Quadro – ONSEMI email&PIR&motion control

Attila Mák
04.12.2017

Revision History

Revision, Date	Editor	Subject(major changes)
Revision 1.0, 04.12.2017	Attila Mák	Initial release
Revision 2.0, 11.10.2018	Attila Mák	Description changed

Contents

1 Scope/ description

This document is guiding to set up Arrow Quadro board with ONSEMI IoT kit. We will create a project which is an alarm system demonstration. If the PIR sensor detects any movement and the shutter is “closed” the alarm message will appear on the screen. And we will make a project with Quadro to send the received data to the cloud via wifi.

The connection between Quadro and ONSEMI IDK is based on UART.

2 Hardware setup

The required hardware to perform the steps described in this application note consists of:

Developer PC:

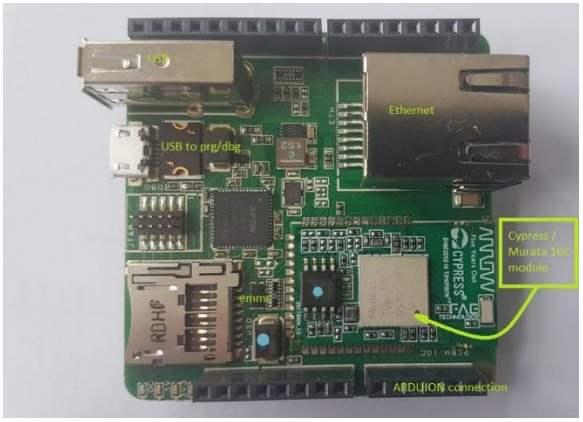
This platform will be used for setup and writing and downloading the two firmware into the microcontrollers. The power supply task will be solved via USB.
Desktop computer or laptop with x86 architecture and USB 2.0

The ARROW’s Quadro board as it is

The ONSEMI IoT kit with LED driver board plus 12V / 2A Supply.

2.1 Arrow Quadro board

https://www.arrow.com/en/campaigns/cypress-wiced-iot

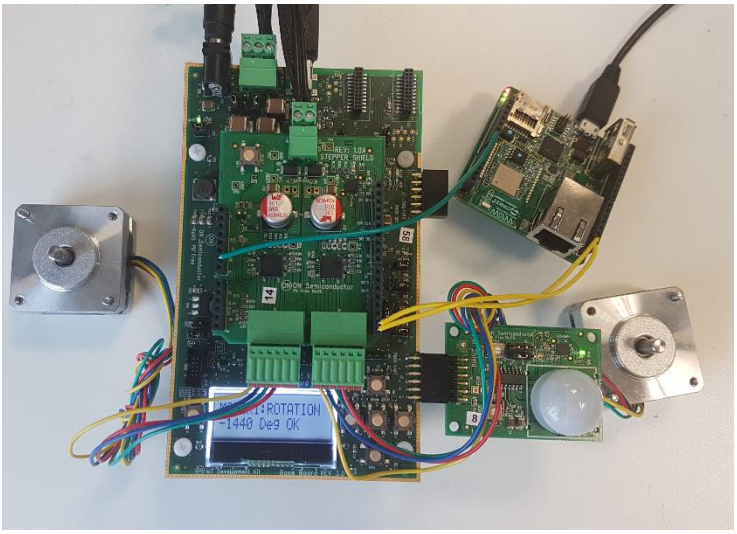


Arduino connection:

X	SCL/D15
X	SDA/D14
X	SCK/D13
NOT CONNECTED	MISO/D12
IOREF	PWM/MOSI/D11
NRST	PWM/CS/D10
NRST	PWM/D9
+3.3V	D8
+5V	NOT CONNECTED
GND	NOT CONNECTED
GND	D7
+VIN	PWM/D6
AN0	PWM/D5
AN1	D4
AN2	PWM/D3
AN3	D2
AN4	UART_TXD_RXuc
AN5	UART_RX_TXuc

2.2 OMSEMI IDK board

http://www.onsemi.com/PowerSolutions/evalBoard.do?id=BB-GEVK



Arduino connection:

	I2C_SCK1
	I2C_SDA
	AREF
	GND
	SPI_CLK1
3V3_KNX	SPI_DATAI1
IOREF	SPI_DATAO2
/RESET	SPI_CS
IOREF	SWO/DIO11
+5V	SWDIO/D13
VCOM	SWDCLK/D12
VCOM	SPI_DATAI2/DIO16
I2C_EXP_GPIO12	INT2
I2C_EXP_GPIO13	SPI_CLK2_DIO14
A3	PWM1
A2	INT0
A1	UART1_TX
A0	UART1_RX

3 Software setup

3.1 IDE installation to the PC

3.1.1 WICED installation

Pay ATTENTION! Some antivirus like BitDefender may cause problems during the installation phase and during normal use of the Wiced Studio

1) Download and install 32bit and 64bit JRE (Java Runtime Environment)

   http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

2) Download and install Cypress WICED Studio 4.1.1

   https://community.cypress.com/community/wiced-wifi/wiced-wifi-documentation

3) Plug the Board with USB Cable

   Verify Driver installation

4) Download and install Quadro BSP file

   BCM943907_QUADRO_w6.zip for WICED 6.0 IDE

5) Download the Quadro IoT Starter Kit Getting Started Guide

   Bluemix IoT LAB

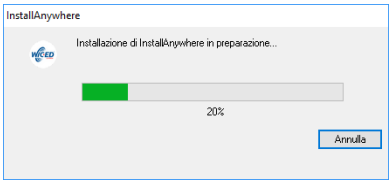
Setup JRE 32 and JRE64 java runtime

1) Check/Fix your JRE (Java Runtime Environment) installation:
   a. 32-bit JRE is needed for Cypress WICED Studio
   b. 64-bit JRE is needed for SDK's installer
      (JRE is designed to allow both 32 and 64 bit variants to be installed on same system)
2) Not normally required, but if you have a JRE related issue, check your Windows PATH. This should include a path to your Java installation:

