A wireless data logger with GSM connectivity and Bluetooth interface between sensors

Keerthiraj Nagaraj,
Electrical and Computer Engineering
University of Florida,
Gainesville, FL 32611, USA
k.nagaraj@ufl.edu

Siddharth Gupta,
Computer Science
University of Florida,
Gainesville, FL 32611, USA
siddg@ufl.edu

Thiago Borba Onofre,
Agricultural and Biological Engineering
University of Florida,
Gainesville, FL 32611, USA
tonofre@ufl.edu

***Abstract*—*It is estimated that 30% of the world food production is lost to pests and diseases. Securing the food supply is a major challenge for farmers. Plant diseases are affected by weather conditions and each disease has its own specific weather requirements. Our wireless data logger system is an attempt to aid farmers in anticipating disease onset in their crops so they can take appropriate measures in time. Disease Alert Systems (DAS) are used to notify farmers when they must spray their crops based on weather conditions. DAS are strongly dependent on weather station data, so it is very important to have a low cost and reliable data logger where farmers would be able to easily add/replace sensors. Commercial data loggers use wired sensors, but this is a limitation due the cable length and connector type. Data loggers have been evolving with the advance in wireless technology such as the GSM and Bluetooth network. In this work, we propose the development of a GSM based datalogger where sensor data flows over a Bluetooth network. The new data logger will replace sensor cables of a conventional datalogger. A concept prototype was developed using low cost and open source resources. Two dataloggers were built: a master and a mini. The mini only has the sensors and a Bluetooth radio. The master data logger is a sink node, as it as two radios: Bluetooth and GSM.*

Keywords*—*GSM Network, Data Logger, Weather Station, Bluetooth Network, Cloud Web Server, InitialState, Arduino, IoT

I. INTRODUCTION

A weather station is a data logger with a collection of sensors to monitor the surrounding environment. Weather stations are extremely important to agriculture, as they are used by researchers to understand how the weather affects daily farm operations. Also, weather station data have been used to predict the occurrence of diseases, as for example, a disease alert system for strawberries was implemented using weather station data.

Weather stations can be classified accordingly with how the user can access the recorded data: locally or wirelessly. Standalone weather stations have an external memory, were real time data is stored with time stamp. Periodically, the user must go the field to manually collect data. Wireless weather stations, report data constantly to a remote server, usually in a 15 minutes' interval, where this data can be accessed using internet. Cell phone is the most popular option to add wireless connectivity to a weather station. Almost all commercial weather stations available nowadays have cell phone connectivity.

A current limitation of current weather stations is the number and the type of sensors. Usually, each manufacturer as its own connector type Reference. Another limitation is the number of inputs, where the user can not add more than a certain number of sensors on the data logger. A third limitation are cables, as they are limited by length.

In this project, two data loggers were designed, as shown in Figure 1. The main data logger contains the cell phone radio with the Bluetooth master. The second data logger contain a Bluetooth radio and the sensors, which will be called mini data logger. The mini data logger, periodically reads and transmits data to the main data logger, which will upload data to the cloud. The main objective of the mini data logger is to replace cables connection adding flexibility to the final user.

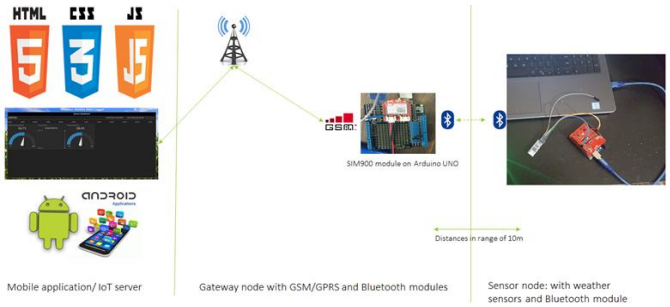

Fig. 1. System Architecture

The content of this paper is organized as follows. In the next Section, we explore the similar existing systems. The following section explains the design of the wireless data logger. The hardware development of cellphone data logger with Bluetooth sensor interface Section IV. Software development, such as cloud and an android application will be carried out in Section V. Section VI presents the future work and the conclusion of this paper.

II. RELATED WORK

A low-cost weather station with Bluetooth connectivity was developed at the Pedagogical and Technological University of Colombia Tunja. The monitored variables were: temperature, relative humidity, soil moisture and wind

direction, and the sensors were calibrated with measuring instruments available locally. Via Bluetooth wireless module, the monitored data was sent to a local computer, [2].

