Hosehold Environment Monitoring System

Evan Orenstein, Scott Melenbrink, Chris McKiernan, Leiquan Pan, Mingyu Cao

January 28, 2018

Abstract

With smart homes becoming increasingly popular, it is important to ensure that the sensors that form the nervous system of the home are as accurate and sophisticated as the applications that they enable. To that end, we have designed, implemented, and tested a full sensor network in a home for the purposes of measuring the precise surface temperature data and a back-end service to monitor the Solar Decatholon house's temperature data.

1 Introduction

The goal of our project was to create a high-quality wireless sensor network with a AWS dataset that we could analyze as well as make available for use and calculation. This was accomplished by creating an indoor/outdoor wireless sensor network that was continuously monitoring several pieces of environmental data. The data collected by that network was sent via a Raspberry Pi TelosB basestation to a AWS database server, where measured temperature stored. We also build a monitoring user interface that various data were calculated and shown.

2 Wireless Sensors

Our wireless sensor network consists of one basestation and multiple nodes. The Raspberry Pi that was destined to be the basestation is responsible for connecting the monitoring back-end service. The basestation connected with a Telosb node which is designed to be the receiver of the wireless sensor network. The receiver was responsible for collect all temperature data, and pass them to the Raspberry Pi3 through serial. Other nodes that were responsible for measuring data send their measured data wireless through a multi-hop network called Collection Tree Protocol based on TinyOS. The first design called for sending a custom packet every few minutes comprised of the ambient temperature and the node number. The second was the basestation app running on the TelosB that acted as a receiver for the Java application running on the RPi itself. The basestation app interprets the packets from the motes using built-in TinyOS utilities and then sends the data by serial to the RPi using another TinyOS tool. The Java application was designed for interpreting serial data and send them to the monitoring back-end.

It also provides a function which enable user to modify the measuring frequencies of the nodes. In order to implement this, we develop a two side transfer throughout the wireless sensor network. When the user attempts to starts the whole network, the user needs to pass a frequency parameter through the Java application mentioned before, then the Java application will update the value of the Dissemination packet on the receiver node, which is a periodically packet that travels throughout the network. Whenever a node detected the updated value of the Dissemination packet, we will use it as the new measuring frequency.

3 System Calibration

In order to achieve precise temperature data, we implement a method called System calibration. Because calibration part usually applied on the Java application, which is on the Receiver side, that means all the temperature measured by different sensors were calibration by a shared equation. A shared calibration equation definitely cannot meet all the temperature error caused by different sensors. Therefore we investigate into the methodology of calibration. Found a more accurate calibration method for wireless sensor network, which is to calibrate the whole system instead of just one sensor. In our first attempt, manage to reduce error by 75 percent. We also researched on the temperature sensor which can give us a more credible temperature data. We asked some professors in the Mechanical Engineering Department, and some of them gave their advice to us. Based on our needs, we pick RTD as our final choice to detect the ground true temperature. We want to minimize the error below:

$$\frac{1}{MN}\sum_{j=1}^{M}\sum_{i=1}^{N}(k*x_{i,y_j}+b-y_j)^2$$

In this equation, i represents the sensor number, y represents the ground true temperature, it calculates the whole system error. Then we use the linear regression the global minimum error for certain k and b. Our measured experimental data as follows:

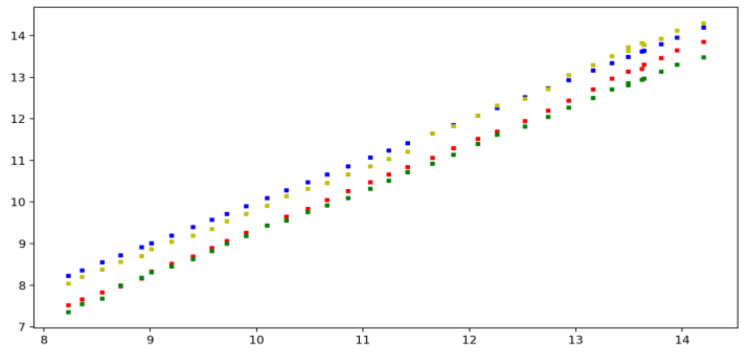

Figure 1: Experimental data for calibration

In Fig.1, x-axis represents the real temperature, y-axis represents the measured temperature. Different color represents different sensor. The blue one is the ground true temperature that measured by RTD temperature sensor. Experimental results shown on Fig.2.

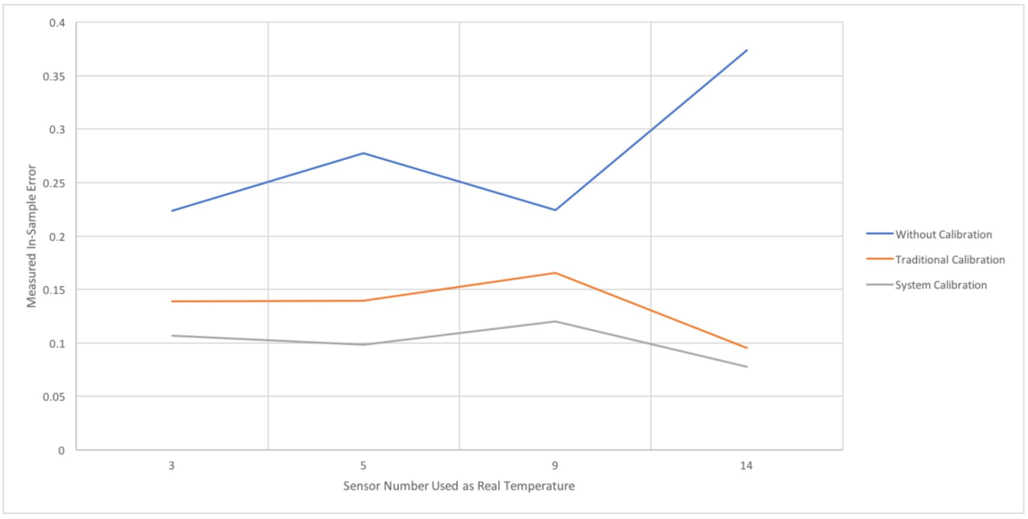

Figure 2: Experimental data for calibration

As the result shows, our system calibration has better accuracy than the original calibration equation done by sensor's development department. We also achieve better accuracy compared to average calibration method, which calculate k and b by averaging.

