

Design of a SmartMesh IP Network with Wireless Motes and GUI Control

Marc Kamsu¹, Shensheng Tang^{2*}, Yi Zheng³

Abstract

SmartMesh IP is an innovative way to connect smart devices with advanced network management and comprehensive security features. SmartMesh IP delivers reliable, scalable, and energy efficient wireless sensor connectivity. In recent years, SmartMesh IP has become the industry's most energy-efficient wireless mesh sensing technology even in harsh and dynamically changing radio frequency (RF) environments. This paper proposes a SmartMesh IP network system that consists of a network manager and a group of motes. The network manager monitors and manages network performance and security, and exchanges data with a host application. The motes are the wireless nodes that have built-in sensors and can collect and relay data from/to other motes or from/to the network manager. The proposed system is implemented the system through hardware integration, firmware design, and software development, as well as system testing. Through various experiments, the software, firmware, and hardware platforms have been shown to be able to fully configure the SmartMesh IP devices and get them to perform real-world applications such as monitoring the environment temperature at different places within the SmartMesh IP network. By using internal or external temperature sensors, the platforms developed in C# programming have shown to successfully achieve the temperature data collection and the network topology viewing. The temperature data can be monitored by the system dynamically with respect to the real time. The developed network topology platform can be used to visualize the communicating nodes and the links among them in the SmartMesh IP network. The proposed design and implementation method can provide practical reference for the development and industrial application of SmartMesh IP networks.

Keywords: SmartMesh IP, Network manager, Mote, C# programming, Hardware integration, Firmware design, GUI.

INTRODUCTION

*Author for Correspondence

Shensheng Tang

E-mail: stang@stcloudstate.edu

¹M.S. Student, Department of Electrical and Computer Engineering, St Cloud State University, St Cloud, MN 56301, USA

²Associate Professor, Department of Electrical and Computer Engineering, St Cloud State University, St Cloud, MN 56301, USA

³Professor, Department of Electrical and Computer Engineering, St Cloud State University, St Cloud, MN 56301, USA

Received Date: July 24, 2021

Accepted Date: August 15, 2021

Published Date: August 20, 2021

Citation: Marc Kamsu, Shensheng Tang, Yi Zheng. Design of a SmartMesh IP Network with Wireless Motes and GUI Control. Journal of Microcontroller Engineering and Applications. 2021; 8(2): 1–20p.

In recent years, a great deal of research is conducted in a variety of scientific areas such as physics, microelectronics, and material science. The invention of Micro-Electro-Mechanical Systems (MEMS) and the advances of radio frequency (RF) technology have drastically evolved within the silicon industry to the point that lower power and higher frequency transceivers are widely implemented on integrated circuits [1]. In addition, the digital circuits are increasingly shrinking in size and lead to the concept of System-On-a-Chip (SoC) [2]. As these technologies become very popular and widely used, the need for combining the capabilities of sensing, actuation, processing, and communication also grows rapidly and leads to further research of the invention of motes which have the ability of sensing, actuating, processing, and communicating

wirelessly [3]. The wireless communication makes it possible for the motes to form wireless sensor networks (WSNs), which consist of data acquisition part to acquire data from different fields and data distribution part to transmit the acquired data to the base station through different techniques such as WiFi, Bluetooth, and cellular.

The idea of designing and using WSNs originated within academic circles as a novel concept, which resulted into a commercially relevant technology to the point that in 1997, the U.S. Defense Advanced Research Projects Agency (DARPA) funded the Smart Dust project [4]. By the middle of the year 2003, the “Berkeley Mote” was born, which integrated sensors and transmitters onto a platform that measured about 5 square millimeters [5]. Massive numbers of these millimeter-scaled motes could be used in self-organizing WSNs for such innovative applications as monitoring seabird nests in remote habitats, pinpointing structural weaknesses in a building after an earthquake, or warning of the presence of biochemical toxins.

As a response to the success and potential of the Smart Dust design, the IEEE started looking at the opportunity and came out in 2003 with the first version of IEEE 802.15.4 [6]. This standard defines both the physical and medium access control (MAC) layers and it is arguably the standard with the most influence on low-power mesh technology. The IEEE 802.15.4 standard has kept evolving, with revisions in 2006 and 2011. Low-power mesh technology is particularly suitable for industrial process applications, where the cost of connecting wired sensors is a real barrier. Their wireless counterparts, however, need to be able to provide wired-like reliability and multi-year battery lifetime. An early response was the proprietary Time Synchronized Mesh Protocol (TSMP) [7], which enables reliable, low power, secure communication in a managed wireless mesh network. Through time synchronization, motes only turn on their radio when they are scheduled to transmit or receive, resulting in ultra-low-power operation.

Following the success of TSMP, in 2008, its core ideas were standardized in WirelessHART [8], the wireless extension of the longstanding wired HART protocol, ubiquitous in industrial process applications. WirelessHART, which is now instantiated as IEC 62591, uses IEEE 802.15.4-compliant radios. In 2012, the IEEE 802.15.4e amendment [9] was published, which replaced the MAC protocol of IEEE 802.15.4 to adopt the core ideas of time synchronization, time slotting, and channel hopping inherent to TSMP and WirelessHART. The IEEE 802.15.4e-based mesh of nodes or motes have necessary hardware and firmware to form a reliable and secure wireless network with collision-free communication; that is, two pairs of motes never transmit at the same time or at the same frequency—resulting in a potentially full utilization of the available bandwidth. Thus, all the motes in the network are synchronized to the same time base and an end-to-end delivery ratio can be achieved over 99.99%. This is the SmartMesh IP technology [10]—an innovative way to connect smart devices with advanced network management and comprehensive security features.

Not much research about SmartMesh IP is presented in the literature, though much similar research has been done. In [11], the authors deployed two low-power wireless networks by using the SmartMesh IP technology to gather sensor values, one in a Smart Agriculture setting, the other in a Smart Building. The performance of both networks were analyzed and compared. In [12], the authors presented a room-level localization solution built on top of the market leading SmartMesh IP industrial low-power wireless mesh network. They re-purpose the blink feature for localization of mobile nodes without any changes to the hardware or protocol stack of the existing product. Initial results indicated that the solution offers over 90% room-level localization accuracy, and at worst adjacent room localization. In [13], a proof-of-concept localization system called WALLSY (Wireless Ad-hoc Low-power Localization SYstem) was implemented for GNSS (Global Navigation Satellite System)-denied environments. WALLSY uses UWB two way ranging to measure distances, which are then sent via the low power SmartMesh IP backbone network to a central hub for calculating coordinates of tracked objects.

In this paper, we propose a SmartMesh IP network system consisting of a network manager and a group of motes. The system implementation involves hardware integration, firmware design, and software development, as well as final system testing. The system implementation involves hardware integration, firmware design, and software development, as well as final system testing. Through various experiments, Combining the software, firmware, and hardware platforms together, our SmartMesh IP network experimental system have been shown to be able to fully configure the SmartMesh IP devices (i.e., the network manager and motes) and get them to perform real-world applications such as monitoring the environment temperature through internal sensors and external sensors as well as viewing the network topology with dynamic adjustment due to the events of new mote joining and existing mote leaving the network. Experimental results are presented with explanation for further understanding.

The remainder of the paper is organised as follows: Section 2 describes the proposed the temperature monitoring system; Section 3 presents the system design including the hardware design and software flowcharts as well as the protocol design for the communication between the GUI (graphical user interface) and Master module and the communication between the master module and sensor module; Section 4 builds up an experimental system based on the previous sections and demonstrates selected experimental results; and Section 5 concludes the paper.

The SmartMesh IP System Description

SmartMesh IP delivers reliable, scalable, and energy efficient wireless sensor connectivity. SmartMesh IP networks use a “triple-play” of wireless mesh technologies—time diversity, frequency diversity, and physical diversity—to assure reliability, resiliency, scalability, power source flexibility, and ease-of-use. A SmartMesh IP network typically consists of a highly scalable self-forming multi-hop mesh of wireless nodes, known as motes, which collect and relay data, and a network gateway or manager that monitors and manages network performance and security, and exchanges data with a host application, as shown in Figure 1. SmartMesh IP networks provide redundant routing to the network gateway or manager as every node in the mesh network has the same routing capability. Each node can transmit data to and receive data from any other network node that is within the range. Individual nodes make routing decisions based upon nodes’ availability, proximity to the network gateway, and current traffic load. This makes the network self-forming, self-healing and highly scalable as devices can transmit data with all of the networked devices over multiple paths and multiple hops without the need for centralized control or a high-power transmitter/receiver.

SmartMesh IP Network Manager/Gateway

The SmartMesh IP network manager (or gateway) has several roles in the network. The first task is to build a schedule to accommodate the motes’ traffic requirements. At any time, a mote can send a “service request” to the manager; a service request is an application-layer packet which contains a declaration of the amount of data the mote is about to produce [10]. The manager then modifies the network schedule to satisfy this request and notifies the requester that the service was granted. To keep the power consumption of the network as low as possible, it is important that the motes request the service which corresponds to the data they send.

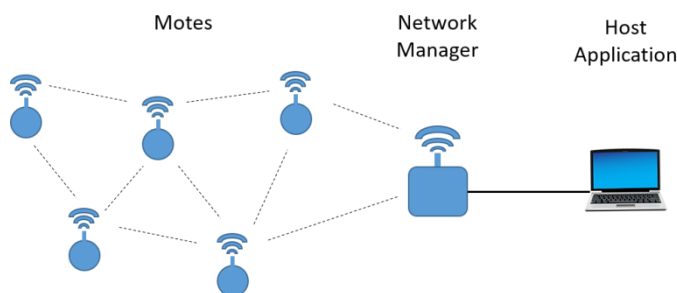


Figure 1. A SmartMesh IP network system with a network manager and various sensor motes.

