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

Abstract— **Floods are one of the most common form 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, operation, and results are described here.**

Index Terms— **Flood management, real-time acquisition, heterogeneous sensors, embedded system, remote deploy.**

I. Introduction

Flooding has many impacts. It damages property and endangers the lives of humans and other species. 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. 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 floodways. 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 U.S. history having cost billions of dollars.

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. 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. 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' [2].

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 [3]. 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 [4], 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.

II. System Architecture

A. Data flow architecture

Flood forecasting and management activities require huge volumes of hydrologic data to be readily available for incorporation into various forecasting models [4]. 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 described here is a solution to this problem, precisely.

The embedded system described here begins with an acquisition block, where hydrologic data from the sites being monitored is obtained. Acquisition occurs at user-defined intervals of time. Collected information is than transmitted via the internet to a storage module, where all the information is stored in an easily accessible form. This data can then be used for all kinds of flood forecasting and management activities.

The acquisition block consists of heterogeneous sensor modules, which capture quantities that represent some or the other hydrologic information. Data collected by these sensors is appropriately converted to represent actual information that was desired to be known. It is the appropriately encoded and forwarded to the transmission block. The transmission block begins with a wireless communication module, which is directly connected to the encoding module of the acquisition block. Here, the data is packed is packed into a packet format, and sent through the internet to a cloud server based data queue. The wireless communication does this by making use of existing mobile communication networks. As the packet reaches the cloud server, it is stored temporarily, until the dedicated flood management server requests it for any received packets. The cloud server based data queue then sends the packets available with it to the dedicated server, and removes it from its memory. The data access queue module, present in the dedicated flood management server, splits the received packets, and sends them individually to a decoding module in the storage block. The storage block, which consists of a decoding module and a database storage module finally stores the acquired information into a database. The decoding module receives packets from the data queue access module in the transmission block and converts it into a form such that it can be stored in the database. The database storage module takes the data and puts it in an appropriate location. The overall data-flow block diagram is shown in Figure *1*.

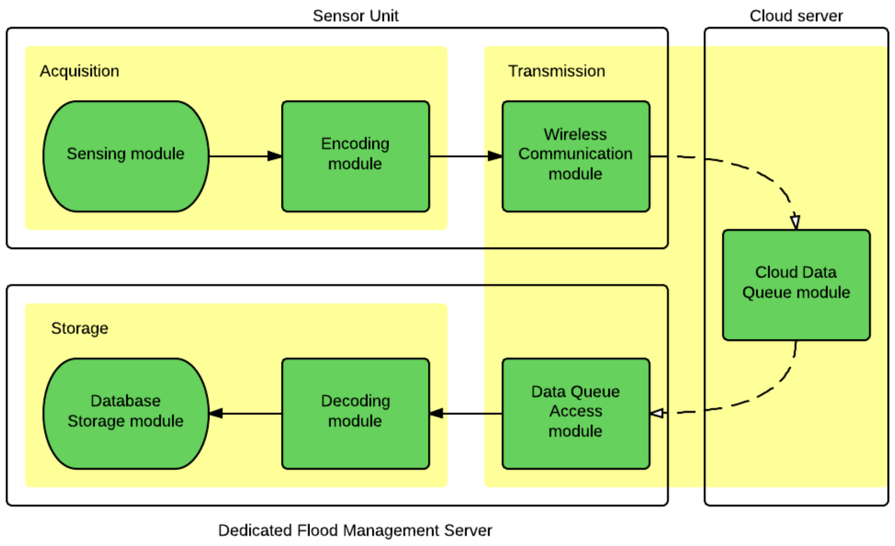


Figure 1: Overall Data-Flow block diagram of the Embedded System.

B. *Hardware architecture*

The hardware part of this embedded system consists of one or more sensor units deployed, per monitoring site. Such sensor can be easily used in multiple sites without any added cost, except the cost of the sensor units themselves. The sensing unit that was designed such that multiple sensors can be easily connected to a single sensor unit. Each sensor unit consists of a low-power microcontroller, a wireless communication module and includes all the sensors connected to it. The sensor unit in itself is very 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 an Arduino Uno microcontroller board, which 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 [5]. 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 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 [6].

The sensor unit also contains a power-supply module. A Lithium-ion rechargeable battery can be used to power the sensor unit. 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 2.

