Visual Design Platform for Wireless Sensor Network

Rosen Silva, Asela Dasanayaka, Roshan Ragel, Asitha Bandaranayake
Department of Computer Engineering, University of Peradeniya
Peradeniya 20400, Sri Lanka
rosensilva@eng.pdn.ac.lk, asela.d@eng.pdn.ac.lk, ragelrg@gmail.com, asithab@gmail.com

***Abstract*—Wireless Sensor Networks (WSN) are being widely used for sensing physical parameters in a broad geographical area. The person who needs WSN will have a pictorial idea of the sensor network. The problem in the traditional method is that the person who needs the WSN should explain the pictorial view of the sensor network to a commercial vendor and buy it from them or they should design it from the scratch. What we proposed in our solution is to develop a platform so that the person who needs the WSN can directly draw the pictorial view on a canvas and then it can automatically generate all the required firmware for the microcontrollers and wiring diagrams. The user is then required only to follow a few instructions to complete the real world implementation of WSNs. This paper is about developing a visual platform to design WSNs. The WSN designing platform was built as a web application, so it can manage a large number of supported sensors and microcontrollers. This means that if one user adds the device driver for any sensor or a microcontroller the other users can directly use it from the WSN design platform without worrying about hardware programming. Further, if anyone needs a new sensor or microcontroller to be supported by this visual design platform, this design tool will have interfaces to directly add new sensors and microcontrollers. The proposed method is affordable for developing custom wireless sensor networks.**

Keywords*—*Micro-controllers; Sensors; Visual WSN design platform; Wireless Sensor Networks

I. Introduction

A Wireless Sensor Network (WSN) [1] is a densely deployed wireless network of small, inexpensive, low-power wireless sensor nodes designed to monitor given phenomena. Wireless Sensor Nodes are devices where you can plug various sensors and transmit the sensor data to a WSN gateway using wireless communication. Sensor networks are extremely useful in, agriculture sector and animal husbandry, air and water quality measurement, wildlife movement tracking and disaster management etc.
It is a fact that sensor networks are really helpful in creating a better human life. Most of the time, the required number of sensor nodes for a given phenomenon is extensive. Given the wireless nature of sensor nodes, they should be powered by batteries. The WSN should have low-power, low-cost and reliable sensor nodes and also protocols that would make the communication between sensor nodes more efficient for both power and computation.
The major problem in the traditional wireless sensor network designing process is that there are no common platforms available to design a wireless sensor network. The term 'common platform' means that a platform which is compatible with any commercially available sensors and microcontrollers. Since there are no common designing platforms available to design sensor networks, it takes a huge amount of time and money to develop a wireless sensor network. Also, there is no way of reusing the existing WSNs. The goal of developing the WSN design platform is to let users easily design custom WSN using a graphical user interface. The wireless sensor network designing platform was designed as a web application so that users can simply log into the web application and use this platform to design the required sensor network. After designing the network, users are provided with automatically generated firmware and instruction to deploy the actual sensor network. Anyone with little computer knowledge can develop a simple wireless sensor network, while professional designers can use this platform to design complex wireless sensor networks with lesser complexity.

II. Related work

WSNs are widely used during last decade for sensing physical parameters and they are really useful in our day-to-day life. Since it has a history of past decade, lots of research have been conducted to find out how to effectively design WSNs.

A. *Other available WSN designing platforms*

IBM Node-RED is a programming tool for wiring together hardware devices [2]. Although this tool is open-source it has limited compatibility with changing firmware of micro-controllers according to users requirement. IBM Node-RED focuses more on integration of web services using visual programming tool.

'Firefly' is a cross-layer platform for WSNs, one of such monitoring platform for sensor networks [3]. Although this platform is commercially available, this platform is limited scalability since the platform is not open for any hardware component available in the market.
Proposed WSN design platform is capable of handling any sensor node hardware if one could write a software device descriptor for the WSN design platform.