A GSM wireless datalogger was presented for monitoring power and water flow rate at a small hydro power generation station in Malaysia. As the Intake of small hydro station is located inaccessible area from the powerhouse, a system composed by a GSM datalogger and a RF wireless network will be installed at the power plant. The gateway will have two radios: GSM and RF. The RF link will be used to receive data from a node installed at the intake. Once, power is measured, the GSM will upload data an online database server where the data will be analyzed, [3].

A wireless data-acquisition system was developed for reading and storing information from 15 weather stations located in zones of difficult access. These stations were located in the north side of the Natural Park of Sierra Nevada, in Huéneja (Granada). The authors used GSM/GPRS cell phone radios, as they are offer considerable advantages in transmitting the information at big distances, specifically due favorable conditions, flexibility and the low costs associated with those modules. A microcontroller based circuit for data acquisition was designed, which scans 8 sensors together at any intervals programmable. Their system was compared with commercial data loggers: Campbell and Hobo H8. The results showed an error of order 1%, which they believe is due to the analog to digital converter quality, [4].

An Automatic Remote Weather Station with a PC-based Data Logger was developed at the University of the South Pacific. The system consists of a weather station prototype that collects weather sensor data such as air temperature, relative humidity, dew point, wind speed, and rainfall. The system performs automatic or unmanned measurements of weather data and sends it wirelessly to the sink node which is a local PC for logging and displaying the data in a graphical user interface. The authors of the paper [5] in the future work describe the possibility of transfer of data over GSM/GPRS networks and to provide weather data to users as text messages.

Continuous development of the emerging technology on data logger has brought the creation of innovation and applications. By using the available resource in the market such Bluetooth and cell phone modem, it is possible to customize data logger with multiple sensors expanding the capabilities of current data loggers. Also, this project incorporates the notions that put forth in the systems explored above, and it provides additional features like: email alerts, cloud server, historical data, and ability to change timelines. The advantages of our system are discussed in detail in a later section.

III. HARDWARE DESIGN

This section discusses about the hardware components used in our project and how they were used to complete this project.

A. Microcontroller

Arduino UNO, Figure 2, is a microcontroller which uses ATmega328P chipset. It is one of the highly-used microcontrollers for academic projects as it is easy to use and supports libraries for various sensors, communication modules and software packages. Programs are compiled and executed on to the board using a software package called as Arduino IDE and the board can be reused as many times as necessary as the functionality of the board can be changed just by changing the code and rewriting it onto the Arduino board. Arduino UNO board has 14 digital I/O pins, 6 analog inputs, a 16MHz quartz crystal, a USB connection to write the programs on to the board, a power jack, an ICSP header, built-in LED, and a reset pin. For Arduino UNO, operating voltage is 5V and the acceptable input voltage range is 7-12 V. Each input I/O pin in the board consumes 20 mA DC current when being used by the controller.[6]

Fig. 2. Arduino Uno microcontroller

B. Cellphone radio

The SIM900 GPRS module from Seeed Studio, Figure 3, was used to establish a cellphone connection between Arduino board and the IoT server, [7]. The features of this board are:

- Quadband support: 850/900/1800/1900 MHz
- Power supply: 5V via 5V pin and 6.5-12V via Vin
- Programmable Baud rate
- Software serial pins: D7 and D8
- Communication support: Standard GSM 7.7/7.5
- SIMCOM AT commands supported

Fig. 3. GPRS SIM900 Arduino Shield

C. Bluetooth Radio

HC-05, Figure 4, is a simple Bluetooth SPP (Serial Port Profile) module designed for simple Bluetooth applications such as data transfer between two microcontrollers, and other transparent wireless serial connections. It works on Bluetooth V2.0+EDR version and uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with Adaptive Frequency Hopping feature (AFH), [8].

Bluetooth Serial Port Profile is similar to the functionality of RS-232 or a UART i.e. mainly serial communication interface but it works with the wireless medium. It sends bursts of data from one device to another which are connected via Bluetooth medium. Features of HC-05 module:

- Sensitivity: -80 dBm
- RF transmit power: +4 dBm
- UART interface with programmable baud rate
- Integrated antenna
- Auto connect to the last device on power as default.
- 5 pins: Vcc, Gnd, RX, TX, Key
- Operating voltage: 1.8-3.6V

Fig. 4. HC05 Bluetooth radio

D. Sensors

The Sparkfun weather shield, Figure 5, was used to collect sensor values such as temperature, pressure, and relative humidity. It is an easy to use Arduino shield and even provides connections for optional sensors such as wind speed, wind direction, rain gauge. However, the optional sensors are not being used in the project and may be considered for future work. The sensors put to use are HTU21D humidity/temperature sensor, MPL3115A2 barometric pressure sensor and ALS-PT19 light sensors.The operating voltage level of this shield is 3.3-16 V and it has built-in voltage regulators. Accuracies of humidity, temperature and pressure are ±2%, ±0.3 °C and ±50 Pa, [9].

