [id], [time], [value] to {encoding module}.
 - 9: Set [pin] = next used pin and go to step 2.
-

Algorithm 2: Algorithm for the encoding module

- 1: Receive [id], [time], [value] from sensing module.
 - 2: Put [id], [time], [value] in a 12-byte [packet].
 - 3: Convert [packet] to a hexadecimal string [hex].
 - 4: Send [hex] to the wireless communication module.
 - 5: Go to step 1.
-



Algorithm 3: Algorithm for the wireless communication module

- 1: Set [ready] = false.
 - 2: Wait for GPRS module to start.
 - 3: Attach to mobile service.
 - 4: Set access point name (APN).
 - 5: Start-up GPRS link.
 - 6: Connect to GPRS proxy server.
 - 7: Receive [hex] from sensing module.
 - 8: Put [hex] in an HTTP packet, [http].
 - 9: Send [http] to the cloud data queue module via internet.
 - 10: Disconnect from proxy server (automatic).
 - 11: Go to step 6
-

Algorithm 4: Algorithm for the cloud data queue module

- 1: Set [packet] = new list.
- 2: Receive HTTP packet, [http] from wireless module.



- 3: Remove HTTP header, and get hexadecimal string [hex].
 - 4: Add [hex] to [packet].
 - 5: Go to step 2.
 - 6: On data queue access module request:
 - 7: Form HTML file [html] from [packet].
 - 8: Clear [packet].
 - 9: Send [html] to the data queue access module via internet.
 - 11: Go to step 6
-

Algorithm 5: Algorithm for the data queue access module

- 1: Wait for user-defined time.
 - 2: Retrieve HTML file [html] from cloud data queue module.
 - 3: Get individual hexadecimal strings [hex] array.
 - 4: Send each [hex] array string to decoding module.
 - 5: Go to step 1
-



Algorithm 6: Algorithm for the decoding module

- 1: Receive [hex] from the data queue access module.
 - 2: Convert [hex] to 12-byte packet, [packet].
 - 3: Get fields [id], [time], [value] from [packet].
 - 4: Send [id], [time], [value] to database storage module.
 - 5: Go to step 1
-

Algorithm 7: Algorithm for the database storage module

- 1: Receive [id], [time], [value] from the decoding module.
 - 2: Connect to the database.
 3. Insert row with [id], [time], [value] to a table.
 - 4: Disconnect from database.
 - 5: Go to step 1
-

4 Results and Discussions

4-1 The Sensor Module

The embedded system was tested with a Piezo-Resistive Pressure Sensor, shown in **Figure 13**, connected to Internal ADC pin of ATmega328 microcontroller in the Sensor Module. It was meant to measure the water level at desired location. Since the Sensor Module was tested without water, the pressure readings taken were those of normal air pressure. However, the main objective of the experiment was to test the communication capability of the designed embedded system, and not its usage in the actual scenario.

