Course: Summer Industrial Training 2015

Lab Report: Extra Credit Project: PSOC BLE-IOT

Report By: Ajinkya Padwad

Omkar Purandare

Executive Summary:

The aim of this project is to implement a real time data logging system using the PSoC BLE and FRDM KL25Z over the ethernet inteface to bring about Internet of Things.

Project Description:

In this final project both the platforms i.e PSOC and mBED were used. This project combines the Home automation project of PSOC BLE with the IOT project of Mbed using phant server. PSOC BLE is connected to LDR and Temperature sensor. KL25z has on board accelerometer. The values of these sensors are sent through IOT to the phant server and displayed on computer. Which values are to be sent is controlled using CySmart application. Thus there will be 5 options in the application corresponding to Lux, temperature, x axis accelerometer, y axis accelerometer and z axis accelerometer. If any of the last three options is used, the PSOC BLE will send an indication to mbed board to transmit accelerometer values. Connection between PSOC BLE and Mbed board is serial. All values are sent through serial. Since IOT infrastructure is less, the demo has to be shown on serial terminal with mbed board. If the project is successfully completed on serial, some limited infrastructure will be provided to put the values on the server.

API Description:

In this project we used the following APIs-

Mbed-

- MMA8451Q- It is the name of the on board accelerometer on the *FRDM KL25Z board.* It 's address is defined initially in the program, so that it is not required to give the address again and again. It is used as the API that is used to get the readings from accelerometer i.e the tilts along various axis (x,y,z) via serial USB communication . We get this readings only in range of -1 to +1.
  Syntax- MMA8451Q name (DATAPIN, CLOCK, MMA8451_I2C_ADDRESS);
- Serial – Takes two parameters transmitter pin and receiver pin and helps in serial communication
- 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.

PSOC-

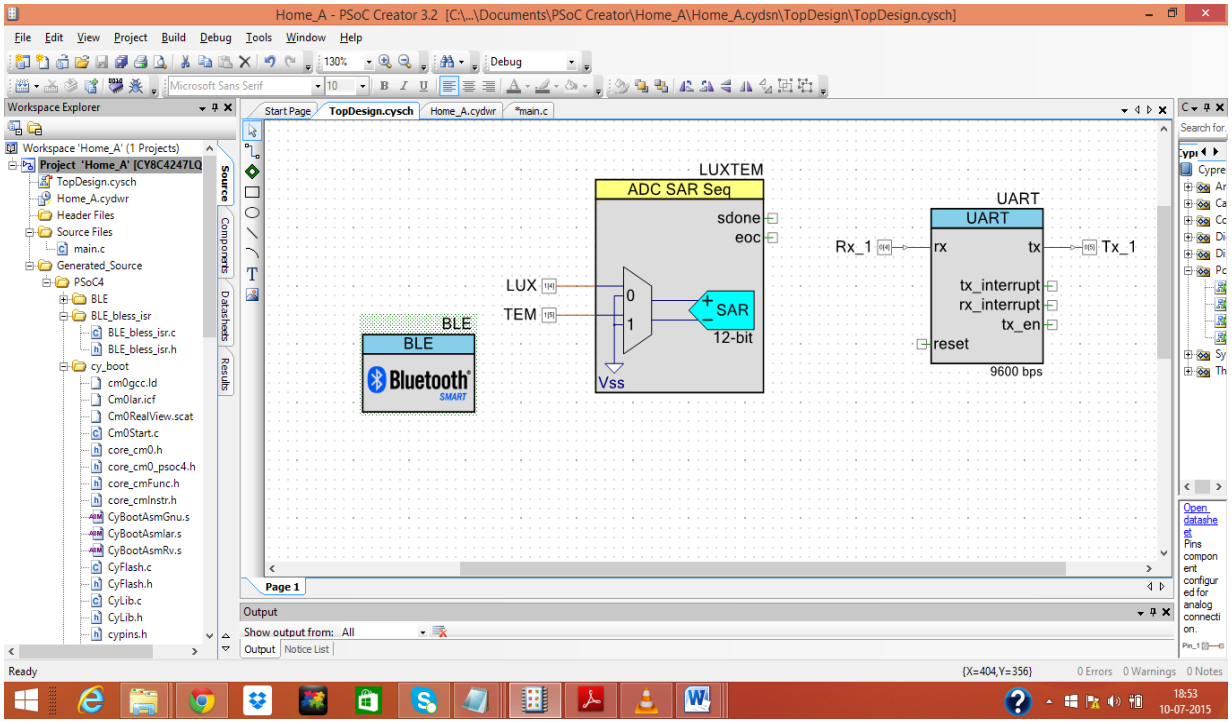
UART_Start();-Initialize and enabling UART.
ADC_Start()- enables ADC and
ADC_StartConvert(); Starts converting analog value
ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);checks if conversion is over or not
ADC_GetResult16(0);Gives 16 bit digital value
ADC_CountsTo_mVolts(0,adcval);Converts adc counts to millivolts
UART_PutString(buffer);sends data serially
CyDelay(1000); delay for 1000mili second

