Department of Electrical and Computer Engineering

CPE 490 Internet of Things

Final Project - Fall 2017

Sensor Network Design, Verification, and Implementation

Katelyn Charbonneau
Email: kc325844@wne.edu
Date: 12/14/17

Contents

Abstract

This project is a Internet of Things (IoT) system which is interested in two environmental factors: light level and temperature. The central piece to this project is the single-board computer BeagleBone Black. The BeagleBone collects data via attached hardware (sensors), performs required data processing, and allows that data to be accessible to devices on the same local network via Node-RED. The network senses when the light level of its environment has decreased below a certain threshold (it is dark) and begins taking temperature data while in this state. During experimentation, while the light sensor was sufficiently covered, the temperature censor began to read accurate data for the environment. In addition, the information was able to be retrieved remotely. As such, it is determined that the goals of the project have been met.

1 Introduction

The project provides a basic functionality: allowing the user to obtain accurate data regarding the light level and temperature of an environment. For example, an ecologist interested only in the temperature of an area at night (while the sun is down) could make use of this system as-is; they could install the sensors around an area and monitor them remotely, without having to worry about having to take them down or set them up during the day.

While the in-depth design will be explored in a later section, a basic flow will be described here. The user interacts with the BeagleBone's Node-RED interface by instructing the system to begin polling for data; the frequency that polls occur is changeable within Node-RED. Each time a poll occurs, a Python script on the BeagleBone is executed. This script first calls a C++ program that collects data from the attached hardware, then processes the raw data into a concise report. Node-RED then reads from this report and displays that information to the user.

The project contains a C++ executable compiled for ARM architecture, a Python script, some JavaScript in Node-RED, and this report was written in LaTeX. The BeagleBone Black runs a Debian Linux OS. The project has a GitHub repository available at:

`https://github.com/kaedec/CPE490-Final`

The Exploring BeagleBone repository created by Derek Molloy [1] was heavily leveraged in the creation of this project:

`https://github.com/derekmolloy/exploringBB`

2 Project Components

2.1 Hardware

In this section, the hardware component to the project will be explained. The first component is the light-dependent resistor (LDR). The LDR is attached to pin analog in 0 (AIN0) on the BeagleBone, and is also connected to a voltage regulator for safety. The temperature sensor, a TMP36, is attached to pin analog in 1 (AIN1) on the BeagleBone. The last component is a simple LED using a transistor switch to control current flow; this is also for safety. As the BeagleBone GPIO pins cannot sink/source very much current, part values had to be carefully controlled. The LDR and LED circuits run off of 5 Volts; the TMP circuit runs off of 3.3 Volts.

The schematic for the system is shown in Figure 1:

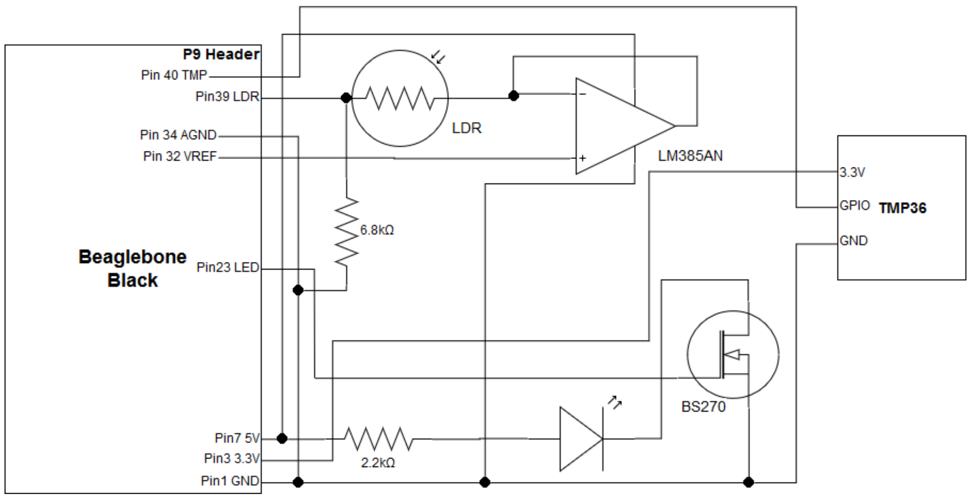


Figure 1: Hardware Schematic

2.2 C++ Program

The C++ program written for this project utilizes the GPIO class and some miscellaneous functions provided by Derek Molloy's repository. The code interacts with the files on the Linux OS related to the GPIO pins used in the project. In software, these are AIN0 (LDR), AIN1 (TMP), and GPIO49 (LED). It handles any I/O operations needed to interface with the hardware; it also performs the system's check to see if it considers the environment's light level dark enough. If it is, then the program will output data and turn the LED on to indicate that it collected data at the last poll. If it is not, then no data is output and the LED is turned off. The code was compiled in the Eclipse IDE on an Ubuntu virtual machine desktop. Because the desktop ran on Intel architecture and the BeagleBone is ARM architecture, a toolchain was used in the project's makefile to cross-compile the code for ARM architecture.