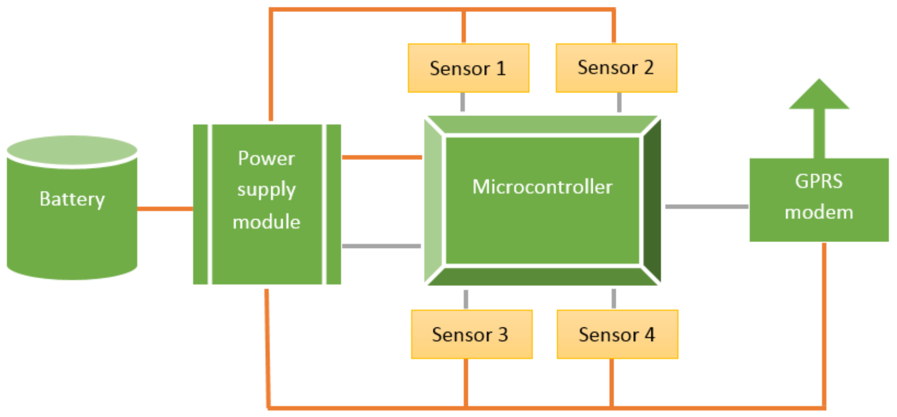


Figure 2: Overall hardware block diagram, of the sensor unit.

C. *Software architecture*

The simplified software architecture of this embedded system is the same as depicted in Figure 1. 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 1 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.

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 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.

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.

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 7. 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.

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 modern, high-level, multi-paradigm, 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
```

III. Results and Discussions

The embedded system was tested without using any sensor for the sensor unit (which means random values were acquired). Hence, this test was a test of the communication capability of the embedded system, and not its usage with a particular sensor. Figure 3 shows the sensor unit, where the Arduino Uno microcontroller board shown on the left, and the wireless communication module shown on the right.

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. Note that, the mobile service providers generally provide internet capability to a GPRS enabled device through a proxy server in their internal network, which is connected to the internet. All web requests are made through this proxy server. Since, this proxy server supports the HTTP protocol, data must be sent to the cloud server using HTTP protocol. In our case, we passed the data in hexadecimal format, and passed it in the "User-Agent" field of the HTTP header of each packet. The cloud server then extracts data from the "User-Agent" field, and stores it, until requested by our dedicated flood management server.

Hexadecimal packets received at the cloud data queue module is shown in Figure 4. Each line in the figure represents one packet in hexadecimal format that was originally encoded by the encoding module and sent by the wireless communication module.

The dedicated flood management server receives data from the cloud server as an HTML file. It then looks for the hexadecimal packets in the HTML file. Extracted data is then converted to binary format, interpreted in appropriate format, and finally stored to a SQLite database. For this purpose, a SQLite library for the C# programming language is used.

Figure 5 shows the dedicated flood management server retrieving data from the cloud data queue module. As can be seen in the figure, we are able to decode each packet, and it is stored to a SQLite database in the background. Figure 6 shows SQLite Studio software which displays the records in the database, and Figure 7 shows a plot of the data. This plot is made by extracting collected information from the database, to Microsoft Excel, and making a 2D scatter plot.

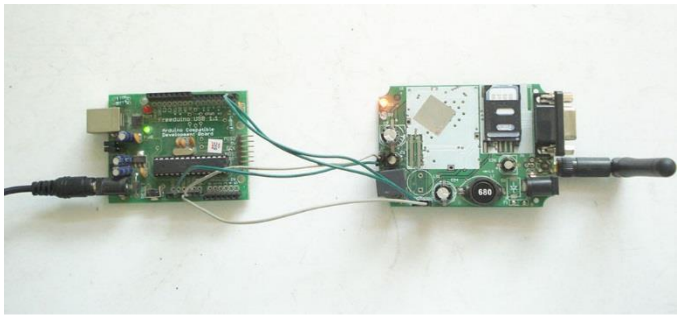
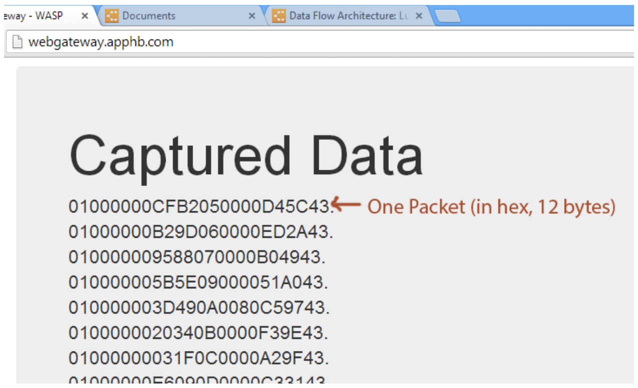


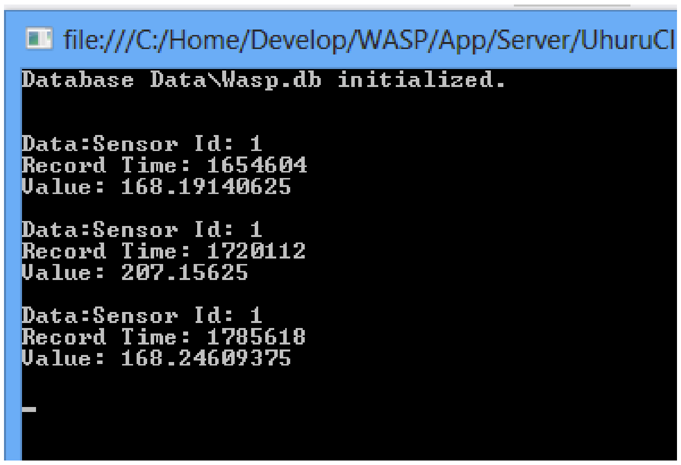
Figure 3: Sensor unit, of the embedded system. Arduino Uno microcontroller board on the left, GPRS module on the right.



Captured Data

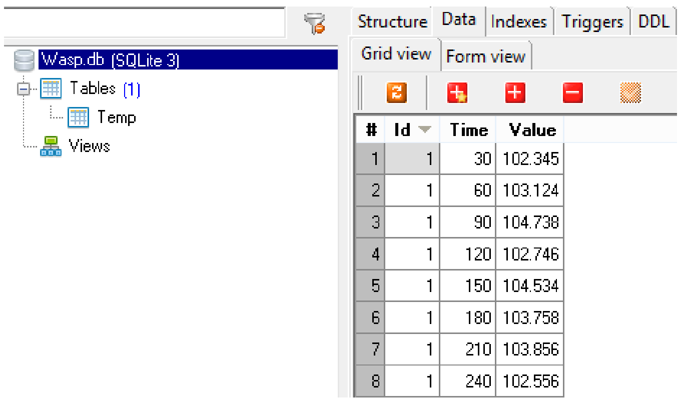
```
01000000CFB2050000D45C43. ← One Packet (in hex, 12 bytes)
01000000B29D060000ED2A43.
010000009588070000B04943.
010000005B5E09000051A043.
010000003D490A0080C59743.
0100000020340B0000F39E43.
01000000031F0C0000A29F43.
```

Figure 4: Hexadecimal packets received at the cloud data queue module