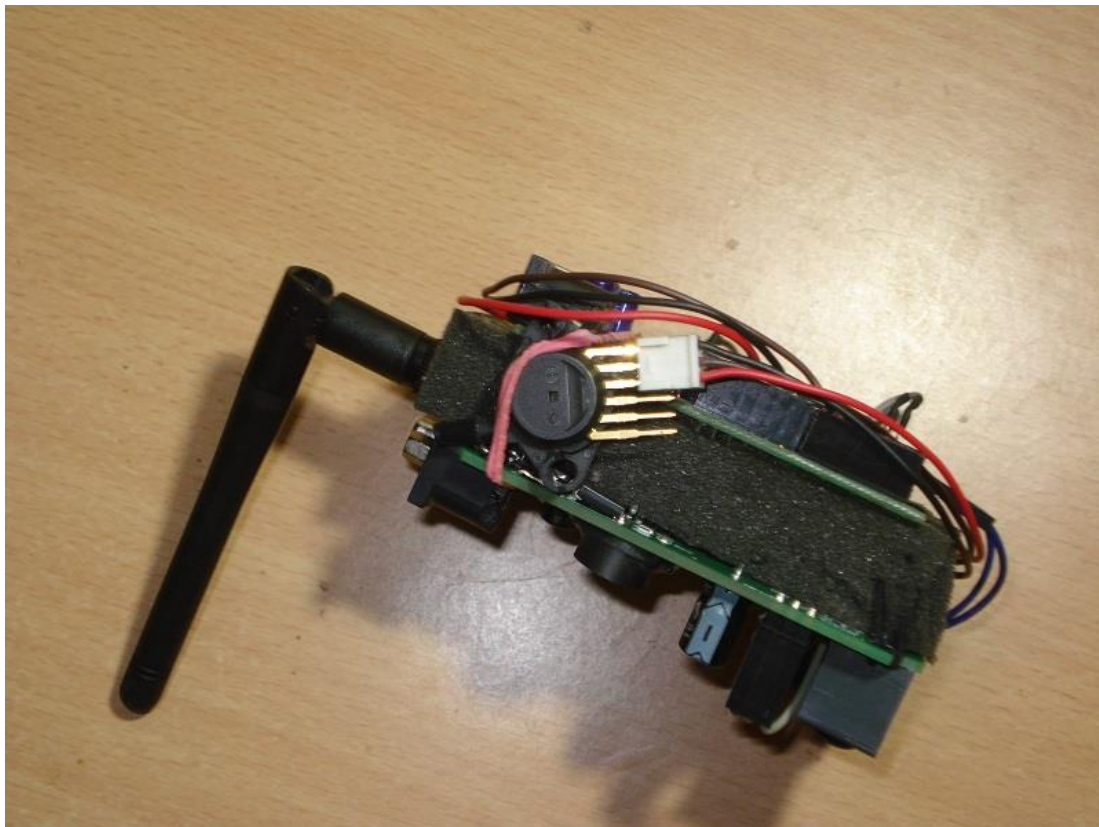


Figure 13: Pressure Sensor connected to the Sensor Module

Figure 14 shows the Microcontroller side of the Sensor Module, which is the Freeduino board. **Figure 15** shows the Wireless unit side of the Sensor Module, which is the GSM/GPRS Modem. The AVR-based Microcontroller controls the GSM/GPRS Modem through the UART Port using standard AT Commands supported by the Modem. The SIM slot of the Modem is loaded with a standard SIM card.



