**Smart Doorlock Laboratory Notebook**

**Primary Author: Sokwhan Huh**

*This space is intentionally left blank*

# August 31st, 2016

Project Requirements

The core project requirements are as follows:

* Demonstrate NFC P2P capability
  + Be able to open the door lock with NFC using a smartphone app under P2P mode. The phone must be registered in advance to open the door lock.
* Demonstrate IoT capability
  + Be able to open or permanently lock (lockdown mode) the door lock over the internet using a smartphone app or by accessing Smart Doorlock’s web server with a browser. The phone must be registered in advance to open the door lock.
* Demonstrate enhanced security system
  + Be able to register phones for access.
  + Be able to clear existing registered phones.

Additional requirements are as follows:

* Implement full logging system for entries.
* Have the door lock function with a battery backup in case of power outage.

Technical Specifications

* Interfacing via NFC in P2P mode using Android Beam technology
* Interfacing via IoT to a cloud server over MQTT Protocol
* NFC communication range: 5 cm maximum (from smartphone to Smart Doorlock)
* Wi-Fi communication range: 50 meters maximum (TX 14.5dBM, RX -74.0dBM at 54 OFDM)
* Microcontroller: Texas Instruments CC3200
* NFC Transceiver IC: Texas Instruments TRF7970A
* AC power with DC battery backup
* Minimum backup battery life of 24 hours
* Keypad: Storm Interface’s Storm 720GFXI Series (7207-121W20)
* LCD: Electronic Assembly’s EA DIP203G-4NLED LCD
* Programming Language: C (Microcontroller), Java (Android Application), C# (Web Server)
* Supported Android versions: 4.0 (Ice Cream Sandwich) and above
* Web Server Framework: ASP.NET MVC 5
* Web Server Database: MySQL

**Source Control (Mercurial):** [**https://bitbucket.org/l46kok/smart-doorlock**](https://bitbucket.org/l46kok/smart-doorlock)

Started a new project and pushed a new source base into repository. To test that CC3200’s basic functionality works as intended, the following simple program was made, which is to be served as the skeleton of the entire project. The following code currently only does two things: initialize the microcontroller and send ASCII strings via UART TX.

|  |
| --- |
| **main.c** |
| /\*  \* Smart Doorlock  \* main.c  \*  \* June 15th, 2016  \*  \* Author: Sokwhan Huh  \*/  // Driverlib includes  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "hw\_memmap.h"  **#include** "hw\_common\_reg.h"  **#include** "hw\_types.h"  **#include** "hw\_ints.h"  **#include** "uart.h"  **#include** "interrupt.h"  **#include** "pinmux.h"  **#include** "utils.h"  **#include** "prcm.h"  // Common interface include  **#include** "uart\_if.h"  **#define** APPLICATION\_VERSION "1.1.1"  **#define** APP\_NAME "Smart Doorlock"  **#define** CONSOLE UARTA0\_BASE  **#define** UartGetChar() MAP\_UARTCharGet(CONSOLE)  **#define** UartPutChar(c) MAP\_UARTCharPut(CONSOLE,c)  **#define** MAX\_STRING\_LENGTH 80  **static** **void** **DisplayBanner**(**char** \* AppName)  {  **Report**("\n\n\n\r");  **Report**("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");  **Report**("\t\t \n\r", AppName);  **Report**("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");  **Report**("\n\n\n\r");  }  **static** **void** **BoardInit**(**void**)  {  /\* In case of TI-RTOS vector table is initialize by OS itself \*/  **#ifndef** USE\_TIRTOS  //  // Set vector table base  //  **#if** defined(ccs)  MAP\_IntVTableBaseSet((**unsigned** **long**)&g\_pfnVectors[0]);  **#endif**  **#if** defined(ewarm)  MAP\_IntVTableBaseSet((**unsigned** **long**)&\_\_vector\_table);  **#endif**  **#endif**  //  // Enable Processor  //  MAP\_IntMasterEnable();  MAP\_IntEnable(FAULT\_SYSTICK);  **PRCMCC3200MCUInit**();  }  **int** **main**(**void**) {  //  // Initailizing the board  //  BoardInit();  //  // Muxing for Enabling UART\_TX and UART\_RX.  //  PinMuxConfig();  //  // Initialising the Terminal.  //  **InitTerm**();  //  // Clearing the Terminal.  //  **ClearTerm**();  DisplayBanner(APP\_NAME);  **return** 0;  } |