The second task is to maintain the communication in the network when topological changes happen. Given the nature of wireless, it is normal that some “good” links become “bad” and some “bad” links become “good” [10]. Consequently, each mote in the network periodically sends information containing statistics about the links they are using, and the motes they are not talking to but are in the vicinity. Based on this information, the manager can, for example, change the routing parent of a mote. At all times, all motes in the network should have at least two routing parents, which yields multi-hop path diversity and thus increases the network’s reliability. The manager can also react to infrequent events, such as a new mote joining, a mote disappearing, or a link being broken.

The third task is to connect the SmartMesh IP network to the outside world. The manager has a main interface and a Command Line Interface (CLI). The former one is an Application Programming Interface (API), running on an HDLC (High-Level Data Link Control)-enabled serial port typically connected to a computer application. The latter is a serial port to which a human can connect a computer and execute commands and debugging. Through the API, the computer application can instruct the manager to send a packet to the motes in the network. The manager also uses the API to publish received data and notification of different events in the network.

SmartMesh IP Mote

SmartMesh IP Motes are the wireless nodes in a SmartMesh IP network. They have built-in sensors and can connect to external sensors/actuators and relay data from/to other motes or from/to the network manager, yet remain low power.

Each mote can both send and receive data simultaneously. Each mote may have a different data reporting rate and the network manager can coordinate individual pair-wise communications to efficiently route the traffic. Each mote may have a different power supply capability.

System Construction and Component Selection

To develop a SmartMesh IP network experimental system, we choose the related circuits for the network manager and motes to construct hardware system, then we develop software application programs to work with the hardware. Our goals are as follows:

- Develop a SmartMesh IP based wireless sensor network system including a SmartMesh IP Network Manager and multiple SmartMesh IP motes.
- Develop C# based GUI applications [14] to control and configure the SmartMesh IP Network Manager and to collect sensor information (e.g., temperature data) using open-source drivers.

We choose the DC2274A-A USB demo board [15] to serve as a network manager and the DC9018B-B SmartMesh IP RF Certified Evaluation/Development Mote [16] to serve as the motes for constructing the SmartMesh IP wireless sensor network system, as shown in Figure 2.

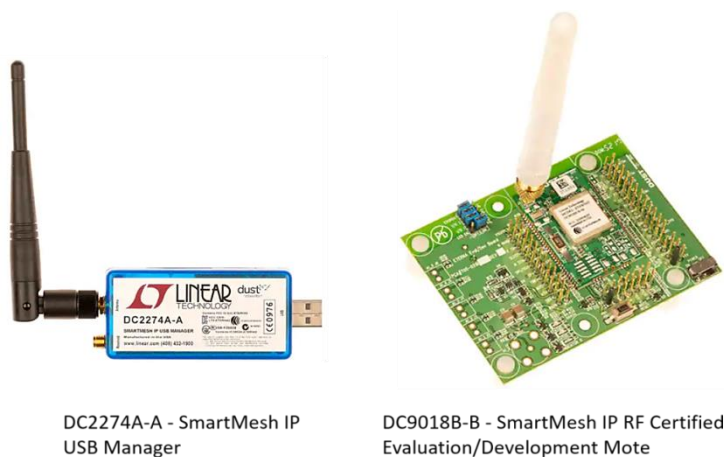


Figure 2. The SmartMesh IP network components (network manager and mote).

The DC2274A-A USB demo board incorporates the LTP5902-IPM SmartMesh IP Module, running EManager firmware with 100-mote network capacity, and provides a USB connection for both power and software access to the manager API UART and CLI UART ports. A DC9018B-B mote incorporates the RF-Certified LTP5902-IPM SmartMesh IP Mote PCB Module (including the MMCX antenna connector) and a CR2032 coin cell. The mote is ultra-low power from the breakthrough Eterna technology (extending battery life to years) and has an advanced array of features. The SmartMesh IP On-Chip SDK (software development kit) is used to gather the real-time temperature data from a temperature sensor. Then, the mote wirelessly transmits that temperature data to either a nearby mote or network manager that monitors and manages network performance and security, and exchanges data with a host application that we plan on designing and implementing within the Microsoft Visual Studio C# integrated development environment (IDE). In addition, a computer is required to run the host application and use the following features: USB capability, Visual Studio, Python [17], and the IAR Embedded Workbench for ARM [18].

To achieve a full SmartMesh IP network, the experimental system may include up to four main communication interfaces as described below.

- **Microprocessor-to-Sensor Interface:** We use the SmartMesh IP Mote's built-in ARM Cortex-M3 microprocessor to perform various sensor-oriented and signal processing data acquisition tasks.
- **Mote-to-Mote Interface:** We have a mote transmit data to another mote over the air through the SmartMesh IP communication protocol in order to prove out the concept of relay transmission in which the data packet from a mote is not directly transmitted to the network manager, but rather first send to another mote, and that latter mote then sends the data to the network manager. This concept is important and necessary for increasing the coverage area of the SmartMesh IP network.
- **Mote-to-Network Manager Interface:** At least one mote sends data directly to the network manager over the air through the SmartMesh IP communication protocol, and vice-versa.
- **Network Manager-to-Computer Interface:** The network manager is connected through USB to a computer which can retrieve all the data received by the network manager and perform various types of signal processing tasks such as result display and data plotting on a GUI.

The SmartMesh IP System Configuration Software

This section describes the design of the SmartMesh IP system configuration platform which can be used to fully configure the SmartMesh IP network manager and the SmartMesh IP mote. This platform is mainly software oriented. On the hardware side, the user simply needs to plug in the SmartMesh IP devices (network manager and mote) to a computer as shown in Figure 3.

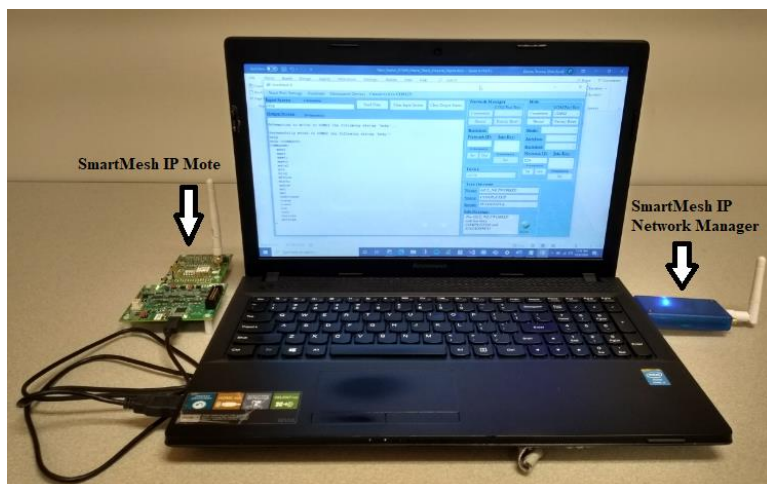


Figure 3. SmartMesh IP Network Manager and Mote system configuration.

On the software side, the SmartMesh IP system configuration GUI program is written in the C# language. It is made up of classes: `Network_Manager_GUI.cs`, `Serial_Port_settings.cs`, `LoadFirmware.cs`, `NetworkManager.cs`, `Mote.cs`, `Applications.cs`, and a few auxiliary classes, where `Network_Manager_GUI.cs` is the main class for system configuration. This program is designed to facilitate the configuration process of the SmartMesh IP network manager and mote. To achieve this facilitation, the GUI is made up of various types of controls which allow the user to perform certain tasks. The configuration software main GUI form is shown in Figure 4. Note that the related firmware programs embedded into the GUI programs. The firmware programs include the existing ones provided when purchasing the hardware circuits and brand new programs that we developed in C programming (however, these are out of the range of the paper). Different GUI configurations are summarized as follows.

Configuration Software CLI Serial Port Settings

To assign the serial/COM CLI port for a SmartMesh IP device, the user can simply click on the PortName box shown in Figure 5 and choose the proper port (here it is COM16) from the list of available ports. Once all the settings have been assigned, the user should press the “Apply Settings” button for the settings to take effect. Once the SmartMesh IP Network Manager is successfully connected to the GUI through serial port communication, it will show the connection on the GUI panel.

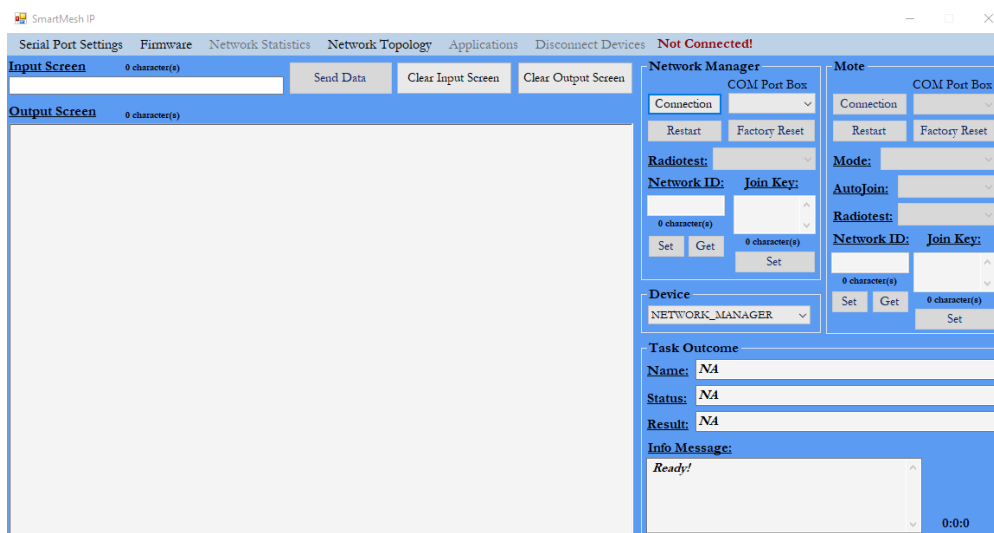


Figure 4. The configuration software main GUI.

