International Journal of Computer Applications in Technology

ISSN online: 1741-5047 - ISSN print: 0952-8091
https://www.inderscience.com/ijcat

Implementation of an IoT system for environment monitoring and remote web control using ARM Mbed cloud and GUI

Shensheng Tang, Yi Zheng

DOI: 10.1504/IJCAT.2023.10056442

Article History:

Received:	07 January 2022
Last revised:	12 February 2022
Accepted:	01 March 2022
Published online:	23 May 2023

Implementation of an IoT system for environment monitoring and remote web control using ARM Mbed cloud and GUI

Shensheng Tang* and Yi Zheng

Department of Electrical and Computer Engineering,
St. Cloud State University,
St. Cloud, Minnesota, USA
Email: stang@stcloudstate.edu
Email: zheng@stcloudstate.edu
*Corresponding author

Abstract: This paper implements an IoT system for environment monitoring and remote web control using ARM Mbed cloud. The embedded system used for environment monitoring is implemented using an ARM Cortex-M4 core-based STM32L4 series development board integrated with multiple sensors. The sensed data (i.e., temperature, relative humidity and atmospheric pressure) can be wirelessly sent to the Mbed cloud managed by the Pelion device management platform. The data values can also be sent to a Graphical User Interface (GUI) with user authentication. We develop three application appliances that can be controlled remotely over Internet through the Pelion platform. The proposed IoT system has been successfully implemented on the STM32L4 series development board with the main application program developed using C++ and the GUI developed by C# programming. The work of hardware and software co-design can be a practical paradigm of engineering education for IoT hobbyists and college students.

Keywords: IoT; ARM Mbed cloud; Pelion device management platform; GUI; C++; C#; Wi-Fi; environment monitoring; web control.

Reference to this paper should be made as follows: Tang, S. and Zheng, Y. (2023) 'Implementation of an IoT system for environment monitoring and remote web control using ARM Mbed cloud and GUI', *Int. J. Computer Applications in Technology*, Vol. 71, No. 1, pp.1–17.

Biographical notes: Shensheng Tang is a Professor in the Department of Electrical and Computer Engineering in St. Cloud State University, USA. He received his PhD degree from University of Toledo, USA. He has eight years of Product Design and Development experience, as Hardware Engineer, System Engineer and Manager, respectively, in the Electronics and Wireless Industry. His current research interests include embedded systems, networking (wireless, wired), Internet of things (IoT) and modelling and performance evaluation. He has served or is serving as editor-in-chief, editor, or guest editor for international journals and program committee chair or member of international conferences. He has produced over 100 peer-reviewed publications in the above areas. He is a Senior Member of IEEE.

Yi Zheng graduated from Iowa State University, joined Faculty of Electrical and Computer Engineering of St. Cloud State University in 1987, served as Department Chair from 1997 to 2004, Full Professor since 1993. His research interest is wave propagation including optical wave in motion, ultrasound and EM waves in tissue and embedded systems. He worked at Very Large Array and Ames Laboratory in 80s. From 1991 to 1992, He was with IBM to apply artificial neural networks for large data processing. From 1993 to 2009, he was with Ultrasound Research Lab of Mayo Clinic to develop neural networks for medical image analysis, and ultrasound vibrometry for measuring tissue elasticity and viscosity. He was with IMI Vision for Embedded System Design and Sensor Design. From 2006 to 2009, he was with Force 10 Network for High-Speed Circuit Research. He worked with Lift-Touch Studio, Motorola, LionPrecision, Medtronic, Born-Fuke and Emerson, etc.

1 Introduction

Technology advances farther and farther with every day. We are entering the era of the Internet of Things (IoT), which enables communication between electronic devices and sensors through the Internet and provides innovative solutions to a variety of challenges in business, government, military and civilian use. The IoT is the interconnection of endpoints (devices and things) that can be uniquely addressed and identified over the Internet.

The IoT is rapidly transforming how we live our daily lives with more and more connected devices and machines over the Internet or cloud-based platforms. These connected devices and machines range from connected equipment in the enterprise and industrial assets such as smart office, smart transportation, and robots to consumer-oriented devices such as smart wearables and smart home solutions. Sensor technologies, embedded systems, and real-time data collection and analysis can track nearly every aspect of the work we do. When used appropriately, the IoT data can allow us to streamline business processes, increase efficiency, improve our health and safety, automate tasks and enable us to gain greater insight into our societies and environments.

Extensive research and experiments have been done and available in terms of scientific articles, technical reports and news communications both on the internet and in the form of printed materials to illustrate the potential effectiveness and applicability of IoT transformations (Zanella et al., 2014; Khajenasiri et al., 2017; Liu et al., 2012; Li et al., 2019; Wang et al., 2014; Qiu et al., 2013; Strielkina et al., 2017; Tang and Xie, 2021; Birje and Hanji, 2020; Chauhan et al., 2021; Heer et al., 2011; Sfar et al., 2018; Hassija et al., 2019). Zanella et al. (2014) provided a comprehensive survey of the enabling technologies, protocols, and architecture for an urban IoT, a technical solution to support the Smart City vision. The technical solutions and best-practice guidelines adopted in the Padova Smart City project were discussed with a proof-of-concept deployment of an IoT island in the city of Padova, Italy. Khajenasiri et al. (2017) introduced a flexible IoT hierarchical architecture model with an overview of each key component for intelligent energy control in buildings for smart cities. Another category of IoT applications is the smart vehicles, where the vehicles equipped with intelligent devices and sensors enable autonomous driving, smart navigation with networked cars and vehicular automation using artificial intelligence. Liu et al. (2012) presented the analysis on the related questions of automobile manufacturing through IoT and described related contents to promote the realisation of automotive informationisation.

Li et al. (2019) unified approach called IoT-CANE (context aware recommendation system) was presented to capture resource configurations in IoT environments, support the recommendation of resource configuration and facilitate the knowledge acquisition with the Cloud/Edge technologies and IoT devices. An end-user case study was provided to evaluate the success of IoT-CANE. Wang et al. (2014) summarised the IoT applications in domestic waste treatment and disposal and presented the current situation and existing problems in a certain condition. Furthermore, the promotion of Information management efficiency were expounded and prospected with the combination of the IoT technology. Qiu et al. (2013) presented an intelligent monitoring platform framework and system structure for a facility agriculture ecosystem based on IoT. The solution consists of four function layers based on the difference in information exchange process and task logical handling, i.e., sensor layer, transmission layer, monitoring layer, application layer. The framework of the facility agriculture ecosystem was experimented to be able to achieve better ecosystem with reduced human intervention.

In Strielkina et al. (2017), a healthcare IoT infrastructure with brief description was presented and a case study of the considered system was modelled using the queueing theory. In Tang and Xie (2021), an availability model of a healthcare IoT system was proposed with two groups of structures described by separate Markov state-space models. The two separate models are analysed and combined to implement the whole IoT system modelling. An Availability Performance Improving (API) method was also proposed for increasing the probability of system full service and decreasing the system unavailability. More introduction of healthcare IoT systems and technologies can be found in a recent review work (Birje and Hanji, 2020), where the authors reviewed state-of-the-art works in IoT-based Distributed Healthcare Systems (DHSs). A comparative study of these systems was made to know the suitability of various healthcare systems. The challenges and open issues associated with existing IoT-DHSs were also discussed. In Chauhan et al. (2021), an IoT based automatic intravenous fluid monitoring system was proposed to monitor the intravenous fluid level in case the caretaker forgets to change the bottle.

In Heer et al. (2011), the applicability and limitations of existing Internet protocols and security architectures were discussed in the context of the Internet of Things. A further IoT security roadmap was presented in Sfar et al. (2018) based on a novel cognitive and systemic approach, where the role of each component of the approach was explained and its interactions with the other components were studied. A case study was presented to highlight the components and interactions of the approach and the security questions about privacy, trust, identification and access control were then discussed. A detailed review of the security-related challenges and sources of threat in the IoT applications was presented in Hassija et al. (2019). Various emerging and existing technologies focused on achieving a high degree of trust in the IoT applications were discussed. Particularly, four different technologies, blockchain, fog computing, edge computing and machine learning were discussed to increase the level of security in IoT.

In this paper, we implement an ARM Cortex-M4 core-based IoT system for environment monitoring and remote web control using the Pelion Device Management Platform. The hardware of the embedded system is implemented using an ARM processor based STM32L4 series development board integrated with multiple on-board sensors and external circuit. The environment monitoring parameters includes

temperature, relative humidity and atmospheric pressure, which can be wirelessly sent to the Mbed cloud managed by the Pelion platform. The sensed data values can also be sent to a GUI panel for local display. The GUI has a security access that needs user authentication. Three application appliances were designed for the purpose of remote web control though the Mbed cloud with an Mbed account. The proposed IoT system is implemented with the main application program developed using C/C++ and the GUI developed by C# programming. The experiment results are presented for further understanding of the implementation. In summary, the major contribution of the work includes:

- Implementation of an IoT system involving hardware and C++/C# programming for environment monitoring with detection parameters (i.e., temperature, relative humidity and atmospheric pressure) transmitted wirelessly to the Mbed cloud as well as a GUI running on a local computer.
- Implementation of an IoT system involving hardware and C++ programming for remote web control using the Pelion Device Management Platform.

The remainder of the paper is organised as follows: Section 2 describes the proposed IoT system and its modules; Section 3 discusses the system design and implementation issues including the local GUI part and the Mbed cloud part; Section 4 presents the experimental results and Section 5 concludes the paper.

2 System description

The proposed IoT system provides environment monitoring and remote web control which consists of five parts: ARM microcontroller module, sensors, application appliances, Cloud Platform and a Graphical User Interface (GUI) running on a laptop, as shown in Figure 1. The sensors sense the environment parameters such as temperature, relative humidity and atmospheric pressure and send the data to the microcontroller. The microcontroller module collects the environmental data and sends them through Wi-Fi to the Mbed cloud (MBED, n.d.) (i.e., Pelion Device Management Platform (Izuma Networks, n.d.)) as well as the GUI. There are a few application appliances (e.g., smart oven, smart cooker and air conditioner) connected to the microcontroller module. The user can remotely control (e.g., turn-on and turn-off) these appliances through cloud over internet. The GUI running on a laptop can display the environmental data in real time in both graphical and digital manners. The following summarises the different parts of the system.

