Project 4: Embedded Systems

PROJECT GOALS

This project will familiarize you with some simple distributed systems characteristics and tools. It will also give you some experience in working with hardware for embedded systems and provide some exposure to basic tools for securing distributed systems. The goals are:

- •primary: build a simple client end of a client/server distributed system.
- •primary: learn about working within the constraints of devices designed to support embedded systems.
- •primary: learn how to use standard tools to provide secure encrypted communications between a client and server
- •secondary: build a program to a specified protocol interface.
- •secondary: obtain experience working with simple peripheral devices.
- •secondary: obtain experience with simple networking debugging.

ASSIGNMENT OVERVIEW

The assignment is divided into three general parts.

1. Building an application that supports the use of a sensor to gather data on an embedded device.
2. Convert the application that interacts with the sensor to become a client using a predefined network protocol to interact with a remote server program.
3. Change the basic client application to make use of SSL/TLS to communicate securely to a remote server that requires cryptographic protection of communications.

In this assignment, you will:

- •Learn how to perform basic operations on the Intel Edison.
- •Learn how to connect simple sensor devices to the Edison and access them from a local application.
- •Implement and demonstrate an application that uses sockets to communicate with a shared server application on a remote machine.
- •Learn how to convert socket communications to use SSL/TLS to provide cryptographic protection of communications.

Your deliverables for this assignment will include:

- •A program that integrates a sensor into the basic Edison platform and gathers readings from that sensor.
- •A test run in which your program interacts with the remote server and performs all supported operations specified in the design for the application. This test run will store a log file on the remote server machine, which will be used in grading your assignment. You will also provide a log file from your Edison of the run.
- •A test run in which the SSL/TLS version of the program interacts with the secure remote server and performs all supported operations specified in the design for the application. This test run will store a log file on the remote server machine, which will be used in grading your assignment. You will also provide a log file from your Edison of the run.

To perform this assignment, you will need to learn about the Edison platform. There are several useful tutorials on working with the Edison that you might find helpful available on line:
Edison tutorials
Also, you will need to use a temperature sensor in the Grove sensor kit for this project. Here's a link that provides some information on using this sensor:
Temperature sensor information

PART 1 - Building a sample Edison embedded device

Summary of Deliverables

- the source for a C program and *Makefile* that cleanly (no warnings) builds using *gcc* on an Edison Linux system, implementing the functionality specified below.
- The contents of a log file showing the program operating on your Edison for at least 60 seconds.

Detailed Instructions

Write a program that uses the Edison to access the temperature sensor included in the Grove sensor kit. The program should read the sensor once per second and output its reading (in Fahrenheit) to a shell attached to the serial port, as the tutorials indicate. Also output these readings to a log file, in the format:

Timestamp Temperature

(Use a space between the timestamp and temperature, not a tab or multiple spaces.) The timestamp should be obtained by running the time() system call on the Edison, and should be converted to an HH:MIN:SEC format. The temperature should be in the format ##.#. (For example, 98.6.) Take measures to ensure that you measure several different temperatures. (Hint: holding your finger on the sensor is likely to produce a hotter reading than the air temperature.)

PART 2 - Integrate your Edison sensor device into a client/server system

Summary of Deliverables

- the source for a C module and *Makefile* that cleanly (no warnings) builds using *gcc* on an Edison system and implements the functionality specified below.
- A log file showing the temperatures your device sent to the server and all commands received from the server.

Detailed Instructions

The behavior of the basic Edison client and its interactions with the server can be found in a document on the class web page.

PART 3 - Convert your Edison sensor client program to use SSL/TLS to protect communications

Summary of Deliverables

•the source for a C module and *Makefile* that cleanly (no warnings) builds using *gcc* on an Edison system and implements the functionality specified below.
•A log file showing the temperatures your device sent to the server and all commands received from the server.

Detailed Instructions

The behavior of the SSL/TLS Edison client and its interactions with the server can be found in a document on the class web page.

GRADING

Part 1 - 20% of overall grade

Value	**Item**
5%	Operational Edison with integrated temperature sensor that can run a new program
5%	Properly written makefile and clean program build
5%	Proper output and timings of temperature readings to log file and log file contents
5%	Proper output of temperature readings to shell

Part 2 - 50% of overall grade

Value	**Item**
2%	Clean build of program and proper makefile
3%	Successful initial connection to server
5%	Basic send of temperatures to the server
5%	Correct log file contents from client side
5%	Correct log file contents from server side
5%	Proper handling of STOP command
5%	Proper handling of START command
5%	Proper handling of SCALE command (including correct temperature conversion)
5%	Proper handling of PERIOD command
5%	Proper handling of DISP command
5%	Proper handling of OFF command

Part 3 – 30% of overall grade

3%	Clean build of program and proper makefile
5%	Successful initial connection to TLS server
5%	Correct log file contents on client side
5%	Correct log file contents on server side
12%	Proper handling of commands (2% each)