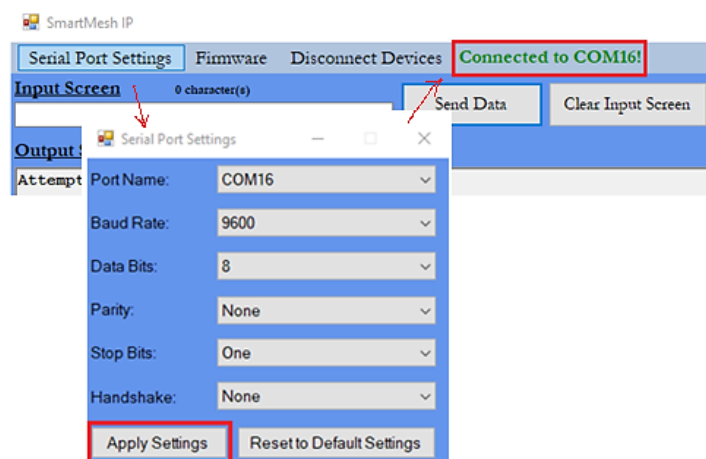


Figure 5. The configuration software main GUI form.

Configuration Software and SmartMesh IP Network Manager Default Firmware Load

The configuration software GUI can be used to load the default firmware on the SmartMesh IP Network Manager. The default firmware resources have been embedded in the configuration software as shown in Figure 6.

Figure 7 shows the steps involved in loading the default firmware on the SmartMesh IP Network Manager. To avoid confusion, before performing these steps, one needs to make sure that the SmartMesh IP Network Manager is the only SmartMesh IP device plugged in to the computer running this default firmware load software.

Configuration Software and SmartMesh IP Mote Default Firmware Load

The configuration software GUI can also be used to load the default firmware on the SmartMesh IP Mote. A similar steps to Figure 7 is involved for the firmware loading and is omitted here. To avoid confusion, before performing these steps, one needs to make sure that the SmartMesh IP Mote is the only SmartMesh IP device plugged in to the computer running this default firmware load software, and also make sure that the SmartMesh IP Mote's power switch is set to ON.

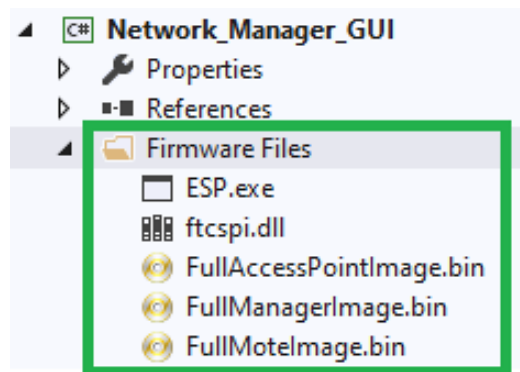


Figure 6. Default firmware files embedded in the configuration software.

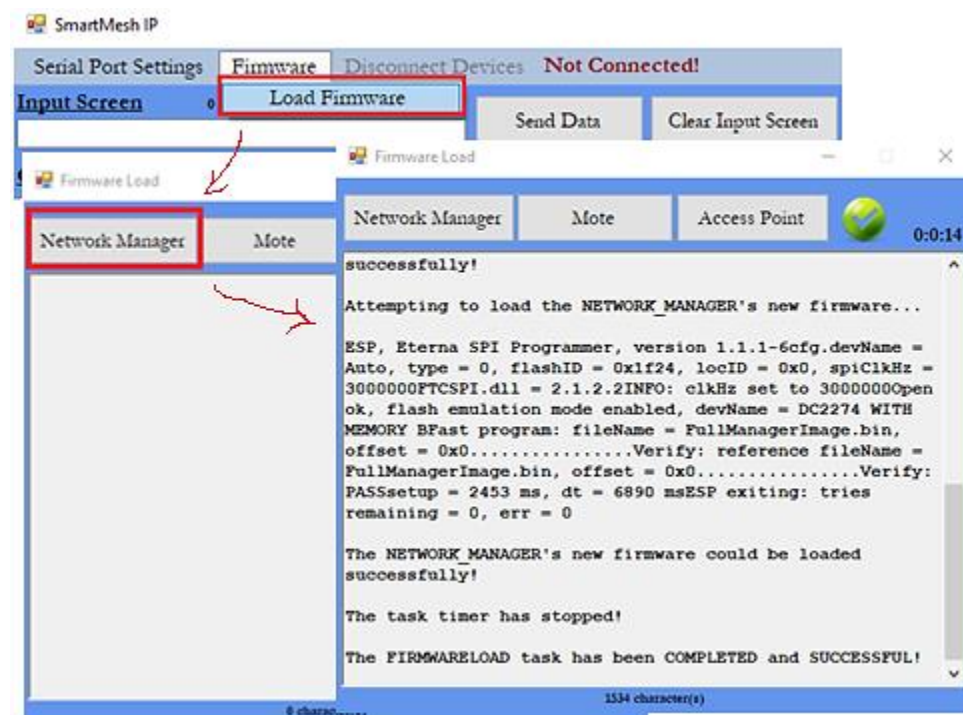


Figure 7. Successful default firmware load on the SmartMesh IP Network Manager.

Configuration Software GUI Controls

As shown in Figure 4 above, the configuration software GUI is made up of multiple controls such as buttons, textboxes, combo boxes, and labels which are meant to help the user effectively and conveniently interact with the GUI to configure the SmartMesh IP devices. These GUI controls are briefly summarized as follows.

- GUI's Serial Port Settings button (to connect the Manager or Mote to the configuration software GUI).
- Firmware button and Load Firmware button (to load the default firmware on the Manager or Mote).
- Disconnect Devices button (to terminate the serial port communication).
- Serial Port Connection Status label (to indicate whether a device connected).
- Input Screen textbox (to enter any compatible CLI command to be executed on a device).
- Output Screen textbox (to display a log of the events including the results of CLI commands execution).
- Connection buttons (for manager and mote) (to establish the serial port communication).
- COM Port Box combo boxes (to choose COM port number).
- Restart buttons (to power-cycle the Manager and Mote).
- Factory Reset buttons (to set all the settings of the device to default).
- Mode combo box (to assign the mode (master or slave) of the Mote).
- AutoJoin combo box (to turn on or off the AutoJoin feature of the Mote).
- Radiotest combo boxes (to turn on or off the Radiotest feature of the device).
- Set and Get Network ID buttons (to assign and view the Network ID values displayed in the Network ID textbox of a device).
- Set Join Key buttons (to assign the Join Key values displayed in the Join Key textbox of a device).
- Device combo box (to set the device type to either Network Manager or Mote).
- Task Outcome Name textbox (to display the current task's name such as CONNECTION, RESTART, FACTORYRESET, MODE, AUTOJOIN, and so on).
- Task Outcome Status textbox (to display the current task's status such as STARTED, CANCELED, and COMPLETED).
- Task Outcome Result textbox (to display the current task's result such as SUCCESSFUL, UNSUCCESSFUL).
- Task Outcome Info Message textbox (to display a message related to the current task).
- Task Outcome Status image (to display the current task's status and result in the form of an image).
- Task Outcome Timer label (to display the elapsed time of the task).

Configuration Software Running Results

Now we show some typical results obtained by executing related tasks on the configuration software GUI. Figure 8 shows the result of performing the Connection task on the Network Manager, where the "connection completed and successful" result is displayed in several related boxes. Similarly, Figure 9 shows the result of performing the Connection task on the Mote.

Some features of the GUI software configuration are listed as follows:

- The network manager's and mote's serial port settings can be independently assigned, viewed, or reset to default.
- Both the network manager and mote can be independently accessed, configured, and restarted.
- Both the network manager's and mote's Network ID and Join Key values can be independently set, retrieved and viewed.
- CLI commands can be typed-in by the user and sent to either the network manager or mote.
- The proper firmware can be independently loaded on the network manager or mote.
- Finally, the system configuration software flowchart is provided in the Appendix Figure A.

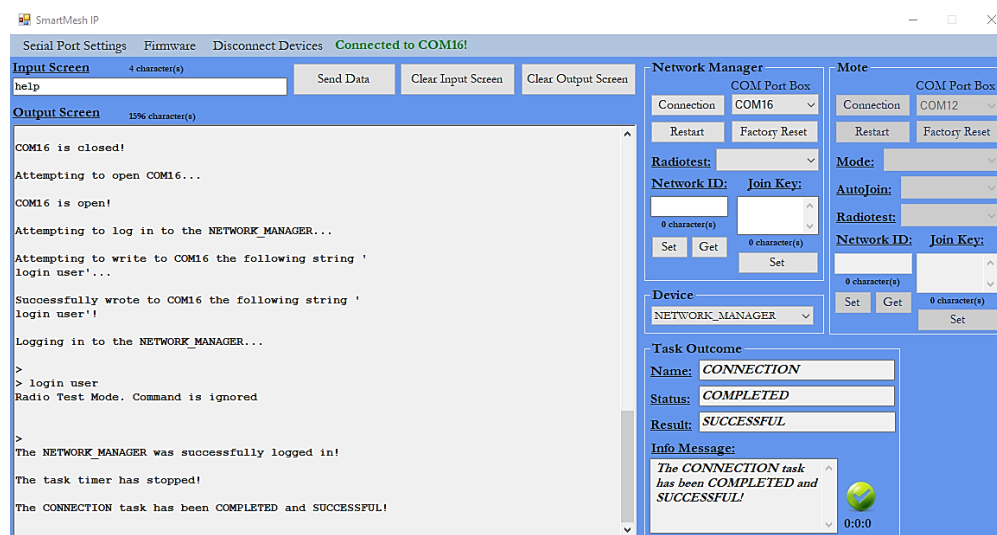


Figure 8. Performing the Connection task on the SmartMesh IP Network Manager.