Figure 14: Microcontroller Side of the Sensor Module

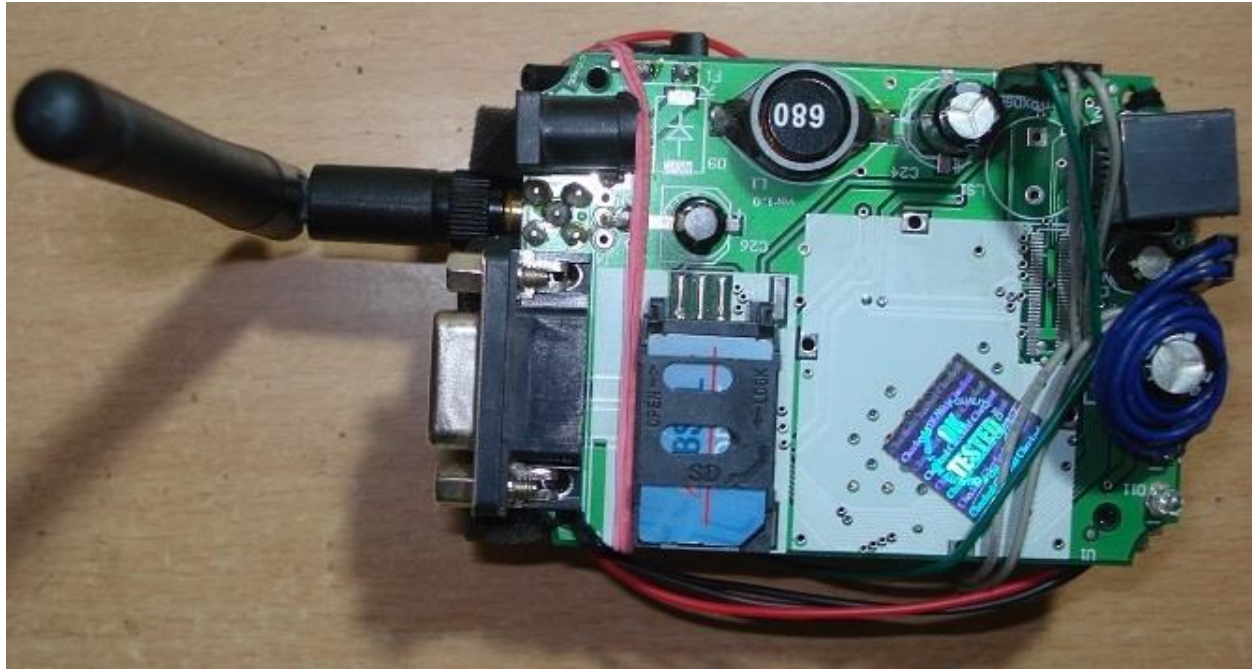


Figure 15: Wireless Unit side of the Sensor Module

A SIM card is inserted into the GPRS modem which allows the GPRS module to register to a mobile network, and use its GPRS service. The GSM/GPRS module with the SIM card is shown in **Figure 15**.

4-2 The Cloud-based Gateway

The data sent from this module is received at the Cloud-based Gateway. **Figure 16** shows the Gateway Application uploaded into the Server, and **Figure 17** shows the ASP.NET Application running and displaying encoded data received from the Sensor Module.