Fig. 5. Sensor board.

E. Bluetooth Network

Generally, in a weather station there will be many sensors such as temperature sensor, pressure sensor, wind sensor, rain sensor, relative humidity sensor etc. All these sensors usually have different wire/pin/port configurations based on their manufacturers. Hence, it is often difficult to replace a sensor or add new sensors to the existing system due to mismatch of wire/pins/ports. This problem arises because of wired connectivity. Using a wired connection for these sensors makes the frequent management of weather station an arduous task and the system is prone to physical damage due to extreme weather conditions or other external agents. Connecting all the sensors to microcontroller in weather station using Bluetooth medium addresses the problems discussed above and hence we use Bluetooth medium to transfer data from sensor nodes to the gateway node. Moreover, this allows the sensors to be placed anywhere as long as they are in the range of gateway Bluetooth node.

In this project, we have used two HC-05 Bluetooth modules, one at the sensor node in the slave mode and the other one at the gateway node in master mode. The master module has been set to initiate, pair and connect operation with slave module as soon as both are switched ON. We use a software serial to connect these Bluetooth modules to Arduino UNO boards. Data collected from Sparkfun weather shield is transmitted from sensor node to gateway node over the bluetooth channel created using HC-05 modules.

IV. Software Design

The software design is split in two parts: data logger and cloud with android application.

A. Data logger

The data logger was programmed using the Arduino IDE v1.6.12. It is an open-source software that makes it easy to write code and upload it to the Arduino board. It runs on Windows, Mac OS X, and Linux. The programming was done using a dialect of features from the programming languages C and C++. The following libraries were used to aid in

development - Software Serial, SparkFunMPL3115A2, SparkFunHTU2ID. Figure 6 shows two Arduinos with Bluetooth radios, were the Arduino on the right side has the weather shield, and the Arduino on the left side (with the shining led) is the master.

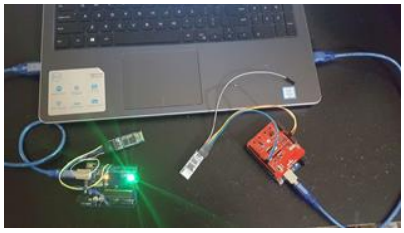

Fig. 6. Bluetooth network

A software serial library was used to establish a communication link between Arduino board and Bluetooth module. Both the master and slave modules were programmed to work at the baud rate of 9600. AT commands were employed for configuration details of baud rate, action mode, and MAC address of slave module using Arduino IDE. The master module was configured using this slave address such that when it is switched on every time it will look for the same slave module and connect to it automatically.

At the sensor node, libraries related to Sparkfun weather shield were installed and used to create an instance through which the sensor values could be read. A timer was used to set a 20 second delay between each set of sensor readings. The sensor values obtained were checked for specific condition in humidity and the status of on-board LED was modified per this condition to notify the user. Sensor reading such as Temperature, Humidity and Pressure were transmitted from sensor node to gateway node. One of the major challenges faced in Bluetooth networks was that we received garbage values at the gateway nod, due to initial mismatch in baud rates of masters and slave modules.

Also, there were innumerous issues with existing libraries for the GSM. In this case, the workaround was working with AT commands programmatically to establish a TCP connection and perform HTTP requests. Firstly, a network Access Point Name (APN), username and password were defined to establish the connection and obtain a working IP. These configuration details are network carrier specific and would vary for each carrier. We use a T-mobile SIM for our project. It is required that a data plan be activated prior to installing the SIM card in the GSM module. Once a TCP connection was established, the server URL and port number (discussed in the Cloud server section) were set. Then the command HTTPACTION=0 (0 signifies a GET request) was used to initiate a GET request to the server.

Summarizing, every 20 seconds, a timer scan the sensors, and the readings are transmitted from sensor node to sink node. The data sink node initiates HTTP GET requests at the same rate to update the values to the cloud web server, where the user can access via web or mobile application.

B. Cloud Web Server and Android Application

The HTTP web server was designed with NodeJS in addition to the Express framework. Node.js is an open-source, cross-platform JavaScript runtime environment that can be used for developing a diverse variety of applications and tools. The runtime environment interprets JavaScript using Google's V8 JavaScript engine. Node.js has an event-driven architecture capable of asynchronous I/O which optimizes throughput and scalability. Express, is a web application framework for Node.js. It is relatively easy to build web applications and APIs and has become the de facto standard server framework for Node.js. It is a minimal framework, many of the add-on features can be availed through plugins.

