Wkaeonreednodefinal project modifications.

How to modify The Node to read and save node id and password over uart, all code should be modified in NodeReedRadiotask.c of the wakonreednodefinal project:

```
/* button driver handle */
PIN_Config buttonPinTable[] = {
    CONFIG_PIN_1 | PIN_INPUT_EN | PIN_PULLUP | PIN_IRQ_NEGEDGE,
    PIN_TERMINATE
};

static PIN_Handle buttonPinHandle;
static PIN_State buttonPinState;
```

1. Declare the above^ before functions declaration, for example line 103. CONFIG_PIN_1 corresponds to the BTN-1 pin on the LAUNCHXL CC1312R, this button pin must be defined in the .syscfg file. The correct path to find which config pin corresponds to the button is wakenoreednodefinal/debug/syscfg/ti_drivers_config.h. Build the project and the syscfg will declare the pin config pins in ti_drivers_config.h.

```
static uint8_t password;
static void buttonCallback(PIN_Handle handle, PIN_Id pinId);
```

2. declare the above along the other function decleration ex line 111.

```
buttonPinHandle = PIN_open(&buttonPinState, buttonPinTable);
    if (!buttonPinHandle)
    {
        System_abort("Error initializing button pins\n");
    }

    /* Setup callback for button pins */
    if (PIN_registerIntCb(buttonPinHandle, &buttonCallback) != 0)
    {
        System_abort("Error registering button callback function");
    }
```

Declare the above^ code in the nodeRadioTaskFunction() before the while loop. Declare the code below wherever along with the other functions. Example line 525.

```
/*
 *  ======== buttonCallback ========
 *  Pin interrupt Callback function board buttons configured in the pinTable.
 */
static void buttonCallback(PIN_Handle handle, PIN_Id pinId)
{
    /* Debounce logic, only toggle if the button is still pushed (low) */
    CPUdelay(8000*50);
    if (PIN_getInputValue(CONFIG_PIN_1) == 0)
                {

                size_t  bytesread;
                uint8_t buffer[32];

                UART2_Handle uart2;
                UART2_Params uartParams;
                UART2_Params_init(&uartParams);
                uartParams.baudRate = 9600;
                uart2 = UART2_open(CONFIG_UART2_0, &uartParams);

            if (uart2 == NULL) {
                    // UART_open() failed
                    while (1);
                }


                    UART2_read(uart2,&buffer,2,&bytesread);
                    UART2_close(uart2);
                    password=buffer[1];
                    nodeAddress=buffer[0];
                    //we need to send uart things here

        }
//button code

     /*
      * we want to take in uart data from the gateway node id and password.
      */


}
```

When the button is pushed, the node address on the node will be set to the node id, The quick workaround to send the password from the node to the gateway is setting the battery to be the password. In the while inside the `nodeRadioTaskFunction(UArg arg0, UArg arg1),` set the `dmSensorPacket.batt = password;` around line 234.

Wakeonreedgatewayfinal project modifications

The code to be edited in the wakonreedgatewayfinal project lies in reedgateway.c file.

The config pin for the button used to trigger the callback must be defined in the .syscfg file of the project along with the second UART.

1.Declare by the header files `#include <time.h> //line 14`

2.At type declarations define:

```
struct nodes_and_passwords{

    uint8_t node_id;
    uint8_t passwords;

};
```

3.Declare before the functions in reedgateway.c:

```
struct nodes_and_passwords confidentialids[255];
volatile uint8_t counter;
PIN_Config buttonPinTable[] = {
     CONFIG_PIN_6  | PIN_INPUT_EN | PIN_PULLUP | PIN_IRQ_NEGEDGE,
    PIN_TERMINATE
```

```
  };//here pin 6/btn-1 will be the trigger for the event, the concentrator will
then give a random node id and password for that specific node
static void buttonCallback(PIN_Handle handle, PIN_Id pinId);
static void passid(void);
```

CONFIG_PIN_6 corresponds to the BTN-1 pin on the LAUNCHXL CC1312R, this button pin must be defined in the .syscfg file. The correct path to find which config pin corresponds to the button is wakenoreedgatewayefinal/debug/syscfg/ti_drivers_config.h. Build the project and the syscfg will declare the config pins in ti_drivers_config.h.