- *Microcontroller module:* The ARM 32-bit Cortex-M4 processor (CPU Cortex-M4, n.d.) provides the combination of high-efficiency signal processing functionality with the low-power, low-cost and ease-of-use benefits of the Cortex-M family of processors, as shown in Figure 2. It has adaptive real-time accelerator allowing 0-wait state execution from flash memory. It provides 1 MB flash, 128 KB SRAM, 1 Quad SPI memory interface, 7 general purpose timers, 2 basic timers, 2 advanced control timers, 2 low-power timers. It also provides multiple types of communication interfaces (UART, I2C, CAN, SPI and USB). In addition, the module has integrated multiple board features such as Bluetooth, Wi-Fi and expansion connectors (Arduino Uno V3, PMOD).
- *Sensors:* Three types of sensors (see Figure 2), temperature sensor, relative humidity sensor and atmospheric pressure sensor are chosen for continuous environment monitoring in the proposed IoT system, though more sensors are integrated in the development board.

Figure 1 A block diagram of the proposed IoT system

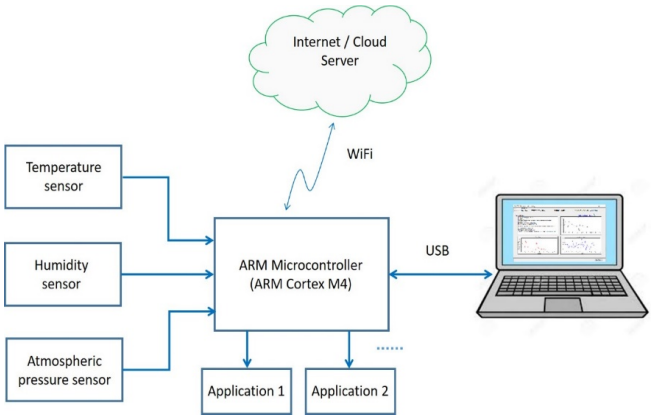
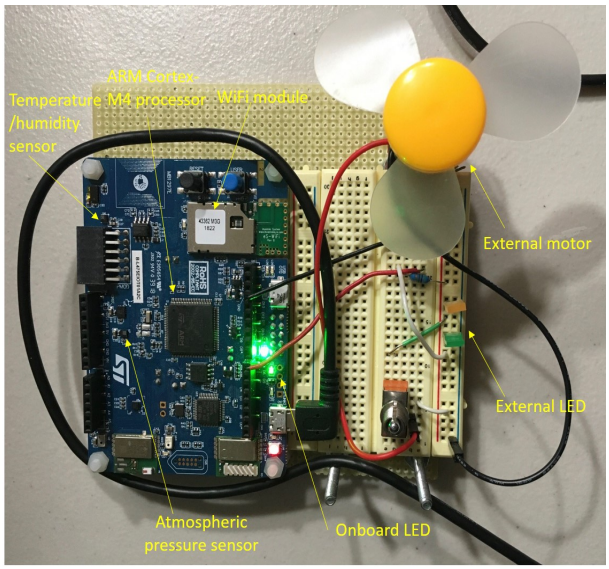


Figure 2 The microcontroller module and its peripheral circuits for the IoT system

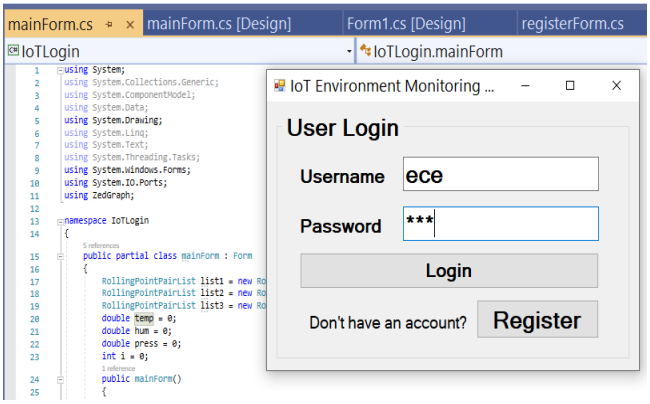


- *Application appliances:* The application appliances are some domestic appliances that can be electronically controlled through Internet-connected systems, which may include setting complex heating and lighting systems in advance and setting alarms and home security controls. They are all connected by a central hub and remote-controlled over the Internet or by a mobile app. In our proposed system, we define three application appliances (see Figure 2): an on-board LED, an external

LED and an external motor, all representing some domestic 'switch-on-off' devices.

- *GUI module:* The GUI module is developed using Visual Studio (Microsoft Corp., n.d.) to communicate with the microcontroller module over serial port. To access the IoT system, the user has to pass the user authentication through a user name and password entering as shown in Figure 3. If the user name or password does not match the information stored in the database, an error message will be popped out. When the user passes the authentication, it will go to the main GUI panel in Figure 4 which has four windows, respectively displaying the system booting information, real-time temperature monitoring, real-time humidity monitoring and real-time atmospheric pressure monitoring, as well as three small text boxes for digital display of the three environment parameters. Some other functions are also considered in the GUI such as screen clearing, serial port and bode rate information. The reset button in the development board can restart the GUI display.

Figure 3 The user authentication for entering the GUI of the IoT system



- *Pelion device management platform:* Pelion is a flexible IoT platform that simplifies the wireless connectivity and secure management of IoT devices. To securely connect an IoT development board to the Internet, one needs to first create an Mbed account, then select her hardware board which is Pelion ready, next use the Mbed online compiler to build her application (The online compiler needs to access her Mbed account and to know which device she is using). Her device needs unique identity and connection parameters to connect to her Pelion Device Management account securely. For remote firmware update, her device also needs an update certificate. This certificate is applied to her project and a corresponding private key is generated. She needs to download and store the private key for the update certificate so that she can sign update manifests for devices with this update certificate in future. She is now ready to build the application and flash it to the device over the USB cable. She also needs to make sure the device to connect to the internet (by editing the mbed_app.json file to provide the application with Wi-Fi credentials). Finally compile the program to create the application binary, which includes the developer-connect-and-update certificates and the device bootloader required to apply firmware updates remotely. The binary is downloaded to the application and the device is now connected to the Pelion Device Management account.

One can use the Device Management Portal to monitor and control the Lightweight Machine to Machine (LwM2M) resource exposed by the application. Here LwM2M is an application layer protocol, which was developed by the Open Mobile Alliance (OMA) to offer a faster time to market by standardising commonly required device management functionality (OMA SpecWorks, n.d.). The IoT devices use the LwM2M protocol to communicate with the server to receive and execute commands. The 'resource' is the name that the LwM2M gives to the readable and controllable aspects of IoT devices such as sensors.

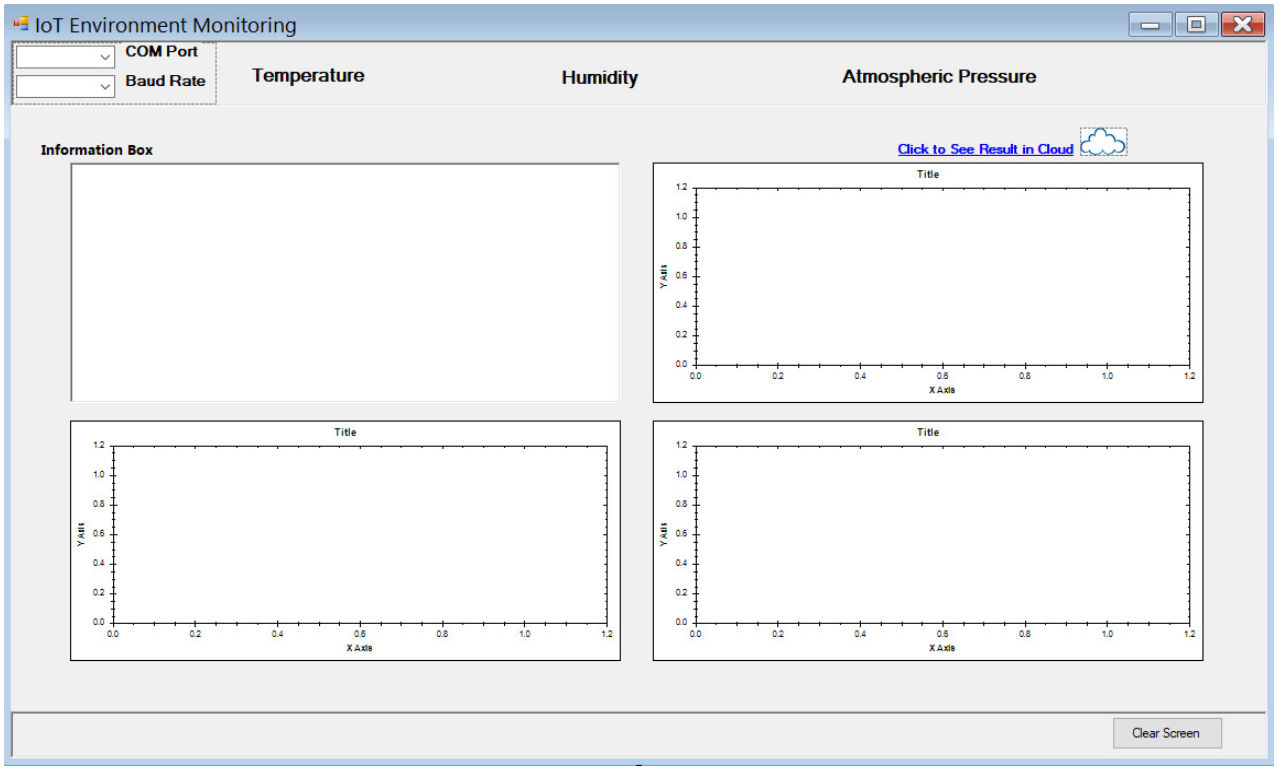
3 System implementation

This section describes the implementation of the IoT system. We implement the system using the L475E-IOT01A Discovery Kit (B-L475E-IOT01A, n.d.) plus a peripheral appliance circuit to communicate with both the Pelion cloud platform and a GUI running on a laptop computer. The L475E-IOT01A Discovery Kit allows users to develop applications with direct connection to cloud servers and enables a wide diversity of applications by exploiting low-power communication, multiway sensing and Arm Cortex-M4 core-based features. It also supports for Arduino Uno V3 and PMOD connectivity to provide unlimited expansion capabilities with a large choice of specialised add-on boards. The communication between the Pelion cloud platform and the hardware circuit is developed by C++ programming. The GUI module is implemented by C# programming, which runs on a laptop and communicates with the hardware circuit using Wi-Fi.