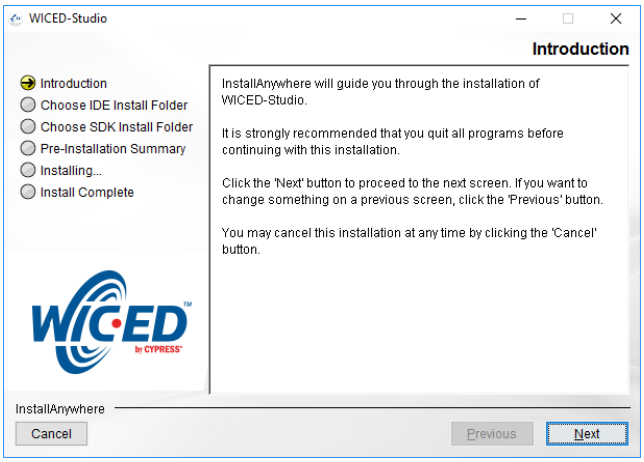
   C:\ProgramData\Oracle\Java\javapath

Setup Wiced Studio (WINDOWS)

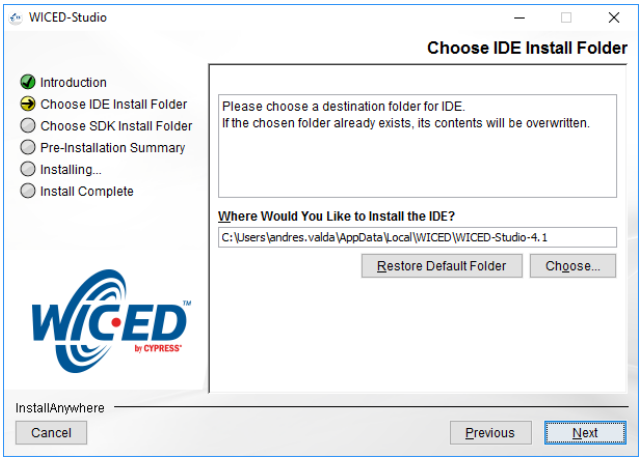
1) install Cypress WICED Studio 4.1.1 development tools by unzip file WICED-Studio-4.1.1.8-IDE-Installer.exe.zip and execute WICED-Studio-4.1.1.8-IDE-Installer.exe
2) When Setup start Appear window below



