**Lab 4 Report** **Hardik Patel**
**EEC 172 - A02 - Spring 2016** **Alex Elkman**

# Introduction to AWS and RESTful API

## Objective:

The purpose of this lab was to familiarize ourselves with the RESTful API (Representational State Transfer) and use it to connect with AWS (Amazon Web Services). To achieve this goal, we were required to obtain an account on AWS and create a 'thing' and using HTTP GET command, we were to retrieve status information about the 'thing'. In the last part of the lab we were required to send a text to our phone via the IR remote, which we used in Lab 2. This would be done using Amazon's Simple Notification Service (SNS) and transmit the push notifications to our phones via SMS (Simple Messaging Service).

## Design and Test Procedure:

Parts 1-2: These parts mainly aimed to acquaint us with the AWS website. This involved setting up an account, going through Amazon's IoT tutorial, creating a 'thing' etc. Our this, as evident from the code attached below, was called 'Schnauzer'. Once the 'thing; was created, we could access the REST API endpoint web address associated with our account and were also able to generate the KEYs (public, private and certificate) associated with the account.

Parts 3-4: In these two parts we used the software 'openSSL' to convert the KEYs that we acquired into the proper type compatible with the CC3200 board. Since the files we downloaded were of type '.pem', they had to be converted to type '.der' before being used. We employed the commands specified in the Lab Manual to achieve this. Once the '.der' files were generated, we used UniFlash to flash all the files onto our board.

Part 5: In this part we simply request access to AWS with the correct address and KEYs and aim to send a string successfully to AWS and receive confirmation. To execute this we changed all the relevant text in the code to make it unique to our 'thing'. This involved changing the Server Name, Post Header, Host Header and the text in the header file 'common.h' to specify the WiFi and the password the we were using. Once the setup was complete we successfully connected to the AWS servers and sent data and received confirmation.

Part 6: Finally, we used the code and modified it to send a text using the IR remote to our phone. For this, we started by importing the IR remote code that we developed in Lab 2 , to the RESTful API code. Noticing a conflict with the pin muxing, we changed our pinmux for the timer to pin 2. In the main of the final code, we put the main of the RESTful API code in the 'Enter' of the IR remote code. This way, we already have the message to send before we connect and send. To make sure our message is in the correct JSON format, we concatenate the first part of the string, with the 'messegeBuffer' to the last part. This is achieved using strcat() and strcpy() functions. Once the code is ready, we set up the SNS service, on the AWS website. As we ran the code, the text generated was sent to the AWS servers and forwarded to our phone automatically by Amazon.

**Problems encountered:** The main problem that we face was connecting to the AWS servers. We couldn't connect to the servers from the EEC 172 lab in Kemper. But we had no problem connecting to the server from a home WiFi. I suspect the reason lied within the 'common.h' header file where we specify the WiFi. But this is speculation.

## Conclusion:

This lab was one of the most interesting, incorporating many different concepts we've learned throughout the quarter. We revived yet again our IR remote algorithms to be used for sending a text over the AWS network. We also had to learn some basics of SSL and RESTful API which will help uswhile working on our final project for the course.