COMPONENTS USED:

Bluetooth Low Energy (BLE) module

- UART module
- SAR ADC Seq
- Serial Receive Pin
- Serial Transmission Pin

PROJECT SCHEMATIC-

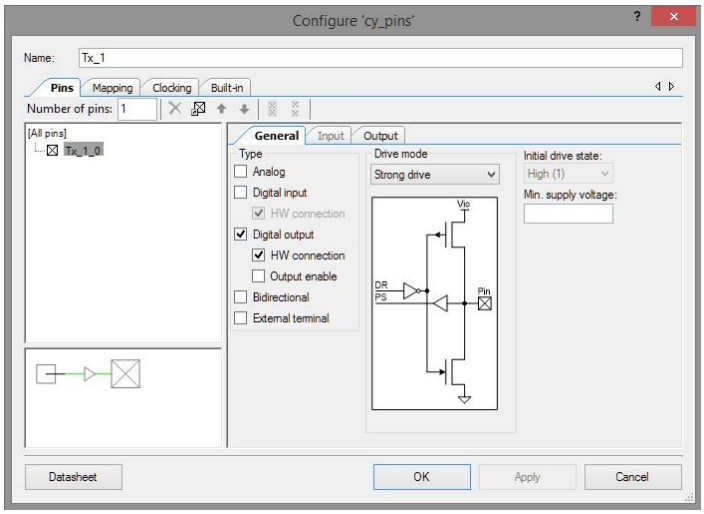


Pin Component Configuration and Pin map:

Digital Output pin -

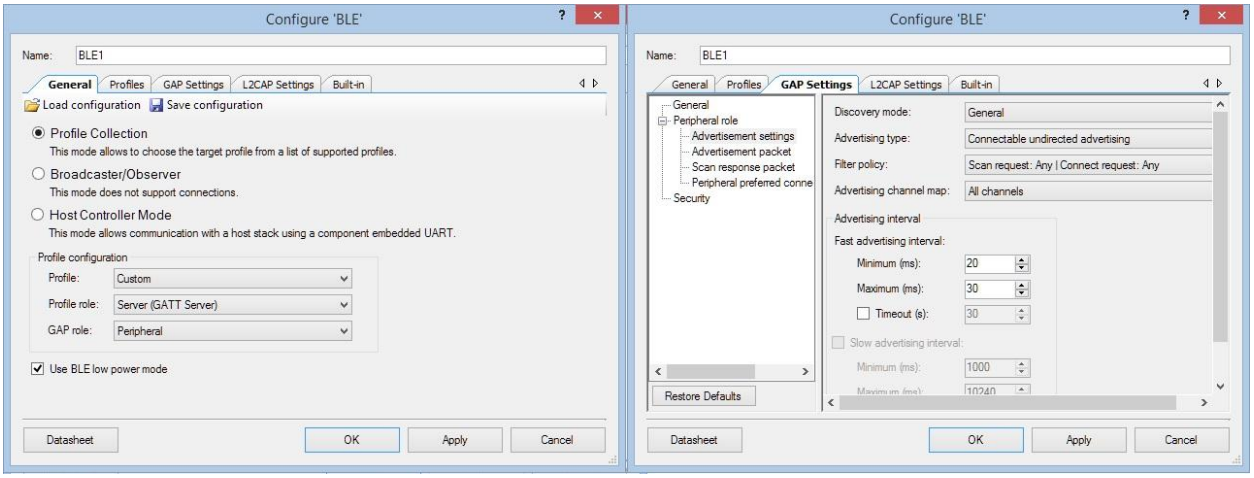
Drive Mode This parameter configures the pin to provide one of the eight available pin drive modes.