The web server also employs the three core technologies of World Wide Web content production; JavaScript, CSS (Cascading Stylesheet Language), and HTML (HyperText Markup Language). JavaScript was preferred owing to its qualities of an untyped, high-level, dynamic, and an interpreted programming language. Additionally, JavaScript is a prototype-based language that has first-class functions. This makes it a multi-paradigm language, that supports imperative, object-oriented, and functional programming styles.

Figure 7, illustrates the web server algorithm developed. Each block will be described on the following paragraphs.

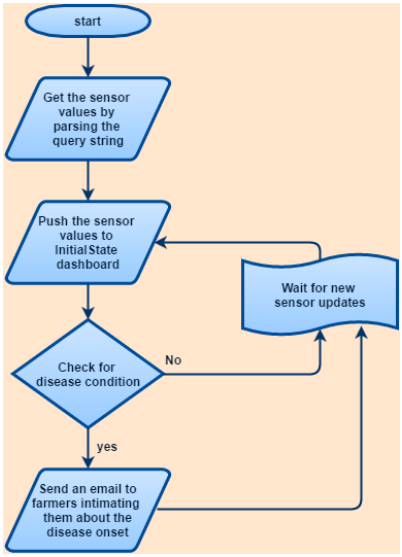

Fig. 7. Web Server Algorithm

Initially, the web server development was done in JetBrains WebStorm Build 2016.3.1 IDE and deployed locally. Later, the web server was pushed to the Cloud9 cloud service. Cloud9 IDE is an online integrated development environment(IDE) that supports hundreds of programming languages, including JavaScript with Node.js. Deploying the server on a cloud service unlocked many advantages. It

circumvented the issue of having a static IP, allowed accessing the dashboard from anywhere, and improved reliability, consistency and manageability of the web server.

The web server provides a URL, Figure 8, at which the GSM node can update the sensor values at a periodic interval. For the server to handle the responses appropriately the interval of sensor readings should be at least 1 second. The URL is composed of the top-level domain(.io), the domain (c9users), the sub domain (wsn-enigmasidd), the port number (8080) and the protocol (http). The query string consists of three key value pairs of temperature, humidity and pressure readings which are separated by ampersand (&). Question mark(?) marks the beginning of the query string.

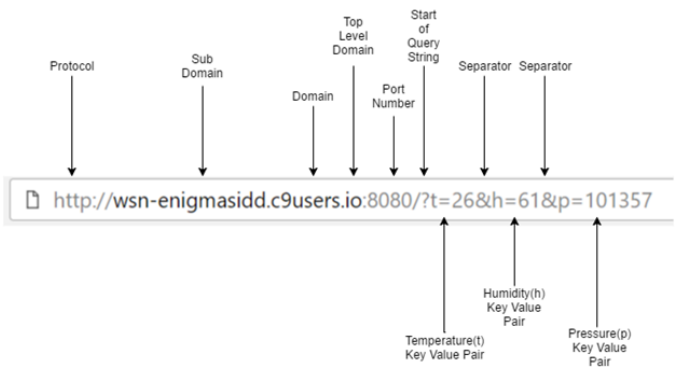

Fig. 8. Web server URL

The GSM module can communicate with the NodeJS server through its RESTful interface. REST stands for REpresentational State Transfer. It is web based architecture that works on the HTTP Protocol. In a REST server, each resource is identified by URIs. GET, POST, PUT, DELETE are the verbs that can be used to interact with the resources. In our scenario, we perform a GET request to the Web Server at the above-mentioned URL and pass the query string along with it. If the values were successfully updated, we get an OK response from the server with an HTTP status code of 200.

Once the web server, figure 9, receives a GET request to update the sensor values the values are sent to InitialState's designated dashboard by using its NodeJS API. To create a dashboard, first a bucket (A container of sensors) is created, and then the user can pass sensor readings as key value pairs. For instance, bucket.push (key, value).

There are a couple of services that are available to handle and visualize IoT data, however, InitialState was chosen for a variety of reasons. It allows dynamic data streaming from sensors which means that there was no need to define data sources. Moreover, this is particularly beneficial in scaling the system to accommodate more sensors. It not only allows visualizing data in real-time, but also allows to store that data forever. The user can view historical data by adjusting the timeline. Additionally, the intuitive visualization tiles are configurable and resizable. Finally, it allows embedding the dashboard into one's webpage and allows to set triggers that can notify the users through SMS. A point to be noted here is that a few of these features are only available with their pro subscription model. More information on pricing tiers and features can be found in the InitialState webpage, [10].