3.1 GUI module implementation

The GUI module is developed to display the monitoring parameters (temperature, relative humidity and atmospheric pressure) in both digital mode and graphical mode, as shown in Figure 4. The information box window is used to display the system booting process and cloud connection through Wi-Fi. The serial communication port and baud rate information are also displayed on the panel. The button of clear screen is used to clear all the four windows.

Figure 4 The GUI panel for the IoT system



The GUI module consists of the user authentication part and the main panel part. They are implemented through the Windows Form App (.Net framework) in C# programming. The user authentication part needs a user to login the GUI panel by entering a user name and password (refer to Figure 3), which are stored in the Structured Query Language (SQL) database. An example of the user credentials stored in the SQL data base is shown in Figure 5. The SQL database is connected by using the SqlConnection object. The following is an example of declaring and instantiating the SqlConnection object at the same time:

SqlConnection cn = new SqlConnection(@'Data Source = (LocalDB)\MSSQLLocalDB; AttachDbFilename = C:\Work\Research\ ...\IoTLogin\dbtest.mdf; Integrated Security=True');

where Data Source identifies the server located in a local computer; the database name is an MDF database file; the third property means the user that runs this application must have access to the target database. With correct credentials, the IoT system brings the user to the main GUI panel that displays the real-time environmental data values. If the credentials do not match what are stored in the database, an error message will be popped out. For the first time user, one can register unique user name and password for user authentication, which are also stored in the SQL database. Figure 6 shows an example of user registration that includes the user's name, password and the full name information (more credential information can be designed for registration).

Figure 5 A list of user credentials stored in the SQL database

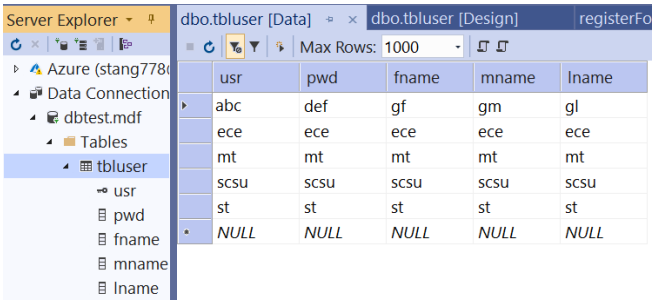
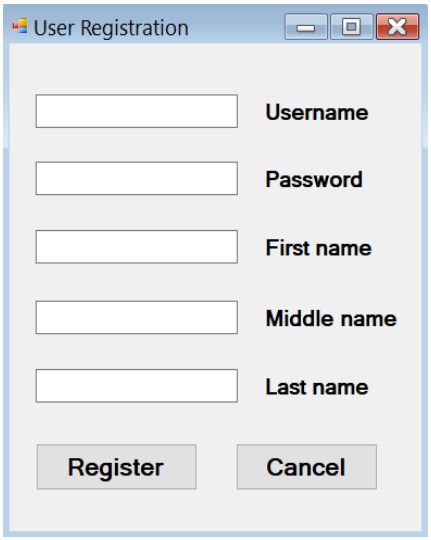


Figure 6 Registration of user credentials for new users



The communication between the GUI and the microcontroller is via a serial port with baud rate of 115200, as shown in Figure 7. Once the serial communication is connected successfully, the sensed data (temperature, humidity, atmospheric pressure) can be transferred into the three textboxes on the GUI, respectively. Every incoming serial data value is defined as string in C#. If the data length is 3, then it is determined that the serial values are of sensor readings and call the functions such as AppendTextBox() to display the sensed data, as shown in Figure 8. If not, it is simple information of MCU operation as Bootloader status, Wi-Fi connection status, displaying IP address, network error status or Wi-Fi connection error status. The function AppendTextBox() is used for the temperature value to be displayed on the textbox, as shown in Figure 9. The other two functions have a similar purpose for displaying humidity and atmospheric.

Next, we describe the implementation issues on plotting the sensed data values graphically using ZedGraph. ZedGraph is a class library, user control, and web control for .net, written in C#, for drawing 2D Line, Bar and Pie graphs of arbitrary data sets (ZedGraph, n.d.). The GraphPane is the primary class for the graph, which includes all other classes as properties and also controls the pane title, the pane frame and axis frame, backgrounds, etc.

To plot the three sensed data values, we need to define three different objects one for each graph. The three objects are defined as zedGraphControl1, zedGraphControl2 and zedGraphControl3. Each graph display is implemented by a function. For example, the temperature display is implemented by the function CreatChart(), as shown in Figure 10. The graph parameters and all essentials like pane, *X*-Axis, *Y*-Axis and LineItem have been defined when the form is loaded. The flowchart of the GUI module design for the IoT system is given in Figure 11.

Figure 7 The serial communication between GUI and microcontroller

```
try
{
    serialPort1.PortName = "COM22";  // Change for specific devices
    serialPort1.BaudRate = Convert.ToInt32("115200");
    serialPort1.Open();
    MessageBox.Show("Automatically Connected");
    comboBox1.Text = "COM22";//for display purpose only
    comboBox2.Text = "115200";//for display purpose only
    comboBox1.Enabled = false;
    comboBox2.Enabled = false;
}
catch (UnauthorizedAccessException)
{
    MessageBox.Show("Unauthorized Acess!!!");
}
```

Figure 8 Display the sensed data values in the textboxes on GUI

```
public void serialPort1_DataReceived(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)
{
    string[] data = serialPort1.ReadLine().Split('^');// read data split by ^
    if (data.Length == 3)
    {
        string tempetureValue = data[0];  //Tempeture data read from the microcontroller
        string humidityValue = data[1];   //Humidity data read from the microcontroller
        string pressureValue = data[2];   //Pressure data read from the microcontroller

        temp = Convert.ToDouble(tempetureValue);
        hum = Convert.ToDouble(humidityValue);
        press = Convert.ToDouble(pressureValue);

        AppendTextBox(tempetureValue);  // call function to dispaly temperature value in textbox
        AppendTextBox2(humidityValue);  // call function to dispaly humidity value in textbox
        AppendTextBox3(pressureValue);  // call function to dispaly air pressure value in textbox
```

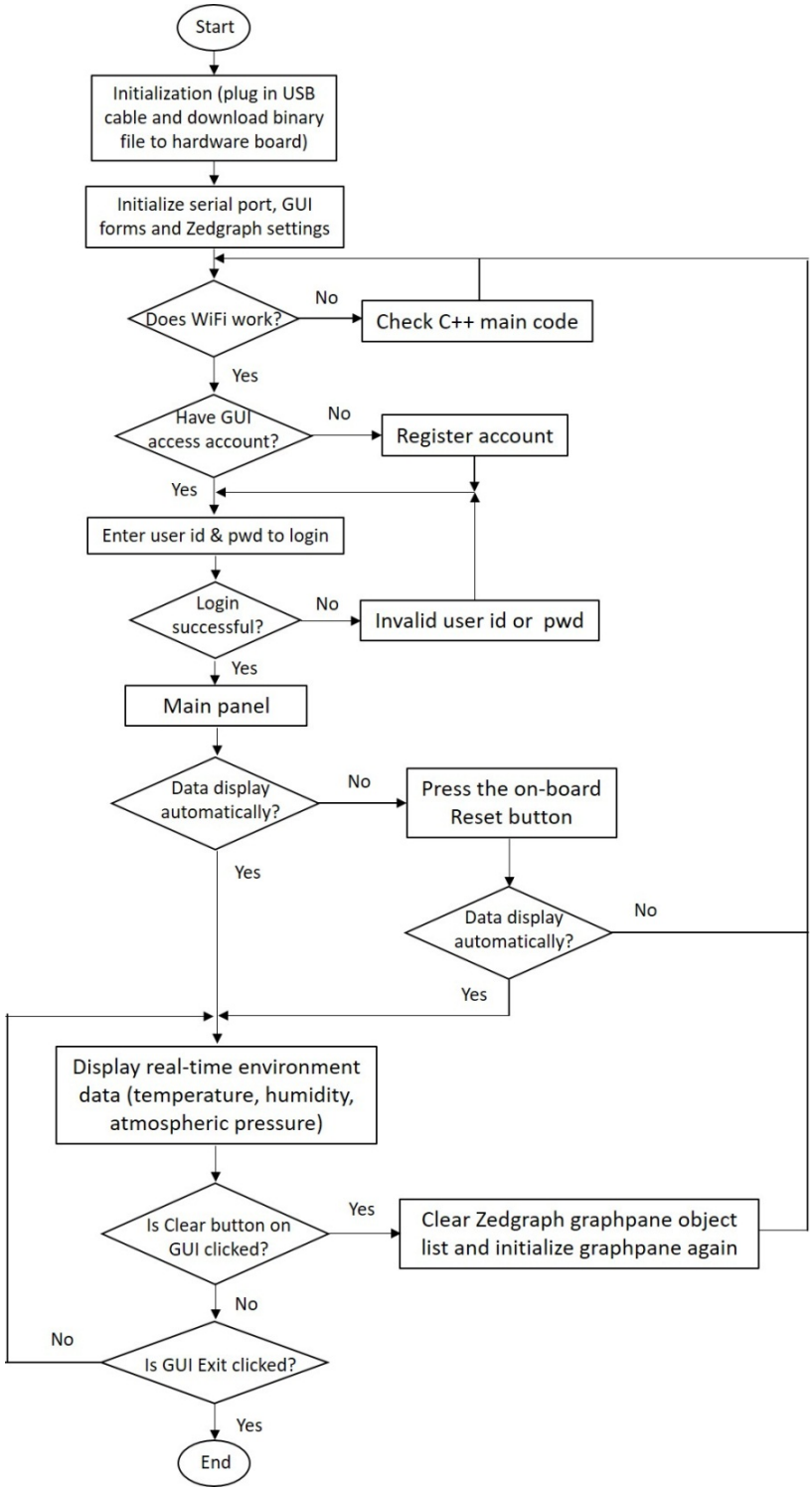
Figure 9 The function AppendTextBox() for temperature display in the textbox

```
public void AppendTextBox(string value)
{
    if (InvokeRequired)
    {
        this.Invoke(new Action<string>(AppendTextBox), new object[] { value });
        return;
    }
    textBox1.Text = "";     // Clears textbox every time the function is called
    textBox1.Text = value;  // Display temperature value in the textbox
    textBox1.Text += " C";  // Add 'C' to represent Celsius
}
```

