NAME: NWUDU OKECHUKWU JEREMIAH

MATRIC NO: 18/ENG04/055

DEPARTMENT: ELECTRICAL/ELECTRONICS ENGINEERING

COURSE CODE: ENG224

COURSE TITLE: STRUCTURED COMPUTER PROGRAMMING

A web based wireless sensor network application using smartphones

SDLC (Software Development Life Cycle) is a life cycle through which a software goes, till it is fully developed and deployed.

It has following Phases:

1. REQUIREMENT GATHERING

A wireless sensor network (WSN) would be used to diagnose the possibly infected users, and Google Maps Web service is used to provide the geographic positioning system (GPS)-based risk assessment to prevent the outbreak. This application would be used to represent each Covid-19 infected user on the Google map that helps the government healthcare authorities to control such risk-prone areas effectively and efficiently. There are a great number of wireless sensor systems available for use in many diverse scenarios. For this project we have implemented a WSN comprised of environmental sensing hardware components and utilizes TinyOS for operation. The TinyOS architecture is event based and has an extremely low system overhead and power consumption. It also has no process management and no dynamic memory allocation, instead allowing only one process at a time and allocating memory at compile time which reduces the strain on the system as well as reducing the amount of used memory. All of the strengths of TinyOS, made it the suitable candidate for this project.

SOFTWARE FEATURES

- IEEE 802.15.4/ZigBee compliant RF transceiver
- 2.4 to 2.4835 GHz, a globally compatible ISM band
- Direct sequence spread spectrum radio which is resistant to RF interference and provides inherent data security
- 250 kbps data rate
- Runs TinyOS 1.1.7 and higher, including a reliable mesh networking stack software module
- Plug and play with all sensor boards, data acquisition boards, gateways, and software Measurement System
- Designed specifically for deeply embedded sensor networks
- Wireless communications with every node as router capability
- Expansion connector for light, temperature, barometric pressure, acceleration/seismic, acoustic, magnetic and other sensor boards

Possible sensor applications:

- Indoor building monitoring and security
- Acoustic, video, vibration and other high-speed sensor data
- Large scale sensor networks (1000+ sensing points)
- ZigBee compliant systems and sensors

HARDWARE FEATURES

To prepare the hardware for use in this project several steps need to be implemented and they are detailed as follows:

Smartphone:

- Needs to be running the Android operating system version 1.6 or later
- Needs to have its wireless capability enabled prior to running the application and be successfully connected to the router or access point of the WSN base station it will monitor
- Needs to have the option ‘install third party applications’ enabled in the phones settings
- Needs to have ample storage for the .apk file of the application, which in our case is approximately 4MB
- Needs to have the application successfully installed

WSN:

- Sensor nodes need to have the appropriate compiled and uploaded source code
- Nodes must have adequate power resources
- The base station must have the appropriate compiled and uploaded source code
- The base station needs to be connected to a gateway; in this case I’ll use the MIB600 Ethernet gateway
- The gateway needs to be connected to a router or access point
- The router or access point needs to be connected to the Internet

Data base:

- Needs to be connected to the Internet

2. DESIGN PHASE

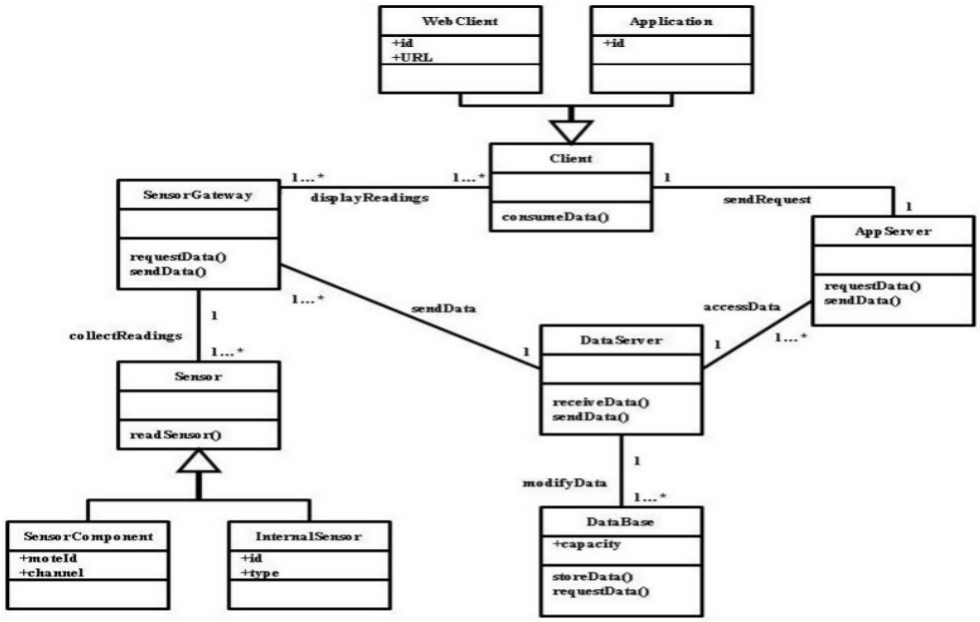

FIG 1: Top down design approach of the Web based wireless sensor network application

