Course: Summer Industrial Training 2015

Lab Report: Extra Credit - Data Acquisition - IOT

Report By: Ajinkya Padwad

Harshit Gandhi

Executive Summary:

Data acquisition of the LDR readings was carried out and logged over the internet at the server through the ethernet interface at the FRDM KL25Z platform.

Project Description:

- The channel was first created online at the server space and necessary field settings were done.
- The API key and the URL were accordingly updated at the program code at the mbed compiler.
- Ethernet interface was carried out with the FRDM KL25Z board and an LDR was attached for reading values.

API Description:

- Ethernet API functions-

init ()	- Initialize the interface with DHCP. Initialize the interface with a static IP address.
connect ()	-Connect Bring the interface up, start DHCP if needed.
disconnect ()	- Disconnect Bring the interface down.
getMACAddress ()	- Get the MAC address of your Ethernet interface.
getIPAddress ()	- Get the IP address of your Ethernet interface.
getGateway ()	- Get the Gateway address of your Ethernet interface.

getNetworkMask () - Get the Network mask of your Ethernet interface.

- Serial API functions -

Baud()	- Set the serial transmission rate value
Printf()	- print the value at the serial terminal

Output Observed:

Variation in the LDR readings were logged onto the server through the ethernet interface and real time data was plotted at the channel on the server.

Test and Debug:

- ✓ Tested the different data transmission rate (baud rate 19200)
- ✓ Sprintf error - data compatibility error - debugged.

.

Learning Outcomes:

- ➢ Ethernet interface and the respective APIs
- ➢ Data conversion for the online server logging and serial transmission
- ➢ Online channel attributes and concepts of IoT .

APPENDIX A

Code:

```
#include "mbed.h"
#include "WIZnetInterface.h"

unsigned char MAC_Addr[6] = {0x00,0x08,0xDC,0x12,0x07,0x0E}; //MAC Address
specified

char* Update_Key = "YTDILMQL53ASDCJ4";  //Unique channel key
char* ServerIP = "184.106.153.149";       //Server internet protocol address
int Count = 15;
float val;

Serial pc(USBTX, USBRX);  //For serial terminal
SPI spi(PTD2,PTD3,PTD1);   //CLK,MISO,MOSI  - SPI protocol
WIZnetInterface ethernet(&spi,PTD0,PTA20);  //SPI, RST,EN
AnalogIn temp(PTC1);
AnalogIn ldr(PTC2);

int i=0;
int main()
{
   //Set serial port baudrate speed: 19200
   pc.baud(9600);
   pc.printf("Start\r\n");

   while(1)
    {

      int ret = ethernet.init(MAC_Addr); //Initialization of the ethernet

      if (!ret) {
         pc.printf("Initialized, MAC: %s\r\n", ethernet.getMACAddress());  // Display
the MAC address
         ret = ethernet.connect();  //Connection initiated
```

```
          if (!ret) {
             pc.printf("IP: %s, MASK: %s, GW: %s\r\n",
                    ethernet.getIPAddress(), ethernet.getNetworkMask(),
ethernet.getGateway());   // Display the IP address, Network mask and Gateway
          } else {
             pc.printf("Error ethernet.connect() - ret = %d\r\n", ret);  // Display error
if not connected
             exit(0);
          }
       } else {
          pc.printf("Error ethernet.init() - ret = %d\r\n", ret);
          exit(0);

       }

   TCPSocketConnection sock;  //Initiate the socket for the connection
   sock.connect("184.106.153.149", 80); //80 is default port for HTTPS
   if(sock.is_connected())
      pc.printf("Socket Connected\n\r");
   else
      pc.printf("Socket NoT Connected\n\r");

   //Code here >>

   val=ldr.read()*3.3;  //Actual value of the LDR voltage

   char buffer[300];
   int ret_t;

   char http_cmd[256];




   sprintf(http_cmd,"GET /update?key=QRMC7D6H5H0XUG8W&field1= %f
HTTP/1.0\n\n",val);  //Puts the LDR value to the server
   printf("Running - %s\r\n",http_cmd);
```

```
    sock.send_all(http_cmd, sizeof(http_cmd)-1);  //data send to server

    ret_t = sock.receive(buffer, sizeof(buffer)-1); //data from server
    buffer[ret_t] = '\0';
    printf("Received %d chars from server:\n %s \r\n", ret_t, buffer);


    sock.close();  // Close the server port

    ethernet.disconnect();  // Disconnect the connection
    printf("Socket Closed");

    //i++;
    wait(30);
  }
}
```