B. *Related firmware*

A firmware is a software that runs on the microcontroller of the sensor node. To develop a low-powered low-cost sensor node it is a primary requirement to develop a stable power efficient firmware. Previous researches have been conducted to develop the firmware that runs on sensor node which is power efficient for battery-powered sensor nodes. One such operating system is TinyOS, which is a specialized component model exploits advanced compiler technology to simultaneously provide efficiency and reliability in microcontroller [4]. Nano-RK

[5] is a reservation-based real-time operating system (RTOS) with multi-hop networking support for use in wireless sensor networks. Arduino Service Interface Programming (ASIP) [6] model is a Service abstraction to easily add new capabilities to microcontrollers, including socket connections, bridging devices, MQTT-based publish-subscribe messaging, discovery services, etc. The WSN design platform was proposed for auto-generating firmware for the nodes. The auto-generated firmware codes are generated with the capability of managing power in the microcontroller efficiently.

III. Methodological Approach to Develop WSN Designing Platform

This paper is about developing a WSN designing platform, the platform was designed to connect any sensor to a supported microcontroller. To achieve such target the platform should be able to manage the details about sensors and microcontrollers. The following subtopics are the main design submodules in the design, following sections will explain how this platform works with any sensor that is supported by a specific microcontroller. The following subtopics will briefly explain the top level view of the whole project.

A. Managing sensors

Since this paper describes creating a common platform to develop wireless sensor networks, it should support a wide range of sensors and microcontrollers. The basic idea here is to maintain a database of sensors descriptors. The WSN design platform was designed to auto-generate all the required firmware for each node in the whole wireless sensor network. In this process, the database entry of a sensor is used to auto-generate the firmware. Users can easily browse through the available sensors and choose one. If the specific sensor is not found, they can simply add a new entry to the database.

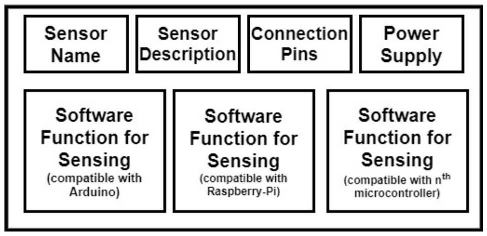

Sensor Name
Sensor Description
Connection Pins
Power Supply
Software Function for Sensing (compatible with Arduino)
Software Function for Sensing (compatible with Raspberry-Pi)
Software Function for Sensing (compatible with n^th microcontroller)

Fig. 1. Database entry for a sensor.

WSN design platform contains interfaces to add new sensors through this WSN design platform. So each new entry of database should contain the required details about the sensors. Fig. 1 will illustrate about the details of database entry which is described in section IV.A.

The database entry for a new sensor can be divided into two parts for ease of understanding. First, the basic information about sensors such as voltage, current, and pins needed to communicate. The second part is the software function which is the most important part when it comes to generalizing the platform to work with almost any sensor. For achieving such a task standard procedures should be used when writing the functions. Basically, the users are advised to write the software functions which deals with the sensor hardware and returning the sensor data according to the predefined format. The most important thing to note is that this database entry can be used by the rest of the world even without knowing how the programming of the sensor with the microcontroller works.

B. Managing microcontrollers

Microcontrollers are acting two roles when it comes to Wireless sensor networks, first as sensor nodes and secondly as WSN gateway. In this WSN design platform, it supports both the settings. The design platform contains a database which stores the information about the supporting microcontrollers, so the users can browse through the available microcontrollers and choose the required microcontroller and start creating a sensor node or a WSN gateway. If the microcontroller that user requires is not found in the existing database they can create a new database entry for the specific microcontroller. Fig. 2 illustrates the structure of the database entry which is described in section IV.B.

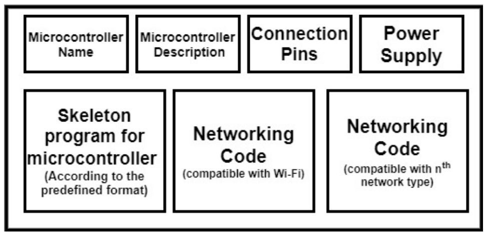

Microcontroller Name
Microcontroller Description
Connection Pins
Power Supply
Skeleton program for microcontroller (According to the predefined format)
Networking Code (compatible with Wi-Fi)
Networking Code (compatible with n^th network type)

Fig. 2. Database entry for a microcontroller.