The figure above shows the diagram for the web based WSN monitoring architecture. For data acquisition, Sensors use a Sensor Gateway (e.g. smartphone, router, etc.) to transmit data over the Internet to a Data Server where data is stored in a Database, from where it can be accessed later, remotely. Sensors that produce data can be Sensor Nodes (e.g. environmental sensors) or they can be Internal Sensors (e.g. smartphone internal sensors). For data consumption, the Client needs to be running a custom application or a web application (e.g. Web Client) that allows them connect over the Internet to the data Server that stores the data. The data Server is running a server application (e.g. AppServer) and it serves client requests by querying the Database. The Client can be for example a smartphone using a web browser application to connect remotely to the data Server using the phones cellular infrastructure. In this case the data Server is using an AppServer to interface with the Web Client application.

3. DEVELOPMENT PHASE

For this project we would be using a medical sensor that detects viral infections. These sensor nodes would be pre-programmed to start collecting information once they are installed. The access to the sensor data is achieved through a Sensor Gateway. The communication between sensor nodes and the Sensor Gateway is done either directly (if the Sensor Gateway is in range) or can be multi hop, using well known protocols such as LEACH or Directed Diffusion. Sensor Node Pattern and Sensor Network Architecture Pattern show details of the implementation of a sensor node and sensor network, respectively. The communication between the Sensors and Sensor Gateway can be done using various wireless standards, such as Bluetooth or ZigBee. Bluetooth is an industry specification for short-range radio connectivity between portable devices. It operates in the 2.4 GHz ISM band and may support up to 3 Mb/s in the enhanced data 60 rate mode with 10-100 m transmission range. The basic piconet configuration is a star topology network with one master and seven slave devices. The Sensor Gateway can be for example a smartphone using Google's Android operating system, which offers open interfaces, availability of quality tools, documentation, and an open source community-based development support. The smartphone has several internal sensors such as 3D accelerometer, digital camera, GPS, and a Bluetooth transceiver. The Sensor Gateway can run an application to process and filter data. Various languages can be used, such as Java with Android SDK. From the Sensor Gateway, sensed data and status are transmitted to the data Server over the Internet, using a cellular network (3G/4G) or a residential/business Internet connection using the phone's wi-fi card. The upload data rate for the current 3G technology exceeds 100 kbps in most coverage areas. Data stored in a database is accessed by clients using web-based protocols (HTTP). The type and nature of the application determines the data type and how frequent packets are transmitted to the data Server. The architecture uses web-based applications. The data acquisition service used for moving sensor data from the wireless network to servers for storage and analysis are technologies such as JavaScript/ Java/ AJAX/ Google APIs are used on the client side and Java/ XML/ PHP on the server side. The Web services approach is the preferred way to build modern distributed applications over the Internet as they simplify the design, integration, and deployment. Data integration is a key feature, i.e. the ability to pull in data from various sources to build complex web applications with plug-and-play components. The web services can be implemented as Java EE servlets or PHP scripts executed on the web server. Two important operations are data acquisition and data consumption. Data acquisition web services receive the flow of sensor measurements coming over the Internet from the sensor Gateway application and save them to a database. Sensor data consumption service allows access and query to sensor data for the clients. The server (or group of servers for reliability), must be connected to the Internet, protected

by a firewall. The server runs a web services framework, such as Apache Tomcat, and it has a relational database, e.g. MySQL database that stores sensor data and executes background programs for analysis on the current sensor data in the database. Access to the database content must be secured with HTTPS and secured authentication, and only authorized users may be given access permission.

4.TESTING

The application is designed to access a local WSN sink or base station via a TCP connection with two main data flows. The first is used to retrieve, display, and post the readings acquired from the smartphones onboard sensing components to the database. The second flow is used for the WSN data. It connects, listens for messages, and posts network data to the database.