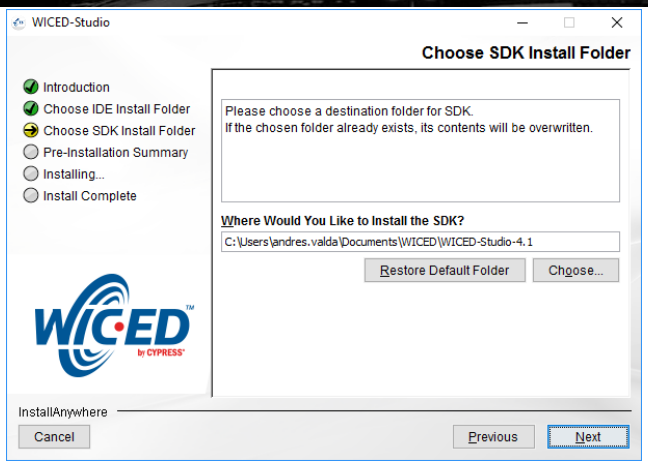
And then



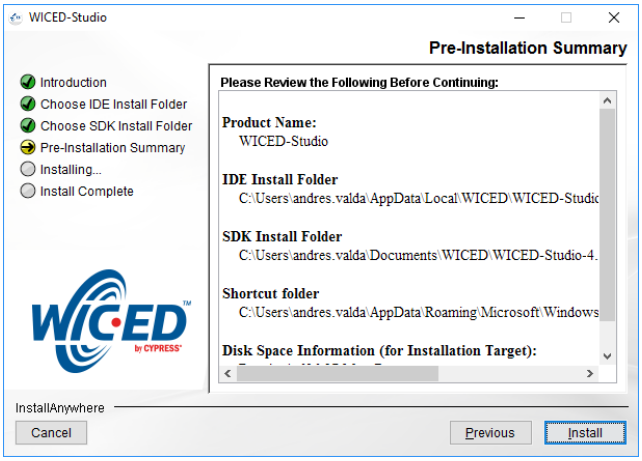
Press Next Button



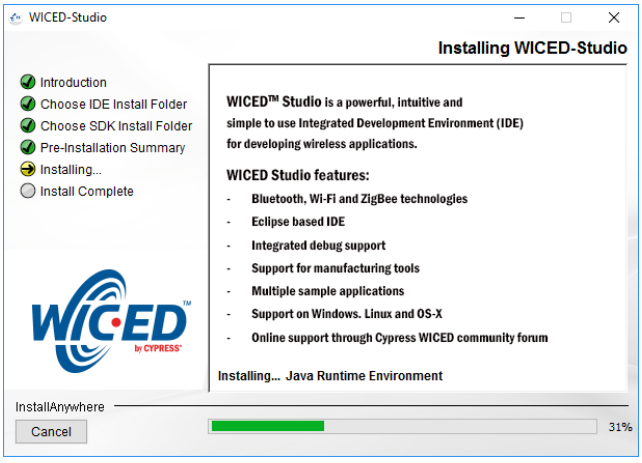
Press Next Button



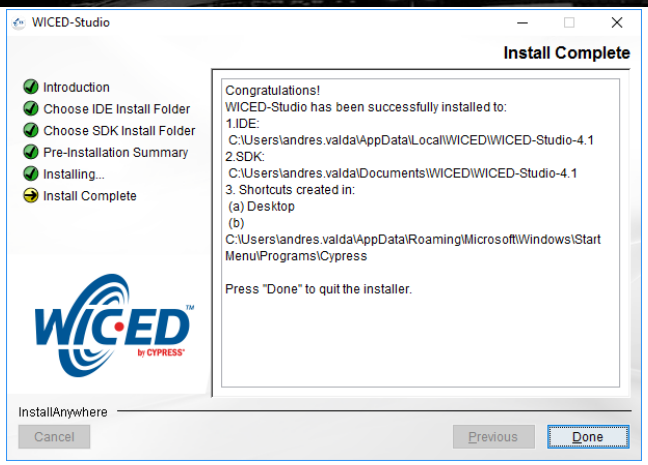
Press Next Button



Press Install

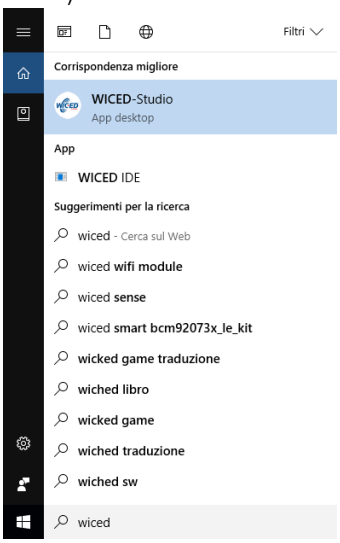


Wait while setup finish

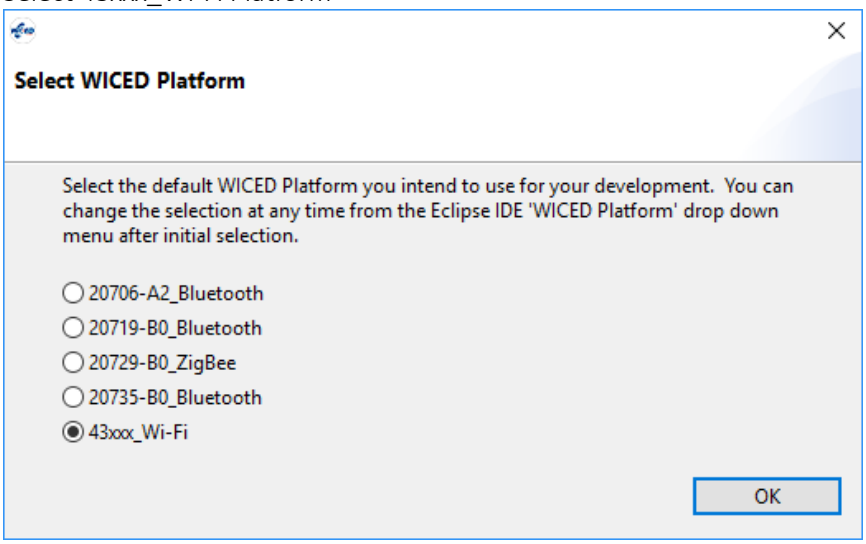


And press Done Button

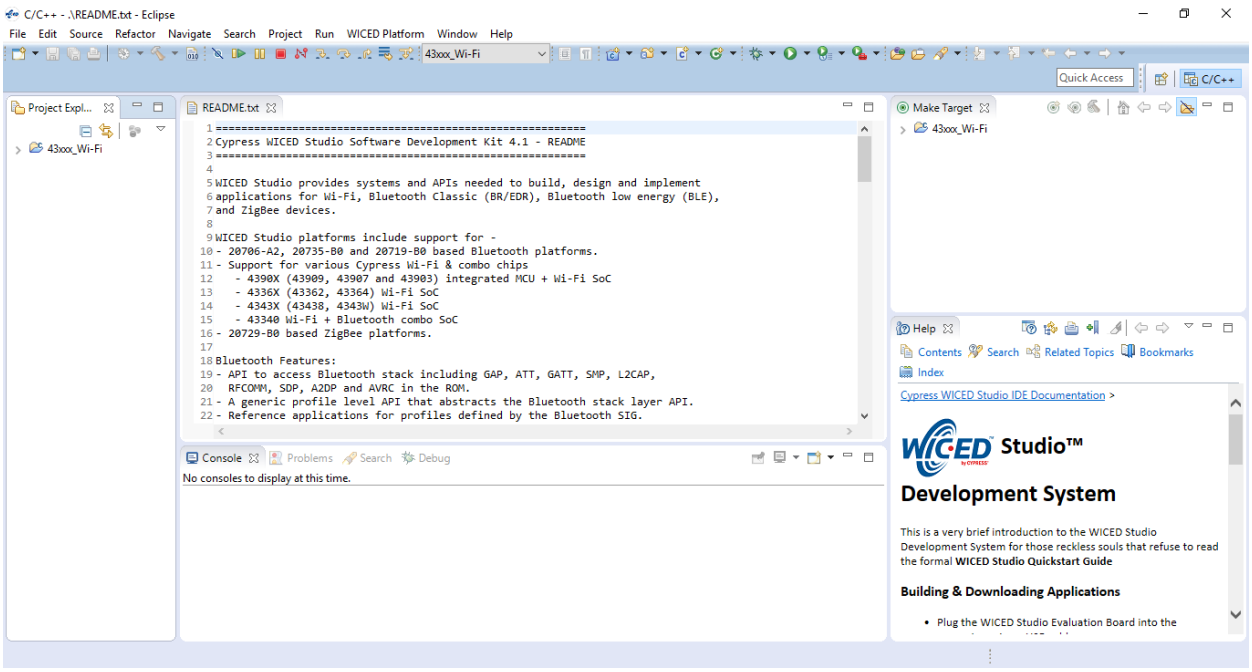
3) Launch the WICED SDK



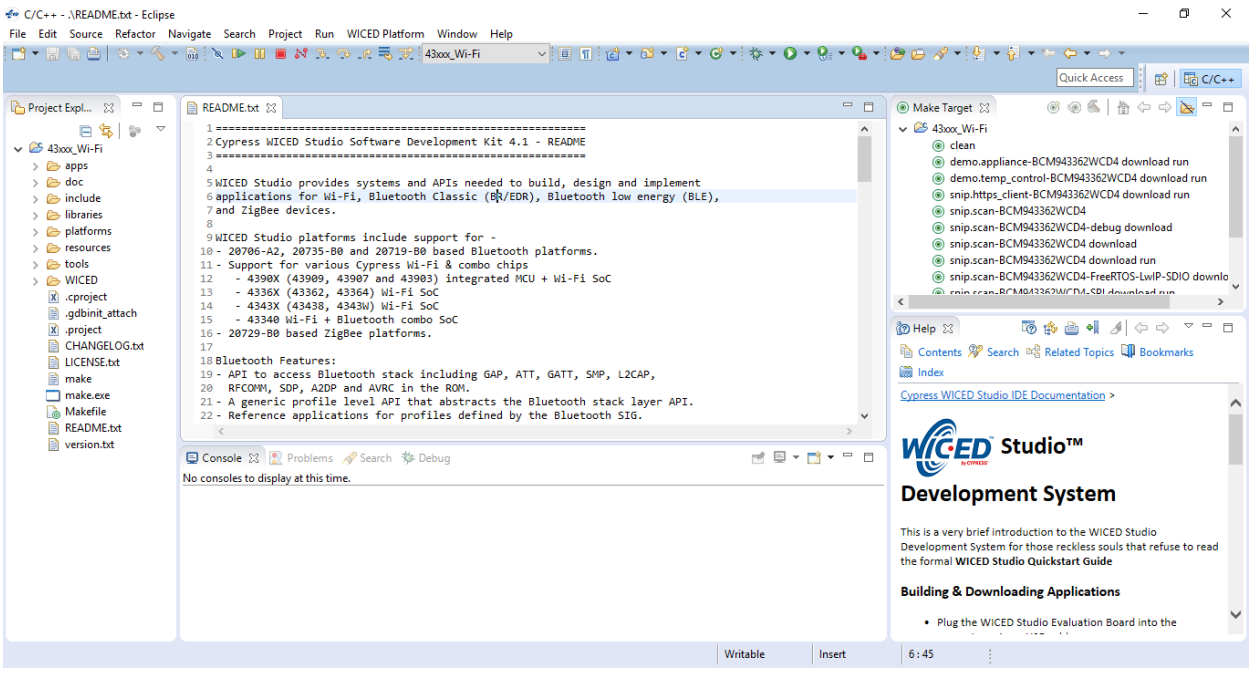
Select 43xxx_Wi-Fi Platform



The WICED IDE must appear as below



