CIT 595 Spring 2018
Group Project

INTRODUCTION

In this project, you will work in a group to develop a system with "embedded" and "web" components. The goal of the project is to create a web application that is able to get data from and send control information to a remote sensor and display driven by an Arduino microcontroller.

REQUIREMENTS

System architecture

The system shall consist of three major components, described as follows:

- *Sensor and Display*: You will use a Gravitech 7-Segment Shield attached to an Arduino microcontroller. This component includes a temperature sensor, a seven-segment display (like a digital clock), and an RGB light. You may start with the Arduino program that is available in Canvas. This program reads the temperature and writes to the serial port once a second.
- *Middleware*: The Arduino shall be connected via USB to a Linux or Mac machine, which will run a C program that handles all communication between the sensor and the user interface.
- *User interface*: The user interface will be a web application using HTML and JavaScript, which communicates with the middleware over the Internet using HTTP.

Note that because of firewall restrictions on the SEAS network, if you are running your middleware component on a Linux machine in the lab, you must use port 3001 for your server, and it can only be accessed by devices on one of the Penn networks. Please be sure to discuss this with a member of the teaching staff if you are unsure.

Functional requirements

Your system must fulfill *all* of the following requirements:

- The user should be able to see the most recent temperature sensor reading.
- The user should be able to see the average, low, and high temperature sensor reading for the past hour; if the sensor has been running for less than an hour, the user should be able to see the statistics for the time since the sensor started running.
- The user should be able to decide whether to see temperature readings and statistics either in Fahrenheit or Celsius. Additionally, the user should be able to change the 7-segment display on the sensor to show the temperature either in Fahrenheit or Celsius.
- The user should be able to put the sensor into a stand-by mode in which it is not reporting temperature readings to the user interface or to the display; however, during this time, the readings should be tracked for determining the high, low, and

average. When the sensor is in stand-by mode, the user should also be able to tell it to resume reporting the readings.
- If the middleware cannot get a reading from the sensor (e.g. because the sensor is disconnected), an appropriate message should be shown to the user the next time the user interface tries to get a reading.
- If the middleware becomes disconnected from the sensor but it is still running, it should be possible to reconnect the sensor and resume normal operations without needing to restart the middleware.

Note that you have complete freedom in how you implement these features in the user interface, i.e. what the user needs to do in order to see the data or send a command, and how it gets displayed. However, all user features need to be accessible via a standard web browser, e.g. Chrome, Firefox, Safari, or IE.

Also, please note that all Arduino code should be implemented in a single Arduino sketch (application), since Arduino can only run one program at a time.

Middleware implementation
You may use the *server.c* file that is available in Canvas as the starting point for your middleware. This is a very simple implementation of a web server that accepts an HTTP request from a web browser, prints the entire request to stdout locally, and then sends an HTTP response back to the browser consisting of some basic HTML.

When you run the program, you need to specify a port number on which the server should listen. This should be given as a command-line argument. Note, however, that you cannot use port numbers below 1024 on most systems.

You can test the server by connecting to the host:port using a web browser. For instance, if you start the server on port 3001, you could open a browser on the same computer, request **http://localhost:3001/test.html**, and see some simple HTML sent back as a response.

After sending back the response, the server then closes the connection and shuts down. You will need to modify the code so that it continues waiting for and handling new requests in a loop. You may need to experiment a bit to figure out what goes into the loop and what should only be done once.

Note! After your program stops, you may see this if you try to restart it right away:
`Unable to bind: Address already in use`
This just means that the operating system hasn't yet realized that your program is done and that it's okay for another process to use that port number. You can choose a different port number, or just wait a few seconds and the port number should become available again.

Also! If you use Firefox or Chrome to test your web server, you may see requests for a file called "favicon.ico". This is automatically requested by the browser so that it can put a little icon in the address bar. If this is causing you problems, it's okay to send nothing

back to the browser when that file is requested, e.g. ignore the request, send back an empty response, etc.