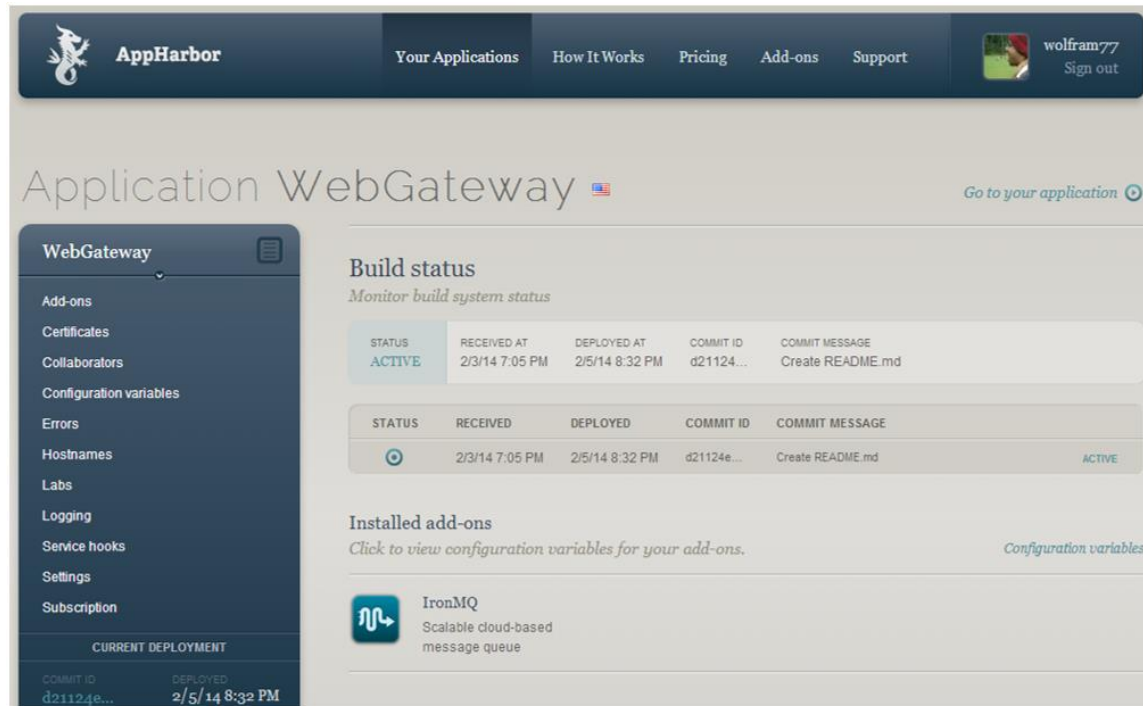


Figure 16: Cloud-based Gateway ASP.NET application uploaded onto the Server

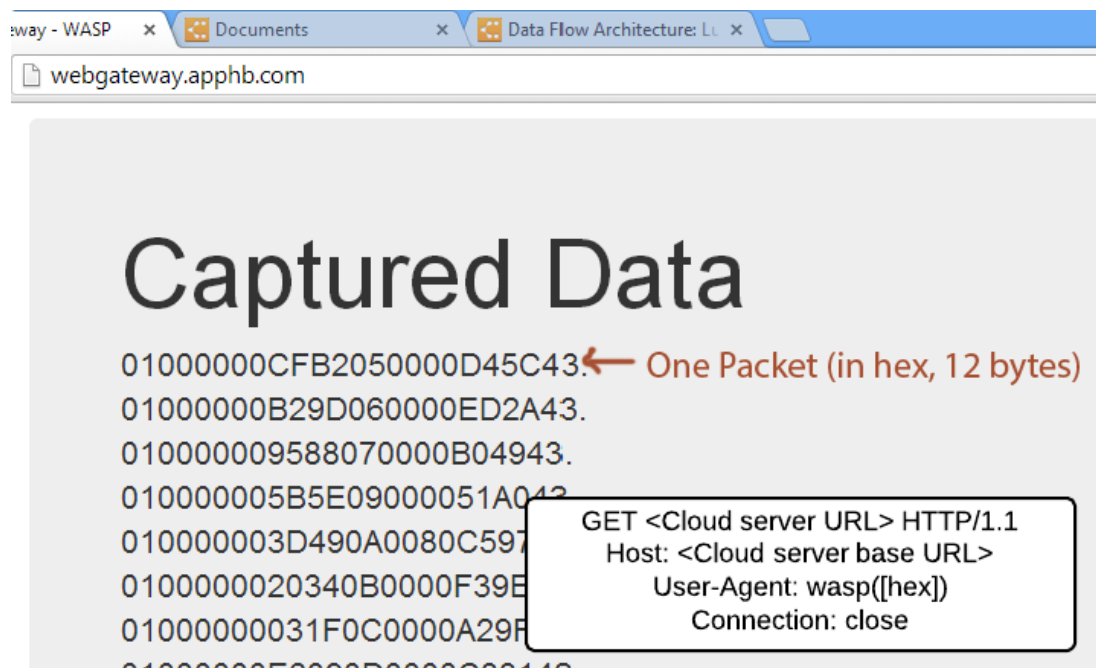
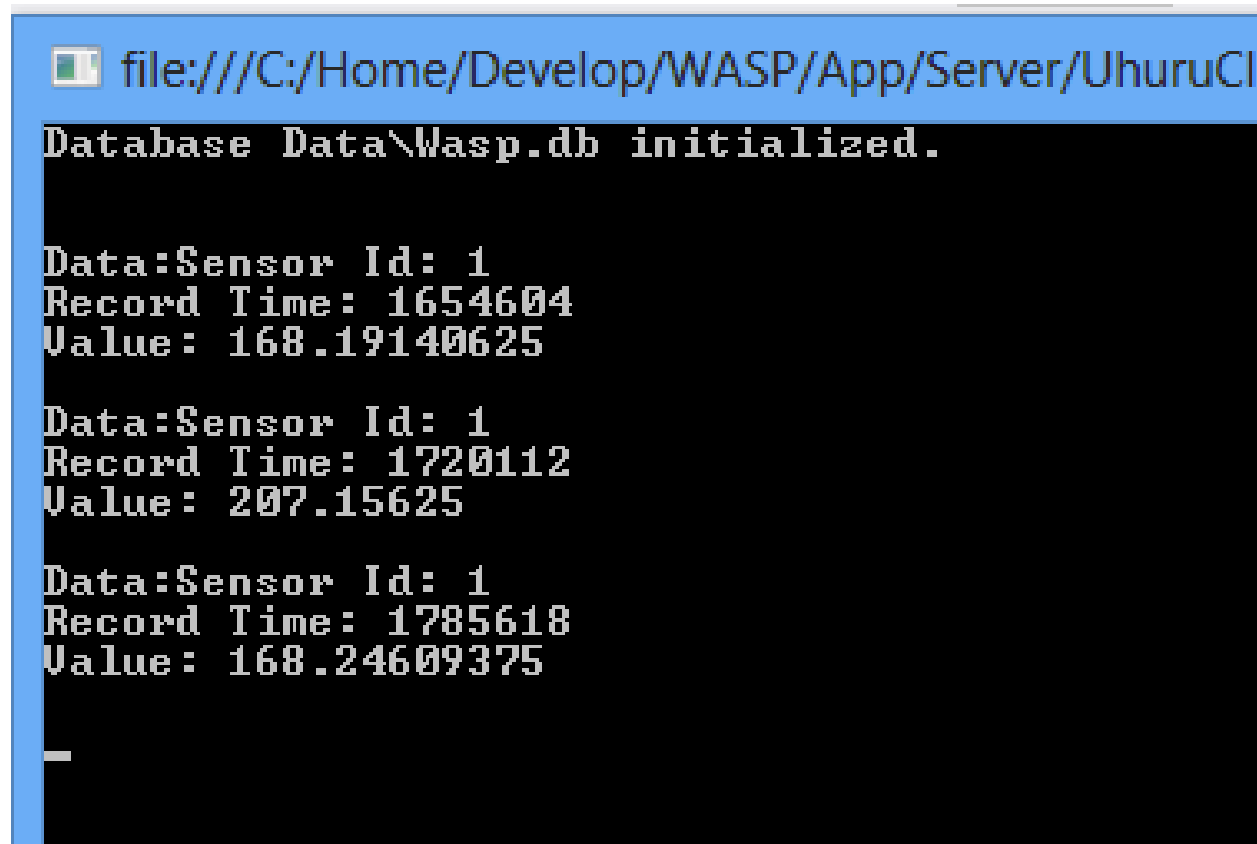