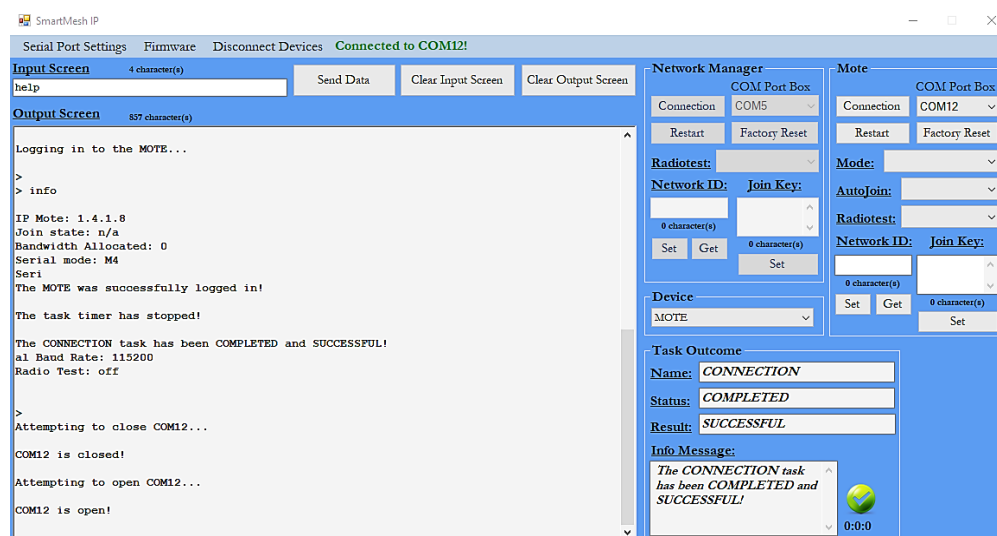


Figure 9. Performing the Connection task on the SmartMesh IP Mote.

The SmartMesh IP System Application Software

This section describes the design of the SmartMesh IP system applications. We develop a SmartMesh IP Temperature Logger platform which can be used to view the temperature recorded by each mote using the mote's built-in temperature sensor. We also develop a SmartMesh IP Temperature Plotter platform which can be used to view the temperature recorded by a mote using an external LM35 analog temperature sensor. Finally, we develop of a SmartMesh IP Network Topology platform which can be used to view the network topology involving all the motes and the network manager. Similar to the previous section, here we only focus on the software development and hardware integration while omit the firmware development. However, the firmware of the SmartMesh IP Network Manager and Mote will be directly embedded into the software programs during the development process. For the hardware integration, the user needs to plug in the network manager to a computer as shown in Figure 10, and make sure that the motes are running in master mode, and the motes and the network manager all have the same Network ID and Join Key values as these parameters are used to establish the SmartMesh IP network. Please refer to the Configuration Software in Section 3 to configure these parameters. As shown in Figure 10, both Status LEDs on the motes are turned on, meaning that the motes are successfully joined to the network manager, thus forming a SmartMesh IP network.

SmartMesh IP Temperature Logger Platform

In this application, we develop a SmartMesh IP Temperature Logger platform to collect the temperature data generated from multiple motes' built-in temperature sensors. The Temperature Logger platform is written in the C# language. It is embedded into the “Applications” menu button of the previous Configuration Software GUI with the button “Internal Temperature Sensor”, as shown in Figure 11. One can click this button to launch the Temperature Logger GUI, as shown in Figure 12.

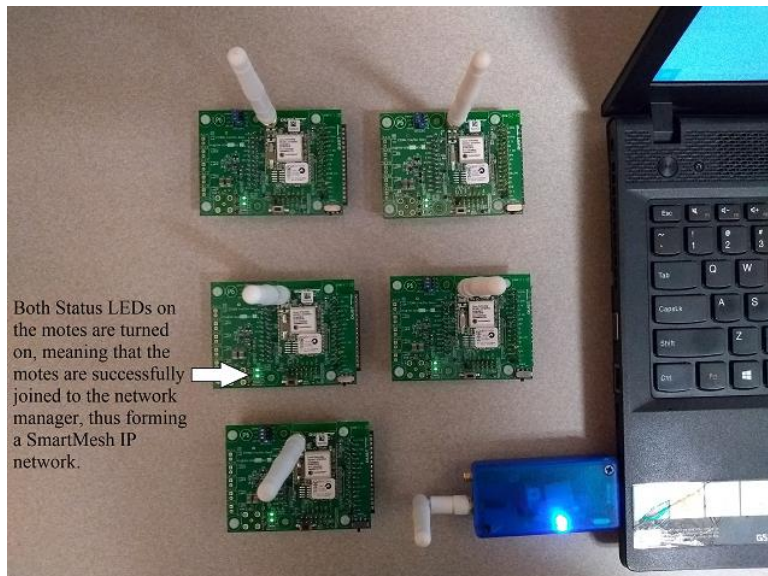


Figure 10. The SmartMesh IP network experimental system with a network manager and five motes.

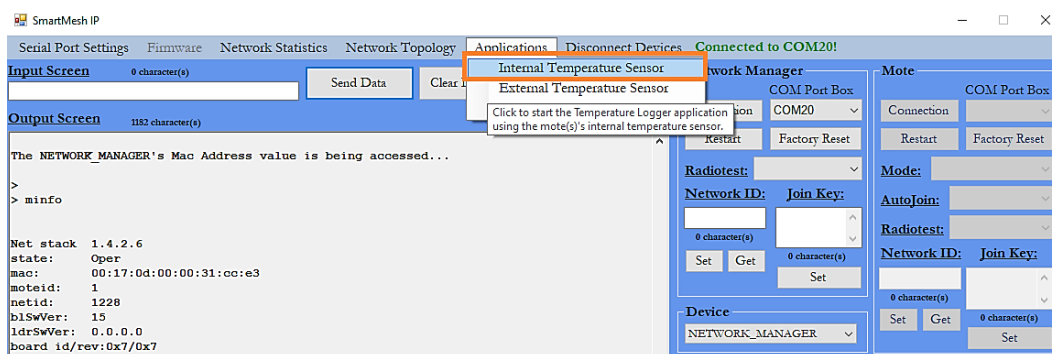


Figure 11. Clicking the “Internal Temperature Sensor” button to launch the Temperature Logger GUI.

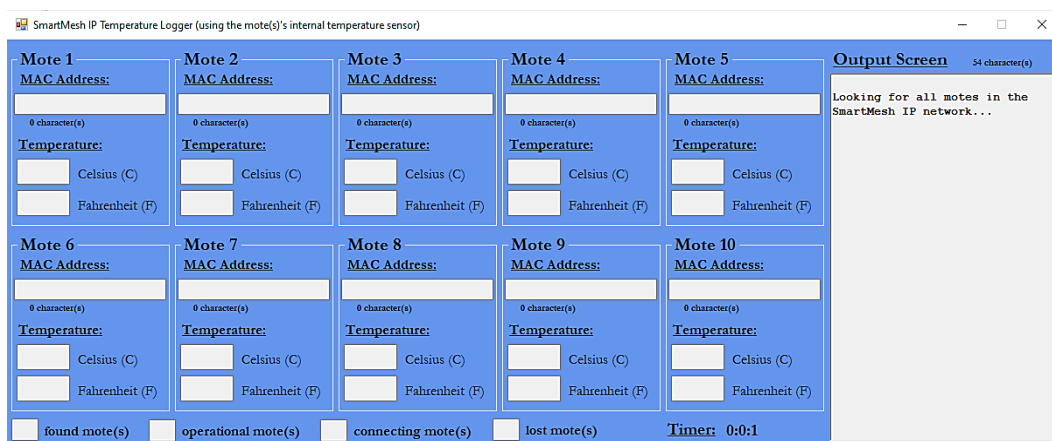


Figure 12. The Temperature Logger GUI after successful launching.

The Temperature Logger GUI is made up of multiple controls such as textboxes and labels which are meant to help the user effectively and conveniently interact with the GUI to view the motes' temperature data. The following briefly introduces each GUI control.

- MAC Address textbox (to display the MAC Address of a SmartMesh IP Mote).
- Mote's Celsius Temperature (to show the temperature recorded by a mote in Celsius).
- Mote's Fahrenheit Temperature (to show the temperature recorded by a mote in Fahrenheit).
- Found Mote(s) textbox (to show the number of motes found in the network, whether they are active or inactive).
- Operational Mote(s) textbox (to show the number of active motes in the network).
- Connecting Mote(s) textbox (to show the number of motes trying to join the network).
- Lost Mote(s) textbox (to show the number of inactive motes in the network).
- Timer label (to show the amount of time that has elapsed since the Temperature Logger GUI is opened).
- Output Screen textbox (to display a log of the events).

The Temperature Logger software flowchart is provided in the Appendix Figure B.