Figure 10 The function CreatChart() for temperature display graphically

```
public void CreateChart(ZedGraphControl zedGraphControl1)
{
    GraphPane myPane = zedGraphControl1.GraphPane;
    myPane.Title.Text = "Temperature";
    myPane.XAxis.Title.Text = "# of samples";
    myPane.YAxis.Title.Text = "Temperature Value in degree C";
    // Generate a red curve for real-time temperature data displ
    LineItem myCurve = myPane.AddCurve(null, list1, Color.Red, SymbolType.None);
    myPane.AxisChange();
}
```

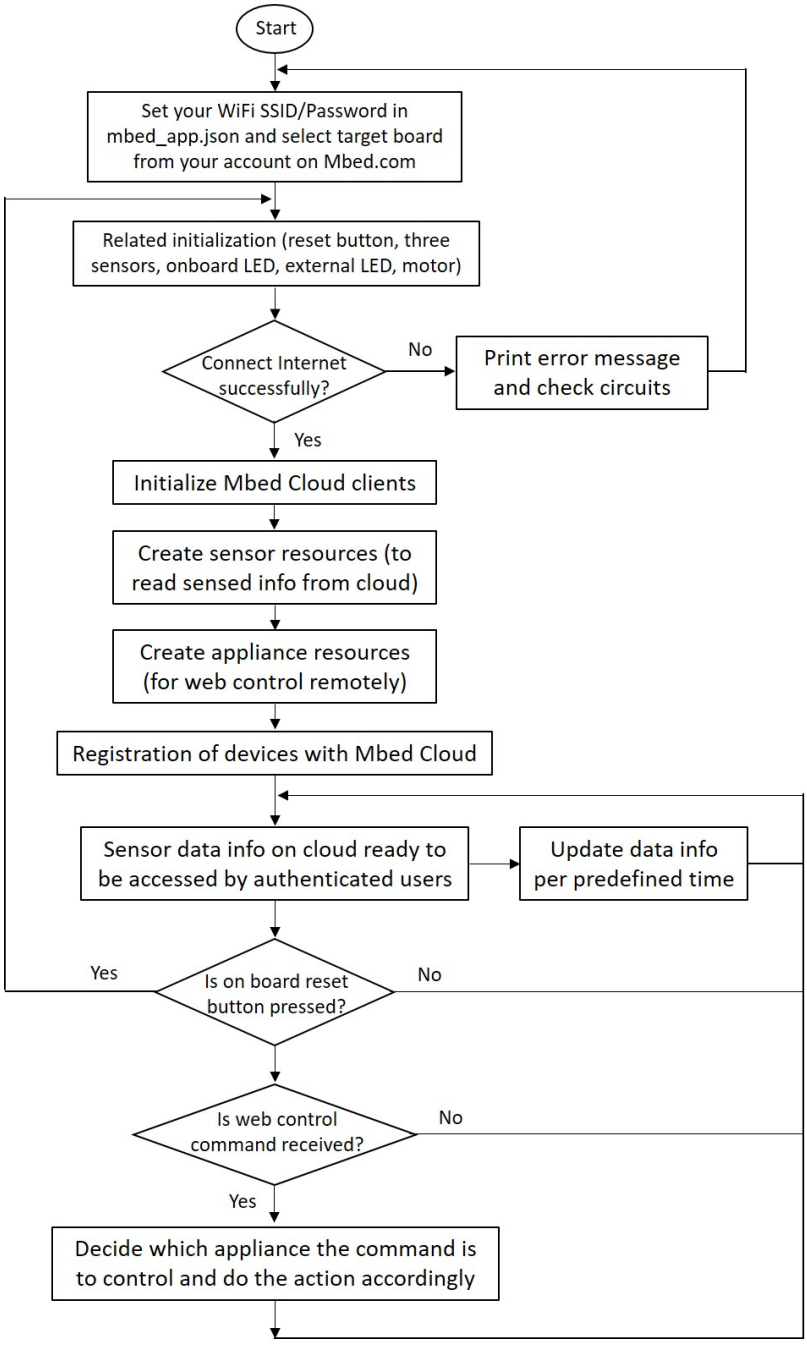
Figure 11 The flowchart of the GUI design for the IoT system



3.2 *Main program implementation involving microcontroller and pelion platform*

The main program of the IoT system is implemented using C++ with Mbed online compiler. The Mbed online compiler enables you to either write your code from scratch or import an existing project and modify it to suit your needs. The flowchart for the main program is shown in Figure 12, which involves the two major functions of the proposed IoT system: environment monitoring (temperature, humidity and atmospheric pressure) on cloud and remote web control through Pelion platform.

Figure 12 The flowchart of the main program for the IoT system



- *Implementation of environment monitoring:* The environment monitoring can be implemented remotely on web through the Pelion Device Management Platform, besides the local monitoring through GUI program. The data transfer between the hardware board and the Pelion cloud is through Wi-Fi. The Wi-Fi SSID (Service Set Identifier) and password is defined in a JavaScript Object Notation (.JSON) file, which is located in the root of the IoT application and can define new configuration parameters and override configuration parameters defined in libraries and the target. You also need to create an account on the Mbed website for using the Mbed online compiler and the Pelion Device Management Platform.

After the Wi-Fi is connected to the Internet successfully, the main program will initialise the three sensors through the function sensors_init() and send the sensed data values (temperature, humidity and atmospheric pressure) periodically through the function sensors_update(). The period is set in the main program. The function sensors_init() is given in Figure 13, where sen_hum_temp() and sen_press() are the functions that belong to the class HTS221Sensor and class LPS22HBSensor respectively. The two classes located in the header files 'HTS221Sensor.h' and 'LPS22HBSensor.h' in the main program have defined the methods init(), enable() and read_id().

The function sensors_update() in Figure 14 is called periodically according to the predefined time in the main program. The methods get_humidity(), get_temperature(), and get_pressure() are defined in the classes HTS221Sensor and LPS22HBSensor, respectively. The variable endpointInfo is a pointer used to receive the name information about the registered device. The res_temperature, res_humidity and res_pressure are the pointers of class type with class MbedCloudClientResource, which can be used to access the class's data members and member functions. The function set_value(float) is defined inside the class MbedCloudClientResource to get the name information of a resource. The class MbedCloudClientResource is defined in the header file 'mbed-cloud-client-resource.h', which is included in the main program.

Figure 13 The code for the initialisation of sensors

```
/* Initialization of three sensors */
void sensors_init() {
   uint8_t id1, id2;
   sen_hum_temp.init(NULL);  // Initialize temperature and humidity sensor
   sen_press.init(NULL);     // Initialize atmospheric pressure sensor
   sen_hum_temp.enable();    // Enable the temperature and humidity sensor
   sen_press.enable();  // Enable the atmospheric pressure sensor
   sen_hum_temp.read_id(&id1);  // get sensor id
   sen_press.read_id(&id2);
   printf("HTS221 temperature and humidity = 0x%X\n", id1); // for debug
   printf("LPS22HB atmospheric pressure = 0x%X\n", id2);
}
```

Figure 14 The code for the update of sensed data

```
/* Update sensors and report their values.
   This function is called periodically.      */
void sensors_update() {
   float temp_value = 0.0, hum_value = 0.0;
   float press_value = 0.0;
   sen_hum_temp.get_humidity(&hum_value);
   sen_hum_temp.get_temperature(&temp_value);
   sen_press.get_pressure(&press_value);
   printf("%2.2f^%2.1f^%3.2f\n",temp_value, hum_value, press_value);
   wait_us(5000); //Wait until the loop_timer reaches 5000us
   if (endpointInfo) { // name information about the registered device
   #ifdef SEND_ALL_SENSORS
      // Get the value of a resource as a string
       res_temperature->set_value(temp_value);
       res_humidity->set_value(hum_value);
       res_pressure->set_value(press_value);
   #endif
   }
}
```

To see the sensed data values from the Mbed cloud, one needs to create sensor resources in the main program. Before that, one needs to create an object of the class MbedCloudClient, which has a parameterised constructor with the same name as the class. When the object is created, the three parameters (net, bd, & fs) will be passed to the constructor. The parameter 'net' represents a connected network interface, 'bd' represents an uninitialised block device and 'fs' represents an uninitialised file system.

The class MbedCloudClient handles data transferring to the Mbed cloud through the application layer protocol LwM2M. The Pelion Device Management Platform enables efficient collection of sensor data and provides remote management capabilities and security features. The Mbed operating system integrates the LwM2M protocol for IoT devices so that the devices can communicate with Mbed cloud. Figure 15 shows a snippet code for creating sensor resources (specifically, the temperature sensor resource, the other sensor resources are omitted). The init() is the initialisation function defined in the class MbedCloudClient. The method function create_resource (const char *path, const char *name) is the pointer of class type with class MbedCloudClientResource. The pointer path is an LwM2M path in the form of 3303/0/5700, where 3303 is the numerical identifier (ID) of the object (temperature sensor), 0 represents the ID of the object instance (single instance) and 5700 represents the ID of the resource (sensor value).

The function set_value() is declared in the class MbedCloudClientResource to set the value of the resource to an integer. The function methods (unsigned int methodMask) is declared in the class MbedCloudClientResource to set the methods that can be applied on this resource. The parameter methodMask is the mask of objects of type M2MMethod, which is a namespace defined in the same header file where the class MbedCloudClientResource is defined. The namespace M2MMethod declares an enumerated type M2MMethod that explicitly defines four elements of enum (GET, PUT, POST, DELETE), which can be accessed from outside the namespace using the scope operator '::', e.g., M2MMethod::GET. The function observable() is declared in the class MbedCloudClientResource to set whether the resource can be observed. Up to now, once the hardware board is registered and connected to the Mbed cloud, one can click the corresponding resources to observe the data values (temperature, humidity, atmospheric pressure).