4. Before the while inside `concentratorTaskFunction(UArg arg0, UArg arg1) declare:`

```
buttonPinHandle = PIN_open(&buttonPinState, buttonPinTable);
        if (!buttonPinHandle)
        {
            System_abort("Error initializing button pins\n");
        }
    /* Setup callback for button pins */
        if (PIN_registerIntCb(buttonPinHandle, &buttonCallback) != 0)
        {
            System_abort("Error registering button callback function");
        }

        if(confidentialids[2].node_id == 0)//if there is 0 in the struct meaning
there is nothing generated yet;
                {
                passid();
                }
```

5. Edit `static uint8_t isKnownNodeAddress(uint8_t address);`
 To `static uint8_t isKnownNodeAddress(uint8_t address,uint8_t password);`

6.Edit the while inside `concentratorTaskFunction(UArg arg0, UArg arg1):`

```
while(1) {
        /* Wait for event */
        uint32_t events = Event_pend(concentratorEventHandle, 0,
CONCENTRATOR_EVENT_ALL, BIOS_WAIT_FOREVER);

        /* If we got a new  sensor value *//* the value */

if(isKnownNodeAddress(latestActiveAdcSensorNode.address,latestActiveAdcSensorNode.
batt)) {
                updateNode(&latestActiveAdcSensorNode);//helper struct for
alternate storage and for future implementing with lcd on collector.
                updateuart2();
            }
            else {
                /* Else add it */

                addNewNode(&latestActiveAdcSensorNode);// if a node has unknown
node adress or unknown password it will be added to the helper struct
            }

     }
```

7. Complete editing isknownnodeadress() in functions definitions to:

```
static uint8_t isKnownNodeAddress(uint8_t address, uint8_t password) {
    uint8_t found = 0;
    uint8_t i;
    for (i = 0; i < Notes; i++)
    {
        if ((confidentialids[i].node_id == address)&&(confidentialids[i].passwords
== password))
        {
            found = 1;
            break;
        }
    }
    return found;
}
```

8. Define at bottom of function definitions:

```
/*
 *  ======== buttonCallback ========
 *  Pin interrupt Callback function board buttons configured in the pinTable.
 */
static void buttonCallback(PIN_Handle handle, PIN_Id pinId)
{
    /* Debounce logic, only toggle if the button is still pushed (low) */
    CPUdelay(8000*50);


       if (PIN_getInputValue(CONFIG_PIN_6) == 0)
           {

           size_t  bytesWritten;
           uint8_t input[16];

           UART2_Handle uart2;
           UART2_Params uartParams;
           UART2_Params_init(&uartParams);
           uartParams.baudRate = 9600;
           uart2 = UART2_open(CONFIG_UART2_1, &uartParams);

        if (uart2 == NULL) {
               // UART_open() failed
               while (1);
           }

            if(counter<255){
                input[0]  = confidentialids[counter].node_id;
                input[1]  = confidentialids[counter].passwords;



               UART2_write(uart2, &input, 2, &bytesWritten);

               //we need to send uart things here
               counter=counter+1; //the counter just iterates through the 0-255
different node and password pair in the confidentialids struct.
            }

            else if(counter>255)
            {
                counter=0;
            }
            UART2_close(uart2);
    }
}

static void passid(void)
{
//this code makes 255 password and node id pairs.

            uint8_t n;
            srand(time(NULL));

           for(n=0;n<255;n++){

            confidentialids[n].passwords = rand() % 255 + 1;
            confidentialids[n].node_id =  n;
           }

}
```

9. When the button is pushed, the node address on the node will be set to the node id, The quick workaround to send the password from the node to the gateway is setting the battery to be the password in the wakonreednodefinal project, see above. Setting two UART2 in sycfg will enable the launchxl cc1312r to use 1 uart channel on the DIO pins to communicate with the esp32 and another uart channel on the dio pins to communicate with the nodes when connected and the buttons pushed at the same time. However currently one must go into the udma.h header file in C:/ti/simpelink_cc13x2_26x2_sdk_3_40_00_02/source/ti/devices/cc13x2_26x2/driverlib/udma.h and define:

```
#DEFINE UDMA_CHAN_UART1_RX      5

#DEFINE UDMA_CHAN_UART1_TX      6
```

As a current workaround, this problem will be fixed in later versions of the SDK for cc13x2_26x2.