```
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 5: Dedicated flood management server retrieving data from the cloud data queue module



#	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
8	1	240	102.556

Figure 6: Acquired data stored in a SQLite database, shown in SQLite Studio

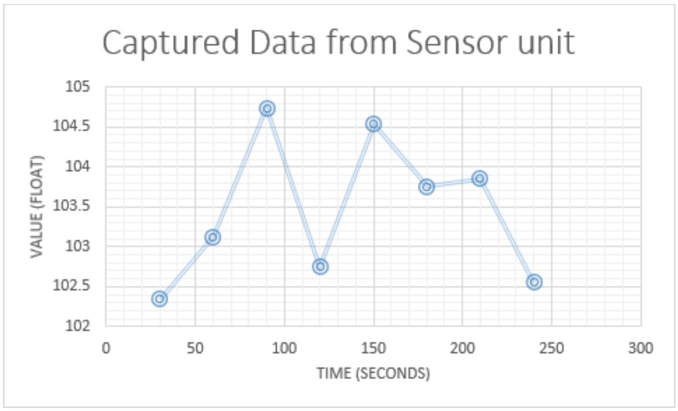


Figure 7: Captured data is taken from the database and plotted in Microsoft Excel

IV. Conclusion

The Heterogeneous Sensor-based Embedded System for Flood Management was designed and implemented and successfully tested using available microcontroller boards and GPRS boards. However, in order to reduce cost and increase the durability and robustness of the developed product, it is highly desirable to implement it in form of a custom PCB. This work can be further extended to implement a generic Internet of Things (IoT) communication system for wireless sensors.

