EE345L – Lab 4: Internet of Things

Ty Winkler and Jeremiah Bartlett
02/24/16

1.0 OBJECTIVE

The objective of this lab was to introduce us to wifi communications using the TM4C123 microcontroller. In this lab we used a CC3100Boost to connect to openweathermap.org and retrieve a TCP payload containing the weather information of Austin, Texas. We then parsed this payload for Austin's temperature which was then printed to the LCD board. We then sent a TCP packet to embsysmooc.appspot.com containing the voltage acquired through the on board ADC.

2.0 ANALYSIS AND DISCUSSION

2.1 Communication Sequence

First the client creates a socket and opens it so that it can send a TCP packet to the server. The client then sends a TCP packet to the server and waits for a reply of the requested information. Upon getting a reply the client then closes the socket.

2.2 DNS

Domain Name Servers allow clients to call upon internet domain names and re-routs them to the respective IP Address. For example, a user could enter google.com and using dns would automatically get re-routed to the IP Address `216.58.192.46`.

2.3 UDP vs TCP

Universal Datgram Protocol and Transmission Control Protocol are two types of IP traffic. UDP is the simpler version of TCP where a client will continuously send packets of data to the server regardless of the information being sent back and regardless of whether that information is arriving to the server. TCP requires the server to respond back to the client confirming that a connection has been established between the client and server. TCP is most often used in web browsing and communication between multiple computers. UDP is faster as it does not require a confirmation and is most commonly used in games and servers that collect data from a large number of users.

3.0 Software & Hardware Design Solutions

3.1 Software Design

```
#define REQUEST "GET /data/2.5/weather?q=Austin%20Texas&units=metric&APPID=955db084d47dd332d35dafe1a3e2881e HTTP/1.1\r\nUser-Agent: Keil\r\nHost:api.openweathermap.org\r\nAccept: */*\r\n\r\n"
int main(void){int32_t retVal;  SlSecParams_t secParams;
  char *pConfig = NULL; INT32 ASize = 0; SlSockAddrIn_t  Addr;
    ADC0_InitSWTriggerSeq3_Ch9();         // allow time to finish activating
  initClk();        // PLL 50 MHz
    Output_On();
    UART_Init();      // Send data to PC, 115200 bps
  Timer1_Init();
  LED_Init();       // initialize LaunchPad I/O
  UARTprintf("Weather App\n");
  retVal = configureSimpleLinkToDefaultState(pConfig); // set policies
  if(retVal < 0)Crash(4000000);
  retVal = sl_Start(0, pConfig, 0);
  if((retVal < 0) || (ROLE_STA != retVal) ) Crash(8000000);
  secParams.Key = PASSKEY;
  secParams.KeyLen = strlen(PASSKEY);
  secParams.Type = SEC_TYPE; // OPEN, WPA, or WEP
  sl_WlanConnect(SSID_NAME, strlen(SSID_NAME), 0, &secParams, 0);
  while((0 == (g_Status&CONNECTED)) || (0 == (g_Status&IP_AQUIRED))){
    _SlNonOsMainLoopTask();
  }
  UARTprintf("Connected\n");
  while(1){
        int i = 0;
        while(i < 10){
            int sendc = 0;
            strcpy(HostName,"openweathermap.org");
            retVal = sl_NetAppDnsGetHostByName(HostName,
                                strlen(HostName),&DestinationIP, SL_AF_INET);
            if(retVal == 0){
                Addr.sin_family = SL_AF_INET;
                Addr.sin_port = sl_Htons(80);
                Addr.sin_addr.s_addr = sl_Htonl(DestinationIP);// IP to big endian
                ASize = sizeof(SlSockAddrIn_t);
                SockID = sl_Socket(SL_AF_INET,SL_SOCK_STREAM, 0);
                if( SockID >= 0 ){
                    retVal = sl_Connect(SockID, ( SlSockAddr_t *)&Addr, ASize);
                }
                if((SockID >= 0)&&(retVal >= 0)){
                    strcpy(SendBuff,REQUEST);
                    sl_Send(SockID, SendBuff, strlen(SendBuff), 0);// Send the HTTP
                    sl_Recv(SockID, Recvbuff, MAX_RECV_BUFF_SIZE, 0);// Receive resp
                    sl_Close(SockID);
                    LED_GreenOn();
                    UARTprintf("\r\n\r\n");
                    UARTprintf(Recvbuff);  UARTprintf("\r\n");
                }
            }
            ST7735_OutUDec(sendc);
            ST7735_OutString("\n");
            i++;
        }

        //while(Board_Input()==0){}; // wait for touch
```

```
        ST7735_OutChar('e');
        ST7735_OutChar('m');
        ST7735_OutChar('p');
        ST7735_OutChar(' ');
        ST7735_OutChar('=');
        ST7735_OutChar(' ');
        for(int i = 0; i < 5; i++){
            ST7735_OutChar(myArray[i]);
        }
        ST7735_OutChar('\n');

        //ADC Part f
        ADC0_SAC_R = ADC_SAC_AVG_64X;    //enable 64 times average before obtaining
    int voltage = ADC0_InSeq3();
        ST7735_OutString("Voltage~");
        ST7735_sDecOut3(voltage);

        char* voltageString;
        char voltageStringNum[5];
        sprintf(voltageStringNum, "%.1d.%.3d", voltage/1000, voltage%1000);
        //ST7735_OutString(voltageStringNum);

        char* sendString;
        char str1[173] = "GET /query?city=Austin%20Texas&id=Ty%20Winkler%20Jeremiah%
    20Bartlett&greet=Voltage%3D";
        strcat(str1, voltageStringNum);
        strcat(str1, "V&edxcode=8086 HTTP/1.1\r\nUser-
    Agent: Keil\r\nHost: embsysmooc.appspot.com\r\n\r\n");


        strcpy(HostName,"embsysmooc.appspot.com");
        retVal = sl_NetAppDnsGetHostByName(HostName,
                            strlen(HostName),&DestinationIP, SL_AF_INET);
        if(retVal == 0){
            Addr.sin_family = SL_AF_INET;
            Addr.sin_port = sl_Htons(80);
            Addr.sin_addr.s_addr = sl_Htonl(DestinationIP);// IP to big endian
            ASize = sizeof(SlSockAddrIn_t);
            SockID = sl_Socket(SL_AF_INET,SL_SOCK_STREAM, 0);
            if( SockID >= 0 ){
                retVal = sl_Connect(SockID, ( SlSockAddr_t *)&Addr, ASize);
            }
            if((SockID >= 0)&&(retVal >= 0)){
                        strcpy(SendBuff, str1);
                        count = 0;
                        sl_Send(SockID, SendBuff, strlen(SendBuff), 0);// Send the
                        sl_Recv(SockID, Recvbuff, MAX_RECV_BUFF_SIZE, 0);// Receive
                        sl_Close(SockID);
                        LED_GreenOn();
                        UARTprintf("\r\n\r\n");
                        //ST7735_OutString(Recvbuff);
                        UARTprintf("\r\n");
                    }
                }
                while(1);
            }
        }
```

3.2 Measurement Data

Percent of lost packets using TCP
0%

Minimum, maximum, and average times from 10 transmissions to openweathermap.org

Seconds
9.27
9.37
13.54
6.83
6.74
10.33
11.47
9.67
16.73
11.28

Minimum Time
6.74 s

Maximum Time
16.73 s

Average Time
10.52 s

Minimum, maximum, and average times from 10 transmissions to EE445L server

Seconds
3.86
3.57
3.99
6.69
4.00
3.82

Minimum Time
3.57 s

Maximum Time
6.69 s

Average Time
4.32 s