4 A Monitoring Back-end Server

Our Monitoring Back-End server use AWS IoT to connect each basestation. AWS IoT considers each basestation as a thing, and each basestation will use MQtt messaging service to publish their measured temperature data to their corresponding thing's shadow. We also use AWS DynamoDB to store all those temperature data for future use, the DynamoDB database will store every record of the temperature data received by the specified thing's shadow. The User Interface of our application is as follows:

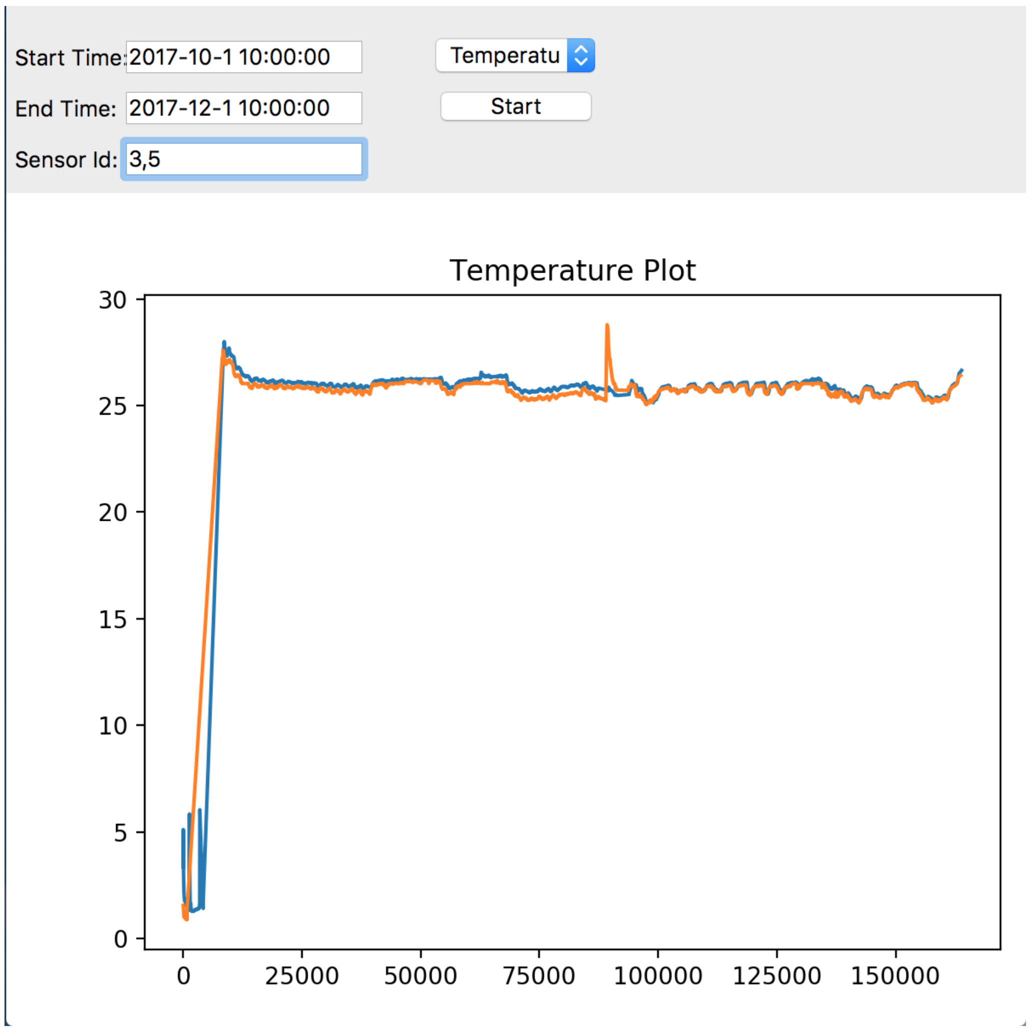
Start Time: 2017-10-1 10:00:00
End Time: 2017-12-1 10:00:00
Sensor Id: 3,5

Figure 3: User Interface

By the way, it's convenient to apply further calculation, as long as we obtain correct temperature data.

5 Solution to Measure Surface Temperature

Since our measuring nodes are using telosb mote module, which is a compacted development kit with radio, sensors and lights etc. combined together. We need some additional auxiliary to enable our sensor detect surface temperature. We use two things, stickers and thermal connector.

6 Conclusion

This project was developed upon a House Hold Temperature Sensing project created by a group of engineers on CSE520S class. Upon their achievements, Mingyu and I extend the wireless sensor network and build a monitoring back-end service. Right now, we have a multi-hop wireless sensor network, that is reliable to measure the surface temperature in a big house. For users, we developed a monitoring back-end service to store historical data and monitoring through a user interface. Furthermore, it's also feasible to implement further calculation for specific purpose.