Figure 17: Gateway displaying encoded data received from the Sensor Module

4-3 The Local-Host Application

Captured data (encoded) is then fetched by the Local-Host Application, decoded and stored into an SQLite database, shown in **Figure 18**. **Figure 19** shows the data stored into the database being viewed using SQLite Studio. The data was copied onto Microsoft Excel and a graph was generated out of it, which is shown in **Figure 20**.



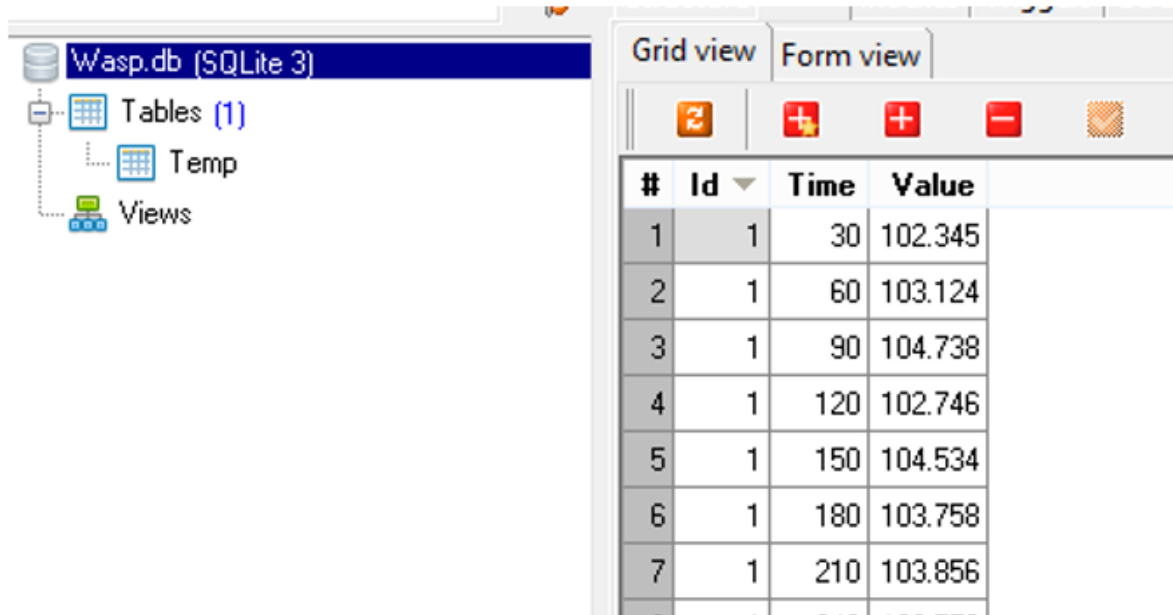
```
file:///C:/Home/Develop/WASP/App/Server/UhuruCI
Database Data\Wasp.db initialized.

Data:Sensor Id: 1
Record Time: 1654604
Value: 168.19140625

Data:Sensor Id: 1
Record Time: 1720112
Value: 207.15625

Data:Sensor Id: 1
Record Time: 1785618
Value: 168.24609375
```