Another limitation of the code we have provided is that if it is handling an HTTP request and another request comes along, the second request needs to wait until the first one is complete before it will be handled. You will need to modify your program so that, after accepting a new incoming request, the program starts a new thread to handle that request and send back the response, and then the first thread goes back to waiting for incoming connections. You will need to consider how to pass the necessary information to the new thread and how to make sure your code is threadsafe.

Shutting down the middleware

In order to gracefully stop your middleware, there should be a thread running on the middleware that waits/blocks until a user sitting at the terminal where the middleware was started enters the letter 'q'. Once it sees that, the middleware program should then terminate. It is okay if you require that one more request come in from the user interface before ending the program.

Additional features

In addition to the functional requirements listed above, your group must come up with *three* other (non-trivial) features to include in your system:

- At least one feature must involve sending a control message from the user interface to the display
- At least one other feature must involve the processing and/or display of the data that is sent from the sensor to the user interface
- At least one feature must involve some non-trivial amount of JavaScript in the web app to handle user interaction and/or the display of data, i.e. there should be some dynamic content in the web app

Your additional features must be approved by a member of the instruction staff, preferably during the Milestone #1 demo described below.

MILESTONES

It is important that you make steady, continuous progress towards your final system. There are various milestones that your group should meet, as described below.

Arduino Tutorial (March 22)

During today's recitation session, your group should complete the Arduino tutorial and ensure that your middleware can receive temperature readings from the sensor.

This will not be graded, but you need to at least get the infrastructure in place so that you can get data sent between the components.

Milestone #1 (March 29)
During today's recitation session, your group will have a 5-10 minute meeting with a member of the instruction staff to show that it is possible to display the current temperature from the Arduino sensor in the user interface, which retrieves it from the middleware using an HTTP request.

Also during the recitation, you will be asked about the additional features that you plan on implementing. These features need to be approved by a member of the instruction staff so please discuss them with your teammates before the start of the recitation.

This milestone is worth 10% of the project grade.

Milestone #2 (April 12)
During today's recitation session, your group will have a 5-10 minute meeting with a member of the instruction staff. By this point, you should have much of the basic functionality completed: in particular, you will be required to demonstrate that you can send some control signal from the web app to the Arduino via the middleware, and have the Arduino take action based on that signal.

During this meeting, be prepared to discuss issues such as:

- where does data processing (e.g., calculating the average temperature) occur?
- what is the structure of your communication protocols between the different components?
- what sorts of error handling have you included in your code?

This milestone is worth 20% of the project grade.

Final Presentation (April 24)
Your group will present your system to members of the instruction staff and discuss your implementation decisions. At this point, the entire system should be working and you should have all features implemented. If not, you may present the project at a later date, but with a late penalty to be decided by the instructor.

This milestone is worth 70% of the project grade.

ACADEMIC HONESTY

You should be working with the other members of your group project team, obviously. Other than that, this project is subject to the same policy as for all other assignments, as described in the Syllabus.

In particular, you may not discuss or share code with students in any other groups, or with students who have taken this course in the past. If you want to use a third-party

library, or use any code that you did not write yourself, you **must** get permission from the instructor. Failure to do either of these will be considered academic dishonesty, and your team will receive a grade of 0 on this project. Which would be really, really bad.

If you run into problems, please ask a member of the teaching staff for help before trying to find solutions online!

SUBMISSION

All deliverables are due in Canvas by **Tuesday, April 24, 11:59pm.** Late submissions will be subject to a penalty determined by the instructor.

One member of your group should submit all Arduino, C, and web app source code in three *separate* zip files. If you use third-party libraries (which have been approved by the instructor!), only submit those in compiled form. Please be sure that all of your code is well-commented so it is easy for the grader to find where different features are implemented. Remember: happy graders give higher grades!

Also, you must return the Arduino board, temperature sensor shield, USB cable, and any other equipment before you can receive a grade for this assignment.

Updated: 21 March 2018, 5:27pm