- *Implementation of remote web control:* The remote web control is to use Mbed cloud to remotely control application appliance to turn on or off. It is also implemented in the main program. Therefore, the prerequisite work (e.g., internet connection through Wi-Fi, related initialisation tasks) is the same as that for the environment monitoring task. In our proposed system, we implement the web control of three application appliances: an on-board LED, an external LED and an external motor, all representing some application appliances such as home oven, microwave, washer and air conditioner. More application appliances can be controlled through the Mbed cloud.

In the following, we focus on the implementation of web control for the on-board LED as an example. The implementation for the others are similar. First of all, we need to map the software code to the specified pins on the hardware board and declare pointers for access to Mbed cloud Client resources, as shown in Figure 16. The DigitalOut is a class defined in the header file 'DigitalOut.h', which uses the DigitalOut interface to configure and control a digital output pin by setting the pin to logic level 0 or 1. The first parameter (LED1, D7 or D13) is the pin name on the hardware board, and the second parameter (1 or 0) is the initial value. The res_led, res_led2, and res_motor are the pointers of class type with class MbedCloudClientResource, which can be used to access the Mbed cloud client resources.

Figure 15 A snippet code for creating sensor resources

```
/* MbedCloudClient handles registering over LwM2M to Mbed Cloud */
MbedCloudClient obj(net, bd, &fs);
int obj_status = obj.init();
if (obj_status != 0) {
    printf("ERROR: Mbed Client initialization failed (%d)\n", client_status);
    return -1;
}
// Sensor resources
res_temperature = obj.create_resource("3303/0/5700", "Temperature");
res_temperature->set_value(0);
res_temperature->methods(M2MMethod::GET);
res_temperature->observable(true);
......
```

Figure 16 The configuration of pins and declaration of pointers of class

```
// Pin configuration
DigitalOut led(LED1, 1);  // On-board LED - to simulate a home appliance
DigitalOut led2(D7, 1);   // External LED - to simulate a home appliance
DigitalOut motor(D13, 0); // External motor - to simulate a home appliance

// Declaring pointers for access to Mbed Cloud Client resources
MbedCloudClientResource *res_led;   // for On-board LED
MbedCloudClientResource *res_led2;  // for external LED
MbedCloudClientResource *res_motor; // for external motor
```

Figure 17 The implementation of a PUT handler

```
/* PUT handler
 * resource: the resource that triggered the callback
 * newValue: updated value for the resource
*/
void put_callback(MbedCloudClientResource *resource, m2m::String newValue) {
   printf("*** PUT received, new value: %s  \n", newValue.c_str());
   led = atoi(newValue.c_str());   // for on-board LED
}
```

Figure 18 The implementation of a PUT handler for the on-board LED control

```
// Creating resources, which can be written or read from the cloud
res_led = obj.create_resource("3201/0/5853", "LED State");
res_led->set_value(1);
res_led->methods(M2MMethod::GET | M2MMethod::PUT);
res_led->attach_put_callback(put_callback);
......
```

Then we implement the PUT handler for command transmission from the Mbed cloud to the remote hardware board. Figure 17 shows an implementation of the PUT handler for processing the on-board LED. The PUT handlers for the other two are similar and omitted here. The PUT handler is implemented by the function put_callback(). The first parameter is the pointer of class type that is used to access the Mbed cloud client resources. The second parameter defines an object 'newValue' from the class String, which and the namespace m2m are both defined in the header file 'm2mstring.h'. Then function c_str() returns a pointer to a C-style, null-terminated string representing the current value of the string object (i.e., newValue). The function atoi() converts the character string to an integer value which is assigned to the on-board LED to perform an action.

Finally, we create resources for the application appliances in the main program so that we can read information and write command at the Mbed cloud, as shown in Figure 18. The method function create_resource ('3201/0/5853', 'LED State') gives the resource name (i.e., LED State) and the three level tree of the LwM2M path (i.e., 3201/0/5853), where the first parameter is the object ID (3201 represents Digital Output), the second parameter represents the Instance ID of the object (0 means a single instance) and the third parameter represents the resource ID (5853 means multiple level output). The function methods() is the same as that in Figure 15 except that the parameter methodMask includes both GET and PUT elements of enum defined in the namespace M2MMethod. In this way we can use the two methods not only to retrieve the representation of the target resource but also to update the current representation of the target resource with the request value. The function attach_put_callback() is declared in the class MbedCloudClientResource to set a callback when a PUT action on this resource happens, which is triggered whenever a command is written to the resource from the Pelion Device Management Platform.

4 Experimental results

In this section, we present experimental results for the proposed IoT system. The normal experiment state should be as follows: the hardware board is connected to a laptop using a micro USB cable and connected to the Mbed cloud through Wi-Fi; the GUI program is run through Visual Studio in the laptop. The GUI program is developed for local environment monitoring. The Mbed cloud can work independently for environment monitoring and remote web control without any help from the GUI program.

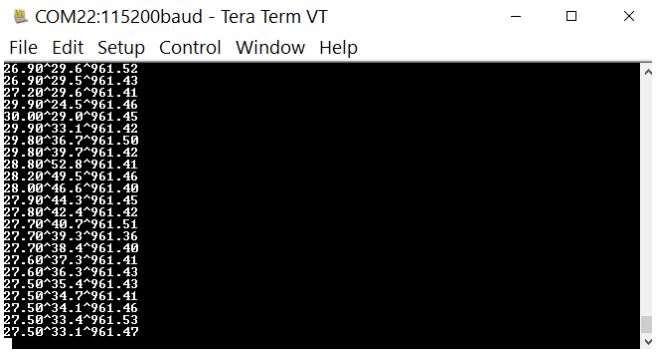
To start the experiment, you first need to use your Mbed account to login to the online Mbed compiler (or Mbed Studio if you have one installed on your laptop), and open your main program under your IoT project. Then you connect your hardware board to the laptop using the micro-USB cable. The online Mbed compiler should recognise the board and allow user to select it for main program (you should be able to see the board name on the top right of your online Mbed workspace. If not, you can manually choose it by the 'Add Board' function).

Next, you need to login to the Pelion Device Management Portal using unique device identity and connection parameters and to create an API key and new developer certificate and add them to your project. To enable remote firmware update with Pelion Device Management, you also need to apply a developer update certificate to your project and generate the corresponding private key for your project and then connect your board to the Pelion Device Management (Note that your Wi-Fi credentials have been set in the file mbed_app.json under your project).

Finally, you can apply the compile function on the online Mbed compiler system to create the application binary. The binary file is downloaded to a local folder and copied to the device folder (which appears as removable storage when the hardware board is connected to the laptop) to install the application. You may press the on-board reset button if needed. At this time, you are ready to see the device resources in both the GUI panel and the Mbed cloud.

In the following, we present selected experimental results for local GUI part and remote Mbed cloud part to demonstrate the functionality of the system. The GUI program has set the correct serial port name and the baud rate (here, it is 115200 bits per second) so that the serial port will automatically be connected when the GUI program is run. You may use any terminal emulator (e.g., Tera Term) to test the serial port communication from the hardware board. Figure 19 shows an example of environment monitoring using Tera Term to receive the sensed data values when the temperature and humidity sensor is touched by a finger occasionally, where the three data values in each row are the temperature, humidity and atmospheric pressure, respectively.

Figure 19 Environment monitoring using Tera Term



When the GUI program is run in Visual Studio, it will first bring you to a user authentication page, where you need to enter user name and password to login. Once the authentication is successful, it will bring you to the main GUI panel and automatically display the hardware board booting process in the Information Box and the environment monitoring parameters in the three graph boxes as well as in the three text boxes on the top, as shown in Figure 20. If it does not display the data values, press the reset button on the hardware board to reset the process.

Figure 20 shows the process of parameter changes when the temperature and humidity sensor is touched two times in the middle. Figure 20 shows the process of parameter changes when the temperature and humidity sensor is touched two times in the middle. We observe from the temperature plot that the temperature display starts at about 28.2°C, and changes to the first peak of 30.3°C when a finger is touched on the sensor and gradually decreases to 28.4°C when the finger is moved away, and then changes to the second peak of 31.4°C when a finger is touched again, and finally gradually decreases to 28.4°C again when the finger is moved away. Since the humidity sensing and the temperature sensing are integrated in the same sensor, we also observe the corresponding humidity changes in the humidity plot during the above temperature change process. The atmospheric pressure sensor is a separate sensor; thus we do not observe much change of the atmospheric pressure in the last plot, which is about 972 mbar.

Besides watching the sensed data values locally, you can also watch them on the Mbed cloud. Figure 21 shows the temperature change history from the Mbed cloud over the same period. We observe that the temperature increases to two peak points (due to the finger's holding) and finally stops at the same value (28.4°C) as that displayed on the local GUI panel. Similarly, we can observe other resources such as humidity and atmospheric pressure from the Mbed cloud. You can watch the change of environment parameters graphically on the Mbed cloud. Figure 22 shows a record of temperature change during a certain amount of time duration.