|  |
| --- |
| **uart\_if.c** |
| /\*  \* Smart Doorlock  \*  \* uart\_if.c  \*  \* Created on: 2016. 8. 27.  \* Author: Sokwhan  \*/  // Standard includes  **#include** <stdarg.h>  **#include** <stdlib.h>  **#include** <stdio.h>  **#include** <string.h>  // Driverlib includes  **#include** "hw\_types.h"  **#include** "hw\_memmap.h"  **#include** "prcm.h"  **#include** "pin.h"  **#include** "uart.h"  **#include** "rom.h"  **#include** "rom\_map.h"  **#if** defined(USE\_FREERTOS) || defined(USE\_TI\_RTOS)  **#include** "osi.h"  **#endif**  **#include** "uart\_if.h"  **#define** IS\_SPACE(x) (x == 32 ? 1 : 0)  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Global variable indicating command is present  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **static** **unsigned** **long** \_\_Errorlog;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Global variable indicating input length  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **unsigned** **int** ilen=1;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Initialization  //!  //! This function  //! 1. Configures the UART to be used.  //!  //! \return none  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **InitTerm**()  {  **#ifndef** NOTERM  MAP\_UARTConfigSetExpClk(CONSOLE,MAP\_PRCMPeripheralClockGet(CONSOLE\_PERIPH),  UART\_BAUD\_RATE, (UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE |  UART\_CONFIG\_PAR\_NONE));  **#endif**  \_\_Errorlog = 0;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Outputs a character string to the console  //!  //! \param str is the pointer to the string to be printed  //!  //! This function  //! 1. prints the input string character by character on to the console.  //!  //! \return none  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **Message**(**const** **char** \*str)  {  **#ifndef** NOTERM  **if**(str != NULL)  {  **while**(\*str!='\0')  {  MAP\_UARTCharPut(CONSOLE,\*str++);  }  }  **#endif**  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Clear the console window  //!  //! This function  //! 1. clears the console window.  //!  //! \return none  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **ClearTerm**()  {  Message("\33[2J\r");  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Error Function  //!  //! \param  //!  //! \return none  //!  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **Error**(**char** \*pcFormat, ...)  {  **#ifndef** NOTERM  **char** cBuf[256];  va\_list list;  va\_start(list,pcFormat);  **vsnprintf**(cBuf,256,pcFormat,list);  Message(cBuf);  **#endif**  \_\_Errorlog++;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Get the Command string from UART  //!  //! \param pucBuffer is the command store to which command will be populated  //! \param ucBufLen is the length of buffer store available  //!  //! \return Length of the bytes received. -1 if buffer length exceeded.  //!  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **int**  **GetCmd**(**char** \*pcBuffer, **unsigned** **int** uiBufLen)  {  **char** cChar;  **int** iLen = 0;  //  // Wait to receive a character over UART  //  **while**(MAP\_UARTCharsAvail(CONSOLE) == false)  {  **#if** defined(USE\_FREERTOS) || defined(USE\_TI\_RTOS)  osi\_Sleep(1);  **#endif**  }  cChar = MAP\_UARTCharGetNonBlocking(CONSOLE);  //  // Echo the received character  //  MAP\_UARTCharPut(CONSOLE, cChar);  iLen = 0;  //  // Checking the end of Command  //  **while**((cChar != '\r') && (cChar !='\n') )  {  //  // Handling overflow of buffer  //  **if**(iLen >= uiBufLen)  {  **return** -1;  }  //  // Copying Data from UART into a buffer  //  **if**(cChar != '\b')  {  \*(pcBuffer + iLen) = cChar;  iLen++;  }  **else**  {  //  // Deleting last character when you hit backspace  //  **if**(iLen)  {  iLen--;  }  }  //  // Wait to receive a character over UART  //  **while**(MAP\_UARTCharsAvail(CONSOLE) == false)  {  **#if** defined(USE\_FREERTOS) || defined(USE\_TI\_RTOS)  osi\_Sleep(1);  **#endif**  }  cChar = MAP\_UARTCharGetNonBlocking(CONSOLE);  //  // Echo the received character  //  MAP\_UARTCharPut(CONSOLE, cChar);  }  \*(pcBuffer + iLen) = '\0';  Report("\n\r");  **return** iLen;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Trim the spaces from left and right end of given string  //!  //! \param Input string on which trimming happens  //!  //! \return length of trimmed string  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **int** **TrimSpace**(**char** \* pcInput)  {  size\_t size;  **char** \*endStr, \*strData = pcInput;  **char** index = 0;  size = **strlen**(strData);  **if** (!size)  **return** 0;  endStr = strData + size - 1;  **while** (endStr >= strData && IS\_SPACE(\*endStr))  endStr--;  \*(endStr + 1) = '\0';  **while** (\*strData && IS\_SPACE(\*strData))  {  strData++;  index++;  }  **memmove**(pcInput,strData,**strlen**(strData)+1);  **return** **strlen**(pcInput);  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! prints the formatted string on to the console  //!  //! \param format is a pointer to the character string specifying the format in  //! the following arguments need to be interpreted.  //! \param [variable number of] arguments according to the format in the first  //! parameters  //! This function  //! 1. prints the formatted error statement.  //!  //! \return count of characters printed  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **int** **Report**(**const** **char** \*pcFormat, ...)  {  **int** iRet = 0;  **#ifndef** NOTERM  **char** \*pcBuff, \*pcTemp;  **int** iSize = 256;  va\_list list;  pcBuff = (**char**\*)**malloc**(iSize);  **if**(pcBuff == NULL)  {  **return** -1;  }  **while**(1)  {  va\_start(list,pcFormat);  iRet = **vsnprintf**(pcBuff,iSize,pcFormat,list);  va\_end(list);  **if**(iRet > -1 && iRet < iSize)  {  **break**;  }  **else**  {  iSize\*=2;  **if**((pcTemp=**realloc**(pcBuff,iSize))==NULL)  {  Message("Could not reallocate memory\n\r");  iRet = -1;  **break**;  }  **else**  {  pcBuff=pcTemp;  }  }  }  Message(pcBuff);  **free**(pcBuff);  **#endif**  **return** iRet;  } |