It is worthy to note that some one-time preliminary software work should be set for the Temperature Logger software to run properly. Here we only describe it briefly due to space limitation. First, the user should install Python and download and unzip the file SmartMeshSDK-1.3.0.1-win [19] to a folder under the local Python folder. Second, the user should start the existing application called "TempMonitor.exe" within the SmartMeshSDK-1.3.0.1-win folder. Once the "TempMonitor.exe" application is started, the user should set the network type to "SmartMesh IP", and set the device type to "manager", then click on "load". Third, the user should enter the network manager's API port name (the API port name is created when the network manager is plugged in to a computer; in our case it is "COM50") and click on "connect". Upon successful connection, the "TempMonitor" application should display the temperature data received from each mote. Figure 13 shows a case of two motes. The user can change the temperature sending rate by entering the desired milliseconds value in the publish rate parameter.

Now that the motes are properly configured, we can monitor the temperature data in detail at the SmartMesh IP TemperatureLogger GUI. Figure 14 shows the result of one deployment scenario that five motes are deployed in different places of different layers in the building, where the communicating distance between a mote and the network manager is up to 30 meters in the experiment. We observe that each mote has a unique MAC address with 23 characters. The temperature data collected from different internal sensors display in both Celsius and Fahrenheit format. Different motes detect a little different temperature values due to different mote locations deployed in the building (e.g., inside room, corridor, hall way). We also observe the information of the number of found motes, the number of operational motes, the number of connecting motes, the number of lost motes, and the amount of elapsed time in the bottom boxes as well as a log of the events in the output screen textbox.

SmartMesh IP Temperature Plotter Platform

In this application, we develop a SmartMesh IP Temperature Plotter platform to collect the temperature data generated from external temperature sensors. The Temperature Logger platform is written in the C# language. It is embedded into the "Applications" menu button of the previous Configuration Software GUI with the button "External Temperature Sensor", as shown in Figure 11. One can click this button to launch the Temperature Plotter GUI, as shown in Figure 15.

The Temperature Plotter GUI is made up of multiple controls such as textboxes and labels which are meant to help the user effectively and conveniently interact with the GUI to view the temperature data from external sensors. These controls are the same as that in Figure 12 except the following two:

- The Mote's Temperature Graph shows the temperature data recorded by a given mote in Fahrenheit. This is a graphical display of the digitized data from an external temperature sensor (e.g., LM35DZ) connected to the analog input pin of a mote.
- The Digital Input Pin textbox shows the value (1-high or 0-low) of the digital input pin of a mote.

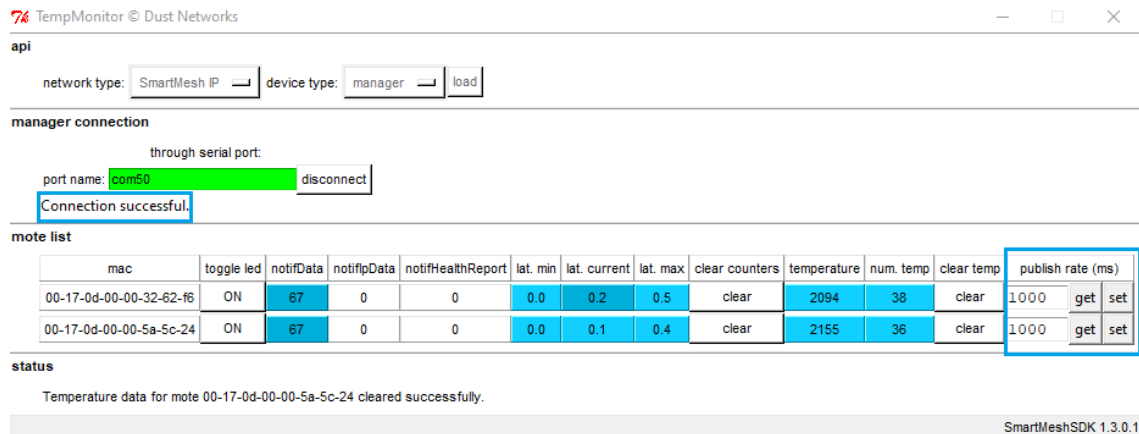


Figure 13. Accessing the temperature data from individual motes.

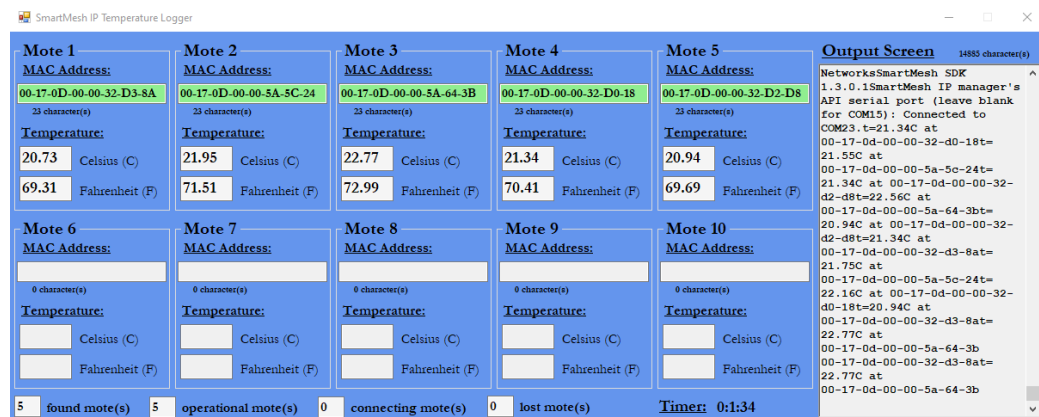


Figure 14. Temperature Logger Software GUI with 5 operational motes.



Figure 15. The Temperature Plotter GUI after successful launching.

The Temperature Plotter software flowchart is provided in the Appendix Figure C.

Similar to Section 4.1, some one-time preliminary software work should be set for the Temperature Plotter software to run properly. The configuration process is simply described as follows. First, the user needs to make sure that the motes are running in master mode, and the motes and the network manager all have the same Network ID and Join Key values. Please refer to Section 3 to complete this step. Second, the user needs to load firmware on a mote from the IAR Embedded Workbench for ARM [18]. Each mote needs to be programmed with a custom System-on-Chip firmware (called 02-gpio_net firmware) developed in the IAR Embedded Workbench. Before loading the 02-gpio_net firmware project on a mote, the user needs to make sure that the mote is connected to the computer running the IAR Embedded Workbench and to the IAR Systems I-jet programmer device, as shown in Figure 16, and also make sure that the mote's power switch is set to ON. The detailed steps of setting up a new project in IAR can be referenced from the dustcloud website [19].

Now it is ready to use the Temperature Plotter GUI to monitor the temperature data from the external sensors. The user should first get the network manager connected to the SmartMesh IP configuration software main GUI by running the "Connection" task as the successful execution of this task enables access to the "Applications" button which grants access to the button for starting the Temperature Plotter GUI.

Figure 17 shows the result of one deployment scenario that two motes with external temperature sensors connected are deployed inside the room for one and in the hall way for the other. We can observe the temperature data collected from the external sensors are displayed in both Celsius and Fahrenheit format in the textboxes. The graphical display of the digitized temperature data is plotted in the Temperature Plotter window in Fahrenheit format. Note that the temperature difference detected is because the two motes are deployed by one inside the room and the other on the hall way. Note also that the temperature in either the digital display or the graphical display changes dynamically according to the ambient temperature, which has been verified by heating devices such as hair dryers.

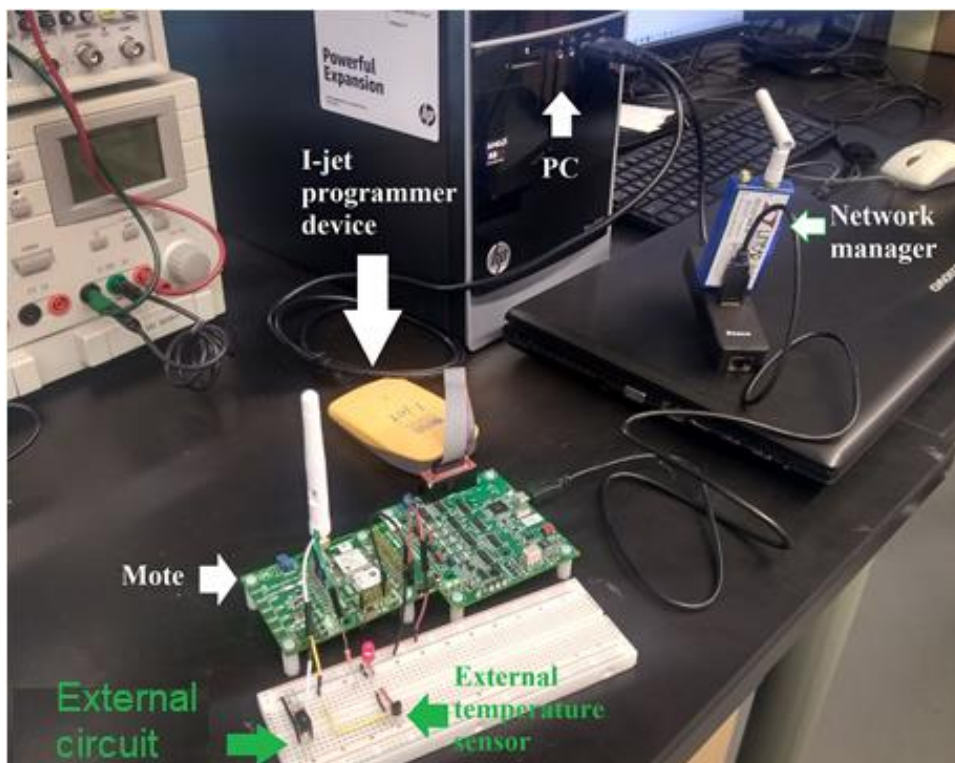


Figure 16. Hardware setup needed to load the 02-gpio_net firmware on a mote.