References

[1] Wikipedia, the free encyclopedia, "Flood control," Wikipedia, the free encyclopedia, 3 February 2014. [Online]. Available: http://en.wikipedia.org/wiki/Flood_control.

[2] Foundation for Water Research, "Assessing the Benefits of Flood Warning: A Scoping Study (UKCC10)," Foundation for Water Research, August 2006. [Online]. Available: http://www.fwr.org/floodde/ukcc10.htm. [Accessed 7 February 2014].

[3] B. Pratt, "Susquehanna Flood Forecast and Warning System," 2010. [Online]. Available: http://www.srbc.net/stateofsusq2010/documents/SFFWSFeatureArticle.PDF.

[4] X.-Y. Li, K. W. Chau, C.-T. Cheng and Y. S. Li, "A Web-based flood forecasting system for Shuangpai region," *Advances in Engineering Software,* no. 37, pp. 146-158, 2006.

[5] Arduino, "Arduino Uno," Arduino, [Online]. Available: http://arduino.cc/en/Main/arduinoBoardUno. [Accessed 8 February 2014].

[6] Flyscale Electronic Co. Ltd., "FLY900 Hardware Manual," 27 April 2013. [Online]. Available: http://www.flyscale.cn/uploads/image/121220055116140626aypxjh16d186.pdf. [Accessed 8 February 2014].

[7] CBSNEWS (AP), "India raises flood death toll reaches 5,700 as all missing persons now presumed dead," CBSNEWS, 16 July 2013. [Online]. Available:

http://www.cbsnews.com/news/india-raises-flood-death-toll-reaches-5700-as-all-missing-persons-now-presumed-dead/. [Accessed 7 February 2014].

[8] India - Water Resource Information System, "CWC National Flood Forecasting Network," India - WRIS, 23 August 2013. [Online]. Available: http://india-wris.nrsc.gov.in/wrpinfo/index.php?title=CWC_National_Flood_Forecasting_Network. [Accessed 7 February 2014].

[9] C. Hodanbosi and J. G. Fairman, "Fluids Pressure and Depth," National Aeronautics and Space Administration, August 1996. [Online]. Available: http://www.grc.nasa.gov/WWW/k-12/WindTunnel/Activities/fluid_pressure.html. [Accessed 8 February 2014].

[10] Freescale Semiconductor, "MPX5700 Datasheet," 2007. [Online]. Available: www.freescale.com/files/sensors/doc/data_sheet/MPX5700.pdf. [Accessed 8 February 2014].

[11] Atmel Corporation, "ATmega328P Datasheet," [Online]. Available: http://www.atmel.com/images/atmel-8271-8-bit-avr-microcontroller-atmega48a-48pa-88a-88pa-168a-168pa-328-328p_datasheet.pdf. [Accessed 8 February 2014].

[12] International Telecommunications Unions (ITU), "The World in 2011: ITC Facts and Figures," Geneva, 2011.

[13] M. Kummu, H. d. Moel, P. J. Ward, O. Varis and M. P. (Editor), "How Close Do We Live to Water? A Global Analysis of Population Distance to Freshwater Bodies," US National Library of Medicine, National Institutes of Health, 8 June 2011. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3110782/. [Accessed 8 February 2014].

[14] Elec Freaks, "EFCom Pro GPRS/GSM Module," Elec Freaks, 27 July 2013. [Online]. Available: http://www.elecfreaks.com/wiki/index.php?title=EFCom_Pro_GPRS/GSM_Module. [Accessed 8 February 2014].

[15] Total Project Solutions, "SIM900A GSM Modem with Serial Interface," Total Project Solutions, [Online]. Available: http://www.onlinetps.com/shop/index.php?main_page=product_info&cPath=67&products_id=1072. [Accessed 8 February 2014].

[16] Flyscale Electronic Co. Ltd., "FLY900 AT Command User Guide," [Online]. Available: www.flyscale.cn/uploads/image/121220055142546877ic2l7408r89d.pdf. [Accessed 8 February 2014].

[17] Indiana University, "What are private IP addresses, and what are the reserved ranges?," Indiana University, 17 October 2013. [Online]. Available: http://kb.iu.edu/data/aijr.html. [Accessed 8 February 2014].

[18] Telecom ABC, "GGSN," Telecom ABC, 2005. [Online]. Available: http://www.telecomabc.com/g/ggsn.html. [Accessed 8 February 2014].