|  |
| --- |
| **uart\_if.c** |
| /\*  \* Smart Doorlock  \*  \* uart\_if.c  \*  \* Created on: 2016. 8. 27.  \* Author: Sokwhan  \*/  // Standard includes  **#include** <stdarg.h>  **#include** <stdlib.h>  **#include** <stdio.h>  **#include** <string.h>  // Driverlib includes  **#include** "hw\_types.h"  **#include** "hw\_memmap.h"  **#include** "prcm.h"  **#include** "pin.h"  **#include** "uart.h"  **#include** "rom.h"  **#include** "rom\_map.h"  **#if** defined(USE\_FREERTOS) || defined(USE\_TI\_RTOS)  **#include** "osi.h"  **#endif**  **#include** "uart\_if.h"  **#define** IS\_SPACE(x) (x == 32 ? 1 : 0)  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Global variable indicating command is present  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **static** **unsigned** **long** \_\_Errorlog;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Global variable indicating input length  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **unsigned** **int** ilen=1;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Initialization  //!  //! This function  //! 1. Configures the UART to be used.  //!  //! \return none  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **InitTerm**()  {  **#ifndef** NOTERM  MAP\_UARTConfigSetExpClk(CONSOLE,MAP\_PRCMPeripheralClockGet(CONSOLE\_PERIPH),  UART\_BAUD\_RATE, (UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE |  UART\_CONFIG\_PAR\_NONE));  **#endif**  \_\_Errorlog = 0;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Outputs a character string to the console  //!  //! \param str is the pointer to the string to be printed  //!  //! This function  //! 1. prints the input string character by character on to the console.  //!  //! \return none  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **Message**(**const** **char** \*str)  {  **#ifndef** NOTERM  **if**(str != NULL)  {  **while**(\*str!='\0')  {  MAP\_UARTCharPut(CONSOLE,\*str++);  }  }  **#endif**  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Clear the console window  //!  //! This function  //! 1. clears the console window.  //!  //! \return none  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **ClearTerm**()  {  Message("\33[2J\r");  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Error Function  //!  //! \param  //!  //! \return none  //!  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **Error**(**char** \*pcFormat, ...)  {  **#ifndef** NOTERM  **char** cBuf[256];  va\_list list;  va\_start(list,pcFormat);  **vsnprintf**(cBuf,256,pcFormat,list);  Message(cBuf);  **#endif**  \_\_Errorlog++;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Get the Command string from UART  //!  //! \param pucBuffer is the command store to which command will be populated  //! \param ucBufLen is the length of buffer store available  //!  //! \return Length of the bytes received. -1 if buffer length exceeded.  //!  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **int**  **GetCmd**(**char** \*pcBuffer, **unsigned** **int** uiBufLen)  {  **char** cChar;  **int** iLen = 0;  //  // Wait to receive a character over UART  //  **while**(MAP\_UARTCharsAvail(CONSOLE) == false)  {  **#if** defined(USE\_FREERTOS) || defined(USE\_TI\_RTOS)  osi\_Sleep(1);  **#endif**  }  cChar = MAP\_UARTCharGetNonBlocking(CONSOLE);  //  // Echo the received character  //  MAP\_UARTCharPut(CONSOLE, cChar);  iLen = 0;  //  // Checking the end of Command  //  **while**((cChar != '\r') && (cChar !='\n') )  {  //  // Handling overflow of buffer  //  **if**(iLen >= uiBufLen)  {  **return** -1;  }  //  // Copying Data from UART into a buffer  //  **if**(cChar != '\b')  {  \*(pcBuffer + iLen) = cChar;  iLen++;  }  **else**  {  //  // Deleting last character when you hit backspace  //  **if**(iLen)  {  iLen--;  }  }  //  // Wait to receive a character over UART  //  **while**(MAP\_UARTCharsAvail(CONSOLE) == false)  {  **#if** defined(USE\_FREERTOS) || defined(USE\_TI\_RTOS)  osi\_Sleep(1);  **#endif**  }  cChar = MAP\_UARTCharGetNonBlocking(CONSOLE);  //  // Echo the received character  //  MAP\_UARTCharPut(CONSOLE, cChar);  }  \*(pcBuffer + iLen) = '\0';  Report("\n\r");  **return** iLen;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Trim the spaces from left and right end of given string  //!  //! \param Input string on which trimming happens  //!  //! \return length of trimmed string  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **int** **TrimSpace**(**char** \* pcInput)  {  size\_t size;  **char** \*endStr, \*strData = pcInput;  **char** index = 0;  size = **strlen**(strData);  **if** (!size)  **return** 0;  endStr = strData + size - 1;  **while** (endStr >= strData && IS\_SPACE(\*endStr))  endStr--;  \*(endStr + 1) = '\0';  **while** (\*strData && IS\_SPACE(\*strData))  {  strData++;  index++;  }  **memmove**(pcInput,strData,**strlen**(strData)+1);  **return** **strlen**(pcInput);  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! prints the formatted string on to the console  //!  //! \param format is a pointer to the character string specifying the format in  //! the following arguments need to be interpreted.  //! \param [variable number of] arguments according to the format in the first  //! parameters  //! This function  //! 1. prints the formatted error statement.  //!  //! \return count of characters printed  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **int** **Report**(**const** **char** \*pcFormat, ...)  {  **int** iRet = 0;  **#ifndef** NOTERM  **char** \*pcBuff, \*pcTemp;  **int** iSize = 256;  va\_list list;  pcBuff = (**char**\*)**malloc**(iSize);  **if**(pcBuff == NULL)  {  **return** -1;  }  **while**(1)  {  va\_start(list,pcFormat);  iRet = **vsnprintf**(pcBuff,iSize,pcFormat,list);  va\_end(list);  **if**(iRet > -1 && iRet < iSize)  {  **break**;  }  **else**  {  iSize\*=2;  **if**((pcTemp=**realloc**(pcBuff,iSize))==NULL)  {  Message("Could not reallocate memory\n\r");  iRet = -1;  **break**;  }  **else**  {  pcBuff=pcTemp;  }  }  }  Message(pcBuff);  **free**(pcBuff);  **#endif**  **return** iRet;  } |