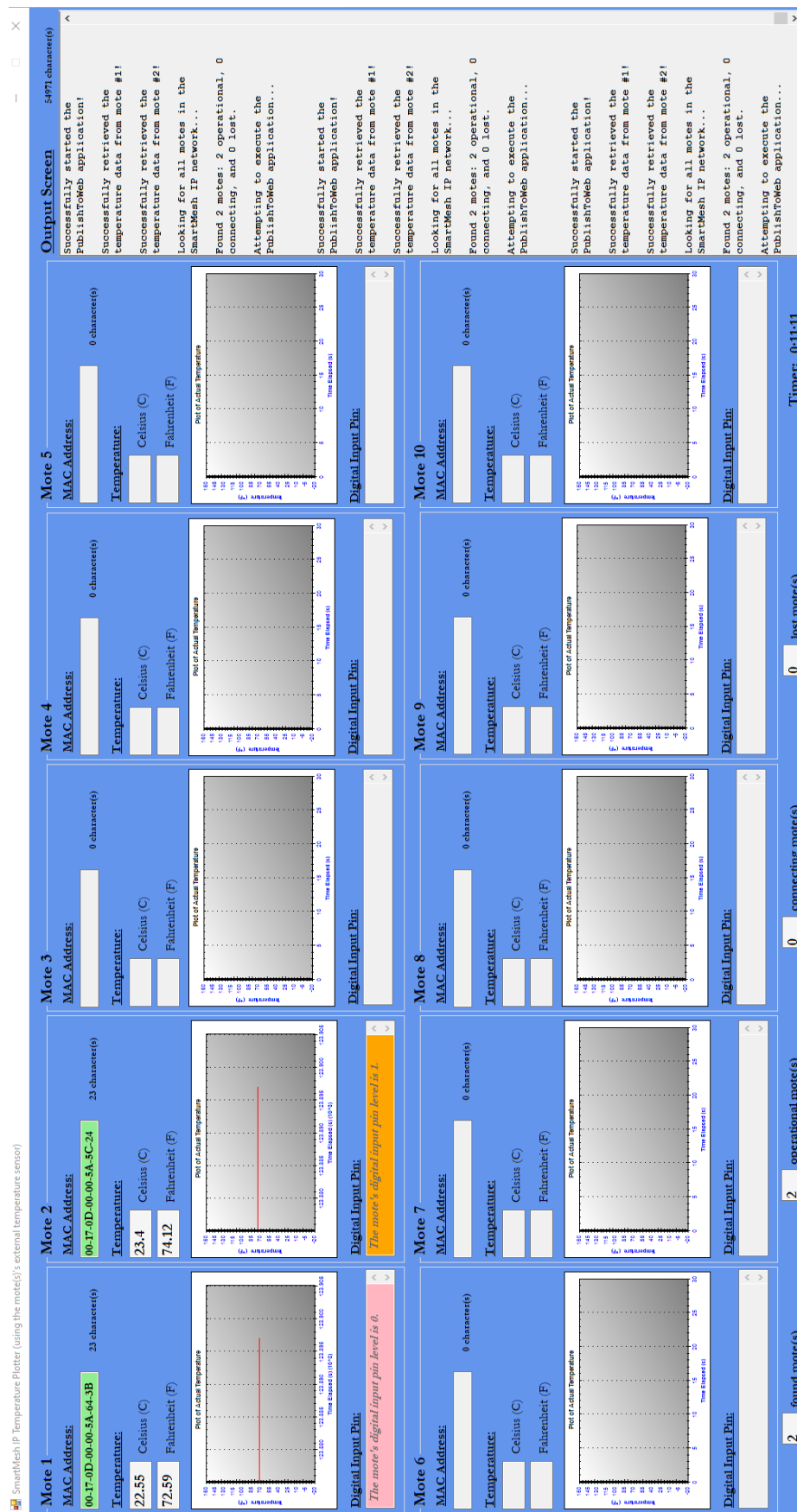


Figure 17. Temperature Logger Software GUI with 5 operational motes.

SmartMesh IP Network Topology Platform

The network topology diagram helps visualize the communicating devices such as the network manager and the motes and the connections between these devices. This platform is mainly software oriented. On the hardware side, the user simply needs to plug in the SmartMesh IP network manager to a computer running the SmartMesh IP Network Topology software, as shown in Figure 10, and make sure that the motes are running in master mode, and all the motes have the same Network ID and Join Key values as the network manager.

On the software side, the SmartMesh IP Network Topology program is written in the C# language, and is part of the class “Network_Manager_GUI.cs”. To access the SmartMesh IP network topology, the “Network_Manager_GUI.cs” program launches, in the background, the SeeTheMesh application (i.e., SeeTheMesh.exe) found in the aforementioned SmartMeshSDK-1.3.0.1-win folder. This application starts a web interface at <http://127.0.0.1:8081> as shown in Figure 18.

The “Network_Manager_GUI.cs” program accesses this web interface which contains the SmartMesh IP network topology and displays it, as shown in Figure 19. Once the web interface has been successfully started, one should be able to obtain the network topology diagram by opening the URL <http://127.0.0.1:8081> in a web browser such as Google Chrome.

To start the Network Topology GUI, the user should first make sure the network manager is plugged in to a computer running the SmartMesh IP configuration software main GUI, then simply click on the “Network Topology” button in Figure 4. After that, the SmartMesh IP network topology diagram should be obtained in a web page (here it is Google Chrome page).

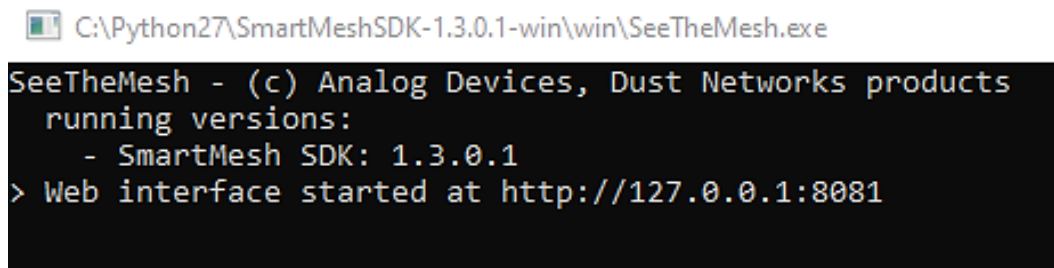


Figure 18. The “SeeTheMesh.exe” application starts a web interface.

```
3382 Output_ScreenText("\n\nStarting the SeeTheMesh application...\n\n");
3383 //Select the cmd.exe program to be run in Windows cmd
3384 Common.commandLine = new CommandLineExecute(@"cmd.exe", 100);
3385 //Send command to run the SeeTheMesh.exe file
3386 if (Common.commandLine.Start("/C " + Common.SeeTheMeshApplicationProcessName, Common.applicationFilesDirectoryPath) == 0)
3387 {
3388     //Append text in output_screen
3389     Output_ScreenText("\n\nThe SeeTheMesh application has successfully started!\n\n");
3390     try
3391     {
3392         /*Open the SmartMesh IP Network Topology in Google Chrome (opening in 2 Google Chrome tabs to "speed up" the loading)*/
3393         Process.Start("chrome.exe", Common.seeTheMeshUrl); //1st tab
3394         Process.Start("chrome.exe", Common.seeTheMeshUrl); //2nd tab
3395     }
3396     catch (Exception ex)
3397     {
3398         //Append text in output_screen
3399         Output_ScreenText("\n\n" + ex.Message + "\n\n");
3400         //Exit function
3401         return;
3402     }
3402 }
```

This line starts the “SeeTheMesh.exe” application

These 2 lines open the “<http://127.0.0.1:8081>” URL in the Google Chrome web browser

Figure 19. Snippet of the “Network_Manager_GUI.cs” code used to access and display the SmartMesh IP network topology.

Figure 20 shows the view of a SmartMesh IP Network Topology diagram with a network manager and two motes. In the network topology design, the network manager is displayed in blue with its MAC Address (the last 8 characters), and each mote is displayed in white with its own MAC Address. Each mote has a direct connection to the network manager (“31-cc-e3”) and/or indirect connection through another mote to the network manager. Furthermore, the network topology diagram automatically gets adjusted with real time changes in the network. For example, Figure 20 shows a network manager (“31-cc-e3”) connected to 2 motes (“5a-64-3b” and “5a-5c-24”); when the user adds another mote (“32-62-f6”) to the network, the network topology diagram will be dynamically redrawn as shown in Figure 21. Figure 21 shows the transition from the situation that the new mote (“32-62-f6”) is joining the network to the situation that the mote has joined the network. Note that it may take time for this topology adjustment. In our experiment, it takes up to about 1 minute for the network topology diagram to fully adjust during a mote joining process.