C. Configuring networking

The networking between the sensor nodes and WSN gateways can be configured dynamically using our WSN design platform. The users can drag and drop the created sensor nodes and WSN gateways to the canvas and build the network. The users are required to draw lines as they want between the nodes that are to be connected. The back-end of the WSN design platform will be responsible for creating the firmware according to the user configurations. The back-end of the WSN design platform will validate each and every connection that is created and it will notify the user if anything is wrong, otherwise, the backend will automatically generate the firmware using the user's configuration. Finally, after user finalizes the WSN network it will make the generated code available for download.

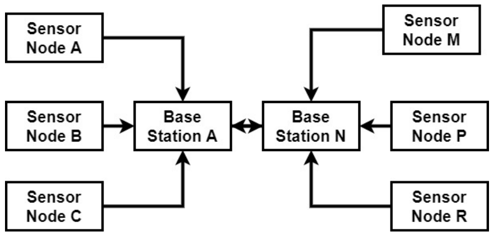


Fig. 3. Sample network configuration.

Fig. 3 shows a diagram of sample WSN. Users are expected to draw a WSN configuration like in Fig. 3 in order to create a WSN.

D. *The automatic firmware code generation*

The core idea of this project is that this WSN design platform will generate all the firmware codes that are needed to deploy the wireless sensor network. Also, the WSN design platform will provide the wiring diagrams that are needed to deploy the WSN. The firmware at each node is generated step by step. The auto firmware generation starts while creating the sensor node. Then it aggregates all the networking configuration. Fig. 4 explains the aggregation of the firmware during the design process of the WSN.

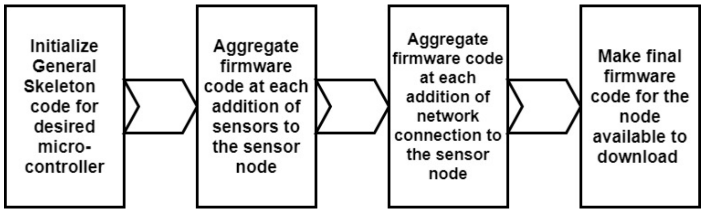

Initialize General Skeleton code for desired micro-controller
Aggregate firmware code at each addition of sensors to the sensor node
Aggregate firmware code at each addition of network connection to the sensor node
Make final firmware code for the node available to download

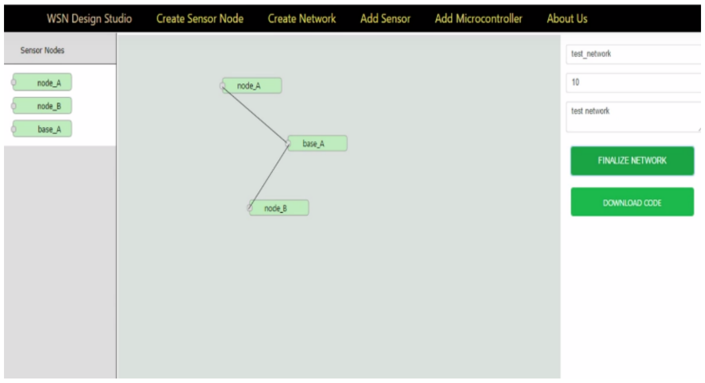
Fig. 4. Automatic firmware code generation.

E. *Upload the firmware to the micro-controller*

Finally, the users will be able to download the firmware. Since the firmware uploading is not uniform across all the micro-controllers we ask users to manually upload the firmware to the micro-controller

IV. Implementation

Fig. 5 is a screenshot of the user interface (UI) of the design platform. The front-end was designed to send a request to the back-end to validating each sensor connection. The users can work on the same UI from the starting of designing the sensor node to deploying the final design of wireless sensor network.



test_network

10

test network

Fig. 5. User interface.