|  |
| --- |
| **uart\_if.h** |
| /\*  \* Smart Doorlock  \*  \* uart\_if.h  \*  \* Created on: 2016. 8. 27.  \* Author: Sokwhan  \*/  **#ifndef** \_\_uart\_if\_H\_\_  **#define** \_\_uart\_if\_H\_\_  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  // If building with a C++ compiler, make all of the definitions in this header  // have a C binding.  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **#ifdef** \_\_cplusplus  **extern** "C"  {  **#endif**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* MACROS \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  **#define** UART\_BAUD\_RATE 115200  **#define** SYSCLK 80000000  **#define** CONSOLE UARTA0\_BASE  **#define** CONSOLE\_PERIPH PRCM\_UARTA0  //  // Define the size of UART IF buffer for RX  //  **#define** UART\_IF\_BUFFER 64  //  // Define the UART IF buffer  //  **extern** **unsigned** **char** g\_ucUARTBuffer[];  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* FUNCTION PROTOTYPES \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  **extern** **void** **DispatcherUARTConfigure**(**void**);  **extern** **void** **DispatcherUartSendPacket**(**unsigned** **char** \*inBuff, **unsigned** **short** usLength);  **extern** **int** **GetCmd**(**char** \*pcBuffer, **unsigned** **int** uiBufLen);  **extern** **void** **InitTerm**(**void**);  **extern** **void** **ClearTerm**(**void**);  **extern** **void** **Message**(**char** \*format);  **extern** **void** **Error**(**char** \*format,...);  **extern** **int** **Report**(**char** \*format, ...);  **void** **UartPutChar**(**unsigned** **char** ch);  **void** **UartPutCrlf**(**void**);  **extern** **void** **UartSendCString**(**char** \*format);  **void** **UartPutByte**(**unsigned** **char** ch);  **void** **UartPutByteHex**(**unsigned** **char** ch);  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  // Mark the end of the C bindings section for C++ compilers.  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **#ifdef** \_\_cplusplus  }  **#endif**  **#endif** |

|  |
| --- |
| **pinmux.c** |
| /\*  \* Smart Doorlock  \*  \* pinmux.c  \*  \* Created on: 2016. 8. 27.  \* Author: Sokwhan  \*/  **#include** "pinmux.h"  **#include** "hw\_types.h"  **#include** "hw\_memmap.h"  **#include** "hw\_gpio.h"  **#include** "pin.h"  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "gpio.h"  **#include** "prcm.h"  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **PinMuxConfig**(**void**)  {  //  // Enable Peripheral Clocks  //  MAP\_PRCMPeripheralClkEnable(PRCM\_UARTA0, PRCM\_RUN\_MODE\_CLK);  //  // Configure PIN\_55 for UART0 UART0\_TX  //  MAP\_PinTypeUART(PIN\_55, PIN\_MODE\_3);  //  // Configure PIN\_57 for UART0 UART0\_RX  //  MAP\_PinTypeUART(PIN\_57, PIN\_MODE\_3);  } |

|  |
| --- |
| **pinmux.h** |
| /\*  \* Smart Doorlock  \*  \* pinmux.h  \*  \* Created on: 2016. 8. 27.  \* Author: Sokwhan  \*/  **#ifndef** PINMUX\_H\_  **#define** PINMUX\_H\_  **extern** **void** **PinMuxConfig**(**void**);  **#endif** /\* PINMUX\_H\_ \*/ |