Figure 20 Environment monitoring using the GUI panel

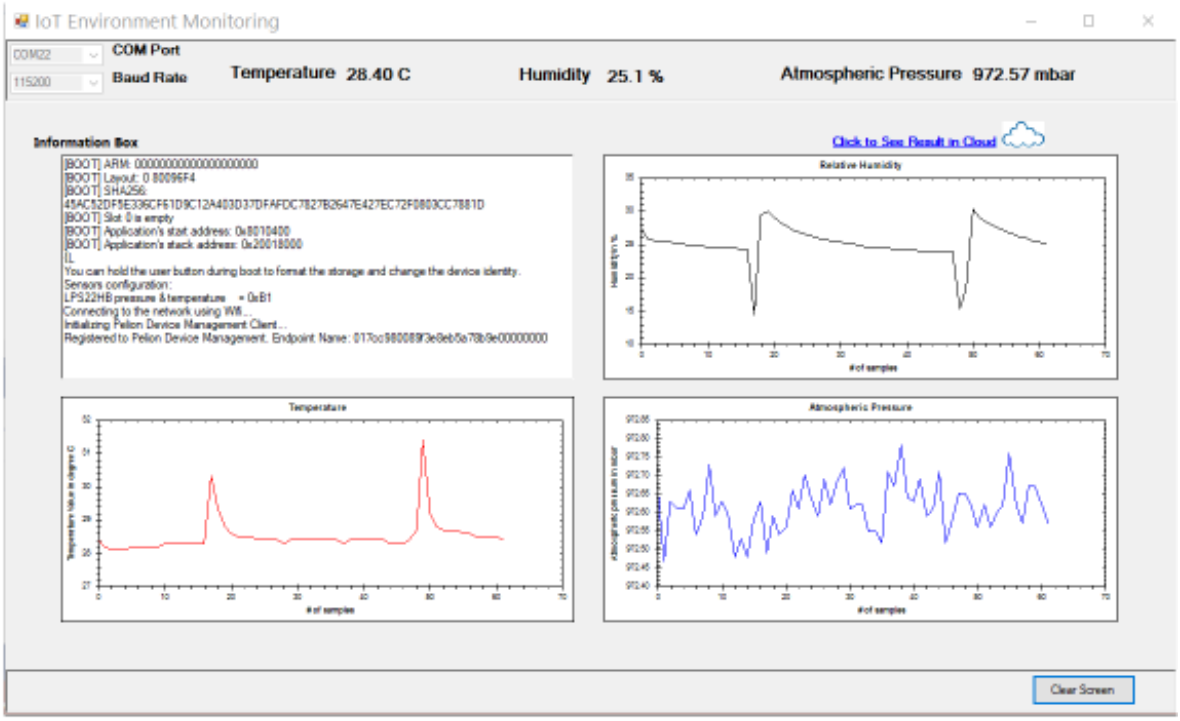


Figure 21 Environment monitoring of text form using the Mbed cloud

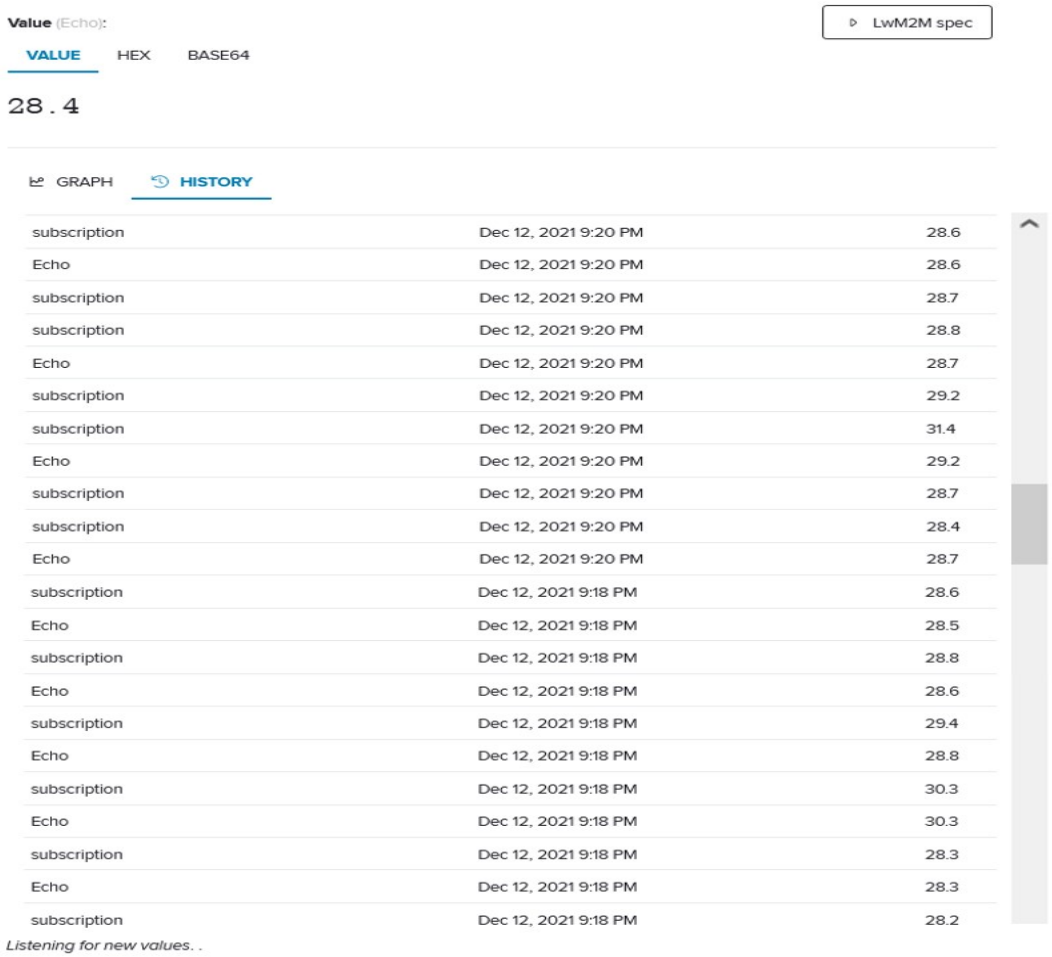
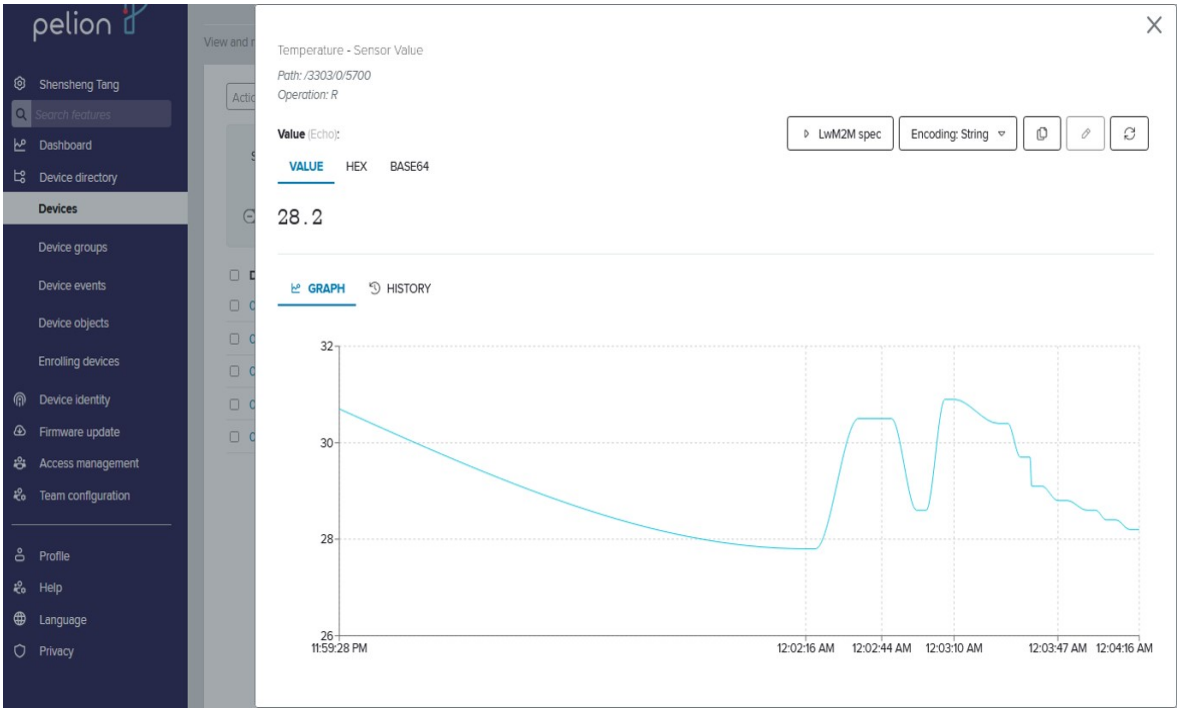


Figure 22 Environment monitoring of graph form using the Mbed cloud



Next, we show some experimental results for the remote web control through Mbed cloud. Figure 23 shows the state that the on-board LED is turned off. After we send a high-voltage command ('logic 1') to the on-board LED from the Mbed cloud, as shown in Figure 24, we observe that the on-board LED is turned on, as shown in Figure 25. Note that the path (3201/0/5853) in Figure 24 means the target resource is the on-board LED, which has been defined in Figure 18.

The remote web control of external LED and motor is shown in Figures 26, 27 and 28. Figure 26 shows the state that both the external LED and the motor are turned off. When we send a high voltage command from the Mbed cloud to each of them, the corresponding action will be performed remotely. Figure 27 shows the state that the external LED is turned on and Figure 28 shows that the external motor is turned on.