Fig. 9. Web User Interface

Furthermore, the server checks the incoming weather sensor values for conditions of disease and triggers a notification via email and SMS to alert the farmers about the disease onset. The background of the webpage is also modified to indicate the onset of the disease. A condition of temperature > 20° C, pressure > 101500 Pa, humidity > 60% was used for demonstration purpose (A condition that can be created in a room). However, the trigger can easily be modified for a real-world scenario if the system is to be deployed in a field. Further, different triggers can be set for different diseases since each disease may have its own specific weather conditions. To send an email notification, the web server uses Nodemailer v2.7.8 module. Nodemailer is a package that allows to send e-mails from Node.js. We use a Gmail account that is configured to allow less secure apps to send emails on users' behalf. Figure 10 is a screenshot of the customized email notification sent to the farmers.

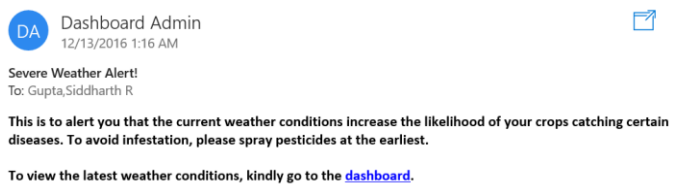

Fig. 10. Notification via email

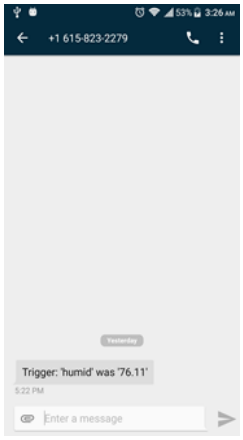

Fig. 11. Notification via SMS

An Android application was developed to adapt and display the webpage with regards to the mobile device layout, Figure 11. The application was coded in Java using the Android Software Development Kit (SDK) in the Android Studio IDE v2.2.2. The application adapts to any screen size and can be installed on any Android device. The application allows the same interaction and features that were available with the web UI.

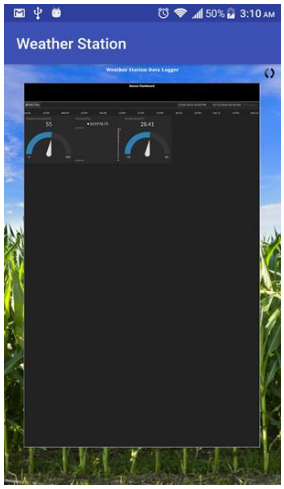

Fig. 12. Android Application

V. Conclusion and Future Work

In this document, a GSM based data logger with sensor Bluetooth network was presented along with cloud web server and an Android application. Battery consumption needs are yet to be explored since the Arduino boards were powered using a USB interface. Additionally, the number of sensors needs may be increased, and the number of Bluetooth nodes may be expanded. Scalability and diagnostic tools need to be added into the system to allow the user to add, remove, and test eventual nodes on the network.

References

[1] Pavan, W., C.W. Fraisse, N.A. Peres. 2011. Development of a web-based disease forecasting system for strawberries. Computers and Electronics in Agriculture 75(1):169-175.

[2] D. F. G. Junco, D. F. Díaz Caro, M. S. C. Forero and I. A. R. Ruge, "Agrometeorological monitoring station based microcontroller and bluetooth communication," 2015 IEEE 2nd Colombian Conference on Automatic Control (CCAC), Manizales, 2015, pp. 1-

[3] M. F. Hunar et al., "GSM wireless datalogger of small hydro power generation system," 2014 4th International Conference on Engineering Technology and Technopreneuship (ICE2T), Kuala Lumpur, 2014, pp. 246-251.

[4] S. Rosiek, F.J. Batlles, A microcontroller-based data-acquisition system for meteorological station monitoring, Energy Conversion and Management, Volume 49, Issue 12, December 2008, Pages 3746-3754

[5] R. V. Sharan 'Development of a Remote Automatic Weather Station with a PC-based Data Logger' by

[6] Arduino Board <https://www.arduino.cc/en/Main/ArduinoBoardUno> accessed in November 12th, 2016

[7] GPRSShield<https://www.seeedstudio.com/GPRS-Shield-V3.0-p-2333.html> accessed in November 12th, 2016

[8] Bluetooth<https://www.amazon.com/Bluetooth-converter-serial-communication-master/dp/B008AVPE6Q> accessed in November 12th, 2016

[9] Sparkfun weather shield <https://www.sparkfun.com/products/12081>

[10] Initial state pricing tiers and features <http://support.initialstate.com/knowledgebase/articles/747096-pricing-tiers> accessed in December 3rd, 2016