Expand 43xxx_Wi-Fi Project Explorer and 43xxx_Wi-Fi Make Target



SETUP WICED STUDIO (OTHER OS)

OSX

https://community.cypress.com/docs/DOC-3988

Linux 64-bit

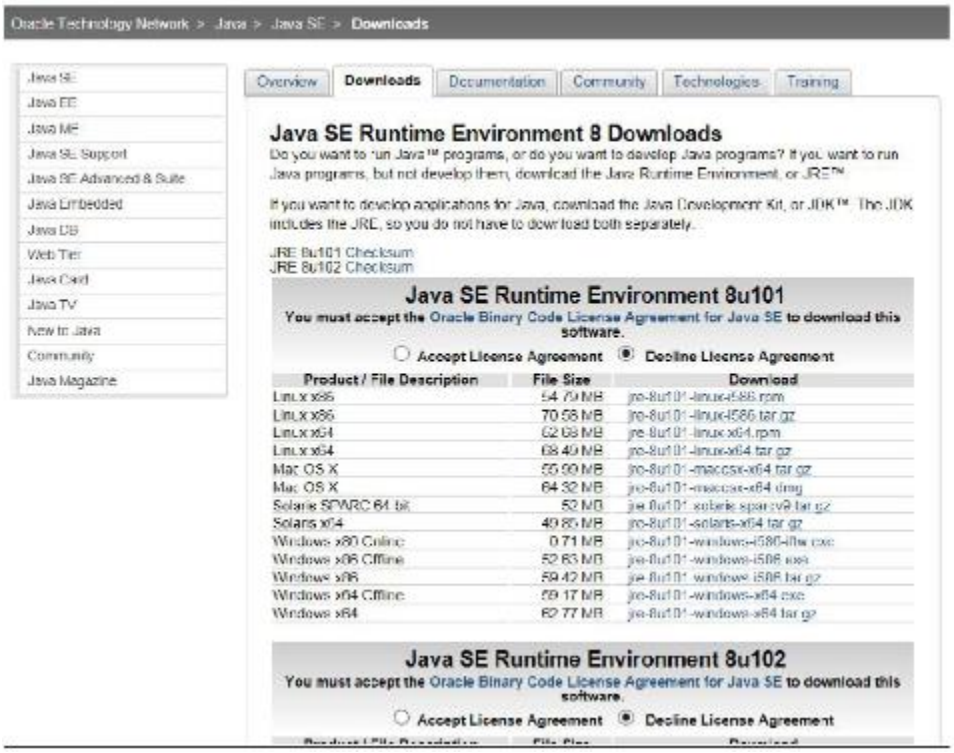
https://community.cypress.com/docs/DOC-3989

Linux 32 bit

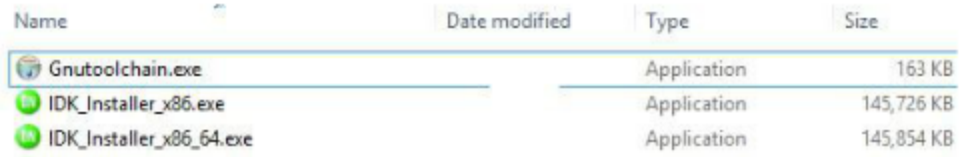
https://community.cypress.com/docs/DOC-3990

3.1.2 ONSEMI IDK installation

Java Installation
JRE/JDK
version 8u101 or above needs to be installed on the PC:
http://www.oracle.com/technetwork/java/javase/downloads/jdk8−downloads−2133151.html