The main flow of events, once the application is started, initializes all the necessary components. If the Wi-Fi connection to the WSN is successful, the application requests permission to access the network data using TinyOS‟s Phoenix Source class. This class provides the automatic reading and dispatching of network data packets (e.g.PacketListener), which is needed to access the sensor data. The MoteIF would be used to build upon the PhoenixSource class and provide a Java interface with which to send and receive messages through a TCP connection. As soon as an instance of the MoteIF is created, the Message Listener objects which is invoked when new messages arrive to the base station.

Upon receipt of a new message, the application buffers the contents into an array. This buffer contains important information like the network id, source id, and the values of the sensor readings. From here the program parses the array into sections, each of which contains the value of one of the attributes previously located in the buffer contents. Once parsed, the application displays on the user interface – the data in usable format for the smartphone client. Simultaneously it sends portions of this parsed data, via its Wi-Fi connection with the Internet, to the data base. It does this by creating a JSON object with the array of measurements, including the network and sensor component values and system time in milliseconds.

Upon receiving the posted JSON object from the application, the database further processes this data and displays it on a password protected web page. The data is displayed in both graphical and textual formats. the graphical representations created in HTML5 and using the Google Chart API. This information is also stored locally for future access and only used by an authorized personnel.

The flowchart below is used for the WSN data. It connects, listens for messages, and posts network data to the database.

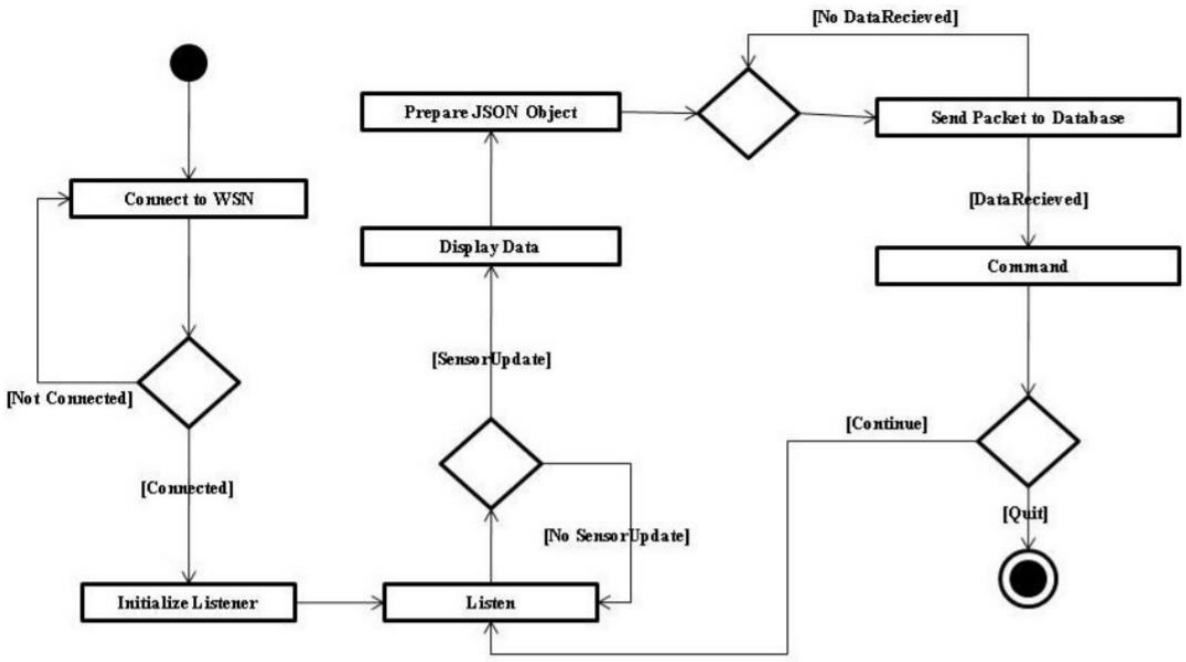

FIG: 2 Activity diagram showing the logic for the WSN

5. DEPLOYMENT

Using the web-based WSN monitoring architecture, individuals are able to track their health status especially patients with chronic respiratory problems changing over time and provide information to authorized medical professionals, family, and caretakers to assess patient's progress. If the patient is using a smartphone, then internal sensors such as acceleration and GPS can be used to monitor the patient's mobility. The smartphone Sensor Gateway component can upload the collected mobility information and location to a server allowing remote access and monitoring for family and caregivers with Internet connection.

The performance and accuracy of the proposed system would be evaluated using dataset for 2 million users. The system would provide high accuracy for initial diagnosis of different users according to their symptoms and appropriate GPS-based risk assessment.