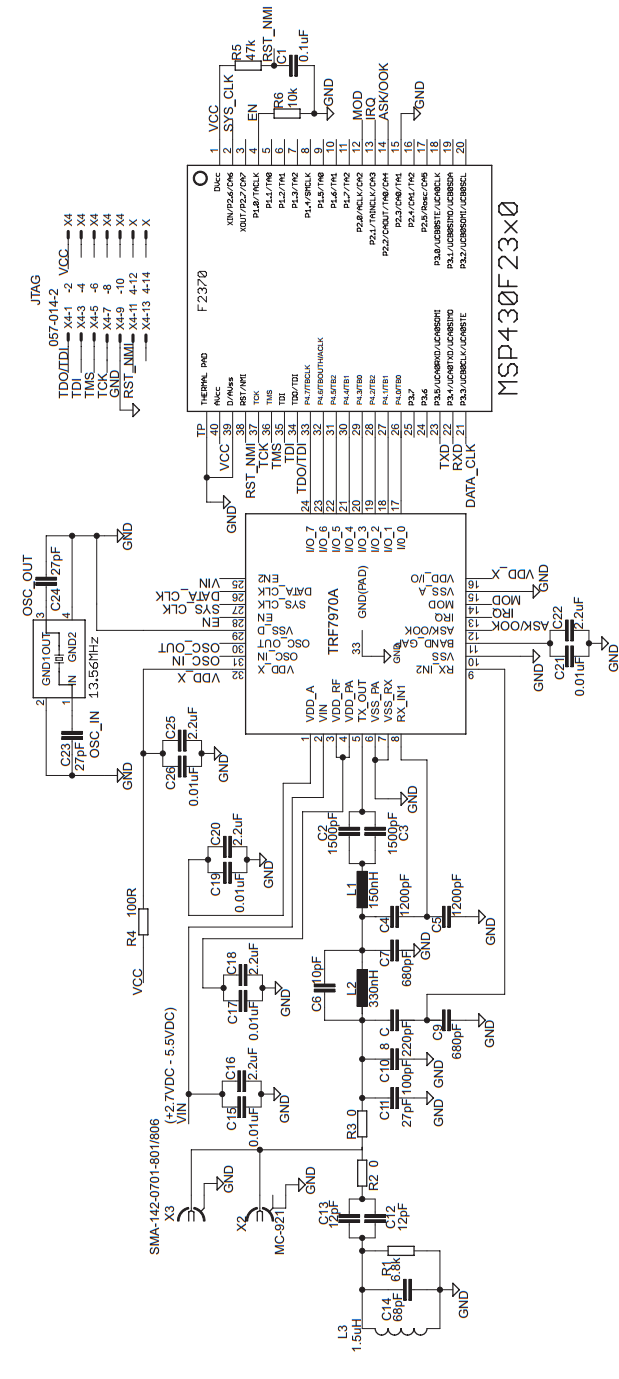
# September 1st, 2016

Further research was performed on NFC communication via tag on TRF7970A. The demo NFCReaderApp from TI would not work for some reason, potentially due to version difference in TRF7970A. Online findings indicated that they’ve changed the pin configurations on the board and the code would have to match the newly set pins. After changing the pin configurations, still no luck so I made a post on TI forums and see if anyone else has an idea. A TI employee answered the post, looked over my code modifications and saw no issues with it. He advised that I solder the jumper pin on IRQ to revert the pin settings back to old version. Will have to try this in lab later. The actual forum link is the following: <http://e2e.ti.com/support/wireless_connectivity/nfc_rfid/f/667/p/518595/1905610#1905610>

Another option is to just jump directly to NFC P2P, but I’m a bit skeptical on whether this approach is sound or not as NFC P2P tends to be a lot more complicated than simple tag read. If I can’t get the NFC tag to work, then I guess this is the alternative route I have to take.

If only I could get my hands on the TRF7970A breakout board, life would be so much easier but they are way too expensive…

# September 4th, 2016

There are couple of things I missed in the initial schematic of this project. Below is a sample schematic of TRF7970A from documentation:

**2**

**1**

First, TRF7970A requires a separate crystal oscillator clocked at 13.56MHz. The development board comes with one, so I mistakenly assumed that the actual chip has an internal clock integrated with it [1]. The second, and probably the bigger problem is that I need to build an entire RF circuitry for the NFC communication to work [2]. I have no prior background in RF so this will pose a huge challenge for me, but it must be catered for before the PCB design. I’m pretty sure CC3200 would require a separate RF circuitry design for Wi-Fi communication but this has not been verified yet. The example seems to be using 50-ohm female-type antenna (with the actual “pole?” thing to be plugged in later), if I am not mistaken, but I will have to find a different schematic as I’ll be using a flat card-like antenna found in most NFC devices.

There are no DIP modules for TRF7970A either so I basically have zero way of testing the RF stuff. I will also have to find out a way of testing this RF circuitry before printing the PCB (if possible).

# September 7th, 2016

RTOS functionality has been included to the project. CC3200 will now spawn a separate thread for executing the main logic of the application. GPIO code was also added for blinking LEDs and handling interrupts in the future.

|  |
| --- |
| **main.c** |
| /\*  \* Smart Doorlock  \* main.c  \*  \* Created on: 2016. 8. 27.  \*  \* Author: Sokwhan Huh  \*/  // C-Library includes  **#include** <string.h>  // Driverlib includes  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "hw\_memmap.h"  **#include** "hw\_common\_reg.h"  **#include** "hw\_types.h"  **#include** "hw\_ints.h"  **#include** "uart.h"  **#include** "interrupt.h"  **#include** "pinmux.h"  **#include** "utils.h"  **#include** "prcm.h"  **#include** "simplelink.h"  // Common interface include  **#include** "uart\_if.h"  **#include** "gpio\_if.h"  **#define** APP\_NAME "Smart Doorlock"  //RTOS Related Defines  **#define** OSI\_STACK\_SIZE 4096 /\* 2048 \*/  **#define** SPAWN\_TASK\_PRIORITY 9  **static** **void** **DisplayBanner**(**char** \* AppName)  {  **Report**("\n\n\n\r");  **Report**("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");  **Report**("\t\t %s \n\r", AppName);  **Report**("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");  **Report**("\n\n\n\r");  }  **static** **void** **BoardInit**(**void**)  {  // Enable Processor  MAP\_IntMasterEnable();  MAP\_IntEnable(FAULT\_SYSTICK);  **PRCMCC3200MCUInit**();  }  /\*\*  \*  \*/  **void** **SmartDoorlockApp**(**void** \*pvParameters) {  GPIO\_IF\_LedConfigure(*LED1*|*LED2*|*LED3*);  **for** (;;) {  GPIO\_IF\_LedOn(*MCU\_RED\_LED\_GPIO*);  **osi\_Sleep**(500);  GPIO\_IF\_LedOff(*MCU\_RED\_LED\_GPIO*);  **osi\_Sleep**(500);  }  }  **int** **main**(**void**) {  // Initailizing the board  BoardInit();  // Muxing for Enabling GPIO, UART\_TX and UART\_RX.  PinMuxConfig();  // Init Terminal  **InitTerm**();  **ClearTerm**();  DisplayBanner(APP\_NAME);  //Start the simplelink host  **VStartSimpleLinkSpawnTask**(SPAWN\_TASK\_PRIORITY);  // Start the SmartDoorlock task  **osi\_TaskCreate**( SmartDoorlockApp,  (**const** **signed** **char**\*)"Smart Doorlock App",  OSI\_STACK\_SIZE, NULL, 1, NULL );  **osi\_start**();  **return** 0;  } |