## Relevant parts of the Final Code

```
//Alex Elkman, Hardik Patel
//EEC172 Embedded Systems
//Spring 2016

//#define SERVER_NAME                "AHMAIFS2X4J4Y.iot.us-west-2.amazonaws.com"
#define SERVER_NAME                             "A3MIY1Q2V62MAH.iot.us-east-1.amazonaws.com"
#define GOOGLE_DST_PORT             8443
#define SL_SSL_CA_CERT "/cert/rootca.der"
#define SL_SSL_PRIVATE "/cert/private.der"
#define SL_SSL_CLIENT  "/cert/client.der"
//NEED TO UPDATE THIS FOR IT TO WORK!
#define DATE                23    /* Current Date */
#define MONTH               5     /* Month 1-12 */
#define YEAR                2016  /* Current year */
#define HOUR                21    /* Time - hours */
#define MINUTE              04    /* Time - minutes */
#define SECOND              0     /* Time - seconds */
#define POSTHEADER "POST /things/Schnauzer/shadow HTTP/1.1\nGET /things/Schnauzer/shadow"
//#define HOSTHEADER "Host: AHMAIFS2X4J4Y.iot.us-west-2.amazonaws.com\r\n"
#define HOSTHEADER "Host: https://A3MIY1Q2V62MAH.iot.us-east-1.amazonaws.com\r\n"
//#define AUTHHEADER "Authorization: SharedAccessSignature
sr=swiftsoftware-ns.servicebus.windows.net&sig=6sIkgCiaNbK9R0XEpsKJcQ2Clv8MUMVdQfEVQP09WkM%3d&
se=1733661915&skn=EventHubPublisher\r\n"
#define CHEADER "Connection: Keep-Alive\r\n"
#define CTHEADER "Content-Type: application/json; charset=utf-8\r\n"
#define CLHEADER1 "Content-Length: "
#define CLHEADER2 "\r\n\r\n"
//#define DATA1 "{\"MessageType\":\"CC3200 Sensor\",\"Temp\":"
//#define DATA2
",\"Humidity\":50,\"Location\":\"YourLocation\",\"Room\":\"YourRoom\",\"Info\":\"Sent from
CC3200 LaunchPad\"}"
#define DATA2
",\"Humidity\":50,\"Location\":\"YourLocation\",\"Room\":\"YourRoom\",\"Info\":\"Sent from
CC3200 LaunchPad\"}"

#define MASTER                      0
//These are the decimal values that corrospont to the binary data from the buttons as decoded
bt our inturrupt handler
#define BUTTON_ONE           66339000
#define BUTTON_TWO           3423416
#define BUTTON_THREE  5520056
#define BUTTON_FOUR          7616696
#define BUTTON_FIVE          9713336
#define BUTTON_SIX           11809976
#define BUTTON_SEVEN  13906616
#define BUTTON_EIGHT  16003256
#define BUTTON_NINE          18099896
#define BUTTON_ZERO          20196536
#define BUTTON_LAST          30728880
```

```
#define BUTTON_MUTE          41162936
static unsigned long g_ulSamples[2];
static char messageBuffer[100]; //Keeps track of the messege in a global variable
static char number[100]; //Keeps track the binary strings of the buttons
static char lastPressed; //Keeps track of the last digit passes.
static char tempString[1]; // Is used for the UARTCharGet functions to store the character in
a temporary variable befor we display it on the OLED
static int keyBuffer[13] = {0}; // An array that helps with the implementation of the
multi-tap aspect of the program.
static int value = 1; // Generates the binary value
static int i = 0;
//static int index = 0;
static int delta = 0;

char* DATA1 = "{\"state\": {\n\r\"desired\" : {\n\r\"message\" :
\"DfongHelloWorldCC3200\"\n\r}}}\n\r\n\r";

static void TimerIntHandler()
{


        g_ulSamples[1] = MAP_TimerValueGet(TIMERA2_BASE,TIMER_A);
        TimerLoadSet(TIMERA2_BASE, TIMER_A,0xffff);

    g_ulSamples[0] = g_ulSamples[1];
    delta = g_ulSamples[1];
        //Delta is used to differentiate betwwn a 0 pulse width and a 1 pulse width.
    if(delta < 54000)
        value = 0;
    else
        value = 1;
    if(i==0)
        value = 0;
        number[i] = (char)value; //Storing a binary value in an array.
        i++;
        MAP_TimerIntClear(TIMERA2_BASE,TIMER_CAPA_EVENT); // Clearing interrupts
}

int main()
{
        //Local variables used for different purposes.
        int y = 0;
        int k = 0;
        int sum = 0;
        int temp = 0;
         char  str2 [1000];
        char* boba = "{\"state\": {\n\r\"desired\" : {\n\r\"message\" : \"";
        char* boba2 = "\"\n\r}}}\n\r\n\r";
        strcpy(str2, boba);
        BoardInit();
```

```
       PinMuxConfig();
       InitTerm();
       DisplayBanner(APP_NAME);
       MAP_PRCMPeripheralClkEnable(PRCM_GSPI,PRCM_RUN_MODE_CLK);
       MAP_PRCMPeripheralReset(PRCM_GSPI);
       MAP_SPIConfigSetExpClk(GSPI_BASE,MAP_PRCMPeripheralClockGet(PRCM_GSPI),
                            SPI_IF_BIT_RATE,SPI_MODE_MASTER,SPI_SUB_MODE_3,
                            (SPI_SW_CTRL_CS |
                            SPI_4PIN_MODE |
                            SPI_TURBO_OFF |
                            SPI_CS_ACTIVELOW |
                            SPI_WL_8));
       MAP_SPIEnable(GSPI_BASE);
       MAP_SPICSEnable(GSPI_BASE);
       MAP_PinConfigSet(PIN_04,PIN_TYPE_STD_PU,PIN_STRENGTH_6MA);
       MAP_TimerIntRegister(TIMERA2_BASE,TIMER_A,TimerIntHandler);
       MAP_TimerConfigure(TIMERA2_BASE, (TIMER_CFG_SPLIT_PAIR | TIMER_CFG_A_CAP_TIME));
        // We are calling interrupts at the negative edges of the input signal
    MAP_TimerControlEvent(TIMERA2_BASE,TIMER_A,TIMER_EVENT_NEG_EDGE);
       //Starting the timer at 0xffff
    MAP_TimerLoadSet(TIMERA2_BASE,TIMER_A,0xffff);
    MAP_TimerIntEnable(TIMERA2_BASE,TIMER_CAPA_EVENT);
    MAP_TimerEnable(TIMERA2_BASE,TIMER_A);

    while(1){
               // i is the nummerof bits generated by our remote. Hence, we run this loop in 26
times
               while(i<26){
               }
               Report("\n\n\r");
               i=0;
               Report("You pressed: ");
               for(k = 0; k<25; k++){
                      sum += (int)number[k]*pow(2, (k+1));
                      Report("%d", (int)number[k]);
               }
               Report("\n\rUnique ID: %d", sum);
               index++; //increment index
               Report("\n\rYou pressed: ");
               //This switch statement is where all the multitap aspect is implemented. Also it
includes our enter button (MUTE) and delete buttons (LAST).
               switch(sum){
                      case(BUTTON_ONE):
                             keyBuffer[1]+=1;
                             Report("1");
                             lastPressed = '1';
                             index-=1;
                             break;
                      case(BUTTON_TWO):
                             Report("2");
```

```
                if(keyBuffer[2] == 3){
                        keyBuffer[2] = 0;
                        delKey("C");
                }
                if(lastPressed != '2')
                        keyBuffer[2] = 0;
                if(keyBuffer[2]==0){
                        Outstr("A");
                }
                else if(keyBuffer[2]==1){
                        delKey("A");
                        Outstr("B");
                }
                else if(keyBuffer[2]==2){
                        delKey("B");
                        Outstr("C");
                }
                lastPressed = '2';
                keyBuffer[2]+=1;
                break;

        case(BUTTON_THREE):
                Report("3");
                if(keyBuffer[3] == 3){
                        keyBuffer[3] = 0;
                        delKey("F");
                }
                if(lastPressed != '3')
                        keyBuffer[3] = 0;
                if(keyBuffer[3]==0){
                        Outstr("D");
                }
                else if(keyBuffer[3]==1){
                        delKey("D");
                        Outstr("E");
                }
                else if(keyBuffer[3]==2){
                        delKey("E");
                        Outstr("F");
                }
                lastPressed = '3';
                keyBuffer[3]+=1;
                break;
        case(BUTTON_FOUR):

                Report("4");

                if(keyBuffer[4] == 3){
                        keyBuffer[4] = 0;
                        delKey("I");
        }

        if(keyBuffer[4] == 3)
                keyBuffer[4] = 0;

        if(lastPressed != '4')
                keyBuffer[4] = 0;

        if(keyBuffer[4]==0){
                Outstr("G");
        }

        else if(keyBuffer[4]==1){
                delKey("G");
                Outstr("H");
        }
        else if(keyBuffer[4]==2){
                delKey("H");
                Outstr("I");
        }

        lastPressed = '4';
        keyBuffer[4]+=1;
        break;

case(BUTTON_FIVE):
        Report("5");
        if(keyBuffer[5] == 3){
                keyBuffer[5] = 0;
                delKey("L");
        }

        if(lastPressed != '5')
                keyBuffer[5] = 0;
        if(keyBuffer[5]==0){
                Outstr("J");
        }
        else if(keyBuffer[5]==1){
                delKey("J");
                Outstr("K");
        }
        else if(keyBuffer[5]==2){
                delKey("K");
                Outstr("L");
        }
        lastPressed = '5';
        keyBuffer[5]+=1;
        break;
case(BUTTON_SIX):
        Report("6");
```

```
                        if(keyBuffer[6] == 3){
                                keyBuffer[6] = 0;
                                delKey("O");
                        }

                        if(lastPressed != '6')
                                keyBuffer[6] = 0;
                        if(keyBuffer[6]==0){
                                Outstr("M");
                        }
                        else if(keyBuffer[6]==1){
                                delKey("M");
                                Outstr("N");
                        }
                        else if(keyBuffer[6]==2){
                                delKey("N");
                                Outstr("O");
                        }
                        lastPressed = '6';
                        keyBuffer[6]+=1;
                        break;
                case(BUTTON_SEVEN):

                        Report("7");

                        if(keyBuffer[7] == 4){
                                keyBuffer[7] = 0;
                                delKey("S");
                        }

                        if(lastPressed != '7')
                                keyBuffer[7] = 0;

                        if(keyBuffer[7]==0){
                                Outstr("P");
                        }

                        else if(keyBuffer[7]==1){
                                delKey("P");
                                Outstr("Q");
                        }
                        else if(keyBuffer[7]==2){
                                delKey("Q");
                                Outstr("R");
                        }
                        else if(keyBuffer[7]==3){
                                delKey("R");
                                Outstr("S");
                        }
```

```
        lastPressed = '7';
        keyBuffer[7]+=1;
        break;

case(BUTTON_EIGHT):

        Report("8");

        if(keyBuffer[8] == 3){
                keyBuffer[8] = 0;
                delKey("V");
        }

        if(lastPressed != '8')
                keyBuffer[8] = 0;

        if(keyBuffer[8]==0){
                Outstr("T");
        }

        else if(keyBuffer[8]==1){
                delKey("T");
                Outstr("U");
        }
        else if(keyBuffer[8]==2){
                delKey("U");
                Outstr("V");
        }

        lastPressed = '8';
        keyBuffer[8]+=1;
        break;
case(BUTTON_NINE):

        Report("9");

        if(keyBuffer[9] == 4){
                keyBuffer[9] = 0;
                delKey("Z");
        }

        if(lastPressed != '9')
                keyBuffer[9] = 0;

        if(keyBuffer[9]==0){
                Outstr("W");
        }

        else if(keyBuffer[9]==1){
                delKey("W");
```

```
                                     Outstr("X");
                              }
                              else if(keyBuffer[9]==2){
                                     delKey("X");
                                     Outstr("Y");
                              }
                              else if(keyBuffer[9]==3){
                                     delKey("Y");
                                     Outstr("Z");
                              }
                              lastPressed = '9';
                              keyBuffer[9]+=1;
                              break;
                      case(BUTTON_ZERO):
                              Report("0");
                              Outstr(" ");
                              lastPressed = '0';
                              break;
                      case(BUTTON_LAST):
                      //We print black over the last pressed as a delete implementation
                              if(index>1)           //since delKey will decrement index, and we
want it to stay > 1
                                     index = index - 1;
                              tempString[0] = messageBuffer[index-1];
                              messageBuffer[index-1] = NULL;
                              Report("\n\rLAST");
                              delKey(tempString);
                              lastPressed = 'l';
                              break;
                      case(BUTTON_MUTE):
                              Report("MUTE");
                              //Running the ENTER command for the MASTER device
                                     Report("\n\rMaster Entered!\n\r");
                      //Put each character from messageBuffer into the TX FIFO (for UART1)
                                     strcat(str2, messageBuffer);
                                     strcat(str2, boba2);
                                     DATA1 = str2;
                                     //Restful API main()
                                 long lRetVal = -1;
                                 //Connect the CC3200 to the local access point
                                 lRetVal = connectToAccessPoint();
                                 //Set time so that encryption can be used
                                 lRetVal = set_time();
                                 if(lRetVal < 0)
                                 {
                                      UART_PRINT("Unable to set time in the device");
                                      LOOP_FOREVER();
                                 }
                                 //Connect to the website with TLS encryption
                                 lRetVal = tls_connect();
```

```
                    if(lRetVal < 0)
                    {
                        ERR_PRINT(lRetVal);
                    }
                    http_post(lRetVal);

                    sl_Stop(SL_STOP_TIMEOUT);
                lastPressed = 'm';
                index-=1;
                break;
            default:
                Report("Unknown code %d", sum);
                index-=1;
                break;
        }
        if(lastPressed == 'l' || lastPressed == 'm' || lastPressed == '1') {}
        else if (lastPressed == '0')
            messageBuffer[index-1] = ' ';
        else{
            temp = 65 + (lastPressed-48-2)*3 + keyBuffer[(int)lastPressed-48] - 1;
//returns ascii value for the char you selected aelkman
            if(lastPressed == '8' || lastPressed == '9')
                temp += 1;
            messageBuffer[index-1] = (char)temp;
        }
        Report("\n\rtemp: %d, index: %d", temp, index);
        sum=0;
        Report("\n\rMessege Buffer: ");
        for(y=0;y<100;y++){
            Report("%c", messageBuffer[y]);
        }
    }
}
```