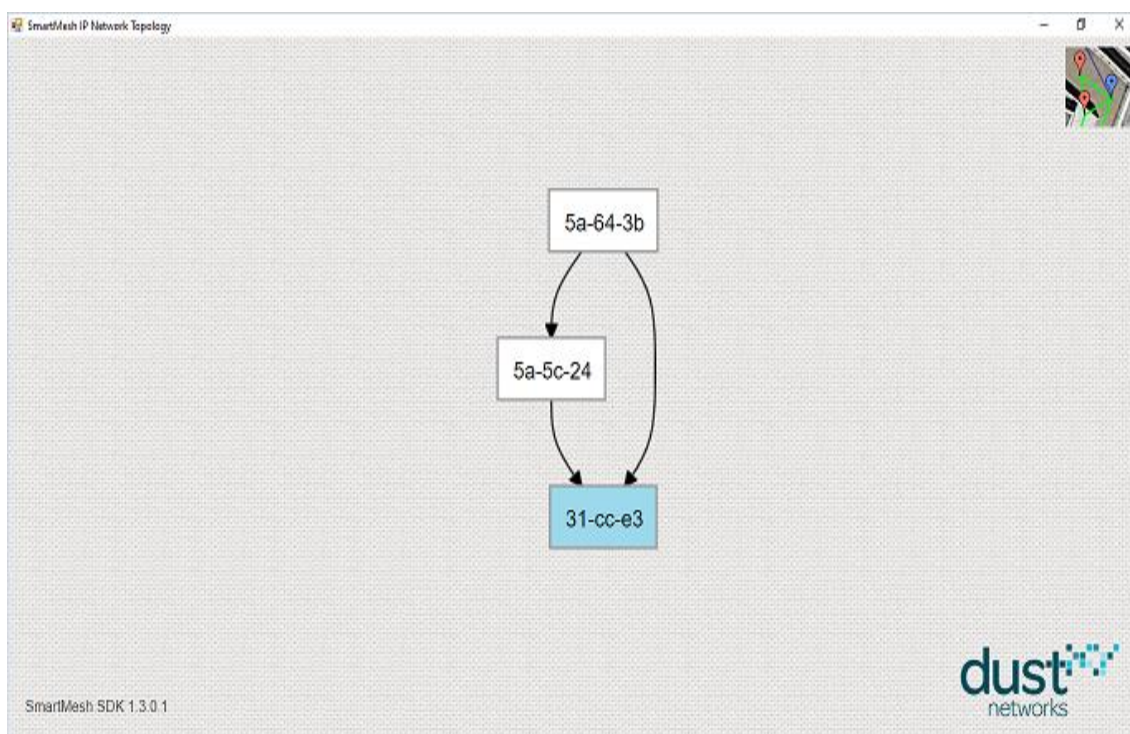
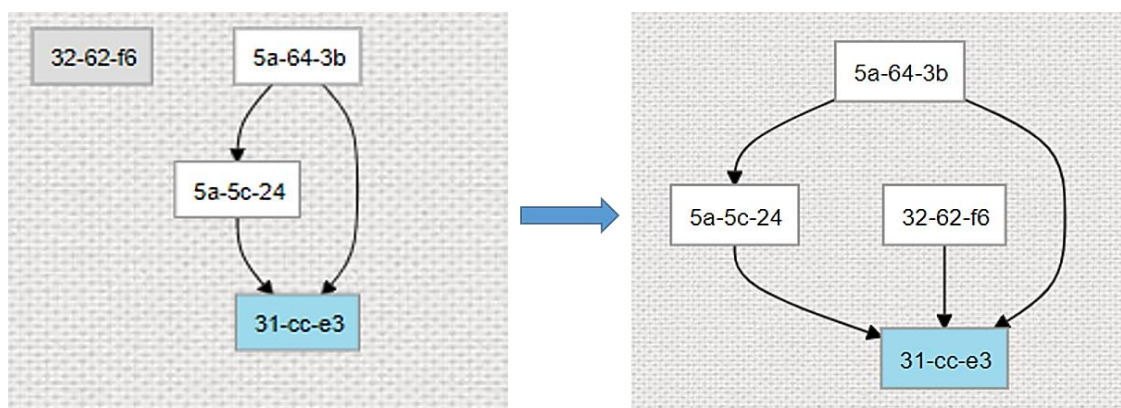


Figure 20. A SmartMesh IP Network Topology with a network manager and two motes.



The mote “32-62-f6” is joining the network

The mote “32-62-f6” has joined the network

Figure 21. The dynamic adjustment of a SmartMesh IP Network Topology with a new mote joining the network.

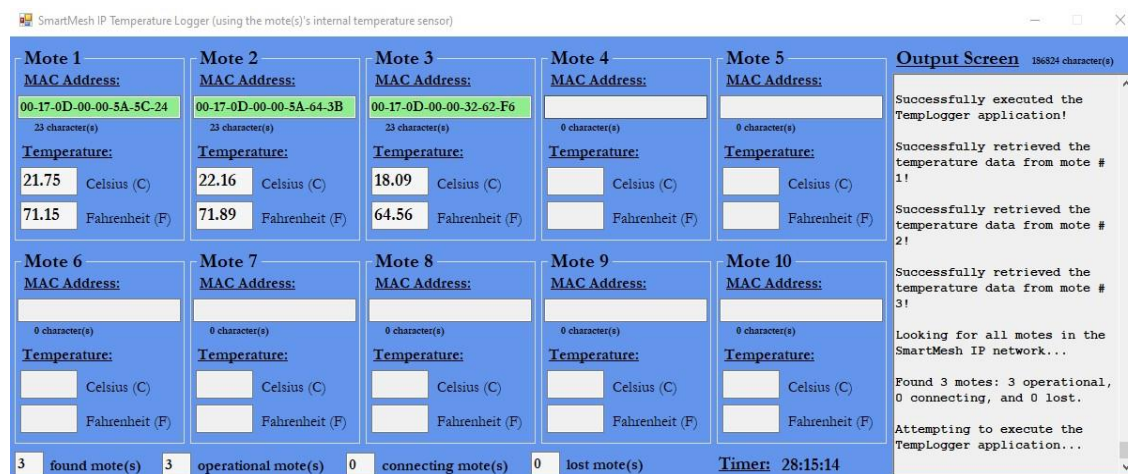


Figure 22. The dynamic adjustment of a SmartMesh IP Network Topology with a new mote joining.

The above testing experiment is performed by using the on-board internal temperature sensors, where the network manager is connected to the host application running on a computer and the motes are programmed with the default firmware so that every second they report to the network manager the ambient temperature data detected by their respective internal temperature sensors. All the devices in this scenario were located within line of sight. As shown in Figure 22 below, each mote has successfully sent the ambient temperature sensed by its internal sensor and our Temperature Logger GUI has displayed the temperature reading in both Celsius and Fahrenheit along with the corresponding mote by the MAC address. Similar to Figure 14, the information of the number of found motes, the number of operational motes, the number of connecting motes, and the number of lost motes are displayed in the respective bottom boxes. Figure 22 also shows that the experiment lasted 28 hours, 15 minutes, and 14 seconds beside the Timer label.

CONCLUSION

We proposed a SmartMesh IP network system with a network manager and a group of motes and implemented it through hardware integration, firmware design, and software development, as well as system testing. Through various experiments, our software, firmware, and hardware platforms have been shown to be able to fully configure the SmartMesh IP devices and get them to perform real-world applications such as monitoring the temperature data collected within the SmartMesh IP network. By using internal or external temperature sensors, our platforms developed in C# programming have shown the successful temperature data collection in the proposed SmartMesh IP network and the network topology viewing. The design and implementation method of this paper can provide practical reference for the development and industrial application of SmartMesh IP networks.

Acknowledgement

The authors would like to acknowledge the partial support from Emerson Automation Solution company through the Grant (No. 629528) and the partial support from the Department of Electrical and Computer Engineering at St. Cloud State University, MN, USA, through providing its equipment and software.

REFERENCES

1. J. Iannacci, RF-MEMS Technology for High-Performance Passives, 2017, IOP Publishing. <http://dx.doi.org/10.1088/978-0-7503-1545-6>
2. S. Furber, Arm System-On-Chip Architecture, 2nd Edition, 2000, Addison-Wesley Professional.
3. T. Arampatzis, J. Lygeros and S. Manesis, "A Survey of Applications of Wireless Sensors and Wireless Sensor Networks," Proceedings of the 2005 IEEE International Symposium on,