|  |
| --- |
| **gpio\_if.c** |
| /\*  \* gpio\_if.c  \*  \* Created on: 2016. 9. 7.  \* Author: Sokwhan  \*/  **#include** <stdio.h>  **#include** "hw\_types.h"  **#include** "hw\_ints.h"  **#include** "hw\_memmap.h"  **#include** "interrupt.h"  **#include** "pin.h"  **#include** "gpio.h"  **#include** "hw\_apps\_rcm.h"  **#include** "prcm.h"  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "gpio\_if.h"  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // GLOBAL VARIABLES  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **static** **unsigned** **long** ulReg[]=  {  GPIOA0\_BASE,  GPIOA1\_BASE,  GPIOA2\_BASE,  GPIOA3\_BASE  };  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Variables to store TIMER Port,Pin values  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **unsigned** **int** g\_uiLED1Port = 0,g\_uiLED2Port = 0,g\_uiLED3Port = 0;  **unsigned** **char** g\_ucLED1Pin, g\_ucLED2Pin, g\_ucLED3Pin;  **extern** **unsigned** **char** g\_heart\_beat, g\_state;  **#define** PIN\_LED1 9  **#define** PIN\_LED2 10  **#define** PIN\_LED3 11  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // LOCAL FUNCTION DEFINITIONS  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! GPIO Enable & Configuration  //!  //! \param ucPins is the bit-pack representation of 3 LEDs  //! LSB:GP09-GP10-GP11:MSB  //!  //! \return None  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **GPIO\_IF\_LedConfigure**(**unsigned** **char** ucPins)  {  **if**(ucPins & *LED1*)  {  GPIO\_IF\_GetPortNPin(PIN\_LED1,  &g\_uiLED1Port,  &g\_ucLED1Pin);  }  **if**(ucPins & *LED2*)  {  GPIO\_IF\_GetPortNPin(PIN\_LED2,  &g\_uiLED2Port,  &g\_ucLED2Pin);  }  **if**(ucPins & *LED3*)  {  GPIO\_IF\_GetPortNPin(PIN\_LED3,  &g\_uiLED3Port,  &g\_ucLED3Pin);  }  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Turn LED On  //!  //! \param ledNum is the LED Number  //!  //! \return none  //!  //! \brief Turns a specific LED Off  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **GPIO\_IF\_LedOn**(**char** ledNum)  {  **switch**(ledNum)  {  **case** *MCU\_GREEN\_LED\_GPIO*:  {  /\* Switch ON GREEN LED \*/  GPIO\_IF\_Set(PIN\_LED3, g\_uiLED3Port, g\_ucLED3Pin, 1);  **break**;  }  **case** *MCU\_ORANGE\_LED\_GPIO*:  {  /\* Switch ON ORANGE LED \*/  GPIO\_IF\_Set(PIN\_LED2, g\_uiLED2Port, g\_ucLED2Pin, 1);  **break**;  }  **case** *MCU\_RED\_LED\_GPIO*:  {  /\* Switch ON RED LED \*/  GPIO\_IF\_Set(PIN\_LED1, g\_uiLED1Port, g\_ucLED1Pin, 1);  **break**;  }  **case** *MCU\_ALL\_LED\_IND*:  {  /\* Switch ON ALL LEDs LED \*/  GPIO\_IF\_Set(PIN\_LED3, g\_uiLED3Port, g\_ucLED3Pin, 1);  GPIO\_IF\_Set(PIN\_LED2, g\_uiLED2Port, g\_ucLED2Pin, 1);  GPIO\_IF\_Set(PIN\_LED1, g\_uiLED1Port, g\_ucLED1Pin, 1);  **break**;  }  **default**:  **break**;  }  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Turn LED Off  //!  //! \param ledNum is the LED Number  //!  //! \return none  //!  //! \brief Turns a specific LED Off  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **GPIO\_IF\_LedOff**(**char** ledNum)  {  **switch**(ledNum)  {  **case** *MCU\_GREEN\_LED\_GPIO*:  {  /\* Switch OFF GREEN LED \*/  GPIO\_IF\_Set(PIN\_LED3, g\_uiLED3Port, g\_ucLED3Pin, 0);  **break**;  }  **case** *MCU\_ORANGE\_LED\_GPIO*:  {  /\* Switch OFF ORANGE LED \*/  GPIO\_IF\_Set(PIN\_LED2, g\_uiLED2Port, g\_ucLED2Pin, 0);  **break**;  }  **case** *MCU\_RED\_LED\_GPIO*:  {  /\* Switch OFF RED LED \*/  GPIO\_IF\_Set(PIN\_LED1, g\_uiLED1Port, g\_ucLED1Pin, 0);  **break**;  }  **case** *MCU\_ALL\_LED\_IND*:  {  /\* Switch OFF ALL LEDs LED \*/  GPIO\_IF\_Set(PIN\_LED3, g\_uiLED3Port, g\_ucLED3Pin, 0);  GPIO\_IF\_Set(PIN\_LED2, g\_uiLED2Port, g\_ucLED2Pin, 0);  GPIO\_IF\_Set(PIN\_LED1, g\_uiLED1Port, g\_ucLED1Pin, 0);  **break**;  }  **default**:  **break**;  }  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! \brief This function returns LED current Status  //!  //! \param[in] ucGPIONum is the GPIO to which the LED is connected  //! MCU\_GREEN\_LED\_GPIO\MCU\_ORANGE\_LED\_GPIO\MCU\_RED\_LED\_GPIO  //!  //!  //! \return 1: LED ON, 0: LED OFF  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **unsigned** **char**  **GPIO\_IF\_LedStatus**(**unsigned** **char** ucGPIONum)  {  **unsigned** **char** ucLEDStatus;  **switch**(ucGPIONum)  {  **case** *MCU\_GREEN\_LED\_GPIO*:  {  ucLEDStatus = GPIO\_IF\_Get(ucGPIONum, g\_uiLED3Port, g\_ucLED3Pin);  **break**;  }  **case** *MCU\_ORANGE\_LED\_GPIO*:  {  ucLEDStatus = GPIO\_IF\_Get(ucGPIONum, g\_uiLED2Port, g\_ucLED2Pin);  **break**;  }  **case** *MCU\_RED\_LED\_GPIO*:  {  ucLEDStatus = GPIO\_IF\_Get(ucGPIONum, g\_uiLED1Port, g\_ucLED1Pin);  **break**;  }  **default**:  ucLEDStatus = 0;  }  **return** ucLEDStatus;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Toggle the Led state  //!  //! \param ledNum is the LED Number  //!  //! \return none  //!  //! \brief Toggles a board LED  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void** **GPIO\_IF\_LedToggle**(**unsigned** **char** ucLedNum)  {  **unsigned** **char** ucLEDStatus = GPIO\_IF\_LedStatus(ucLedNum);  **if**(ucLEDStatus == 1)  {  GPIO\_IF\_LedOff(ucLedNum);  }  **else**  {  GPIO\_IF\_LedOn(ucLedNum);  }  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Get the port and pin of a given GPIO  //!  //! \param ucPin is the pin to be set-up as a GPIO (0:39)  //! \param puiGPIOPort is the pointer to store GPIO port address return value  //! \param pucGPIOPin is the pointer to store GPIO pin return value  //!  //! This function  //! 1. Return the GPIO port address and pin for a given external pin number  //!  //! \return None.  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **GPIO\_IF\_GetPortNPin**(**unsigned** **char** ucPin,  **unsigned** **int** \*puiGPIOPort,  **unsigned** **char** \*pucGPIOPin)  {  //  // Get the GPIO pin from the external Pin number  //  \*pucGPIOPin = 1 << (ucPin % 8);  //  // Get the GPIO port from the external Pin number  //  \*puiGPIOPort = (ucPin / 8);  \*puiGPIOPort = ulReg[\*puiGPIOPort];  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Configures the GPIO selected as input to generate interrupt on activity  //!  //! \param uiGPIOPort is the GPIO port address  //! \param ucGPIOPin is the GPIO pin of the specified port  //! \param uiIntType is the type of the interrupt (refer gpio.h)  //! \param pfnIntHandler is the interrupt handler to register  //!  //! This function  //! 1. Sets GPIO interrupt type  //! 2. Registers Interrupt handler  //! 3. Enables Interrupt  //!  //! \return None  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **GPIO\_IF\_ConfigureNIntEnable**(**unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin,  **unsigned** **int** uiIntType,  **void** (\*pfnIntHandler)(**void**))  {  //  // Set GPIO interrupt type  //  MAP\_GPIOIntTypeSet(uiGPIOPort,ucGPIOPin,uiIntType);  //  // Register Interrupt handler  //  MAP\_GPIOIntRegister(uiGPIOPort,pfnIntHandler);  //  // Enable Interrupt  //  MAP\_GPIOIntClear(uiGPIOPort,ucGPIOPin);  MAP\_GPIOIntEnable(uiGPIOPort,ucGPIOPin);  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Set a value to the specified GPIO pin  //!  //! \param ucPin is the GPIO pin to be set (0:39)  //! \param uiGPIOPort is the GPIO port address  //! \param ucGPIOPin is the GPIO pin of the specified port  //! \param ucGPIOValue is the value to be set  //!  //! This function  //! 1. Sets a value to the specified GPIO pin  //!  //! \return None.  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **GPIO\_IF\_Set**(**unsigned** **char** ucPin,  **unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin,  **unsigned** **char** ucGPIOValue)  {  //  // Set the corresponding bit in the bitmask  //  ucGPIOValue = ucGPIOValue << (ucPin % 8);  //  // Invoke the API to set the value  //  MAP\_GPIOPinWrite(uiGPIOPort,ucGPIOPin,ucGPIOValue);  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Set a value to the specified GPIO pin  //!  //! \param ucPin is the GPIO pin to be set (0:39)  //! \param uiGPIOPort is the GPIO port address  //! \param ucGPIOPin is the GPIO pin of the specified port  //!  //! This function  //! 1. Gets a value of the specified GPIO pin  //!  //! \return value of the GPIO pin  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **unsigned** **char**  **GPIO\_IF\_Get**(**unsigned** **char** ucPin,  **unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin)  {  **unsigned** **char** ucGPIOValue;  **long** lGPIOStatus;  //  // Invoke the API to Get the value  //  lGPIOStatus = MAP\_GPIOPinRead(uiGPIOPort,ucGPIOPin);  //  // Set the corresponding bit in the bitmask  //  ucGPIOValue = lGPIOStatus >> (ucPin % 8);  **return** ucGPIOValue;  }  **void** **GPIO\_Set**(**unsigned** **char** ucGPIONum){  **int** ucGPIOPin, uiGPIOPort;  ucGPIOPin = 1 << (ucGPIONum % 8);  uiGPIOPort = ulReg[ucGPIONum / 8];  MAP\_GPIOPinWrite(uiGPIOPort,ucGPIOPin,0xff);  }  **void** **GPIO\_Clear**(**unsigned** **char** ucGPIONum){  **int** ucGPIOPin, uiGPIOPort;  ucGPIOPin = 1 << (ucGPIONum % 8);  uiGPIOPort = ulReg[ucGPIONum / 8];  MAP\_GPIOPinWrite(uiGPIOPort,ucGPIOPin,0x0);  }  **void** **GPIOIntInit**(**unsigned** **long** ulPort, **unsigned** **char** ucPin, **unsigned** **long** ulInterrupt, **void** (\*pfnHandler)(**void**), **unsigned** **long** ulIntType, **unsigned** **char** ucPriority) {  **IntRegister**(ulInterrupt, pfnHandler);  **IntPrioritySet**(ulInterrupt, INT\_PRIORITY\_LVL\_1);  **GPIOIntTypeSet**(ulPort,ucPin,ulIntType);  **GPIOIntClear**(ulPort,0xFF);  **GPIOIntEnable**(ulPort,ucPin);  **IntEnable**(ulInterrupt);  } |