HW Connection parameter determines whether the digital input terminal for an input pin is displayed in the schematic. It may or may not be API controlled depending upon the option being selected.



Bluetooth Low Energy (BLE) module-

The *Profile role* parameter configuration depends on the chosen Profile, and the Profile role selection affects the GAP role parameter.
The *toolbars* contain navigation options and a means to add or delete Services, Characteristics, and Descriptors.
The *GAP parameters* define the general connection settings required when connecting Bluetooth devices



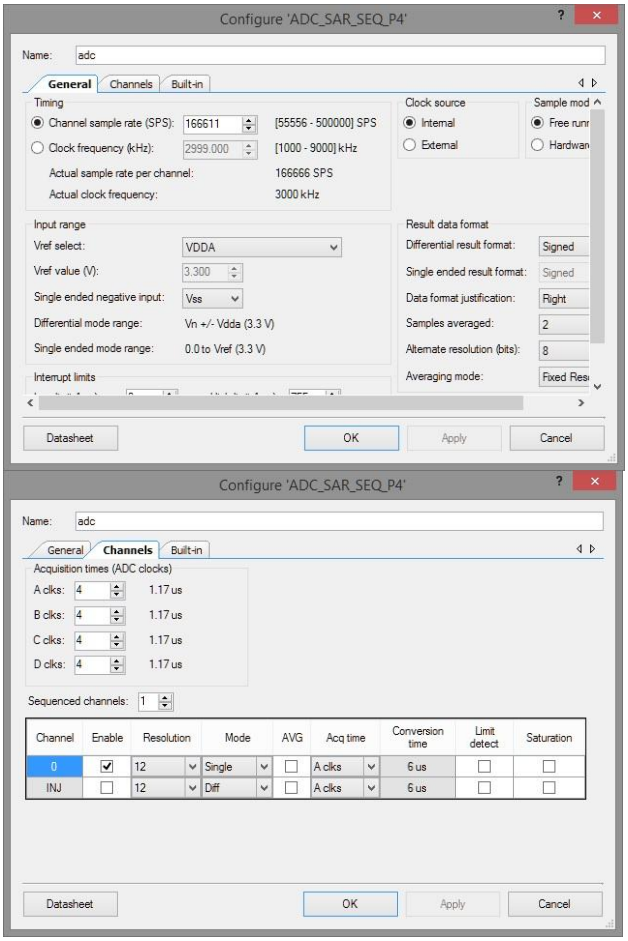
SAR ADC module

Clock rate is automatically calculated based on the number of channels, averaging, resolution, and acquisition time parameters to meet the entered sample rate.

Clock frequency can be anywhere between 1 MHz and 18 MHz (14.508 MHz in CY8C41).

Vref value parameter displays the reference voltage value that is used for the SAR ADC reference.