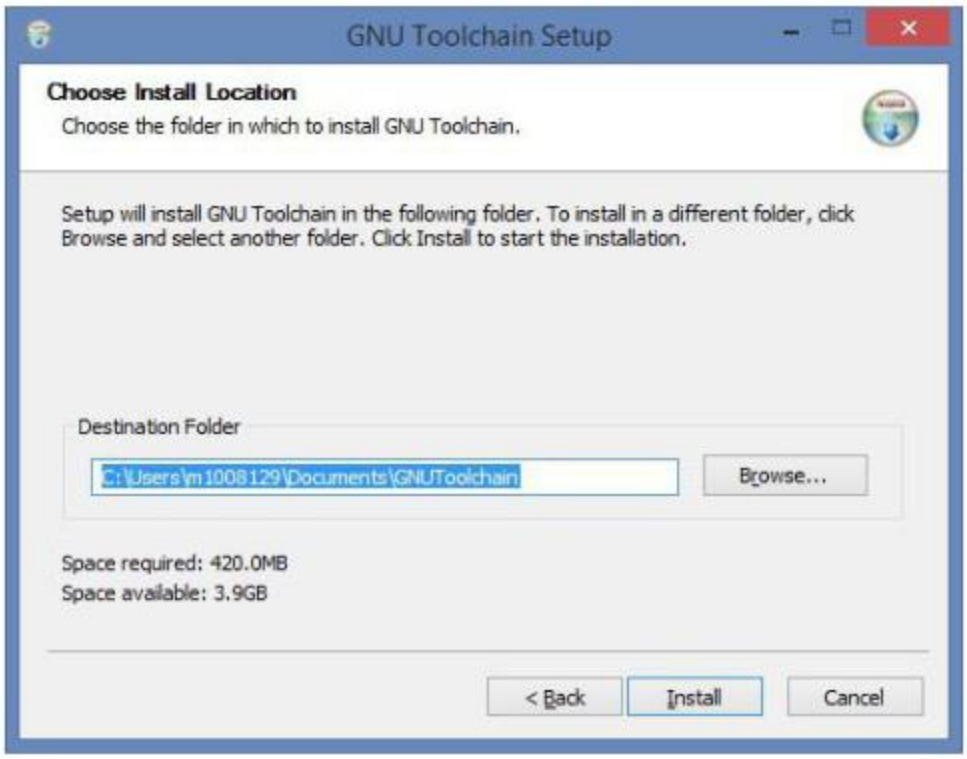


GNUToolchain
The GNU cross compiler needs to be installed to compile the IDK application. Double click on the GNUToolchain.exe to install the cross compiler. Internet connection is mandatory to install the cross compiler.

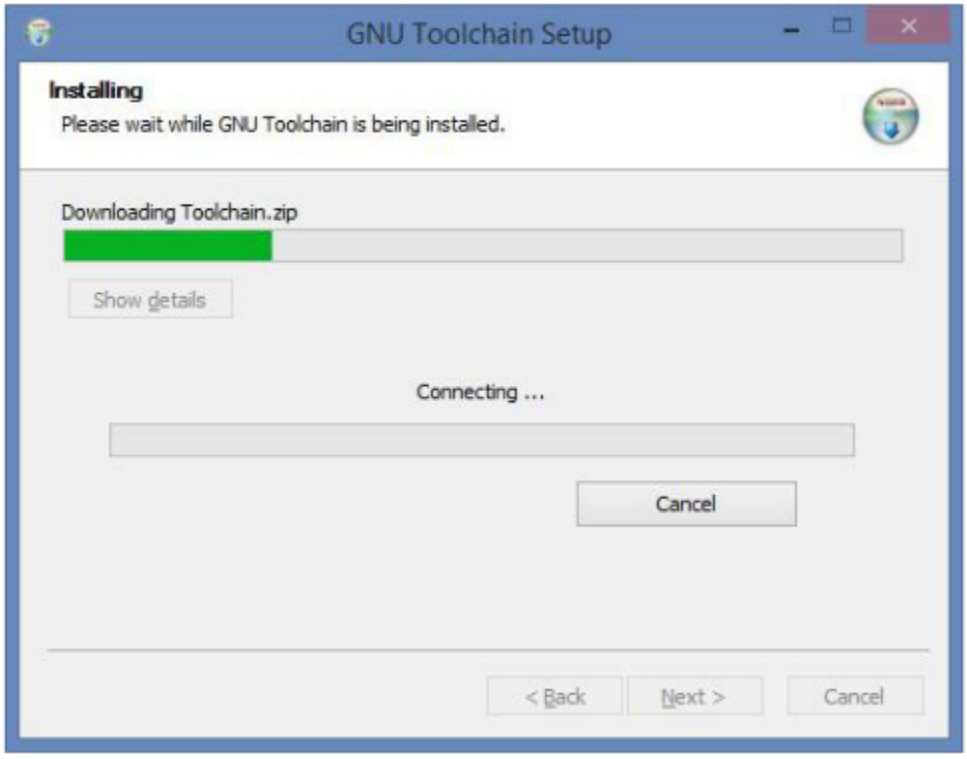


Select the “GNU Toolchain” checkbox and click Next.

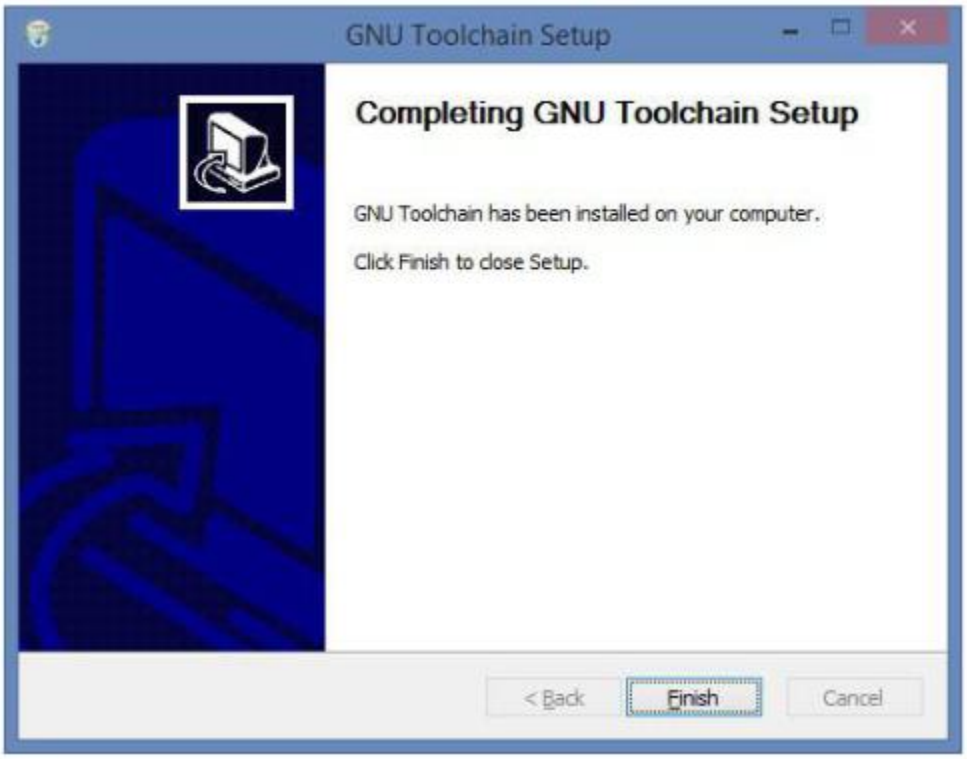
Select Destination folder and click Next. It is recommended to not change installation path.