|  |
| --- |
| **gpio\_if.h** |
| /\*  \* gpio\_if.h  \*  \* Created on: 2016. 9. 7.  \* Author: Sokwhan  \*/  **#ifndef** \_\_GPIOIF\_H\_\_  **#define** \_\_GPIOIF\_H\_\_  **typedef** **enum**  {  *NO\_LED*,  *LED1* = 0x1, /\* RED LED D7/GP9/Pin64 \*/  *LED2* = 0x2, /\* ORANGE LED D6/GP10/Pin1 \*/  *LED3* = 0x4 /\* GREEN LED D5/GP11/Pin2 \*/  } ledEnum;  **typedef** **enum**  {  *MCU\_RED\_LED\_GPIO* = 9, /\* GP09 for LED RED as per LP 3.0 \*/  *MCU\_ORANGE\_LED\_GPIO* = 10,/\* GP10 for LED ORANGE as per LP 3.0 \*/  *MCU\_GREEN\_LED\_GPIO* = 11, /\* GP11 for LED GREEN as per LP 3.0 \*/  *MCU\_ALL\_LED\_IND* = 12  } ledNames;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  // API Function prototypes  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **extern** **void** **GPIO\_IF\_GetPortNPin**(**unsigned** **char** ucPin,  **unsigned** **int** \*puiGPIOPort,  **unsigned** **char** \*pucGPIOPin);  **extern** **void** **GPIO\_IF\_ConfigureNIntEnable**(**unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin,  **unsigned** **int** uiIntType,  **void** (\*pfnIntHandler)(**void**));  **extern** **void** **GPIO\_IF\_Set**(**unsigned** **char** ucPin,  **unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin,  **unsigned** **char** ucGPIOValue);  **extern** **unsigned** **char** **GPIO\_IF\_Get**(**unsigned** **char** ucPin,  **unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin);  **extern** **void** **GPIO\_IF\_LedConfigure**(**unsigned** **char** ucPins);  **extern** **void** **GPIO\_IF\_LedOn**(**char** ledNum);  **extern** **void** **GPIO\_IF\_LedOff**(**char** ledNum);  **extern** **unsigned** **char** **GPIO\_IF\_LedStatus**(**unsigned** **char** ucGPIONum);  **extern** **void** **GPIO\_IF\_LedToggle**(**unsigned** **char** ucLedNum);  **extern** **void** **GPIO\_Set**(**unsigned** **char** ucGPIONum);  **extern** **void** **GPIO\_Clear**(**unsigned** **char** ucGPIONum);  **extern** **void** **GPIO\_Toggle**(**unsigned** **char** ucGPIONum);  **void** **GPIOIntInit**( **unsigned** **long** ulPort,  **unsigned** **char** ucPin,  **unsigned** **long** ulInterrupt,  **void** (\*pfnHandler)(**void**),  **unsigned** **long** ulIntType,  **unsigned** **char** ucPriority);  **void** **GPIOs3IntHandler**(**void**);  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  // Mark the end of the C bindings section for C++ compilers.  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **#ifdef** \_\_cplusplus  }  **#endif**  **#endif** // \_\_GPIOIF\_H\_\_ |