A. *Implementation of adding new sensor feature*

As explained in the Methodology section III.B each sensor will have a database entry. The details of the sensor database entry will be discussed in the following section. If a sensor is not already registered in the database, users can add a new entry to the database. The steps of adding a new sensor to the WSN design platform is illustrated in Fig. 6. If a user fills the following details in order as in Fig. 6 correctly and saves the sensor, then that sensor will be added to the WSN designing platform.

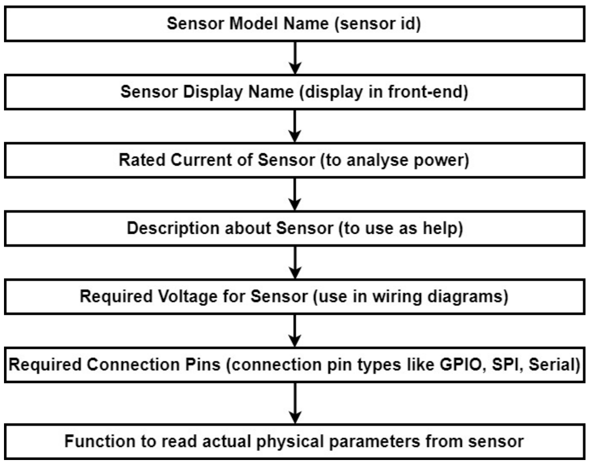

Sensor Model Name (sensor id)
Sensor Display Name (display in front-end)
Rated Current of Sensor (to analyse power)
Description about Sensor (to use as help)
Required Voltage for Sensor (use in wiring diagrams)
Required Connection Pins (connection pin types like GPIO, SPI, Serial)
Function to read actual physical parameters from sensor

Fig. 6. Requirements for adding new sensor.

All the detailed requirements will be available in the application itself. The users should pursue the following standards,

- Include libraries that are needed should be uploaded with the names of the libraries.
- If there is a requirement to initialize the sensor, that code should enter in the given field while adding the new sensor.
- Should enter a function(s) for each microcontroller which measures and output the sensor values in 'String' data type (If there are multiple outputs, the function should return all the outputs separated by '&&' sign).
- The user may input multiple functions for microcontrollers as per the requirement.

B. *Implementation of adding new microcontroller feature*

WSN design platform supports Arduino, ESP8266-12E and Raspberry Pi boards by default, which means that the WSN design platform already contains three entries in the microcontrollers' database. If a user needs to integrate a new microcontroller with the WSN design platform, the user can use the built-in feature to add a new microcontroller to the design platform.
To register a new microcontroller with the WSN design platform users should fill the following requirements illustrated in Fig. 7.The last two steps in Fig. 7 will further discuss in section IV.B.1 and IV.B.2 respectively.

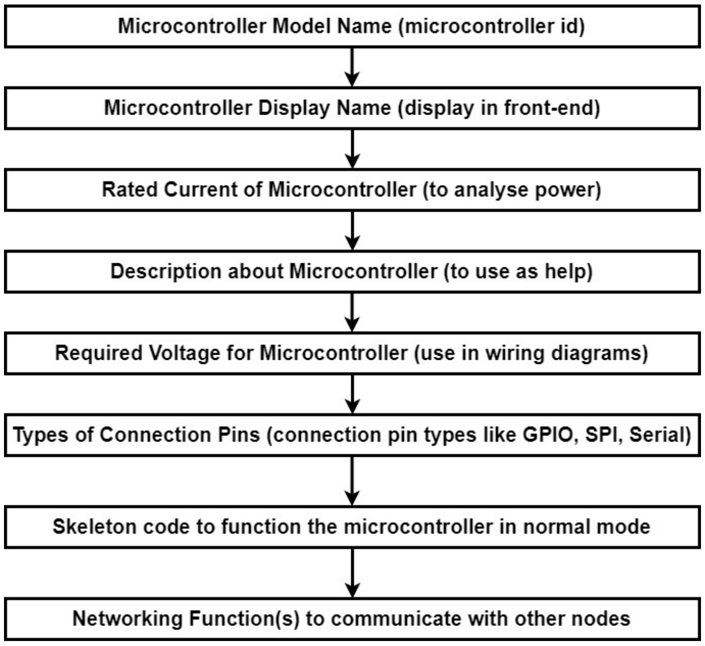
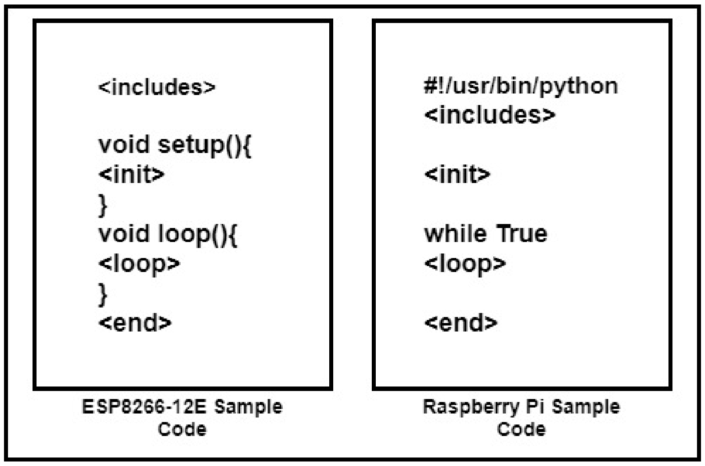

