Daniel Diamont – dd28977

Robert Noe – rn6678

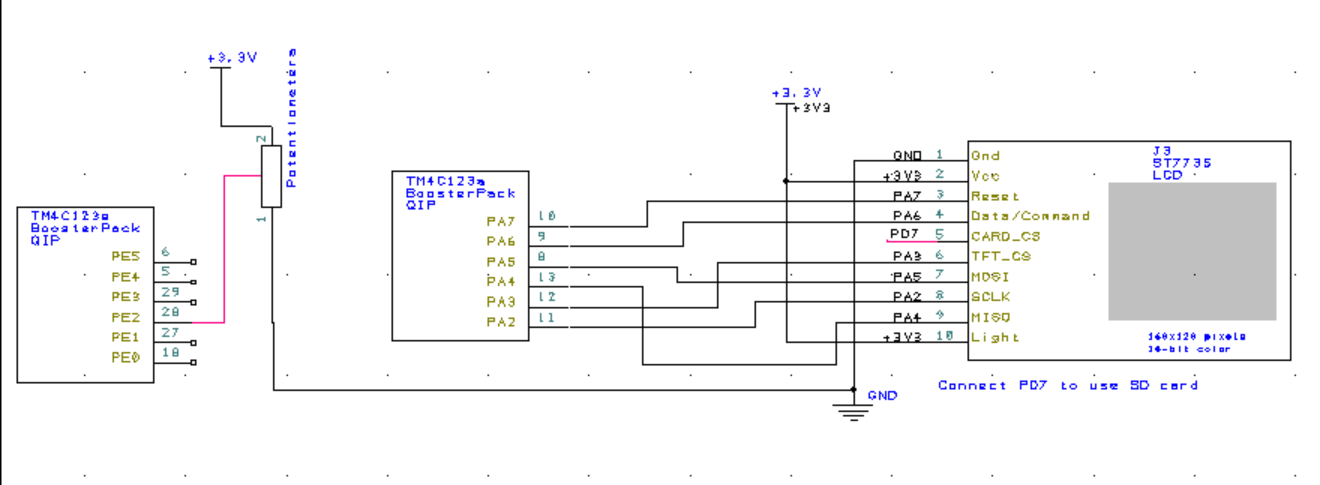
EE445L

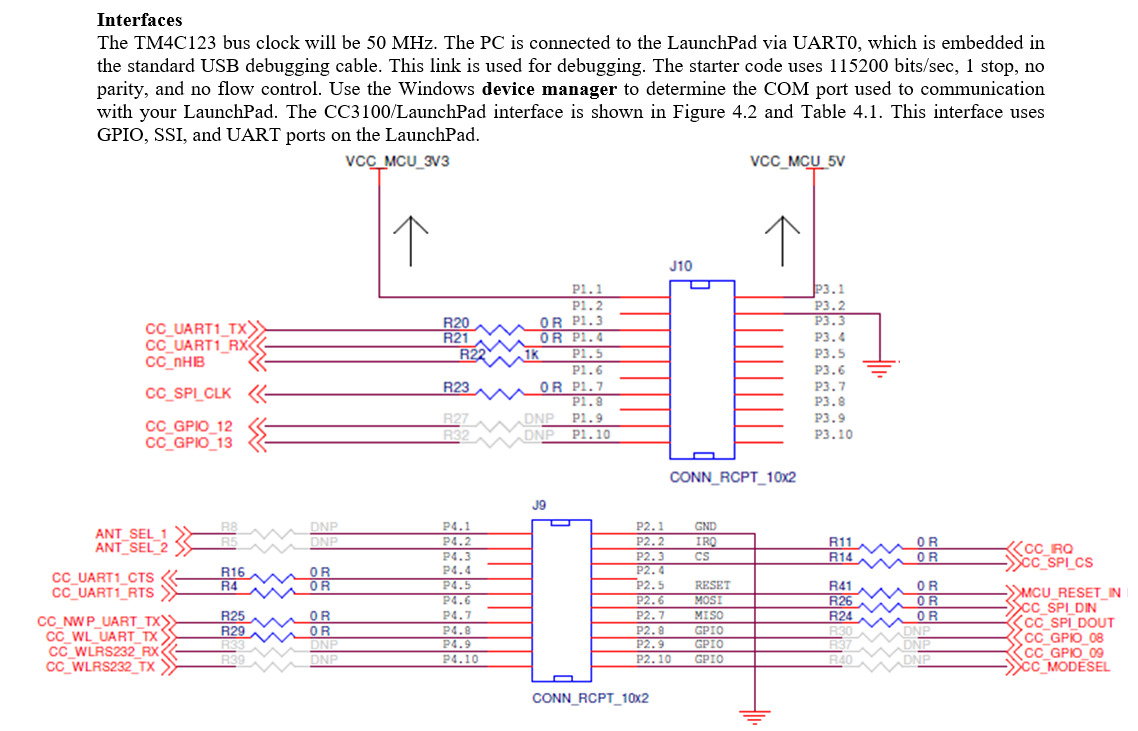
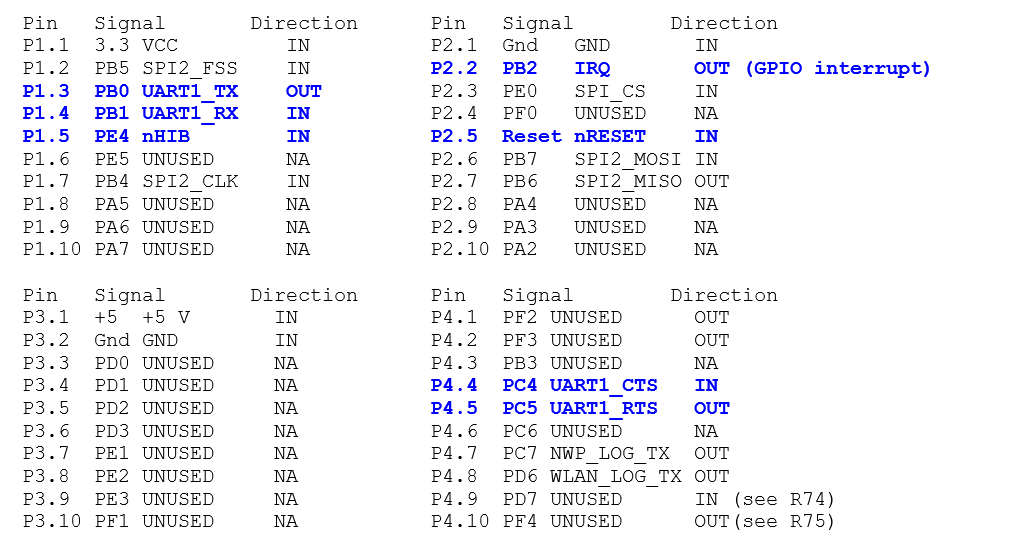
Lab 4C – Wi-Fi