Figure 23 Remote web control with the on-board LED OFF

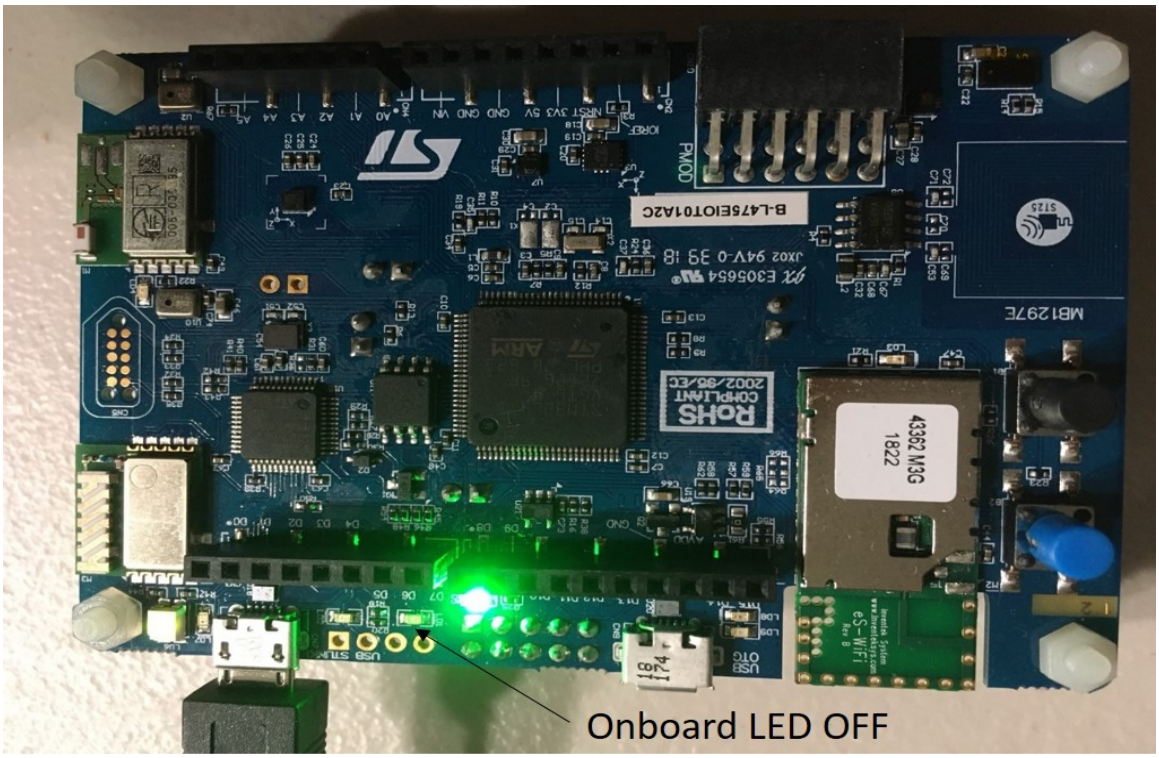


Figure 24 Remote web control with "ON" command sent to the on-board LED

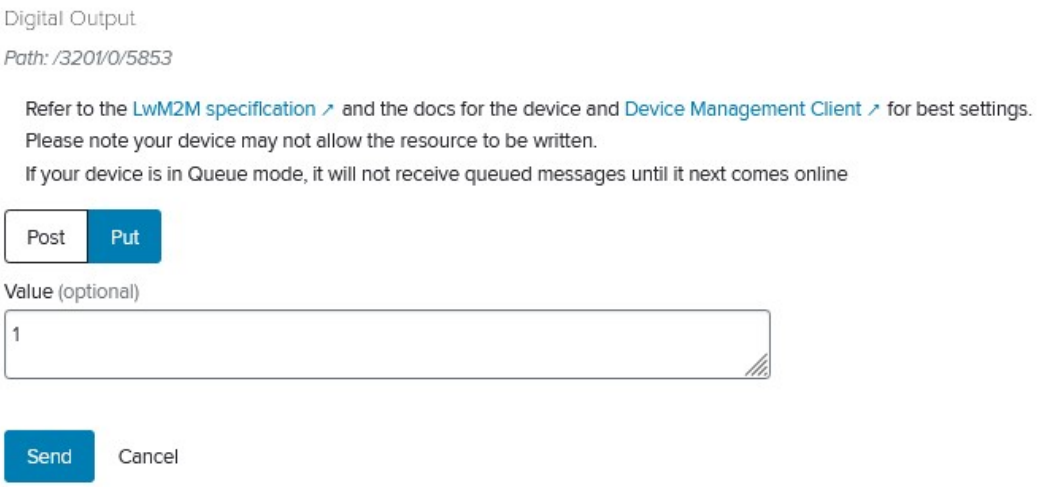


Figure 25 Remote web control with the on-board LED ON

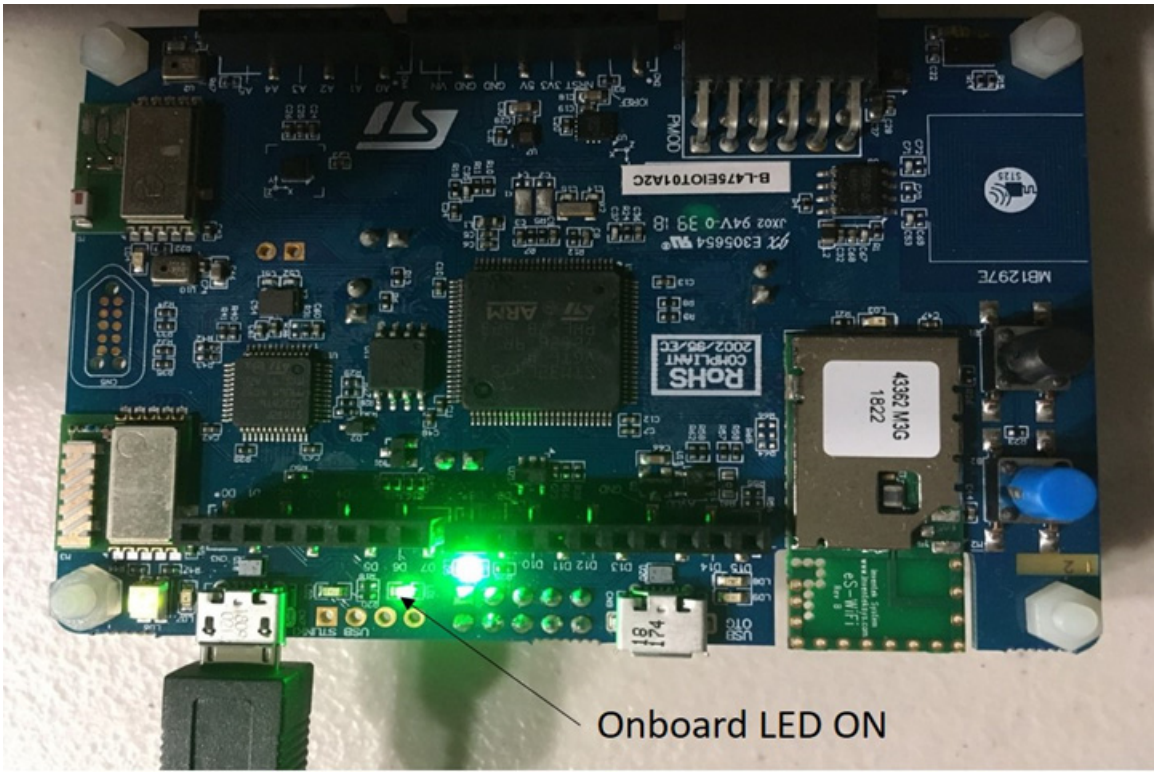


Figure 26 Remote web control with both the external LED and the motor OFF

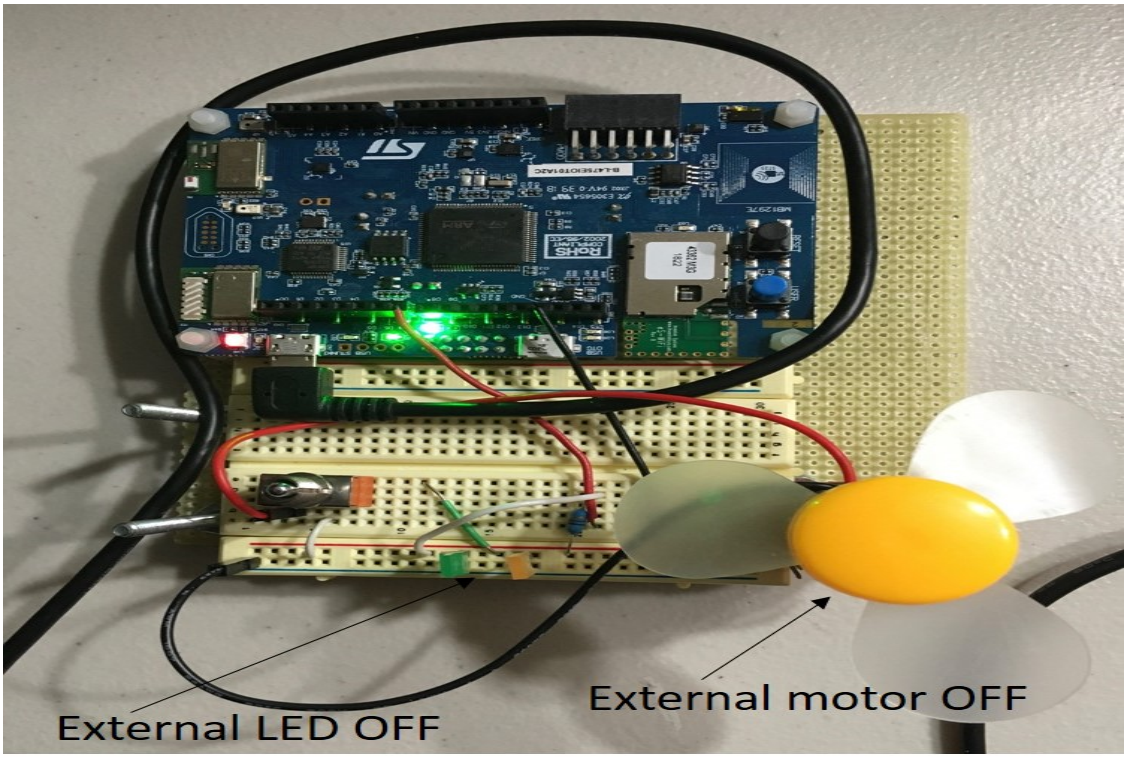


Figure 27 Remote web control with the external LED ON

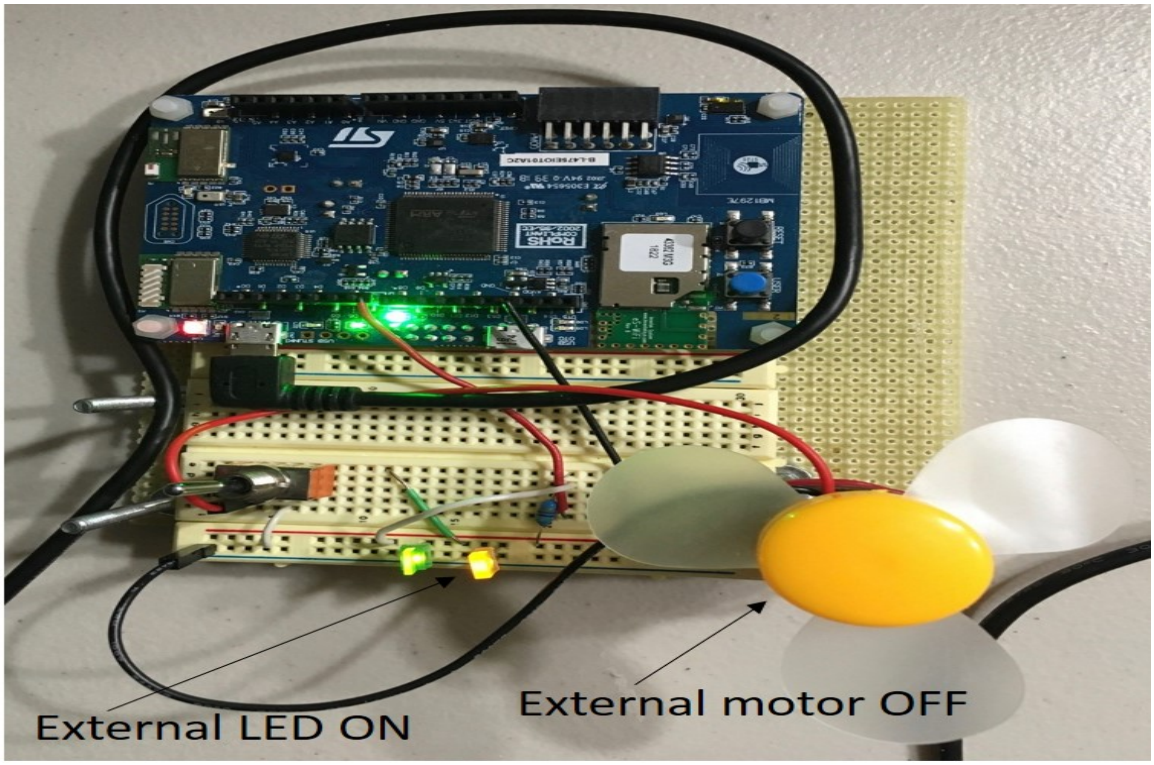
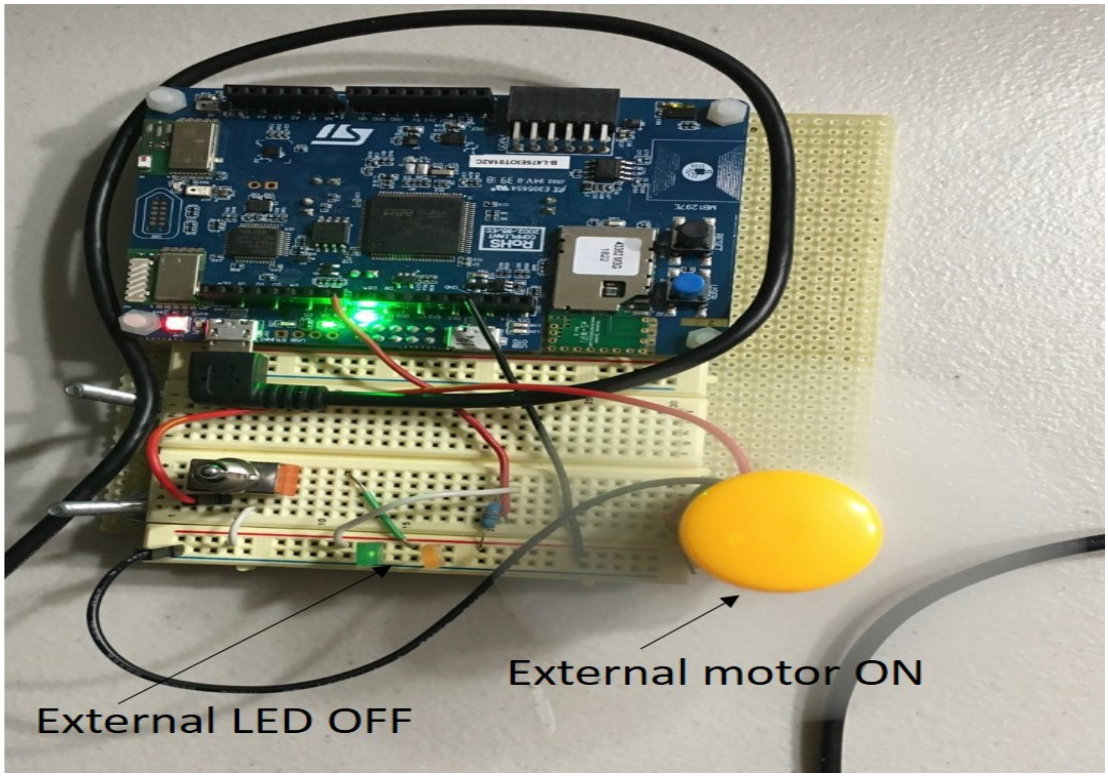


Figure 28 Remote web control with the external motor ON



5 Conclusion

We implemented an ARM Cortex-M4 core-based IoT system for environment monitoring and remote web control using the Pelion Device Management Platform. The hardware part of the embedded system is implemented using an ARM processor based STM32L4 Series development board integrated with multiple sensors and external circuit. The environment monitoring parameters (i.e., temperature, relative humidity and atmospheric pressure) can be wirelessly sent to the Mbed cloud managed by the Pelion platform. The sensed data values can also be sent to a GUI panel for local display. The GUI has a security access that needs user authentication. Three application appliances were designed for the purpose of remote web control though the Mbed cloud with a user account. The proposed IoT system has been successfully implemented with the main application program developed using C/C++ and the GUI developed by C# programming. The related experiment results were presented for further understanding of the implementation. For the future work, we consider to apply more application appliances to the proposed IoT system and try to implement the system on different cloud computing platforms such as Amazon Web Services (AWS). The work of hardware and software co-design can be a practical paradigm of engineering education for IoT hobbyists and college students.

Acknowledgement

The authors would like to acknowledge the partial support from St. Cloud State University, MN, USA through Faculty Improvement Grant No. 211208 and Emerson Automation Solutions Grant No. 629528.

References

Birje, M.N. and Hanji, S.S. (2020) 'Internet of things based distributed healthcare systems: a review', *Journal of Data, Information and Management*, Vol. 2, pp.149–165. Doi: 10.1007/s42488-020-00027-x.

B-L475E-IOT01A (n.d.) *STMicroelectronics L475E-IOT01A discovery kit data sheet*. Available online at: https://www.st.com/resource/en/data_brief/b-l475e-iot01a.pdf

Chauhan, H., Verma, V., Gupta, D. and Gupta, S. (2021) 'IoT-based automatic intravenous fluid monitoring system for smart medical environment', *International Journal of Computer Applications in Technology*, Vol. 66, No.2, pp.154–164.

CPU Cortex-M4 (n.d.) *ARM Cortex-M processor series*. Available online at: https://www.arm.com/products/silicon-ip-cpu/cortex-m/cortex-m4

Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P. and Sikdar, B. (2019) 'A survey on IoT security: application areas, security threats, and solution architectures', *IEEE Access*, Vol. 7, pp.82721–82743. Doi: 10.1109/ACCESS.2019.2924045.

Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S.L., Kumar, S.S. and Wehrle, K. (2011) 'Security challenges in the IP-based internet of things', *Wireless Personal Communications*, Vol. 61, pp.527–542. Doi: 10.1007/s11277-011-0385-5.

Izuma Networks (n.d.) *Pelion device management portal*. Available online at: https://portal.mbedcloud.com/

Khajenasiri, I., Estebsari, A., Verhelst, M. and Gielen, G. (2017) 'A review on internet of things for intelligent energy control in buildings for smart city applications', *Energy Procedia*, Vol. 111, pp.770–779.

Li, Y., Alqahtani, A., Solaiman, E., Perera, C., Jayaraman, P.P., Buyya, R., Morgan, G. and Ranjan, R. (2019) 'IoT-CANE: a unified knowledge management system for data-centric internet of things application systems', *Journal of Parallel and Distributed Computing*, Vol. 131, pp.161–172. Doi: 10.1016/j.jpdc.2019.04.016.

Liu, T., Yuan, R. and Chang, H. (2012) 'Research on the internet of things in the automotive industry', *Proceedings of the International Conference on Management of e-Commerce and e-Government*, Beijing, China, pp.230–233. Doi: 10.1109/ICMeCG.2012.80.

MBED (n.d.) *ARM Mbed cloud*. Available online at: https://os.mbed.com/

Microsoft Corp. (2019) *Visual studio*. Available online at: https://visualstudio.microsoft.com/

OMA SpecWorks (n.d.) *OMA lightweight M2M (LwM2M) protocol*. Available online at: https://omaspecworks.org/

Qiu, T., Xiao, H. and Zhou, P. (2013) 'Framework and case studies of intelligence monitoring platform in facility agriculture ecosystem', *Proceedings of the 2nd International Conference on Agro-Geoinformatics (Agro-Geoinformatics)*, pp.522–525. Doi: 10.1109/Argo-Geoinformatics.2013.6621976.

Sfar, A.R., Natalizio, E., Challal, Y. and Chtourou, Z. (2018) 'A roadmap for security challenges in the internet of things', *Digital Communications and Networks*, Vol. 4, No. 2, pp.118–137. Doi: 10.1016/j.dcan.2017.04.003.

Strielkina, A., Uzun, D. and Kharchenko, V. (2017) 'Modelling of healthcare IoT using the queueing theory', *Proceedings of the 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Bucharest, Romania.

Tang, S. and Xie, Y. (2021) 'Availability modeling and performance improving of a healthcare internet of things (IoT) system', IoT 2021, Vol. 2, No. 2, pp.310–325. Doi: 10.3390/iot2020016.

Wang, J.Y., Cao, Y., Yu, G.P. and Yuan, M.Z. (2014) 'Research on application of IOT in domestic waste treatment and disposal', *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pp.4742–4745. Doi: 10.1109/WCICA.2014.7053515.

Zanella, A., Bui, N., Castellani, A., Vangelista, L. and Zorzi, M. (2014) 'Internet of things for smart cities', *IEEE Internet of Things Journal*, Vol. 1, No. 1, pp.22–32.

ZedGraph (n.d.) *Open source software*. Available online at: https://sourceforge.net/projects/zedgraph/