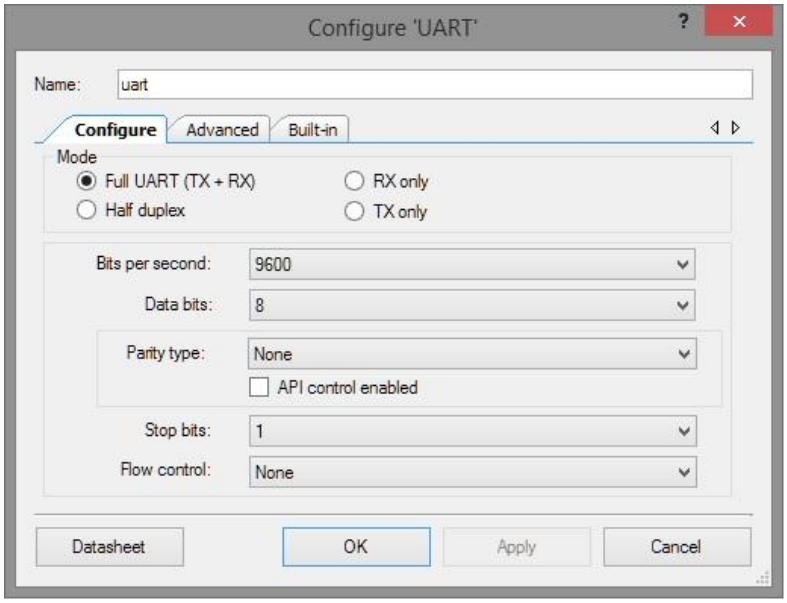
Channels parameter selects how many input signals are scanned. The maximum number of channels is either 8 or 16 depending on the device.



UART Module-

Mode parameter defines the functional components you want to include in the UART. This can be setup to be a bidirectional Full UART (TX + RX) (default), Half Duplex UART (uses half the resources), Receiver (RX Only) or Transmitter (TX Only).

Bits per second parameter defines the baud-rate or bit-width configuration of the hardware for clock generation.



Output Observed:

The toggling of signals from the CySmart app was correctly detected by the PSoC BLE and the FRDM KL25Z boards for the subsequent values of the accelerometer, Lux and temperature values. The data was observed online on the serial terminal first and on the online server through the ethernet interface.

Test and Debug:

PSoC BLE-

- ✓ Debugged a data type incompatibility error while transmitting data serially over the boards.
- ✓ Appropriate scaling for the lux values and the temperature values at ouput.
- ✓ Sprintf error - debugged using the header file.

Mbed -

- ✓ Character to integer conversion using ASCII for the transmission, since data logging needs integer values
- ✓ MAC address settings problem - ethernet connection problem.
- ✓ API key and URL settings.

Learning Outcomes:

- ➢ Learnt the basic interface of PSoC BLE with FRDM KL25Z boards using UART
- ➢ Ethernet interface for the FRDM KL25Z board and data logging over the online server
- ➢ Basic troubleshooting for the ethernet interface and the UART