Figure 18: Local-Host Application fetching data from Gateway and storing it



| # | Id | Time | Value |
|---|----|------|---------|
| 1 | 1 | 30 | 102.345 |
| 2 | 1 | 60 | 103.124 |
| 3 | 1 | 90 | 104.738 |
| 4 | 1 | 120 | 102.746 |
| 5 | 1 | 150 | 104.534 |
| 6 | 1 | 180 | 103.758 |
| 7 | 1 | 210 | 103.856 |

Figure 19: Data stored in the SQLite database being looked up using SQLite Studio

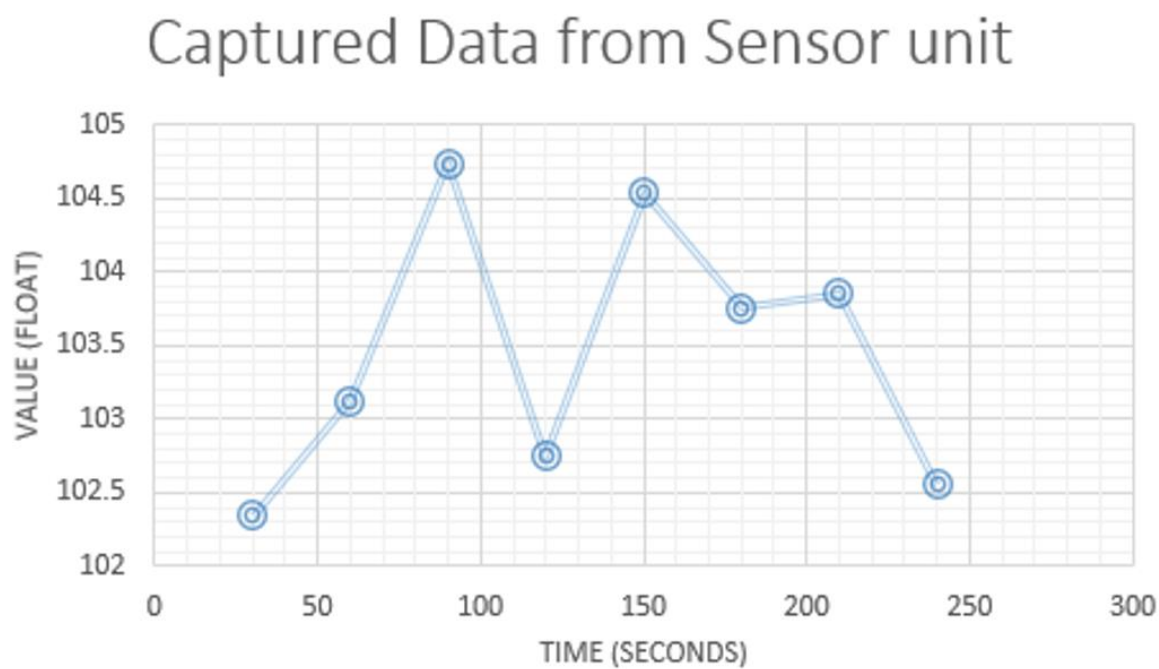


Figure 20: Data loaded from database being displayed as a graph in Microsoft Excel

5 Conclusion and Future Work

5-1 Conclusion

A Heterogeneous Sensor-based Embedded System for Flood Management was designed, implemented and successfully tested. This system has the capability to measure water level information at multiple locations in user-defined intervals of time. The captured data is then appropriately encoded and transmitted wirelessly through a GPRS connection, to a Cloud-based Gateway. A Local-Host program installed on the user computer then automatically extracts the received data from the Cloud-based Gateway, decodes it and finally stores it into an SQLite database. Once the data is collected into a database, it can be used for applying into flood forecasting models by the user.