Microcontroller Model Name (microcontroller id)
Microcontroller Display Name (display in front-end)
Rated Current of Microcontroller (to analyse power)
Description about Microcontroller (to use as help)
Required Voltage for Microcontroller (use in wiring diagrams)
Types of Connection Pins (connection pin types like GPIO, SPI, Serial)
Skeleton code to function the microcontroller in normal mode
Networking Function(s) to communicate with other nodes

Fig. 7. Requirements for adding new microcontroller.

1) The following standards should be followed by the users while writing skeleton code for microcontrollers: Skeleton code for the microcontroller is the most important part of the microcontroller database entry. This skeleton code should contain the following tags in the relevant places,

1) <includes> - Where to insert the other 3rd party libraries that are needed for the functionality of the sensors and microcontroller.
2) <init> - Where the initialization codes should reside.
3) <loop> - Location of the main loop.
4) <end> - Where the end code locate.

Without these tags, the skeleton code should compile and run on the relevant microcontroller and it should run in an infinite loop without any functionality. Fig. 8 shows the examples written for ESP8266-12E and Raspberry Pi.



```
<includes>

void setup(){
<init>
}
void loop(){
<loop>
}
<end>
```

ESP8266-12E Sample Code

```
#!/usr/bin/python
<includes>

<init>

while True
<loop>

<end>
```

Raspberry Pi Sample Code

Fig. 8. Sample code configuration.

2) Networking function(s) that is written to connect with different types of networking technologies: Network function should be written using three parts

1) Network initialization code - this code should run while the microcontroller is at initialization stage and it should basically connect to the given network.
2) Data sending function - this code should run in the loop and should be capable of sending the given string through the network.
3) Data receiving function - This code should also run in the loop and should be capable of receiving any String through the network.

Note that if the user needs another type of networking technology (for example if the user wants to use RF communication instead of WiFi), the user should write the network function separately and insert it to the WSN design platform while adding the new microcontroller.

C. Implementation of automatic code generation module

This is the most important module of the WSN design platform, this module handles the logic of generating the firmware including all the user configurations. The following steps are the process that generates the firmware codes that are required to generate the WSN.

1) Initializing the node firmware: Sensor nodes usually contain microcontroller and sensors. If a user drag and drop a microcontroller into the sensor node designing canvas, then the automatic code generation module will call the database and retrieve the skeleton code. Then the skeleton code is loaded into the memory.

2) Developing the firmware when a new sensor is added: Since there is a microcontroller in the sensor node designing canvas, it is already loaded with the skeleton code in the memory. As user keep on adding new sensors the code generator will aggregate the firmware by adding each sensor to the firmware of the node. This procedure is illustrated in Fig. 9.

