AN IMPLEMENTATION OF FRAMEWORK FOR REMOTE ALARM SYSTEM USING AN ARDUINO MICROCONTROLLER

[1]HITESH PAMBHAR, [2]JINSUK BAEK

Department of Computer Science, Winston-Salem State University, Winston-Salem, NC, USA
E-mail: [1]Hpambhar116@rams.wssu.edu, [2]baekj@wssu.edu

Abstract - The Internet of Things (IoT) technology allows physical, electronic devices to be sensed, monitored, or controlled remotely across the available network infrastructure. Those devices are embedded with software, sensors, and actuators, which are able to inter-operate within the existing network domain. Since its autonomous nature results in economic benefit, the IoT is expected to be widely deployed in many applications requiring integration of physical digital world into computer-based systems. In this paper, we introduce our contribution to IoT revolution with a framework implementation to support remote alarm system using a most widely used IoT device, calledArduino.

Index Terms—Alarm system, Arduino, IoT, temperature sensor

I. INTRODUCTION

The IoT refers to the connection of things other than traditional computing devices to the Internet [1]. Currently, the IoT applications has been adopted to many fields and it can be broadly categorized into three different fields – 1) Smart home, 2) Smart car, and 3) Smart city.

- Smart Home – The consumer electronics have stimulated the rapid growth of applications in smart home environments. The home is now not just a place where a number of traditional digital appliances carry on simple operations and tasks, but a distributed system with many entities working together. Web TVs, home appliances, and even kitchen appliances can be connected through the IoT.
- Smart Car – The smart car is a next-generation vehicle that enhances convenience by integrating mobile devices and mobile communication technologies. These enhancement allows the car autonomously manage the vehicle information in real time manner. More importantly, along with artificial intelligent technology, it can greatly reduce car accidents or minimize damage from the accidents. Also, it will make possible to reduce the environmental pollution.
- Smart City – The extension of the IoT technologies to an urban context is an ultimately desirable, since it results in better use of the public resources, increasing the quality of the services offered to the citizens, while reducing the operational costs of the public administrations. At this point and at the beginning phase, a few cities are trying to adopt IoT solutions in the management of public affairs.
-

In this paper, we introduce a general-purpose, IP-based, client-server communicationframework to support remote monitoring systems that can be applied to many environment-sensitive applications in all above mentioned categories. Although the proposed framework is general-purpose system, we implement a temperature alarm system to show an example how the system can work with those applications.

II. SYSTEM STRUCTURE

As shown in Fig. 1, the proposed system consists of three components including 1) Arduino board equipped with a temperature sensor, 2) client module, and 3) server module.

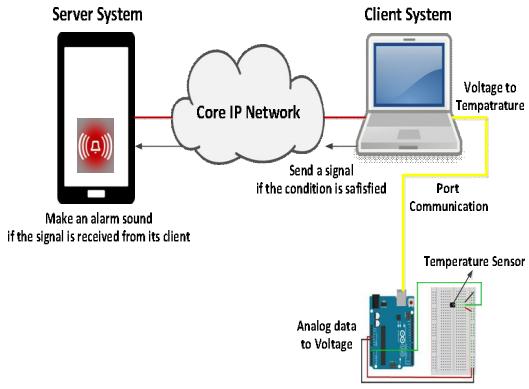


Fig. 1. System structure

With the system, we utilize Arduino board [2] equipped with temperature sensor as an IoT device. The analog temperature sensor measures raw temperature of the given environment and outputs a changing voltage depending on the temperature it senses. This operation is performed with the built-in ADC unit on the Arduino board. After converting the ADC reading to voltage, we further convert the voltage to temperature in degrees Celsius. The client system gets temperature information from the Arduino output and allows the system administer to autonomously set up threshold-based conditions depending on the measured temperature variations. Once the condition is satisfied, the client system will make notice to the remote server system which will make alarm sounds. This operation between the client and server is performed with IP-based communication.

III. SYSTEM COMPONENTS

A. Arduino Board

Arduino is a small microcontroller board with a USB plug to connect to computer systems. A simple interactive logic circuit of the board for our intended application is shown in Fig. 3 using a temperature sensor and a piezo. Accordingly, the corresponding board can be configured as shown in Fig. 4. Arduino operates as a power source for this circuit. For making a local alarm sound, we additionally connect piezo, which is a small element that vibrates and creates sound waves by displacing air around it when it receives electricity.