The hardware developed for this project is a GPRS-based Wireless Sensor Module which consists of an ATmega328 based Freeduino Microcontroller Board along with a wireless GSM/GPRS Modem with an attached SIM card. A Piezo-Resistive Pressure Sensor with high liquid pressure measuring capability was used, so as to be able to measure the water level at the desired location, where the sensor module has been placed.

The software developed for this project consists of the Firmware for the Sensor Module, the program for the Cloud-based Gateway and the Local Host program. The Sensor Module Firmware is a set of source files written in C++, for AVR Architecture using the Arduino library. It has been built into a Hex file using the Arduino IDE and programmed



into the Freeduino Microcontroller using the IDE itself. The Cloud-based Gateway is developed as a Web Application in Visual Studio using ASP.NET in Visual C#. The Web Application is then uploaded to a PAAS Cloud Server, which runs it at a designated URL. Finally, the Local Host program is developed as a Windows Console Application using Visual C# and “System.Data.SQLite” library for C#. This Console application can be directly executed on the user machine and it automatically initializes a database, and starts storing collected data into it.

The developed system works autonomously and wirelessly, through the internet, which makes it a good candidate for replacing the existing manual flood monitoring sites. Since GPRS is used as the wireless communication technology for the system, the widespread availability of GPRS (which is available through the Mobile Network) allows the Sensor Modules to be easily deployed at the chosen flood monitoring sites.

5-2 Future Work

A developmental project such as this always has a lot of scope for future work. If the Sensor Module can be made cost effective, it would allow for deployment of a larger number of Sensor Modules for the same investment, and greater number of monitoring systems normally implies better prediction. Again, cheaper cost of Sensor Modules



would mean that they would be less susceptible to theft and easily replaced, if damaged.

Coming to the Software Architecture, the methodology used here is somewhat primitive in the sense that it does not support bi-directional communication and does not leave any room for future evolution of software on one side, if the other side is kept the same. This work can be further extended to implement a generic Internet of Things (IoT) communication system for all kinds of wireless sensors.

References

- [1] G. Robinson, "Flash Floods in France," 17 June 2010. [Online].
- [2] Wikipedia, "Flood control," 21 April 2014. [Online].
- [3] UKCC10, "Assessing the Benefits of Flood Warning: A Scoping Study," August 2006. [Online].
- [4] Wikipedia, "Natural disasters in India," 9 February 2014. [Online].
- [5] Water Resources Information System of India, "CWC National Flood Forecasting Network," 23 August 2013. [Online].
- [6] National Institute of Disaster Management, India, "Flood Forecasting and Warning System - NIDM," [Online].
- [7] India Meteorological Department, "Monsoon Report 2013," [Online].
- [8] CBS News, "India raises flood death toll reaches 5,700 as all missing persons now presumed dead," 16 July 2013. [Online].
- [9] Hindustan Times, "Rs 1000 cr relief for flood hit Uttarakhand announced," 19 June 2013. [Online].
- [10] B. Pratt, "Susquehanna Flood Forecast and Warning System," 2010. [Online].
- [11] K. W. C. C. T. C. Y. S. L. X. Y. Li, "A Web-based flood forecasting system for Shuangpai region," *Advances in Engineering Software*, no. 37, pp. 146-158, 2006.
- [12] E. L. M. Dale R. Shelton, "Modernized hydrologic forecast operations at National Weather Service Weather Forecast Offices," NOAA's National Weather Service, 2 February 1992. [Online].
- [13] Freescale Semiconductor, "Integrated Silicon Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated," October 2012. [Online].
- [14] Atmel Corporation, "ATmega328P Datasheet," [Online].
- [15] Flyscale Electronic Co. Ltd., "FLY900 Hardware Manual," [Online].
- [16] Flyscale Electronic Co. Ltd., "FLY900 AT Command User Guide," [Online].
- [17] Arduino, "Arduino Uno," [Online].
- [18] Flyscale Electronic Co. Ltd., "FLY900 Hardware Manual," April 2013. [Online].