2.3 Python Program

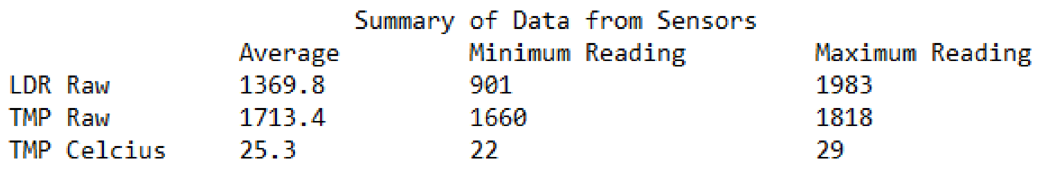
The Python program handles all of the data processing for the project. It also executes the C++ script. If the C++ file sends a data output, that information is snagged by the Python script and added to a raw data .csv file. The script then opens that file and calculates the average, minimum, and maximum values for each of the following variables: LDR Raw, TMP Raw, TMP Celsius. Raw values are on a scale from 0 to 4095; they are effectively the scaled voltage measurements. The Celsius value is the read temperature converted to Celsius. The script finally writes the processed data to a report file.

2.4 Node-RED

Node-RED is used as the user interface to the system. as described in the Introduction, the user requests (polls) the Python script to run at a defined interval. Node-RED then reads the report file written by the script and prints it into its debugger menu. While not very glamorous, the necessary information is conveyed over a wireless network.

3 Results

In this section there are results from a sample of 103 polls taken during testing. Note that during this experiment, the interval at which polls were done was changed for testing purposes, and is not reflected in Figure 3. As such, as the apparent curves in the graph should not be taken as literal. However, they can still be quite useful: cases where the temperature data increased were when the sensor was being covered by the tester's fingers, thus increasing its temperature. Large gaps in the curve are interpreted as moments when the LDR was not covered enough to be taking data. In this case, the temperature sensor is still changing values, but no data is being taken.



```
                     Summary of Data from Sensors
            Average         Minimum Reading        Maximum Reading
LDR Raw     1369.8          901                    1983
TMP Raw     1713.4          1660                   1818
TMP Celcius 25.3            22                     29
```

Figure 2: Output file from experiment which is read by Node-RED

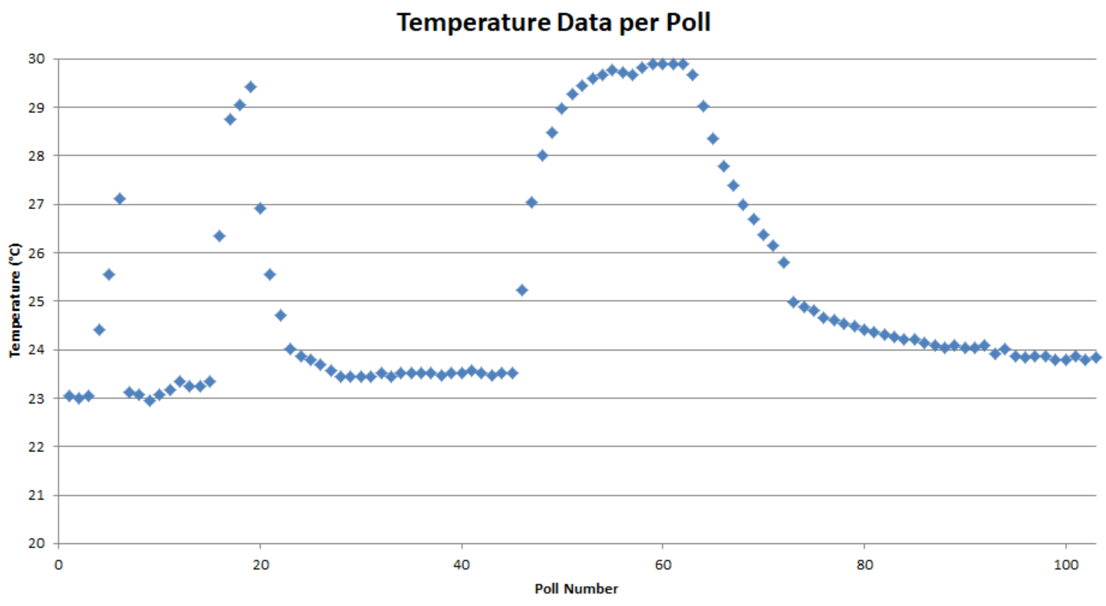


Figure 3: Graph of the temperature data taken in experiment

4 Conlusion

While the project was successful in what it set out to do, it could also be made significantly better. One issue is that Node-RED requires the use of absolute paths for directories. This project works on the system it was tested on, and no others, unless they have the same directory structure. Finding a way to use relative paths with Node-RED would be desirable. Also, the output in Node-RED is only in the debugger. This is not a very clean or easy way to read the data, so some visual improvement or creation of a graphical user interface would fix this issue.

The project does not particularly do anything too useful. A large improvement could be adding even more actuation onto the temperature sensor, such as activating a devices that will raise or lower the temperature if it desired to be in a certain range. However, as is, the project accomplishes its goal.

References

[1] Derek Molloy, *Exploring BeagleBone: Tools and Techniques for Building with Embedded Linux*, Wiley, 2014, ISBN:9781118935125.