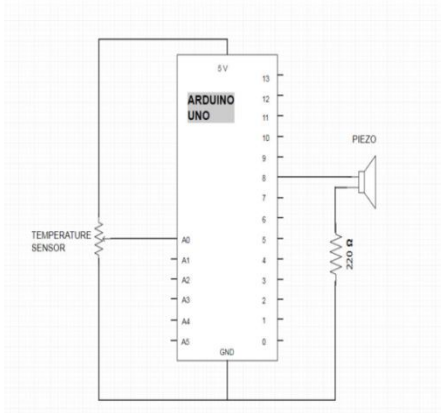


Fig. 3. Logic circuit of the Arduino board

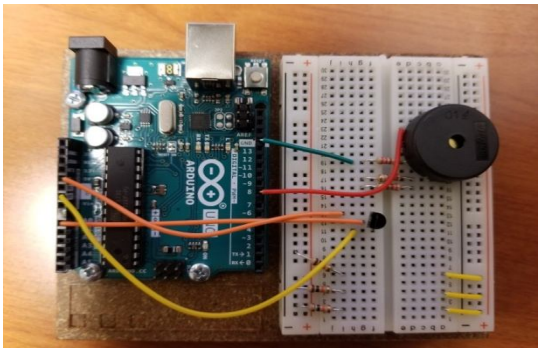


Fig. 4.Real Arduino board with a sensor and a piezo

The analogRead(A0) function take a voltage reading on. The value, which is between 0 and 1023, is a representation of the voltage on the analog reading pin. We now need to figure out what the real voltage on the A0 input pin is. The real voltage will be a value between 0 and 5 volts since the Arduino board provide maximum 5 volts between its 5V pin and its ground pin. We can calibrate this by

voltage = (value on the A0 pin/1024) × 5.0(1)

With the real voltage, we get the accurate temperature in degrees Celsius by Equation (2), which will be displayed on the Arduino IDE monitor.

temperature = (voltage – 0.5) × 100 (2)

In a nutshell shown in Fig. 5, we list the piece of code that reads the current temperature value from the Arduino board.

```
void loop() {
intsensorVal = analogRead (A0);
float voltage = (sensorVal/1024.0)*5.0;
float temperature = (voltage–0.5)*100;
Serial.println(temperature);
// customize code depending on the value of
temperature
delay (1);
 }
```

Fig. 5.Example code of temperature reading in analog

B. Client Module

Since Arduino microcontroller sends message to serial port and display it to the serial monitor, the client module should use a Java serial communication library to get the serial messages. For port communication between the Arduino board and client module,we need to adjust the serial port names according to the corresponding Arduino board, which is “COM3” in case of our Windows machine which is shown in Fig. 6.

```
private static final String PORT_NAMES[] = {
 "COM3"
};
```

Fig. 6. Port tuning for client module to the Arduino board

Fig. 7lists the piece of code that actually reads the serial port content from the Arduino IDE console and variable inputLine. Depending on the values of the inputLine, the client module connects to the server module. In our example, we set up the threshold to 5 degrees Celsius and use the UDP [3] communications for fast transmission. In order to invoke send() call, the client module sets the four parameters including 1) the message to send, 2) its length, 3) IP address of server, 4) port number at which server is listening.Since our goal is providing a framework, users can customize this part. For example, if the given application requires reliable transmission and guarantee of the in-order delivery, it might benefit from running TCP [4] instead of UDP.

```
public synchronized
voidserialEvent(SerialPortEventoEvent) {
if(oEvent.getEventType() ==
SerialPortEvent.DATA_AVAILABLE) {
try{ String inputLine=input.readLine();
//In our application, we implement a UDP datagram
socketprogram
                    // customize code depending on the
value of inputLine
} catch (Exception e) {
System.err.println(e.toString());
}
}
}
```

Fig. 7. Example of client module

```
DatagramSocket socket = newDatagramSocket
(1234);
```

```
byte[] buffer = new byte [256];
DatagramPacket alarm = newDatagramPacket (buffer,
buffer.length);
Socket.receive(alarm);
//customize code depending on the value of alarm
InetAddressclientAddress = alarm.getAddress();
intclientPort = alarm.getPort();
String data = “Acknowledgement”;
buffer = data.getBytes();
DatagramPacket ACK = newDatagramPacket (buffer,
buffer.length, clientAddress, clientPort);
socket.send(ACK);
```

Fig. 8. Example of server module

C. Server Module

Since our application handles time-sensitive information, we want to transmit data packets in a fast manner not worrying about the loss-tolerant aspect. Therefore, our data transmission is based on the UDP DatagramSocket. In real life, many applications use a hybrid mechanism,transmitting the critical data over TCP and rest of the data via UDP. In our application, for receiving a packet via UDP, the server module creates the socket object for invoking receive() call. Whenever it extracts the data from the received packet, it will make an alarm sound.For providing at least some levels of reliability, we require the server to give a feedback such as an acknowledgment.Hence, the server also needs to know client’s address and port number, which can be obtained from the packetpreviously received from the client. Thisoperation is shown in Fig. 8 with a java code.

CONCLUSION

We have proposed a framework to support remote monitoring system. The system demonstrates how the IoT devices can contribute to the next-generation IT world by seamlessly attaching it to the well-established, IP-based client-server communications. In order to validate the extensibility of the proposed framework, we further implement a temperature alarm system as a time-sensitive application.Based on the proposed framework, we plan to develop many other practical applications.

REFERENCES

[1] M. G. Kwak, J. Baek, and P. S. Fisher, “A Degree Constrained Spanning Tree-Based Device Discovery Scheme for Lightning Control Networks,” Proc. of 2015 International Conference on Internet of Things, pp. 1–2, Seoul, Korea, October 2015.

[2] Arduino, https://www.arduino.cc/ (last retrieved in August 2018)

[3] J. Postel, “User Datagram Protocol,” RFC 768, Aug. 1980.

[4] Information Science Institute, University of Southern California, “Transmission Control Protocol,” RFC 793, Sep. 1981.

★★★