Installer automatically downloads toolchain and installs.



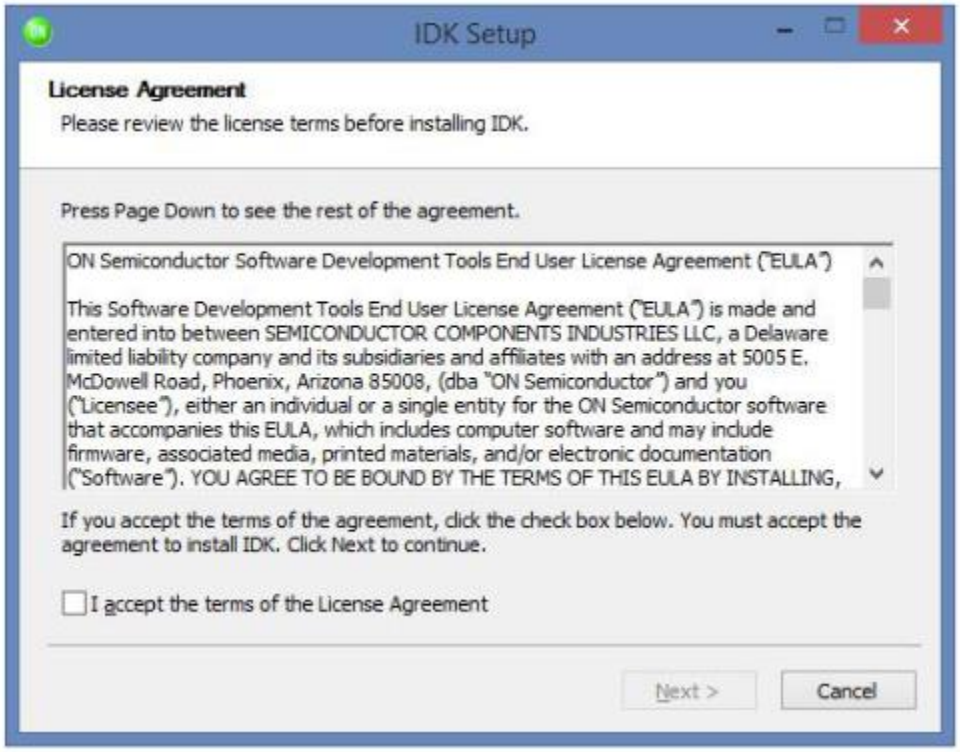
GNU Tool chain installation complete.



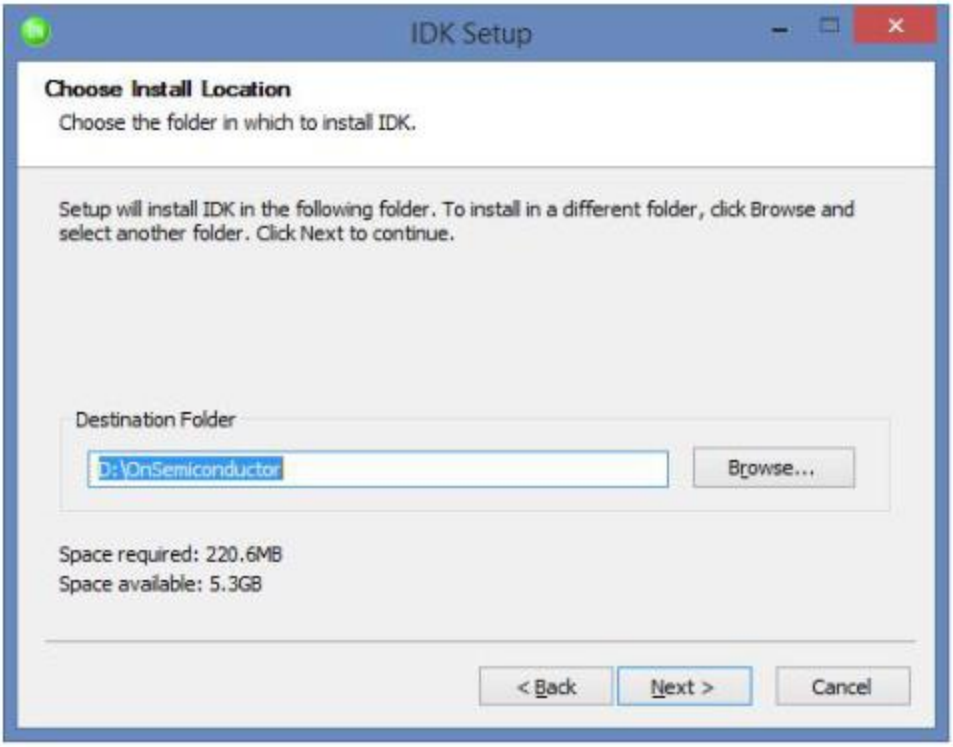
IDK Installation

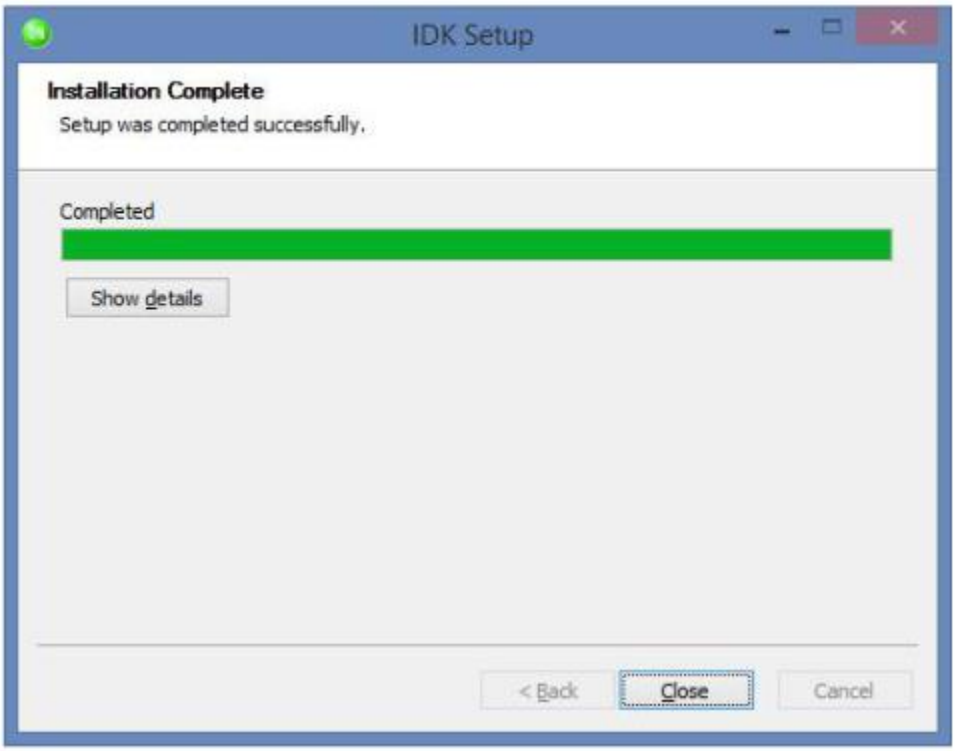
Double click on the installer downloaded from ON Semiconductor. For 32 bit machines, install IDK Installer x86.exe. For 64 bit machines, install IDK Installer x86 64.exe