|  |
| --- |
| **pinmux.c** |
| /\*  \* Smart Doorlock  \*  \* pinmux.c  \*  \* Created on: 2016. 8. 27.  \* Author: Sokwhan  \*/  **#include** "pinmux.h"  **#include** "hw\_types.h"  **#include** "hw\_memmap.h"  **#include** "hw\_gpio.h"  **#include** "pin.h"  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "gpio.h"  **#include** "prcm.h"  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **PinMuxConfig**(**void**)  {  // Enable Peripheral Clocks  MAP\_PRCMPeripheralClkEnable(PRCM\_UARTA0, PRCM\_RUN\_MODE\_CLK);  // Reserved for Future Use (Probably IRQ)  MAP\_PRCMPeripheralClkEnable(PRCM\_GPIOA0, PRCM\_RUN\_MODE\_CLK);  // Used for LED1,2,3  MAP\_PRCMPeripheralClkEnable(PRCM\_GPIOA1, PRCM\_RUN\_MODE\_CLK);  // Configure PIN\_55 for UART0 UART0\_TX  MAP\_PinTypeUART(PIN\_55, PIN\_MODE\_3);  // Configure PIN\_57 for UART0 UART0\_RX  MAP\_PinTypeUART(PIN\_57, PIN\_MODE\_3);  // Configure PIN\_64 for LED1 (GPIO9)  MAP\_PinTypeGPIO(PIN\_64, PIN\_MODE\_0, false);  MAP\_GPIODirModeSet(GPIOA1\_BASE, 0x2, GPIO\_DIR\_MODE\_OUT);  // Configure PIN\_01 for LED2  MAP\_PinTypeGPIO(PIN\_01, PIN\_MODE\_0, false);  MAP\_GPIODirModeSet(GPIOA1\_BASE, 0x4, GPIO\_DIR\_MODE\_OUT);  // Configure PIN\_02 for LED3  MAP\_PinTypeGPIO(PIN\_02, PIN\_MODE\_0, false);  MAP\_GPIODirModeSet(GPIOA1\_BASE, 0x8, GPIO\_DIR\_MODE\_OUT);  } |