1. **Objectives**

* Implement a system that connects to the internet via an IEEE 802.11 – **WiFi** module, CC3100
* Use DNS to convert name to IP address
* Configure a smart object that can retrieve data from a weather server using TCP
* Design a smart object that can store data onto an internet server using TCP
* Implement a web server to log data from your smart object

1. **Hardware Design**

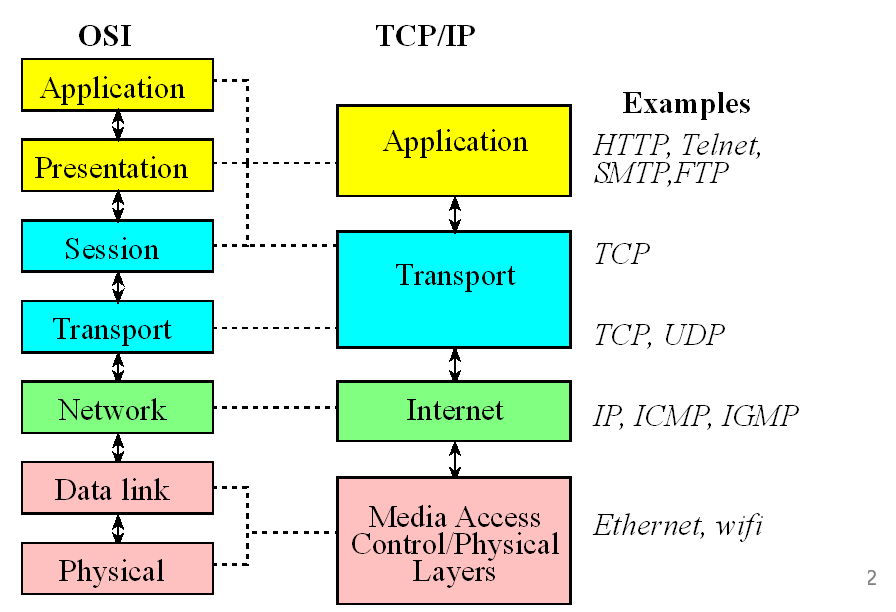




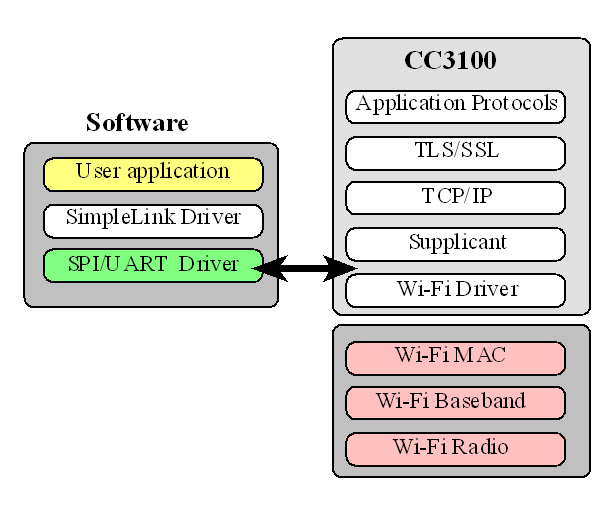
1. **Software**

Please see software attached to .zip file. The rest of this section shows graphically how our software interfaces with the CC3100 to send data to external servers.

Our software is the application layer of the TCP/IP stack, and we use the SimpleLink framework and SPI/UART drivers to send data to the CC3100, which can wrap our data in a TCP packet and add an IP header, and actually transmits our data over the airwaves to the Access Point specified in the Application Layer. In this manner,

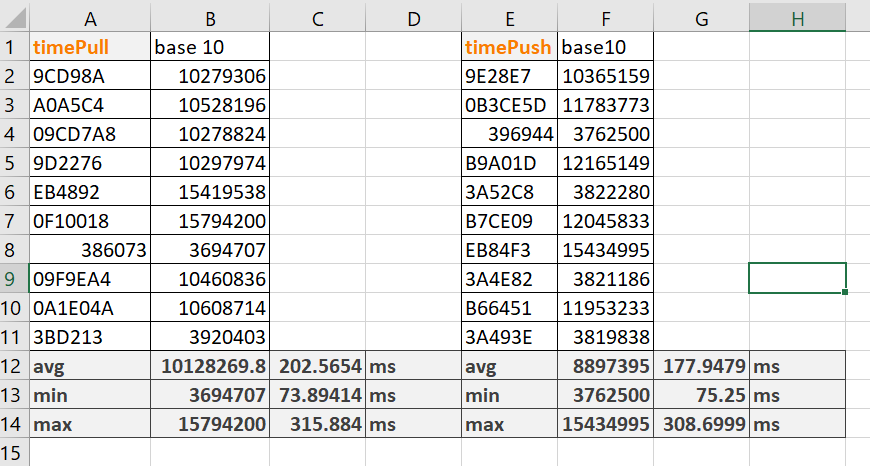


From Dr. Jonathan Valvano’s “*aLec14\_Client\_Server*” presentation



From Dr. Jonathan Valvano’s “*aLec15\_CC3100WeatherServer*” presentation

1. **Measurement Data**



0% of packets lost.

The *timePull* buffer indicates the SysTick timer values after weather data was requested from api.openweathermap.org.

The *timePush* buffer indicates the SysTick timer values after potentiometer position data was sent to our own server at URL: <ee445l-ourproject.appspot.com>

Below each buffer is the average, minimum, and maximum time that the microcontroller spent in communicating with the servers.

1. **Analysis and Discussion**
2. In the client server paradigm, explain the sequence of internet communications sent from client to server and from server to client as the client saves data on the server. Assume the client already is connected to the WiFi AP and the client knows the IP address of the server.

The server opens a connection socket on port 80 and waits for a client’s request. The client opens a socket on port 80 and connects to the server. The server opens a thread to serve the client. The server reads the client’s request and opens up its own server socket to process the client’s request, issue a response, and then closes the socket. The client uses its socket to receive and process the response, and then closes its own socket.

2) What is the purpose of the DNS?

The purpose of the Domain Name System is to have a decentralized naming and addressing system for computers to aid users in finding websites. Users can request websites by entering a plaintext domain name (e.g. [www.howstuffworks.com](http://www.howstuffworks.com)), and a DNS server will match the string to an IP address of a How Stuff Works server, and will return the IP address to the user such that the user can now directly request the website from the desired webserver.

3) What is the difference between UDP and TCP communication? More specifically when should we use UDP and when should we use TCP?

The difference between is that UDP does not require the sender to wait for authentication from the receiver before sending data. In comparison, TCP forces the sender to wait for an acknowledgement – a handshake – from the receiver, before information is sent. We use UDP in applications where the integrity of the information does not have to be guaranteed, and/or when a constant stream of data is preferred, (e.g. streaming videos or music). TCP is used when we wish for the transfer of data to be reliable, and we do not want to lose any packets of information, (e.g. bank transfers).