Read the license, check the box and click Next.



Choose the destination directory to install the IDK. It is recommended to have IDK installed under
C:\OnSemiconductor or D:\OnSemiconductor.
If a previous workspace is being retained, then make sure that metadata folder inside Workspace directory is deleted.


IDK Setup
Choose Install Location
Choose the folder in which to install IDK.
Setup will install IDK in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
D:\OnSemiconductor
Browse...
Space required: 220.6MB
Space available: 5.3GB
< Back
Next >
Cancel

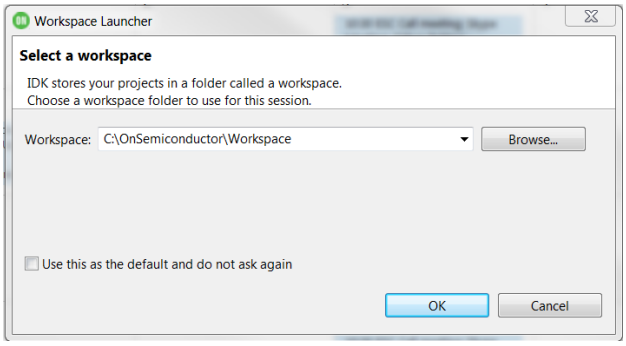

IDK Setup
Installation Complete
Setup was completed successfully.
Completed
Show details
< Back
Close
Cancel

Once in is successfully installed, a shortcut will be created on the desktop. Double click on the IDK shortcut on the desktop to launch the IDK IDE. The ON Semiconductor splash screen will launch, followed by the Welcome Screen.

4 Using ONSEMI IDK

First Start the 

Then select the desired workspace name



Here the example Workspace name is the default.

```
//Initialize Serial instance
Serial screen(USBTX,USBRX);
Serial Quadro(p0,p1);

// create a PIR class object.
NCS36000 pir;
// create a LCD class object.
NHD_C0216CZ lcd;
//butons
IOXP_BTN bb_keys;
//motor driver panel
AMIS30543D stepper2(MOTOR2);



#define INT_SET     1
#define INT_CLEAR   0
//variables
unsigned int button_status = BUTTON_INVALID;
bool buttonPress = BUTTON_FALSE;
bool alarmState = false;
bool PIR_sensed = false;
bool ToggleSwitch = false, Closed = false;
```

Includes, variables for the project

```
void exit_demo (void)
{
  ;//stop here
}
```

Exit demo routine

```
void button_routine(uint8_t button)
{
    int ret=0;

   if( button == BUTTON_LEFT )
   {
       lcd.displayString("LEFT");
       Quadro.printf("LEFT");


   }
   else if (button == BUTTON_DOWN)
   {

       lcd.displayString("DOWN");
       Quadro.printf("DOWN");
       if(!Closed)
       {
           ret = stepper2.rotateAngle(-1440);
           if(ret != STPR_RET_SUCCESS){
               screen.printf("motor driver 2 : rotating motor -1440 degrees failed, error : %d !!!\r\n", ret);
               lcd.displayString("MOTOR1:ROTATION\n-1440 Deg failed");
               wait(1);
               lcd.displayString("ERROR !!!\nEXITING DEMO");
               exit_demo();
           }else{
               screen.printf("motor driver 2 : rotating motor -1440 degrees success ...\r\n");
               lcd.displayString("MOTOR1:ROTATION\n-1440 Deg OK");
           }
           Closed = true;
           wait(1);

       }

   }
```

```
    else if (button == BUTTON_OK)
    {

        if(ToggleSwitch==true)
        {
            ToggleSwitch=false;

            lcd.displayString("Disalarmed");
            Quadro.printf("Disalarmed");

        }
        else
        {
            ToggleSwitch=true;

            lcd.displayString("Alarmed");
            Quadro.printf("Alarmed");

        }
    }
    else if (button == BUTTON_UP)
    {


        lcd.displayString("UP");
        Quadro.printf("UP");
        if(Closed)
        {
            ret = stepper2.rotateAngle(1440);
                    if(ret != STPR_RET_SUCCESS){
                        screen.printf("motor driver 2 : rotating motor 1440 degrees failed, error : %d !!!\r\n", ret);
                        lcd.displayString("MOTOR1:ROTATION\n1440 Deg failed");
                        wait(1);
                        lcd.displayString("ERROR !!!\nEXITING DEMO");
                        exit_demo();
                    }else{
                        screen.printf("motor driver 2 : rotating motor -1440 degrees success ...\r\n");
                        lcd.displayString("MOTOR1:ROTATION\n1440 Deg OK");
                    }
                    Closed=false;
                    wait(1);
        }
```

```
    }
    else if (button == BUTTON_RIGHT)
    {


        lcd.displayString("RIGHT");
        Quadro.printf("RIGHT");


    }
    else
    {
        lcd.displayString("Invalid");
    }
}


void pir_routine()
{
    if((ToggleSwitch==true)&&(Closed))
    {
        lcd.displayString("Movement Detected");
        PIR_sensed=true;
        screen.printf("Movement Detected\r\n");
        Quadro.printf("Move\n");
    }

}
```