APPENDIX A

1. PSOC-BLE CODE(TRANSMITTER):-

```
#include <project.h>
#include "stdio.h"

CYBLE_CONN_HANDLE_T connectionHandle;
CYBLE_GATTS_WRITE_REQ_PARAM_T *wrReqParam;
uint8 LIST;
int16 adcval;
uint16 v;
char buffer[20];
char acl;
int temp1,temp2;
int i;
int b;
int m;                  //variable for slope
int lux1=0;
int lux2=0;         // variable for incident lux

float voltage[12] = {0.1,0.15,0.16,0.17,0.18,0.22,0.25,0.28,0.35,0.5,3.3};// array of LDR voltages used for
calibrating

float lux[12] = {10000,9000,8000,7000,6000,5000,4000,3000,2000,1000,0};// LUX at the corresponding voltage

void HandleDevice()
{
   switch(LIST)
   {
   case 0x01: //LUX CALCULATION CONVERSION AND TRANSMISSION
      ADC12_StartConvert();
      ADC12_IsEndConversion(ADC12_WAIT_FOR_RESULT);
      adcval = ADC12_GetResult16(0);
      v = ADC12_CountsTo_mVolts(0,adcval);
      for(i=0;i<12;i++)// loop for converting voltage to lux
      {
         v = v/100;
       if(v <voltage[i]) //loop to calibrate and find new lux value using
            slope
              {
          UART12_PutChar('v');
          m = (lux[i]-lux[i-1])/(voltage[i]-voltage[i-1]);// Calculating
                                    slope between two points
          lux1= (m*(v-voltage[i]))+lux[i];//Calculating the lux
                        break;
             }
              }
      lux2 = lux1/100;          //creating 1st two digits of value
      UART12_PutChar(lux2);  //serially passing 1st two digits of value
      b = lux1%100;                        //creating last two digits of value
      UART12_PutChar(b);  //serially passing last two digits of value
      break;
```

```
    case 0x02:   //TEMPERATURE CALCULATION CONVERSION AND TRANSMISSION
      ADC12_StartConvert();
      ADC12_IsEndConversion(ADC12_WAIT_FOR_RESULT);
      adcval= ADC12_GetResult16(1);
      v =ADC12_CountsTo_mVolts(1,adcval);

      temp1 = v * 100;
      temp2 = temp1/100;    //creating 1st two digits of value
            UART12_PutChar(temp2);  //serially passing 1st two digits of value
      b = temp1%100;                        //creating last two digits of value
      UART12_PutChar(b);     //serially passing last two digits of value
       break;

case 0x03:   //X VALUE CALCULATION CONVERSION AND TRANSMISSION
       { acl='x';

         UART12_PutChar(acl); //serially passing x value
       }
      break;

case 0x04:   //Y VALUE CALCULATION CONVERSION AND TRANSMISSION
       {
         acl='y';
         UART12_PutChar(acl); //serially passing y value
      }
      break;

case 0x05:   //Z VALUE CALCULATION CONVERSION AND TRANSMISSION
      {
         acl='z';
         UART12_PutChar(acl); //serially passing z value
      }
      break;

default: break;
      }
   }

void CustomEventHandler(uint32 event, void * eventparam)
{
   switch(event)
   {
      case CYBLE_EVT_STACK_ON:
         CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);
            break;

      case CYBLE_EVT_GAP_DEVICE_DISCONNECTED:
         CyBle_GappStartAdvertisement(CYBLE_ADVERTISING_FAST);
            break;
      case CYBLE_EVT_GATT_CONNECT_IND:
         connectionHandle = *(CYBLE_CONN_HANDLE_T *)eventparam;
            break;
```

```
      case CYBLE_EVT_GATTS_WRITE_REQ:
         wrReqParam = (CYBLE_GATTS_WRITE_REQ_PARAM_T *)eventparam;
         if (CYBLE_HOME_CONTROL_DEVICE_CONTROL_CHAR_HANDLE==wrReqParam-
                                          >handleValPair.attrHandle)
         {
           LIST = (uint8)wrReqParam->handleValPair.value.val[0];

            HandleDevice();
            CyBle_GattsWriteRsp(connectionHandle);
         }
            default: break;
   }
}

// MAIN CODE

int main()
{
   CyGlobalIntEnable; /* Enable global interrupts. */
    UART12_Start();
   ADC12_Start();

   CyBle_Start(CustomEventHandler);

   for(;;)
   {
      CyBle_ProcessEvents();

      /* Place your application code here. */
   }
}
```