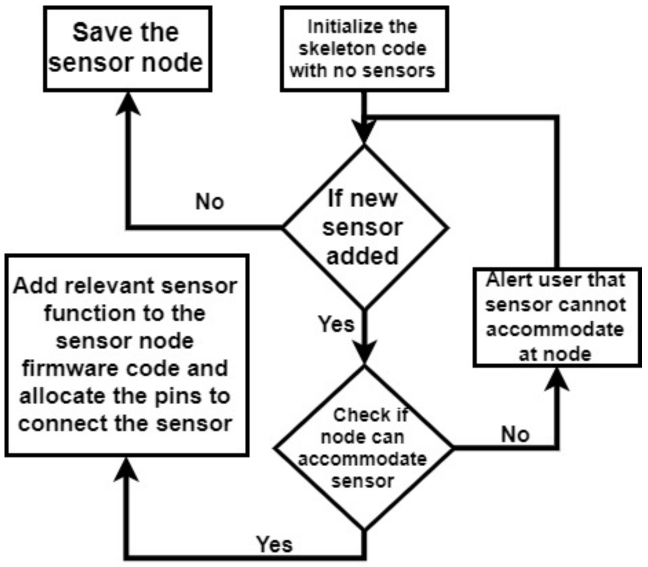

Save the sensor node
Initialize the skeleton code with no sensors
No
If new sensor added
Yes
Add relevant sensor function to the sensor node firmware code and allocate the pins to connect the sensor
Alert user that sensor cannot accommodate at node
Check if node can accommodate sensor
No
Yes

Fig. 9. Code generation procedure.

As shown in Fig. 9, there is a step for validating whether each sensor can be really accommodated at the node and it is a really interesting problem to solve. A mathematical approach was used to solve this problem. The mathematical solution will be scalable for any microcontroller and also for any sensor. Also, the wiring diagrams can be obtained from this mathematical method. Basically, there is a matrix generated each time when a sensor node is initialized, that matrix contains the available number of pins in different categories. Initially, the matrix would be the same as the available pins in the microcontroller, then while adding each sensor the main matrix is subtracted with the sensor's required pins matrix. If any column gives a negative number, then it is clear that the sensor cannot be accommodated at the microcontroller. If both matrices subtract without giving a negative number, it means that the sensor can be accommodated. Fig. 10 shows a simple example of successful and an unsuccessful sensor addition.

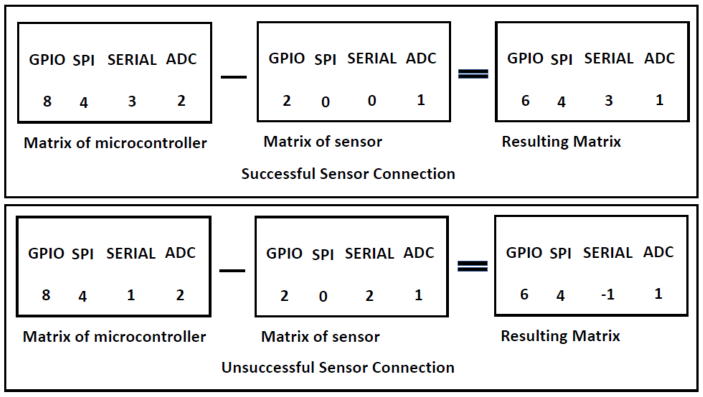


Fig. 10. Sensor addition example.

3) Generate node firmware for the networking: After the user created a Wireless Sensor Node, the memory already has the firmware with sensor functions. Now the user will able to design a network between sensor nodes and WSN gateways, for that user can use the already created sensor nodes. The user is expected to drag and drop the created sensor nodes into the canvas and design the network by drawing lines in between them. While the user keeps on dragging and dropping the nodes into the canvas and connecting them with wires, the program backend will validate each network connection and keep aggregating the code with the network configuration setting at the firmware code. Fig. 11 shows a networking setup for simple sensor network with four nodes.

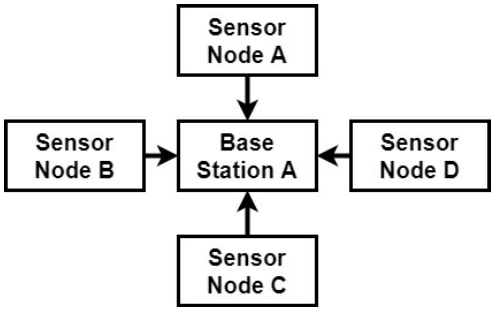


Fig. 11. Networking setup.

V. Results and Analysis

The auto-generated firmware codes will be discussed in the results and analysis section. Table I shows the sensors that are supported by the WSN design platform by default.

TABLE I. Summary for sensors

Sensor name	Hardware interface	Supported micro-controllers
DS18B20	GPIO	ESP8266, Arduino, Raspberry Pi
LIDAR-Lite v3	I2C, PWM	Arduino, Raspberry Pi
HX711	GPIO pins	Arduino, Raspberry PI, esp8266
MPU-6050	I2C	Arduino, Raspberry PI, esp8266
DHT-11	GPIO	Arduino, Raspberry PI, esp8266

A. Results

An experiment was carried out for measuring humidity using a WSN. The firmware for sensor node with DH11 humidity sensor was generated from WSN design platform. ESP8266 was used as the microcontroller for this experiment. The user is expected not to write any code but, draw the network configuration in the WSN design tool canvas. The network was designed using WiFi connections. Then the auto-generated firmware was then compiled and loaded into the microcontroller.

B. Analysis

The automatic code generation module uses the database entries of each sensor that is being used. The functions that are written for each sensor contains XML tags. The connection pins are not hard-coded in the sensor function (in the skeleton sensor function the pins are assigned as <DIGITALINPUT1>, <DIGITAL-INPUT2>, <ANALOG-INPUT1>). These tags are replaced by the automatic code generation module. Fig. 12 illustrates the macros that were generated by the WSN design platform. From this method, it is possible to make any sensor works with any supported micro-controller

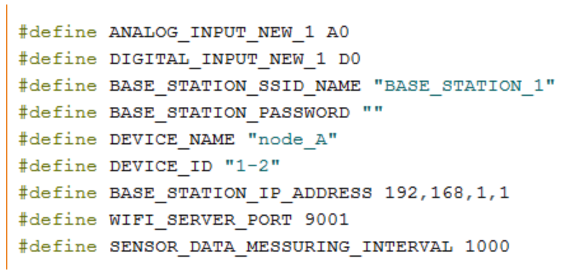
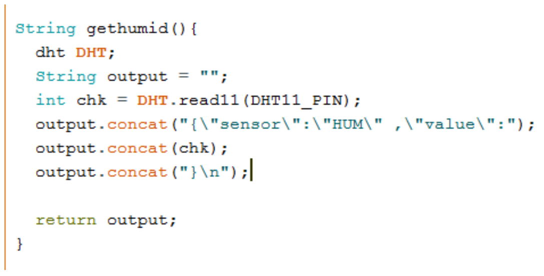

```
#define ANALOG_INPUT_NEW_1 A0
#define DIGITAL_INPUT_NEW_1 D0
#define BASE_STATION_SSID_NAME "BASE_STATION_1"
#define BASE_STATION_PASSWORD ""
#define DEVICE_NAME "node_A"
#define DEVICE_ID "1-2"
#define BASE_STATION_IP_ADDRESS 192,168,1,1
#define WIFI_SERVER_PORT 9001
#define SENSOR_DATA_MESSURING_INTERVAL 1000
```

Fig. 12. Auto-generated defines.

Fig. 12 shows how the tagged pins are replaced using real pins (D0, A0, and D1). Fig. 13 shows how a function from database entry (DH11 humidity sensor) is used to generate the firmware code for the sensor node. Fig. 14 shows an auto-generated loop code with the tag of <SENSOR DATA MEASURING INTERVAL>. The data measuring interval will directly replace according to the user's preferences.

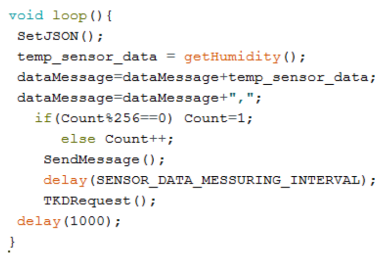


```
String gethumid(){
  dht DHT;
  String output = "";
  int chk = DHT.read11(DHT11_PIN);
  output.concat("{\"sensor\":\"HUM\" ,\"value\":");
  output.concat(chk);
  output.concat("}\n");

  return output;
}
```

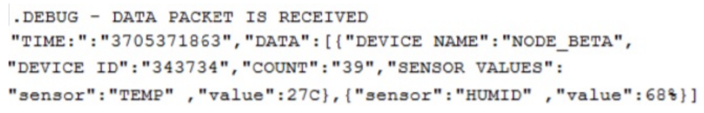
Fig. 13. A sensor function is inserted to firmware.

Also, the function call is done within this loop. Finally, Fig. 15 shows the sample outputs from the WSN. The output format is a standard JSON object. The output JSON object can be used to transfer the sensor data to any remote location for further processing.



```
void loop(){
 SetJSON();
 temp_sensor_data = getHumidity();
 dataMessage=dataMessage+temp_sensor_data;
 dataMessage=dataMessage+",";
   if(Count%256==0) Count=1;
      else Count++;
    SendMessage();
    delay(SENSOR_DATA_MESSURING_INTERVAL);
    TKDRequest();
 delay(1000);
}
```

Fig. 14. Auto-generated code for the main loop.



```
.DEBUG - DATA PACKET IS RECEIVED
"TIME:":"3705371863","DATA":[{"DEVICE NAME":"NODE_BETA",
"DEVICE ID":"343734","COUNT":"39","SENSOR VALUES":
"sensor":"TEMP" ,"value":27C},{"sensor":"HUMID" ,"value":68%}]
```

Fig. 15. Structure of the output data format.

VI. Conclusion

The primary aim of the project was to build a common platform to design and develop WSNs. Although there are a lot of commercial wireless sensor nodes available in the current market, still there is no common platform to connect sensors and microcontrollers through a graphical user interface. This project is developed to make the development of wireless sensor networks easy. This WSN design platform is designed in a way that, anyone with moderate computer literacy can design their own WSN network within a short period of time.

WSN design platform has basically developed as a web application, a web application can be deployed on a server and users can log into the WSN design platform from anywhere and start developing the desired wireless sensor network. Since the web application is centralized, the design platform can increase the number of supporting microcontrollers and sensors as users add new sensors and microcontrollers to the system. The web application has designed using responsive front-end and RESTful backend.

Acknowledgment

This research was supported by Department of Computer Engineering, University of Peradeniya, Sri Lanka. We thank our lecturers and colleagues from the University of Peradeniya, Sri Lanka who provided insight and expertise that greatly assisted the research.

References

[1] C. Guy, "Wireless sensor networks," in *Sixth International Symposium on Instrumentation and Control Technology: Signal Analysis, Measurement Theory, Photo-Electronic Technology, and Artificial Intelligence*, vol. 6357. International Society for Optics and Photonics, 2006, p. 63571I.

[2] A. Node-Red, "visual tool for wiring the internet-of-things," *Spletni vir: http://nodered. org/(zadnji dostop: 21.7. 2016)*, 2016.

[3] R. Mangharam, A. Rowe, and R. Rajkumar, "Firefly: a cross-layer platform for real-time embedded wireless networks," *Real-Time Systems*, vol. 37, no. 3, pp. 183–231, 2007.

[4] J. Hill, M. Horton, R. Kling, and L. Krishnamurthy, "The platforms enabling wireless sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 41–46, 2004.

[5] A. Eswaran, A. Rowe, and R. Rajkumar, "Nano-rk: an energy-aware resource-centric rtos for sensor networks," in *Real-Time Systems Symposium, 2005. RTSS 2005. 26th IEEE International*. IEEE, 2005, pp. 10–pp.

[6] G. Barbon, M. Margolis, F. Palumbo, F. Raimondi, and N. Weldin, "Taking arduino to the internet of things: the asip programming model," *Computer Communications*, vol. 89, pp. 128–140, 2016.