-
- Mediterranean Conference on Control and Automation Intelligent Control, 2005, pp. 719-724, doi: 10.1109/2005.1467103
4. J.M. Kahn, R.H. Katz and K.S.J. Pister, "Mobile Networking for Smart Dust", ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99), Seattle, WA, August 17-19, 1999.
 5. M.A. Horton, S. Glaser, and N. Sitar, "Wireless Networks for Structural Health Monitoring and Hazard Mitigation", Proc. of the US-Europe Workshop on Sensors and Smart Structures Technology, pp. 19-23, 2002.
 6. IEEE 802.15.4-2003-IEEE Standard for Telecommunications and Information Exchange Between Systems-LAN/MAN Specific Requirements-Part 15: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPAN). https://standards.ieee.org/content/ieee-standards/en/standard/802_15_4-2003.html
 7. K. Pister and L. Doherty, "TSMP: Time synchronized mesh protocol", IASTED International Symposium on Distributed Sensor Networks (DSN 2008), Nov. 16–18, 2008, Orlando, Florida, USA.
 8. J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt, "WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control," 2008 IEEE Real-Time and Embedded Technology and Applications Symposium, 2008, pp. 377-386, doi: 10.1109/RTAS.2008.15.
 9. IEEE 802.15.4e-2012-IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer. https://standards.ieee.org/standard/802_15_4e-2012.html
 10. T. Watteyne, L. Doherty, J. Simon and K. Pister, "Technical Overview of SmartMesh IP," 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2013, pp. 547-551, doi: 10.1109/IMIS.2013.97.
 11. K. Brun-Laguna, A. L. Diedrichs, D. Dujovne, C. Taffernaberry, R. Léone, X. Vilajosana, and T. Watteyne, "Using SmartMesh IP in Smart Agriculture and Smart Building applications", Computer Communications, Vol. 121, pp. 83-90, 2018. <https://doi.org/10.1016/j.comcom.2018.03.010>.
 12. Y. Tanaka, B.H. Le, V. Kobayashi, C. Lopez, and T. Watteyne, "Demo: Blink–Room-Level Localization Using SmartMesh IP", in Proceedings of the 2020 International Conference on Embedded Wireless Systems and Networks, pp. 198–199, Feb. 2020.
 13. S. Gheorghiu, K. Nagy-Betegh, R. Molnar, and R. Grammenos, "WALLSY: The UWB and SmartMesh IP enabled Wireless Ad-hoc Low-power Localization SYstem", in Proceedings of the 2021 International Conference on Localization and GNSS (ICL-GNSS), 1-3 June 2021.
 14. M. Price, C# 7 and .NET Core: Modern Cross-Platform Development, 2nd Edition, Packt Publishing, March 2017.
 15. Analog Devices, DC2274A-A, SmartMesh IP USB Network Manager. Available: <https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/dc2274a-a.html#eb-overview>
 16. Analog Devices, DC9018B-B, SmartMesh IP RF Certified Evaluation/Development Mote. Available: <https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/dc9018b-b.html#eb-overview>
 17. M. Lutz, Learning Python, 5th Edition, June 2013, O'Reilly Media, Inc.
 18. IAR Systems, IAR Embedded Workbench for ARM, Software Development Tool. Available: <https://www.iar.com/>
 19. Dust Networks, Inc., SmartMesh SDK-a Python package. Available: <https://dustcloud.atlassian.net/wiki/spaces/SMSDK/overview?homepageId=1015834>

APPENDIX

Flowcharts

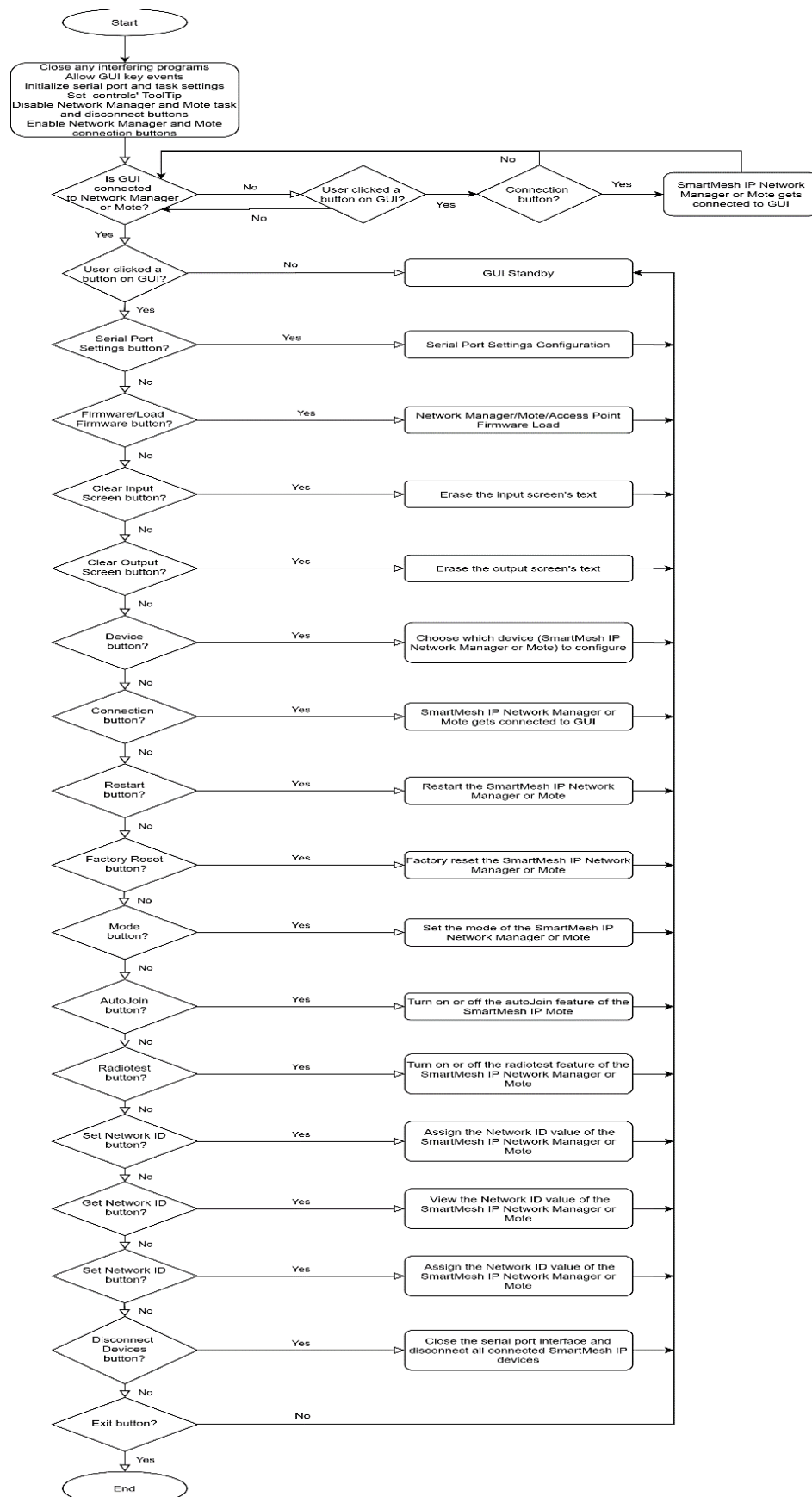


Figure A. The system configuration software flowchart.

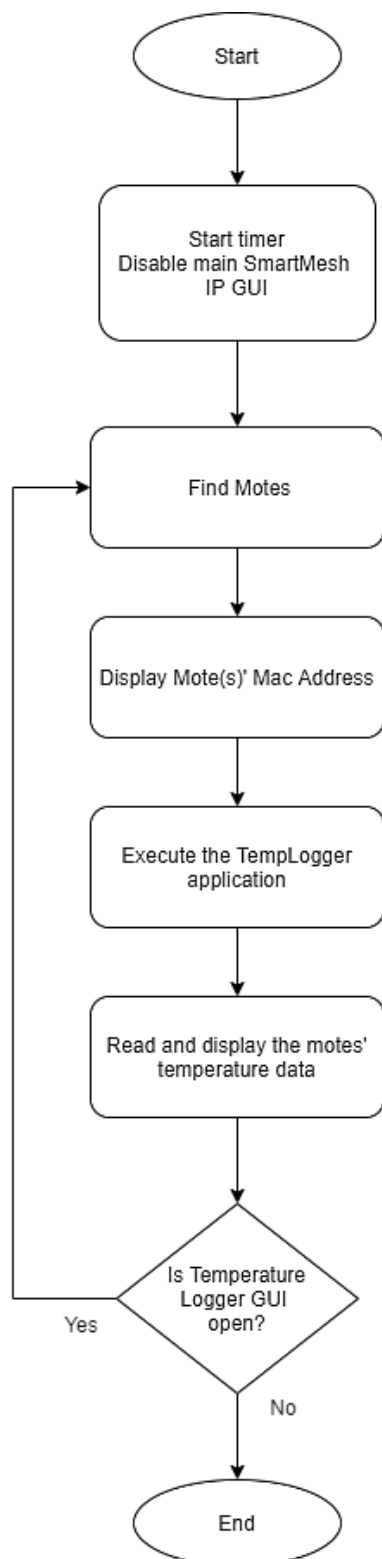


Figure B. The Temperature Logger software flowchart.

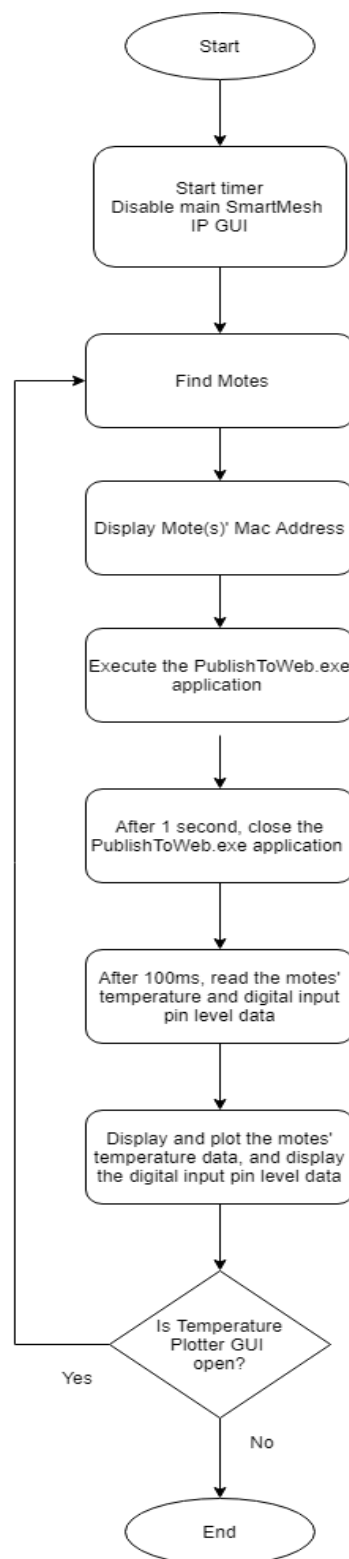


Figure C. The Temperature Plotter software flowchart.