2. FRDM KL25Z CODE(RECEIVER) ALONG WITH ETHERNET SHIELD:-

```
#include "mbed.h"

#include "MMA8451Q.h"

#include "WIZnetInterface.h"

#define MMA8451_I2C_ADDRESS (0x1d<<1)

unsigned char MAC_Addr[6] = {0x00,0x08,0xDC,0x12,0x07,0x0F};

char* Update_Key = "YTDILMQL53ASDCJ4";

char* ServerIP = "184.106.153.149";

int Count = 15;

SPI spi(PTD2,PTD3,PTD1);

WIZnetInterface ethernet(&spi,PTD0,PTA20); //API USED WHILE TRANSFERRING DATA ON THE
INTERNET VIA ETHERNET PORT OR SHIELD
```

```
AnalogIn temp(PTC1);

DigitalOut led(LED1);

Serial pc(USBTX, USBRX); //TX,RX FOR THE COMPUTER SERIAL TERMINAL

Serial my(PTE0, PTE1);    //USER DEFINED TX AND RX FOR UART COMMUNICATION

int16_t x,y,z,v;

int a,b,c;

int main(void)

{    pc.baud(9600);

   pc.printf("Start\r\n");

   MMA8451Q acc(PTE25, PTE24, MMA8451_I2C_ADDRESS);//API TO USE ACCELEROMETER
        FUNCTIONS

while (true)

 {   if(my.readable())       //IF ANY DATA IS READ BY DEFINED TX RX PINS THEN ONLY PROCEED

          {     v = my.getc();

               int ret = ethernet.init(MAC_Addr);

               if (!ret) {

               pc.printf("Initialized, MAC: %s\r\n", ethernet.getMACAddress());

               ret = ethernet.connect();

                    if (!ret) {

                              pc.printf("IP: %s, MASK: %s, GW: %s\r\n", ethernet.getIPAddress(),
                                   ethernet.getNetworkMask(), ethernet.getGateway());

                              }

                    else {

                         pc.printf("Error ethernet.connect() - ret = %d\r\n", ret);

                         exit(0);

                        }

                    }

      else {
```

```
        pc.printf("Error ethernet.init() - ret = %d\r\n", ret);
        exit(0);
      }
   TCPSocketConnection sock;
   sock.connect("184.106.153.149", 80);
   if(sock.is_connected())
      pc.printf("Socket Connected\n\r");
   else
      pc.printf("Socket NoT Connected\n\r");
   int ret_t;
      char http_cmd[256];
if(v == 'L')
       {
       a = my.getc(); //RECEIVE 1ST TWO DIGITS
       b = my.getc(); // RECEIVE LAST TWO DIGITS
       c= a*100 + b; //CREATING LUX VALUE
       sprintf(http_cmd,"GET /update?key=523SQ64652WCEK07&field1=%d HTTP/1.0\n\n",c);
       pc.putc(c); //DISPLAY ON THE TERMINAL
       }
        if(v == 'T')
        {
           a = my.getc();//RECEIVE 1ST TWO DIGITS
           b = my.getc();// RECEIVE LAST TWO DIGITS
           c = a*100 + b; //CREATING TEMP VALUE
           pc.putc(c); //DISPLAY ON THE TERMINAL
           sprintf(http_cmd,"GET /update?key=523SQ64652WCEK07&field2=%d HTTP/1.0\n\n",c);
//PASSING VALUE TO DISPLAY IT ON INTERNET
       }
```

```
        if(v == 'x')
        {
           x = 25 * acc.getAccX();
           pc.printf("%d",x); //DISPLAY ON THE TERMINAL
           sprintf(http_cmd,"GET /update?key=523SQ64652WCEK07&field3=%d HTTP/1.0\n\n",x);
      }
if(v == 'y')
      {
         y = -25 * acc.getAccY();
         pc.printf("%d",y); //DISPLAY ON THE TERMINAL
         sprintf(http_cmd,"GET /update?key=523SQ64652WCEK07&field4=%d HTTP/1.0\n\n",y);
      }
      if(v == 'z')
      {
         z = 25 * acc.getAccZ();
         pc.printf("%d",z); //DISPLAY ON THE TERMINAL
         sprintf(http_cmd,"GET /update?key=523SQ64652WCEK07&field5=%d HTTP/1.0\n\n",z);
     }
      sock.send_all(http_cmd, sizeof(http_cmd)-1);
     ret_t = sock.receive(buffer, sizeof(buffer)-1);
      buffer[ret_t] = '\0';
      sock.close();
      ethernet.disconnect();
   printf("Socket Closed");
wait(30);
           }
}
```