How to handling the buttons on the board

```
void pir_routine()
{
    if((ToggleSwitch==true)&&(Closed))
    {
        lcd.displayString("Movement Detected");
        PIR_sensed=true;
        screen.printf("Movement Detected\r\n");
        Quadro.printf("Move\n");
    }

}
```

PIR sensor routine

```
int main()
{
    int ret=0;

    lcd.init();
    lcd.displayString("ALARM demo with QUADRO ");
    wait(2);
    pir.registerCallback(pir_routine);
    lcd.displayString("PIR initialized");
    wait(2);

    ret = stepper2.enable();
    if(ret != STPR_RET_SUCCESS){
        screen.printf("motor driver 2 : enabling the stepper motor 2 failed, error : %d !!!\r\n", ret);
        lcd.displayString("MOTOR2:\nEnable Failed");
        wait(1);
        lcd.displayString("ERROR !!!\nEXITING DEMO");
        return (-1);
    }else{
        screen.printf("motor driver 2 : enabling the stepper motor 2 success ...\r\n");
        lcd.displayString("MOTOR2:\nEnable success");
    }
    wait(1);

    //This API should be called in the while loop, if any, in the main application during each iteration of loop.
            //Should be called at least once in an application without loop.It should not be called inside any ISR.
            //Clears any interrupt caused by ERR out pin in stepper shield gpio expander.
             ret = stepper2.checkStprErrorOut();
            if(ret == true){
                screen.printf("ERR OUT set low, check status registers for exact reason of error !!!\r\n");
                lcd.displayString("MOTOR2:\nERR OUT LOW");
            }else{
                screen.printf("ERR OUT set HIGH, No errors ...\r\n");
                lcd.displayString("MOTOR2:\nNO ERR OUT");
            }

            //enable motor2 peak coil current as 715mA.
            ret = stepper2.setMcPeakCurr(9);
            if(ret != STPR_RET_SUCCESS){
                screen.printf("motor driver 2 : setting the peak current of motor 2 coil to 715mA failed, error : %d !!!\r\n", ret);
                lcd.displayString("MOTOR2: Coil\nPeak cur not set");
                wait(1);
                lcd.displayString("ERROR !!!\nEXITING DEMO");
                return (-1);
            }else{
```

```
                screen.printf("motor driver 2 : setting the peak current of motor 2 coil to 715mA success ...\r\n");
                lcd.displayString("MOTOR2: Coil\nPeak cur set");
            }
            wait(1);

            //set the desired stepper mode.
            ret = stepper2.setStepperMode(MICRO_1_4);
            if(ret != STPR_RET_SUCCESS){
                screen.printf("motor driver 2 : setting stepper mode to micro step 1/4 failed, error : %d !!!\r\n", ret);
                lcd.displayString("MOTOR2:MICRO1/4\nMODE not set");
                wait(1);
                lcd.displayString("ERROR !!!\nEXITING DEMO");
                return (-1);
            }else{
                screen.printf("motor driver 2 : setting stepper mode to micro step 1/4 success ...\r\n");
                lcd.displayString("MOTOR2:MICRO1/4\nMODE set");
            }
            wait(1);


    /*Initialize the button library*/
    bb_keys.init();

    /*Register button/key events*/
    bb_keys.registerCallback(button_routine);

    while(1)
    {
        lcd.clearDisplay();
        wait(3);
    }
}
```

This is the main function.

5 Using Quadro board

Here we use the email sample /snip program. Just we add there some functions.

```
101
102 wiced_result_t        ret = WICED_SUCCESS;
103 wiced_result_t        result;
104
105 wiced_ring_buffer_t rx_buffer;
106 uint8_t              rx_data[RX_BUFF];
107
108 wiced_ring_buffer_t Q_rx_buffer;
109 uint8_t              Q_rx_data[RX_BUFF];
110
111/******************************************************
112 *                      Structures
113 ******************************************************/
114 wiced_uart_config_t uart_handle =
115 {
116     .baud_rate    = 115200,
117     .data_width   = DATA_WIDTH_8BIT,
118     .parity       = NO_PARITY,
119     .stop_bits    = STOP_BITS_1,
120     .flow_control = FLOW_CONTROL_DISABLED,
121 };
122
```

Structs for serial communication.

```
140 void application_start( )
141 {
142     uint32_t             expected_data_size = 1;
143     int                    retries = 0,i=0;
144     int                    count = 0, cnt=0;
145     uint8_t              incomingDATA[RX_BUFF];
146     uint8_t              outDATA[RX_BUFF];
147     char                   newmsg[1024];
148     char c;
149
150
151     /* Initialise WICED device */
152     wiced_init( );
```

Inside the application_start function the variables what we need. And the wiced init.

```
153     ring_buffer_init(&rx_buffer, rx_data, RX_BUFF );
154     ring_buffer_init(&Q_rx_buffer, Q_rx_data, RX_BUFF );
155
156     /* Initialize Uart */
157     wiced_uart_init( WICED_UART_2, &uart_handle, &rx_buffer );
158     wiced_uart_init( WICED_UART_1, &uart_handle, &rx_buffer );
159
```

```
        while ( wiced_uart_receive_bytes( WICED_UART_2, &c, &expected_data_size, WICED_NEVER_TIMEOUT ) == WICED_SUCCESS )
        {
            /* Email requires timestamp.
             * Enable automatic time synchronisation and configure to synchronise once a day.
             */
            wiced_uart_transmit_bytes( WICED_UART_1, &c, 1 );

            outDATA[i]=rx_data[0];
            incomingDATA[i]=c;
            i++;



            if (c=='\n')
            {

                if((incomingDATA[0]=='M')&&(incomingDATA[1]=='o')&&(incomingDATA[2]=='v')&&(incomingDATA[3]=='e'))
                {

                    if(count==0)
                    {
                        send_email( );
                        WPRINT_APP_INFO(("email sending\r\n"));
                    }
                    count++;
                    if(count==24)
                        count=0;

                    WPRINT_APP_INFO(("Movement detected\r\n"));

214                 }
215
216                 i=0;
217             }
218
219
220         }
221
222
223         // Clean-up
224         sntp_stop_auto_time_sync();
225         /* Disable roaming to other access points */
226
227
228         sntp_stop_auto_time_sync();*/
229 }
230
231
```

This is how its looks when working: