# Infrared Image Processing for IoT Module using NIR Spectroscopy

**A PROJECT REPORT**

*Submitted in partial fulfillment of the requirements for the award of the degrees*

***of***

**BACHELOR OF TECHNOLOGY**

**in**

**ELECTRICAL ENGINEERING**

*Submitted by:*

**Addepalli Hari Narayana**

*Guided by:*

| | |
|---|---|
| **Dr. Ram Bilas Pachori** | **Mr. Pravin Kumar Angolkar** |
| **Associate Professor, EE** | **Lead, Emerging Businesses & Tech** |
| **IIT Indore** | **Analog Devices, Bengaluru** |

**INDIAN INSTITUTE OF TECHNOLOGY INDORE**

**DECEMBER 2017**

# CANDIDATE'S DECLARATION

I hereby declare that the project entitled **"Infrared Image Processing for IoT Module using NIR Spectroscopy"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Electrical Engineering' completed under the supervision of **Dr. Ram Bilas Pachori, Associate Professor, Discipline of Electrical Engineering, IIT Indore** and **Mr. Pravin Kumar Angolkar, Lead Emerging Businesses and Tech, Analog Devices, Bengaluru** is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

**Addepalli Hari Narayana**
**140002002**
**B.Tech 4th year**
**Discipline of Electrical Engineering**
**IIT Indore**

# <u>CERTIFICATE by BTP Guides</u>

It is certified that the above statement made by the student is correct to the best of our knowledge.

**Mr.Pravin Kumar Angolkar**
**Lead, Emerging Businesses & Tech**
**Analog Devices,**
**Bengaluru.**

**Dr. Ram Bilas Pachori**
**Associate Professor**
**Discipline of Electrical Engineering**
**IIT Indore**

# Preface

This report on “**Infrared Image Processing for IoT Module using NIR Spectroscopy**" is prepared under the guidance of Dr. Ram Bilas Pachori, Associate Professor, Discipline of Electrical Engineering, IIT Indore and Mr. Pravin Kumar Angolkar, Lead Emerging Businesses and Tech, Analog Devices, Bengaluru.

Through this report, I have tried to give a detailed description about the projects I have worked on, in Analog Devices, Bengaluru for 6 months as a part of my B.Tech project. I have explained about the architecture and the data flow of the products which I have developed at Analog Devices, Bengaluru. I further have explained about the working of these products with the help of the results that were generated using a working prototype.

I have tried to the best of my abilities and knowledge to explain the content of my project in a lucid manner. I have also added figures explaining the working and photographs of the setup, I have used, to make my description more illustrative.

**Addepalli Hari Narayana**
**140002002**
**B.Tech. 4th Year**
**Discipline of Electrical Engineering**
**IIT Indore**

## <u>Acknowledgements</u>

I would like to thank my BTP Supervisor **Dr. Ram Bilas Pachori** for his kind support. I am thankful for his constant support and guidance that has helped me in structuring this project. His valuable feedback has helped me a lot in making this project report. Without his support this report would not have been possible.

I am especially grateful to **Mr. Pravin Kumar Angolkar**, who was my supervisor in Analog Devices, Bengaluru, he explained me the concepts and provided the initial pathway for the project.

I would also like to thank my Head of the Department **Dr. Trapti Jain**, for her help and support throughout my B.Tech project.

I would like to acknowledge all well-wishers who helped me in the completion of my project and submitting the report.

**Addepalli Hari Narayana**
**140002002**
**B.Tech. 4th Year**
**Discipline of Electrical Engineering**
**IIT Indore**

## Abstract

The invention of the mobile phone has made the human life very easy. The mobile phone has combined the functions of different machines like camera, audio player, video player, radio etc. It can be also used to access the internet, social media and many more. Now, to increase its functions beyond its capabilities and without changing the existing hardware smart phone cases are being introduced. With the smart phone cases equipped with modules that serve different purposes, the ability of the smart phone has raised to newer heights.

Food which is the basic necessity of humans constitutes different types of vegetables, fruits, milk, meat etc. Growing these food components has always been a challenge to the human race and we humans have introduced various methods of cultivation of crops with the advancement in the fields of science and technology. To determine the quality of the food varieties that we use in our day to day life, a food sensing device named SCiO is introduced which lets people to scan the food component they are using and find out the food values present in them.

During my project at Analog Devices, Bengaluru, I have worked on the development of boot loader, micro controller development kit (MDK) libraries, drivers and corresponding tiles of the modules that are connected to the smart phone cases. I have also worked on the development of SCiO module which involves, collecting data from ADSC100 image sensor that performs near infra-red (NIR) spectroscopy on an object, then applying dead pixel algorithm, image processing algorithm and AES encryption on the data and sending the processed image data from the micro controller to the phone app via Bluetooth low energy (BLE).

# Table of Contents

# List of Figures

# Abbreviations

| | | |
|---|---|---|
| **MCU** | - | Micro controller unit |
| **SPI** | - | Serial peripheral interface |
| **PPI** | - | Parallel peripheral interface |
| **I2C** | - | Inter integrated circuit |
| **UART** | - | Universal asynchronous receiver and transmitter |
| **API** | - | Application programming interface |
| **BLE** | - | Bluetooth low energy |
| **MDK** | - | Microcontroller development kit |
| **AES** | - | Advanced encryption standard |
| **CBC** | - | Cipher block chaining |
| **SHA1** | - | Secure hash algorithm |
| **HMAC** | - | Hash based message authentication code |
| **ADC** | - | Analog to digital converter |
| **CCES** | - | Cross core embedded studio |
| **SRAM** | - | Static random access memory |
| **UUID** | - | Universally unique identifier |
| **BSL** | - | Boot strap loader |
| **IoT** | - | Internet of things |
| **NIR** | - | Near infra-red |
| **RGB** | - | Red, Green, Blue |
| **LED** | - | Light emitting diode |
| **PWM** | - | Pulse width modulation |

| | | |
|---|---|---|
| **DSP** | - | Digital signal processor |
| **EZ** | - | Evaluation board |
| **RF** | - | Radio frequency |
| **R/W** | - | Read or write |
| **R** | - | Read only |
| **RAM** | - | Random access memory |
| **GAP** | - | Generic access profile |
| **GATT** | - | Generic attribute profile |
| **Wi-Fi** | - | Wireless Fidelity |

# Chapter 1

# Introduction

In this chapter, we will discuss about the basic introduction, background and motivation for the project. The problem statement is also described and towards the end of the project a possible solution has been be derived.

## 1.1 Background

The most revolutionary invention of this century is mobile phone. With the introduction of mobile phone man's life has become very easy and communication between any two places was quicker than ever. The mobile phone which was first introduced had the primary job of helping people in different places communicate with each other, but recent technologies had advanced so greatly that communication is just one of the many jobs that a smart phone can do. Now, a smart phone can be used as a camera, radio, video player, audio player, to access internet, social media and many more.

To enhance the capabilities of a smart phone beyond its capacity without any changes in its existing hardware, smart phone cases are being introduced. These smart cases are equipped with modules which help in increasing the phone's ability to newer levels. Theses smart cases are connected externally to the phone and have slots to connect the Internet of things (IoT) based modules which can be controlled by a phone app. The smart phone cases are shown in Figure 1.1.

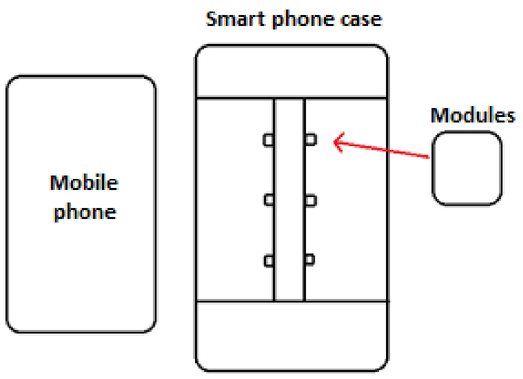


Figure 1.1: Smart Phone Cases

Food which is the basic need of humans, constitutes different types of vegetables, fruits, milk, meat etc. It is essential for us to determine the food values that are present in the food components that we are consuming. So, to determine the quality of the food varieties that we use, a food sensing module named SCiO is being introduced which lets people to scan the food component they are using and find out the food values like carbohydrates, vitamins, nutrients etc. that are present in them. This SCiO module is interfaced with smart phone case and the information about the food component that has been scanned can be obtained from the phone app.

## 1.2 Objectives

Analog Devices is the technological partner of a company called Moduware. The smart phone cases are developed by Moduware [1] and the modules that are connected to the phone case are to be developed by Analog Devices.

In Analog devices, I was a part of Analog Garage Team and I had two main objectives in my project and they were as follows

1. To develop the boot loader and micro controller development kit (MDK) libraries for the micro controller unit (MCU) of the modules. The micro controller had many digital peripherals and offered different types of utilities. This project also involved developing drivers for all the features that the micro controller was offering like Analog to digital converter (ADC), timers, and different types of communication protocols like serial peripheral interface (SPI), inter integrated circuit (I2C), universal asynchronous receiver and transmitter (UART), and the corresponding tiles that are to be uploaded into the phone app to control the functioning of each of these features.
2. I also worked on developing material sensing module (SCiO) which captures the near infrared (NIR) image of a food component, then apply the dead pixel algorithm, image processing algorithm, and Advanced encryption standard (AES) encryption on the image data and send the processed image data to the phone app via Bluetooth low energy (BLE).

# Chapter 2

# Embedded Firmware Development

In this chapter, we will discuss about the firmware development of the module which includes boot loader, MDK libraries, driver and tile of the module.

## 2.1 Tools and Hardware used

2.1.1 The MCU of the modules that are being developed by Analog Devices is ARM cortex M-3, based micro controller ADuCM3029 [3], which has a core clock frequency of 26 MHz and system clock frequency of 6.5 MHz. For the development of the firmware on ADuCM3029, its evaluation board ADICUP3029 [4] has been used that is shown in Figure 2.1.

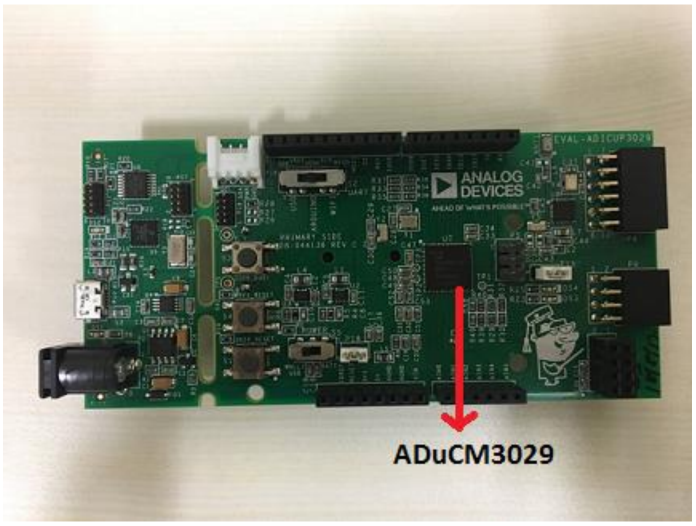


Figure 2.1: ADICUP3029

2.1.2 The MCU of the smart phone case is MSP430 and it was developed by Moduware. The mini developer board is shown in Figure 2.2, was used in the development of the setup which is a smaller version of the smart phone case

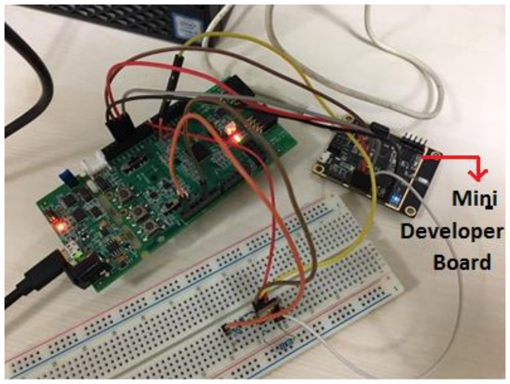


Figure 2.2: Mini Developer Board

2.1.3 The micro controller coding platform that I have used is Cross Core Embedded Studio (CCES).

## 2.2 Connection between different parts of the device

2.2.1 The smart phone case is connected to the phone through BLE.

2.2.2 The ADPAQ modules are connected to the smart phone case in the slots provided. The MCUs of the module and the smart phone case communicate through SPI protocol, where the MCU of the smart phone case is the master and the MCU of the modules is the slave.

2.2.3 The sensor corresponding to each module is connected to the MCU through digital peripherals like SPI, I2C, UART etc.

The Moduware architecture is shown below in Figure 2.3

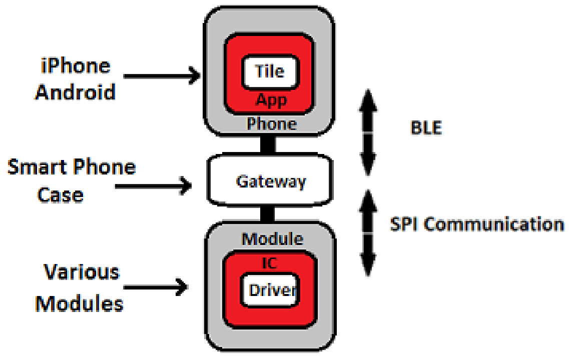


Figure 2.3: Moduware Architecture

## 2.3 Data flow

2.3.1 The phone app sends the command to collect data to the smart phone case.

2.3.2 The smart phone case sends the command to the ADPAQ module.

2.3.3 The MCU of the ADPAQ module collects the data from the sensor, processes it and sends it to the smart phone case which sends it to the phone and the data is displayed in the Moduware phone app.

The data flow of Moduware is shown in Figure 2.4

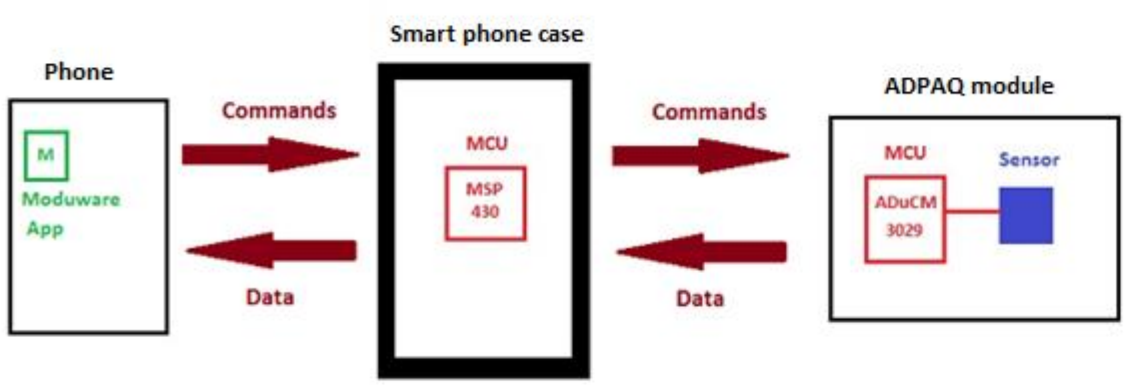


Figure 2.4: Data flow of Moduware

## 2.4 Memory allocation

The ADuCM3029 micro controller has 256 KB of flash memory, 4 KB of cache memory, 32 KB of instruction static random access memory (SRAM) and 32 KB of data SRAM. The cache memory is a part of instruction SRAM. The memory map of flash and RAM memory is shown in Figure 2.5

The flash memory was partitioned into 4 sections and the code for the boot loader, MDK libraries and the driver were written inside the flash memory which had access to 4KB of cache and 32KB of data SRAM. The four sections are

1. Interrupt vectors and security features – 2KB
2. Boot loader – 60KB

3. MDK libraries and driver of the module – 192KB
4. Information memory – 2KB

The interrupt vectors and security features part of the flash has the interrupt vector table which consists of the list of commands that are to be executed when an interrupt is generated. It also includes the security features that are required for the booting process of the MCU (ADuCM3029).

The boot loader section of the flash consists of the boot loader code of ADPAQ module.

The MDK libraries and driver of the module section of the flash comprises of both the MDK libraries code and the driver code of the particular module. This section has been allotted the maximum memory because it consists the driver code of a specific peripheral.

The information memory of the flash stores the universally unique identifier (UUID) of the device. The UUID of the device is the unique identification id which is used inside the boot loader code and is sent to the smart phone case to identify the module's MCU and connect to it.

Similarly, 32 KB of data SRAM is also divided into three sections,

1. Common SRAM – 32B
2. Boot loader RAM – 16KB
3. MDK libraries and driver of the module – 16KB

The common SRAM is accessible by both the boot loader and MDK libraries and driver of the module codes which contains data like NODE ADDRESS and RESET ADDRESS that are required for establishing the connection between module and smart phone case.

In the boot loader RAM, all the variables and data corresponding to the boot loader code is stored.

In the MDK libraries and driver of the module RAM, all the variables and data corresponding to the MDK libraries and driver of the module code is stored.

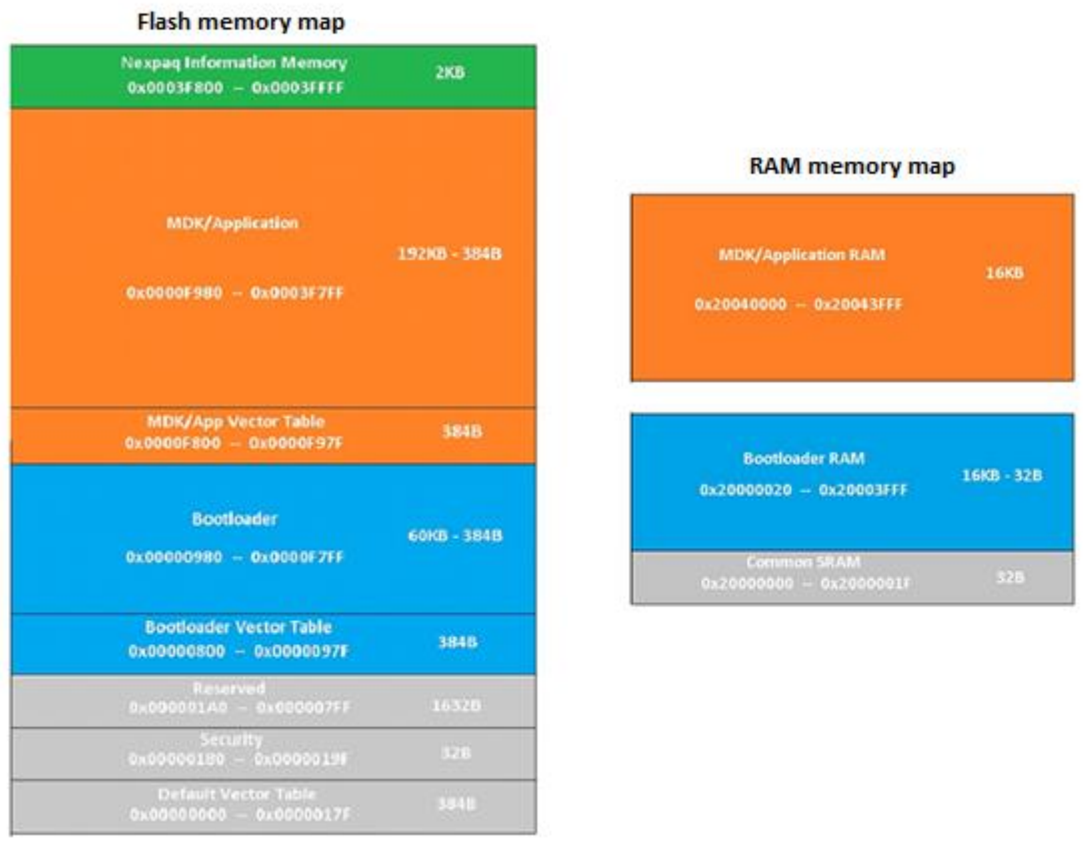


Figure 2.5: Flash and RAM Memory map

## 2.5 Working of the setup

Bluetooth in the smart phone is enabled and it is connected to the smart phone case through BLE. When the ADPAQ module is connected to the smart phone case, the module is reset.

The module’s firmware comprises of boot loader, MDK libraries and the driver of a particular peripheral of the ADuCM3029.

As soon as the module is reset, the boot loader code is executed and it establishes a connection between the ADuCM3029 which is the module’s MCU and the smart phone case.

After the boot loader code is executed, the program counter jumps to the reset handler of the MDK libraries code and executes the MDK libraries code. The driver corresponding to the module is executed along with the MDK libraries. The MDK libraries establish connection between the driver and the tile that is present inside the moduware phone app. Data is transferred between the driver and tile using MDK Library application programming interface (API).

The binary file (.bin) of the boot loader and the library file (.lib) of the MDK libraries are attached to the driver files which makes up the firmware of the particular peripheral.

Setup during development is shown in Figure 2.6.

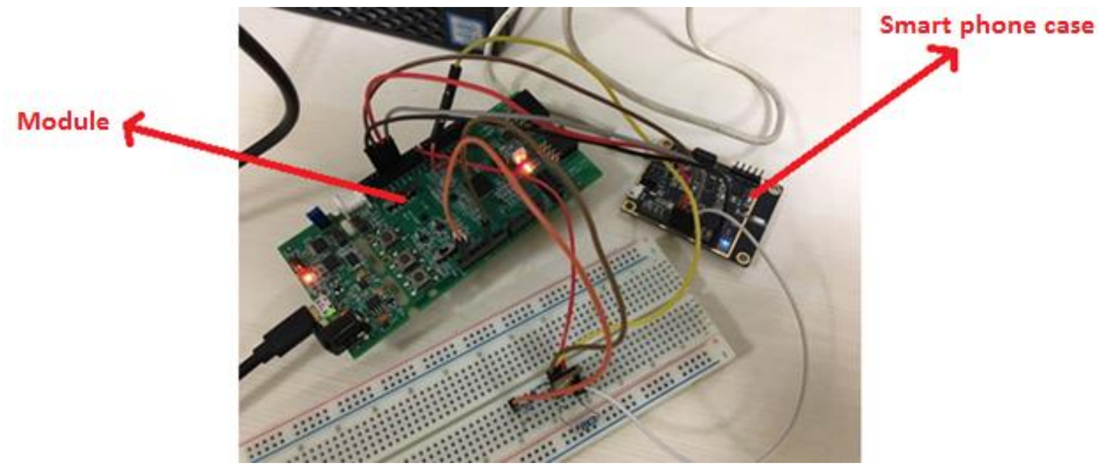


Figure 2.6: Working Setup

Setup after the product is released, is shown in Figure 2.7.

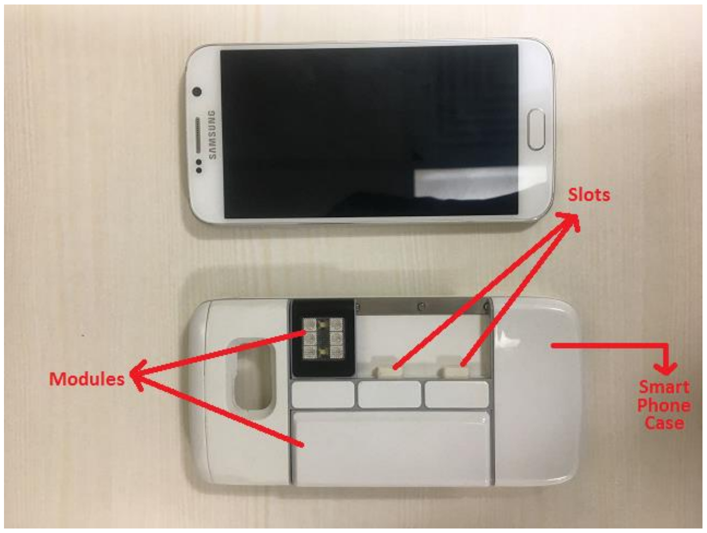


Figure 2.7: Smart Phone Case with Modules

Once the connection is established between smart phone case, phone and the boot loader, MDK/driver codes of the module are executed, the modules are displayed in the dash board of the moduware phone app as shown in Figure 2.8.

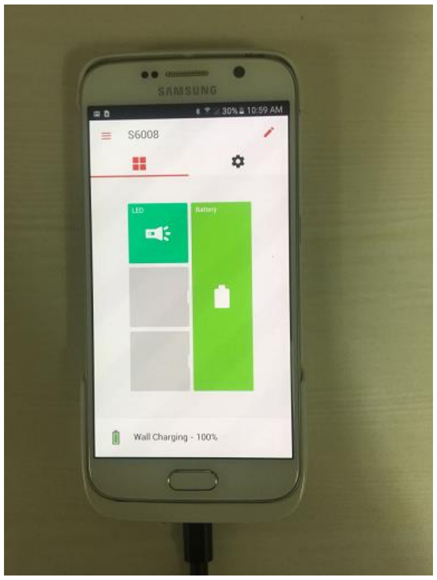


Figure 2.8: Tiles on Moduware phone app

## 2.6 ADPAQ module boot loader

Upon reset, the compiler starting compiling the code from the boot loader. We have used a mini developer board manufactured by moduware which is a smaller version of the smart phone case as the master in the SPI communication with ADuCM3029 (MCU of modules) which acts as the slave.

Inside the boot loader main function, the following are initialized

1. Core and system clocks.
2. Boot strap loader (BSL) version
3. Flash memory, SPI (slave mode) initialization.
4. UUID is written inside information memory section of flash
5. Inside the common SRAM few addresses are stored.

After initialization, there is a while 1 loop inside which “np_app_spi_event” function is called continuously. This function executes the state machine in coordination with the mini developer board. The mini developer board and the ADuCM3029 communicate via SPI protocol and exchange messages. The state machine is executed in the following way.

The mini developer board which is the master in the SPI communication sends a message, the ADuCM3029 which is the slave decodes the message and executes the corresponding command by switching to the case pointed by the command. After executing the command function, the ADuCM3029 sends back a response to the master.

Each message sent from the mini developer board consists of command, data and data length.

The following messages are sent by the mini developer board to which ADuCM3029 responds.

1. ASK BSL INFO – Sending the Boot strap loader (BSL) version
2. REQ INTO APP – Jump into the MDK libraries code by pointing the program counter to the reset handler of the MDK libraries code.
3. REQ ERASE FLASH – Erase the complete flash memory contents
4. REQ ERASE APP AND INFO – Erase the driver code along with the information memory which stores the UUID.
5. REQ UPGRADE FLASH INFO – Upgrade the contents of the flash.
6. REQ WRITE FLASH – Write into the flash memory.
7. ASK STATUS – Ask the status of the MCU of the module.
8. ASK UUID – Ask for the UUID of the MCU, to pair.
9. REQ ALLOT NODE – Request the node number of the MCU.

After executing the commands sent by the master, the program counter of the boot loader jumps to the reset handler of the driver/MDK libraries to execute the MDK libraries code along with the driver code.

## 2.7 MDK libraries

The MDK libraries are used along with the driver to establish a connection between the driver and the tile present inside the phone app. Most of the MDK libraries work on the basis of interrupts generated by SPI. Inside the main function of the MDK libraries following are initialized:

1. MDK version.
2. Initialization of SPI interrupts.
3. Initialization of timer

After these functions are initialized, an extern function is called “np_api_setup” which is defined inside the driver and it initializes the settings required by the driver. Finally there is a while 1 loop inside which “np_api_loop” is called continuously which is also an extern function, defined inside the driver code which continuously collects the data from the driver and sends it to the tile and vice versa.

## 2.8 Drivers

Each module will be associated with a particular sensor that is connected to the micro controller through a peripheral. Drivers are developed for some of the features that are available on ADuCM3029 like timers, ADC, communication interfaces like SPI, I2C, UART etc.

1. Timers:
   One of the features of a micro controller is the general purpose timers that are based on the core clocks. They are used when a program has to be performed at regular intervals. The timers are used in developing the drivers for the following modules.

   a. In the module of RGB LED, a RGB LED sensor is connected to the MCU and the intensity of the LED is controlled from the moduware app. The intensities of red, green, blue are set from the app. The intensities corresponding to each color are sent to their corresponding pins through pulse width modulation (PWM) signals that generated with the help of timers.

The tile of RGBLED module is shown in Figure 2.9

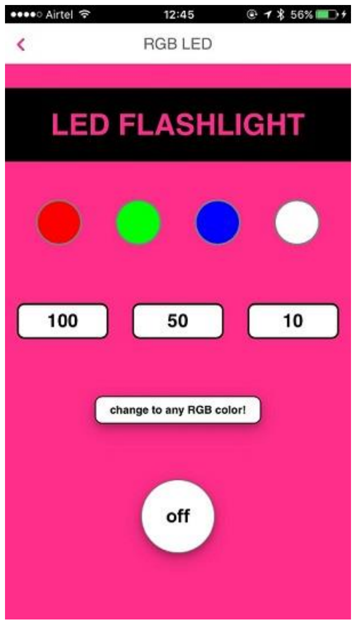


Figure 2.9: RGB LED Module Tile

b. In the module of ultrasonic sensor, the senor that is connected to the MCU measures the distance of an object and displays it in the phone app by calculating the time elapsed between the transmitted signal and the echo signal which can be achieved with the help a timer. The tile of the ultrasonic sensor is shown in Figure 2.10

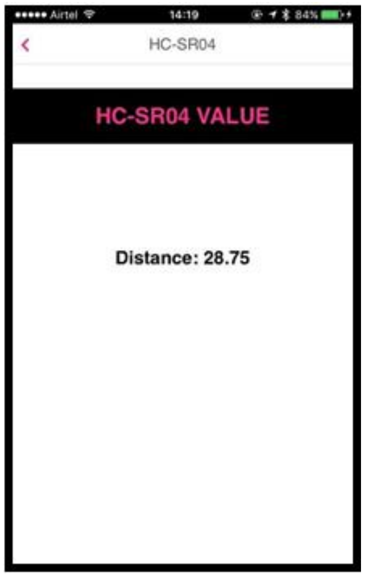


Figure 2.10: Ultrasonic Module Tile

c. In the module of the servo motor, the motor is connected to the MCU and the frequency of the motor is set from the app. The frequency determines the PWM duty ratio that has to be set for the motor which can be achieved with the help of timers. The tile of the servo motor module is shown in Figure 2.11

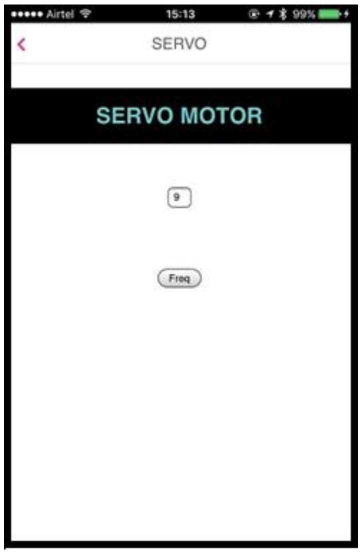
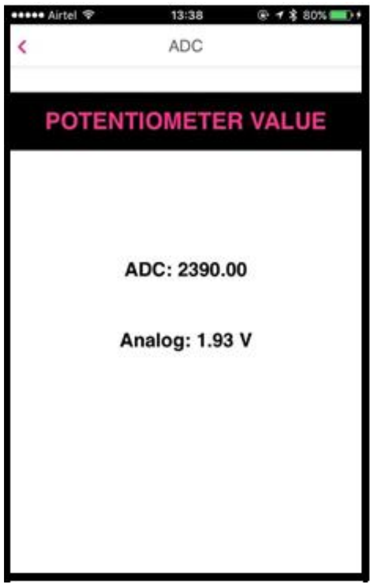


Figure 2.11: Servo Motor Module Tile

2. Analog to digital converter (ADC):

   Many sensors give an analog output and to convert the analog value into digital value with a specified bit resolution, ADC is used. The ADuCM3029 has provision to convert the analog value into 12, 14, and 16 bit resolution digital values. The tile of the potentiometer module is shown in Figure 2.12



ADC: 2390.00

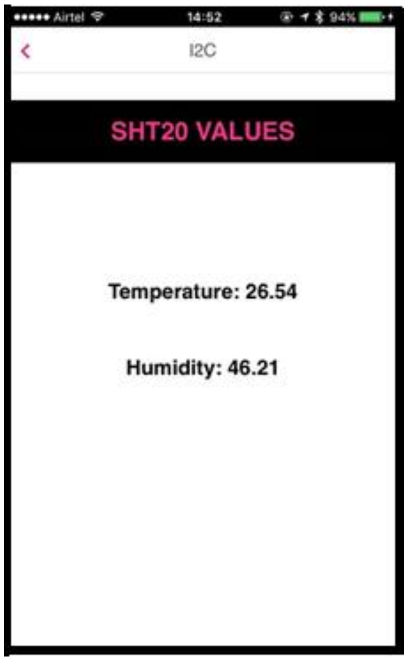
Analog: 1.93 V

Figure 2.12: Potentiometer Module Tile

ADuCM3029 MCU has digital peripherals for communicating with different sensors using various communication protocols like SPI, I2C, UART etc. Modules are developed individually for sensors which use these different communication protocols. Some of them are as follows:

1. Accelerometer module – Uses SPI communication protocol.
2. Temperature and humidity module – Uses I2C communication protocol.
3. LCD module – Uses I2C communication protocol.
4. Arduino module – Uses UART communication protocol.

The tile of these modules that display the sensor output is shown in Figure 2.13



Temperature: 26.54

Humidity: 46.21

Figure 2.13: Temperature and Humidity Module Tile

## 2.9 Tiles

For each driver, a corresponding tile has to be uploaded into the moduware App, so that the functionality of the driver can be controlled from the phone app.

The basic structure of the tile is shown in Figure 2.14 and it consists the following files:

1. Styles.css – This file included all the colors and graphics which appear on the tile.
2. Normalize.css – It is used in connecting the tile with the phone app.
3. Script.js – This file performs the calculations on the data received from Driver.json file.
4. NepaqHeader.js – This is a header file which is used in connecting the tile to the app.
5. Driver.json – It is the link between the app and the firmware, all the data from the firmware comes into this file.
6. Icon.svg – This file denotes the icon which is shown on the dash board of the app.
7. Index.html – This file describes about how the tile should look like and all the values and buttons that are visible on the tile. This file receives data from the Script.js file and displays it.
8. Manifest.json – This file describes about the details of the tile like name, version etc.

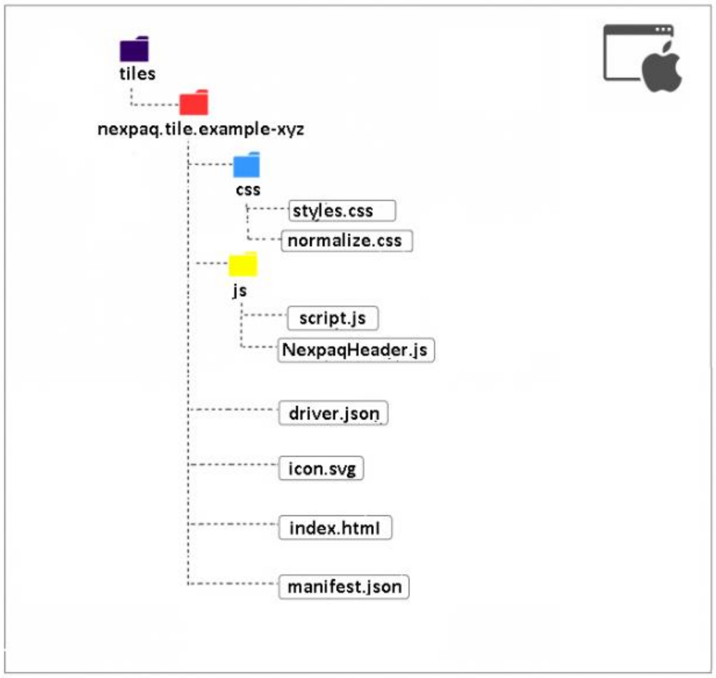


Figure 2.14: Tile structure

# Chapter 3

# Material Sensing using NIR Spectroscopy

In this chapter, we will discuss about the material sensing module (SCiO) [2]. This chapter includes the architecture, data flow, and the working of the SCiO. It describes about the Image processing algorithm and other algorithms that are applied on the NIR image data.

## 3.1 Tools and Hardware used

3.1.1. An image sensor ADSC100 [8] equipped with a LED driver and spectrometer, shown in Figure 3.1 is used to capture the infrared image of the object (eatable).

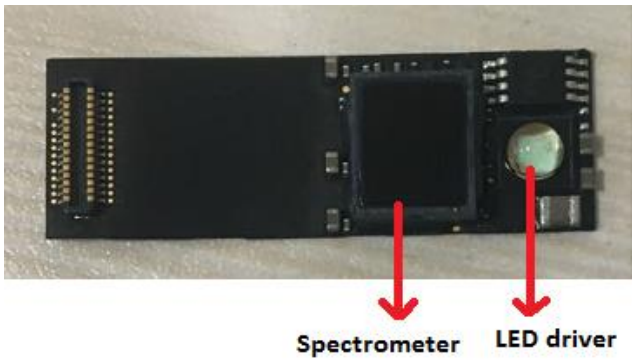


Figure 3.1 ADSC100 Image Sensor

3.1.2 The blackfin digital signal processor (DSP) micro controller collects the data from the image sensor. The DSP micro controller we have used is ADSP-BF592 [5] [6], and to use the MCU peripherals, its evaluation board (EZ) kit, ADSP-BF592 EZ kit, is used that is shown in Figure 3.2. The ADSP-BF592 micro controller has a core clock of 400 MHz and a system clock of 100 MHz. It has instruction SRAM of 32 KB and data SRAM of 32 KB.

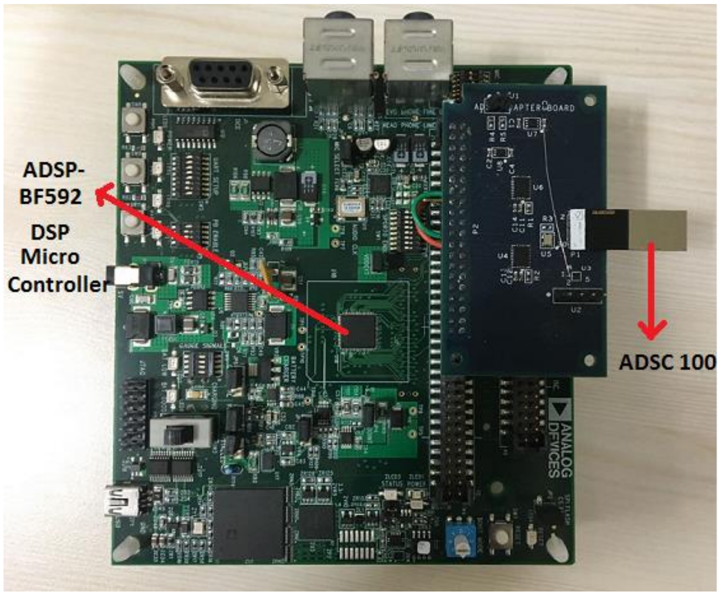


Figure 3.2 ADSP-BF592 EZ kit

3.1.3 The BLE communication between the BLE chip and the phone is achieved by considering 2 smart radio frequency (RF) boards [7]. One of them is BLE peripheral and the other is BLE central that are shown in Figure 3.3 which will be replaced with the mobile phone in later stages.

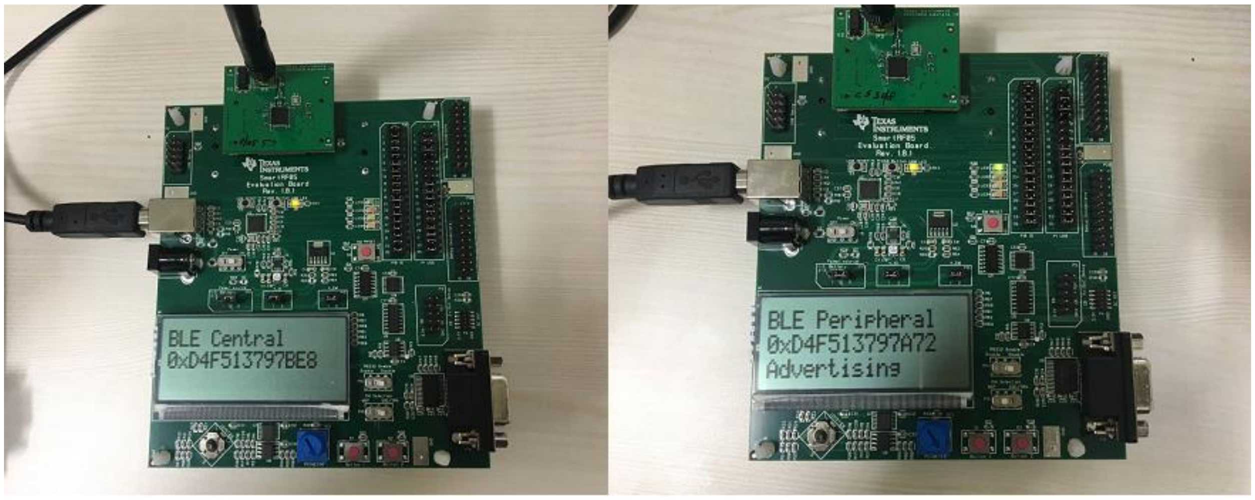


Figure 3.3 Smart RF Boards-BLE peripheral and BLE central

## 3.2 Connection between different parts of the device

3.2.1 The image sensor is connected to the blackfin DSP through parallel peripheral interface (PPI).

3.2.2 The DSP micro controller sends the image data to BLE peripheral through UART communication.

3.2.3 BLE peripheral connects to the BLE central and sends the image data to the BLE central over BLE.

3.2.4 BLE central (Phone) sends the image data to the cloud, in the cloud this image data is compared with the database that is already present in the cloud and the cloud sends the information of the object that has been scanned to the BLE central (Phone) through Wireless fidelity (Wi-Fi).

The material sensing (SCiO) module architecture is shown in Figure 3.4

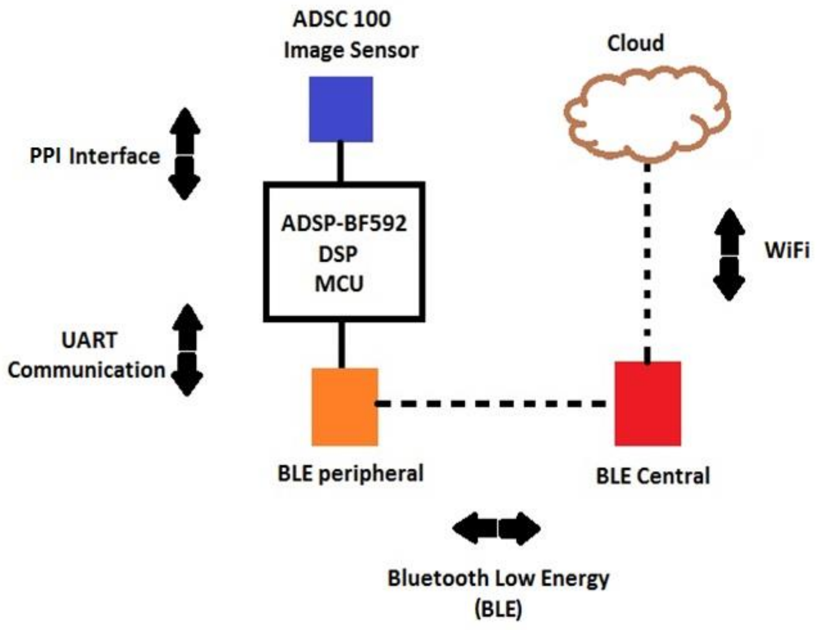


Figure 3.4: Material sensing (SCiO) module architecture

## 3.3 Data flow:

3.3.1 The BLE peripheral connects with the BLE central using its generic access profile (GAP).

3.3.2 BLE central sends the command to capture an NIR image to the BLE peripheral.

3.3.3 BLE peripheral receives the message from BLE central and sends it to the DSP micro controller.

3.3.4 DSP micro controller gives the command to the image sensor to collect the data and the image sensor performs NIR spectroscopy by sending near infrared waves on to the object we are scanning, and it captures the reflected NIR waves data and sends it to the DSP micro controller.

3.3.5 The DSP performs applies the image processing algorithm and sends the processed image data to BLE peripheral which in turn sends the data to BLE central.

3.3.6 The BLE central sends the processed image data to the cloud to compare with the database already present in it and the cloud returns the information about the scanned object to be displayed in the SCiO app

The data flow of SCiO module is shown in Figure 3.5

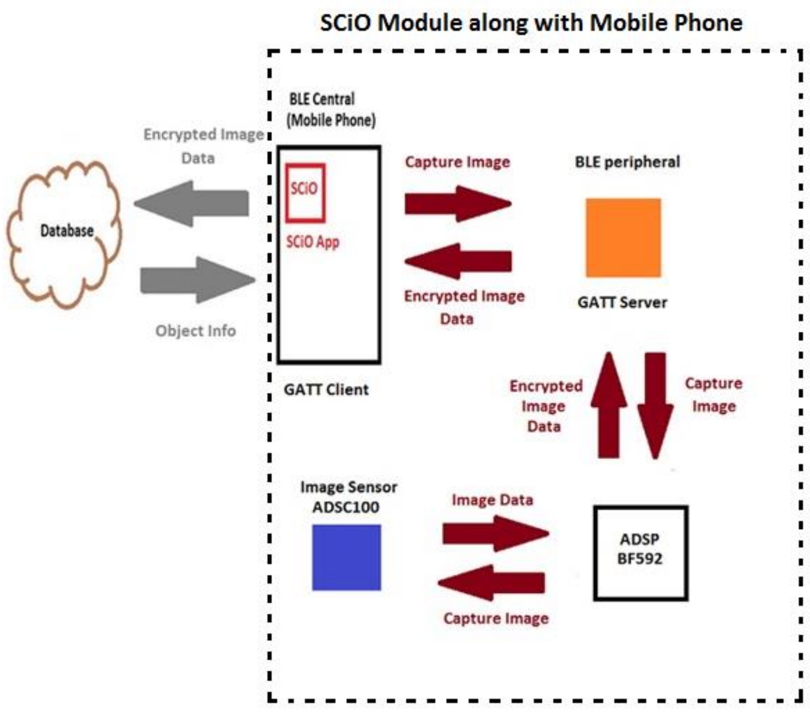


Figure 3.5: Data flow of SCiO module

## 3.4 Setup

3.4.1 During the development stage of the device, smart RF board is used as the BLE central which will later be replaced by the phone as shown in Figure 3.6.

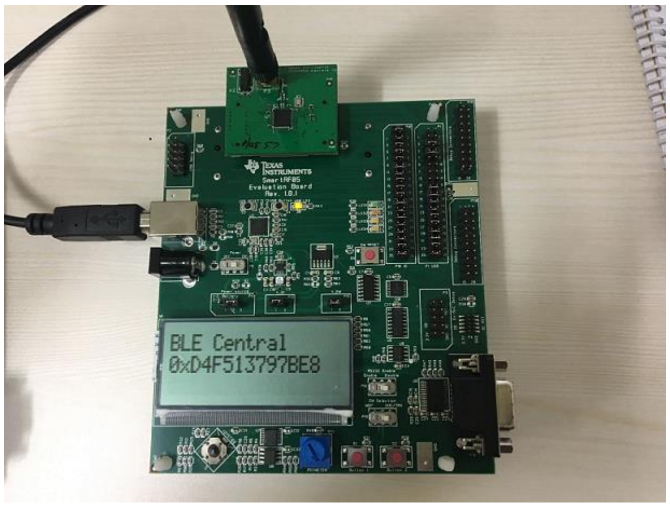


Figure 3.6: BLE central

3.4.2 The ADSC100 EZ kit that is shown in Figure 3.7, consists of BLE chip that acts as the BLE peripheral and the ADSP-BF592 DSP micro controller

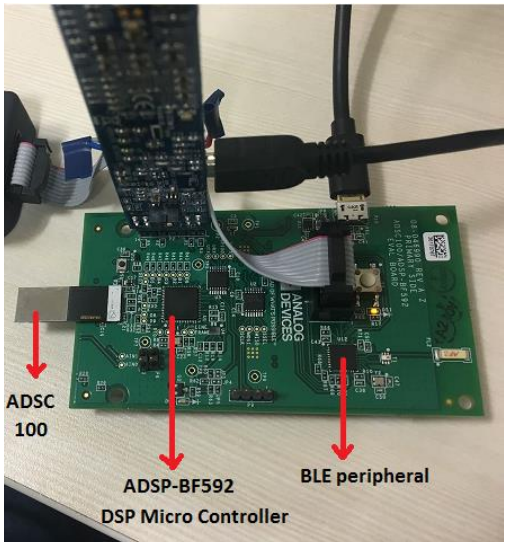


Figure 3.7: ADSC100 EZ kit

3.4.3 The code is flashed into the DSP micro controller with the help of a debugger ICE1000 and the code is flashed into the BLE peripheral chip using the CC debugger, shown in Figure in 3.8.

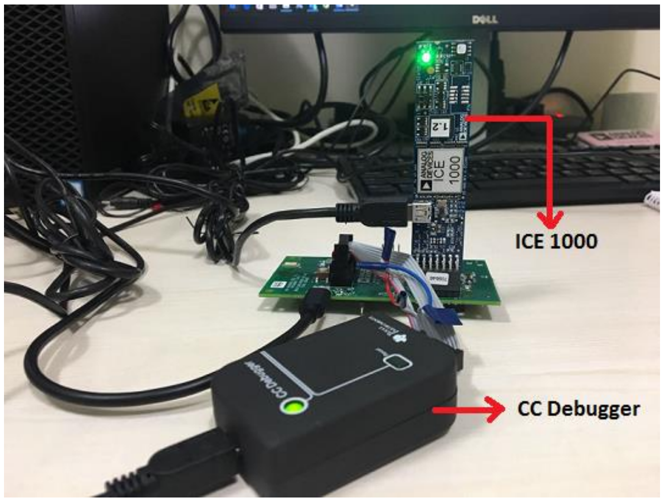


Figure 3.8: ICE1000 and CC Debugger

Once the BLE central and the ADSC100 EZ kit are powered on, the BLE peripheral starts advertising and the BLE central starts searching for nearby BLE devices to pair.

## 3.5 BLE communication

Any BLE chip has 2 kinds of profiles

1. Generic access profile (GAP)
2. Generic attribute profile (GATT)

1. The GAP profile determines the role of the BLE chip in connection:

a. BLE central
b. BLE peripheral
c. BLE observer
d. BLE advertiser

Based on the GAP profile, one BLE chip is the BLE central which acts as the master and the other is the BLE peripheral which will act as a slave. The BLE peripheral will start advertising itself and the BLE central discovers BLE peripheral and connects to it as shown in Figures 3.9, 3.10, 3.11.

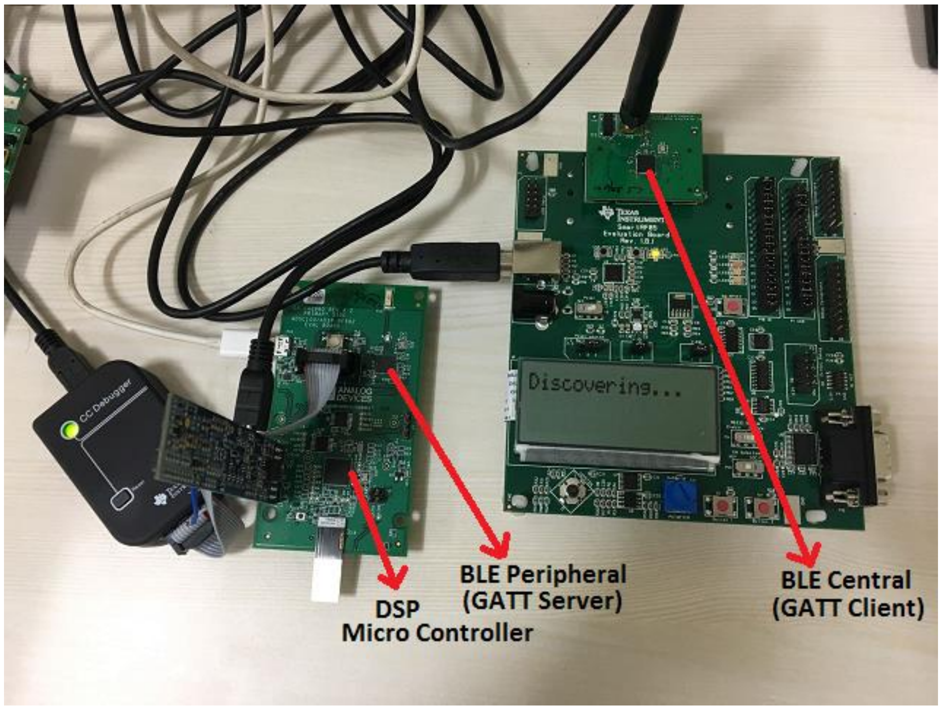


Figure 3.9: BLE central discovering BLE peripheral

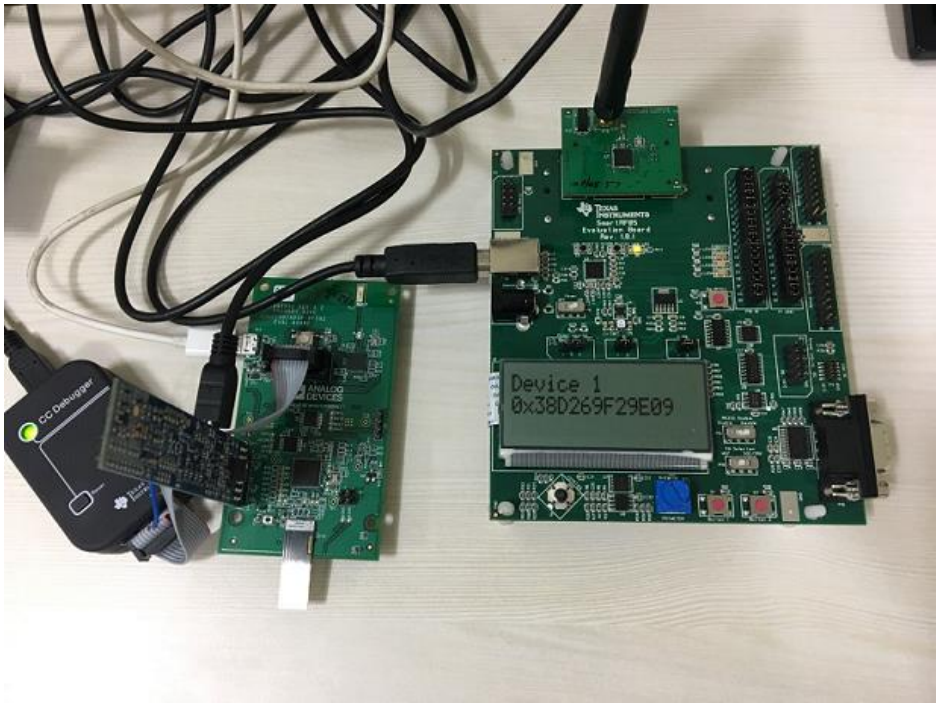


Figure 3.10: BLE central discovered BLE peripheral

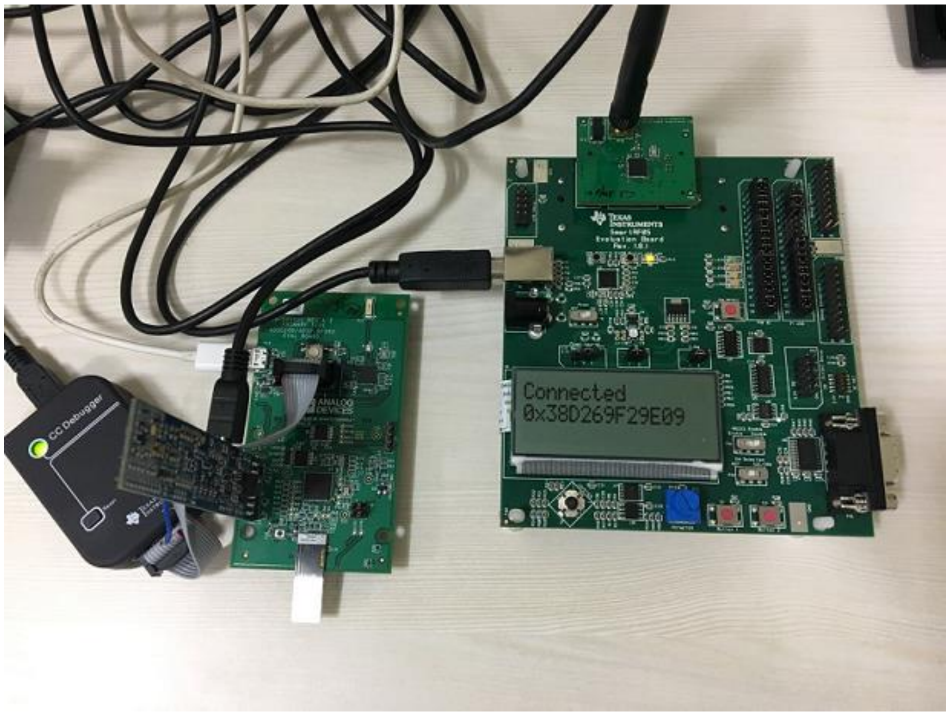


Figure 3.11: BLE central connected to BLE peripheral

2. The GATT profile determines the role of BLE chip in data transfer and based on this profile the BLE chip's role in data transfer can be classified into 2 roles and they are:

a. GATT client
b. GATT server

Based on the GATT profile of the setup we are using, the BLE peripheral acts as the GATT server and the BLE central chip acts at the GATT client.

The GATT server contains an attribute table which consists of characteristics with different attributes. In our device, the attribute table of the GATT server has 2 characteristics. The length of each of these characteristics is 16 bytes, this is the maximum length of data that can be transferred between the GATT client and the GATT server at a time. The 2 characteristics are

1. Controller characteristic: This characteristic has the permission to let the GATT client to both read and write (R/W) data into the GATT server.
2. Reporter characteristic: This characteristic has the permission to let GATT client to only read (R) data from the GATT server.

After connection is established between the BLE peripheral and BLE central, few messages are exchanged between both the BLE chips and the GATT server sends the handle addresses of both the characteristics to the GATT client as shown in Figure 3.12. The GATT client uses these handles to send commands to the GATT server.

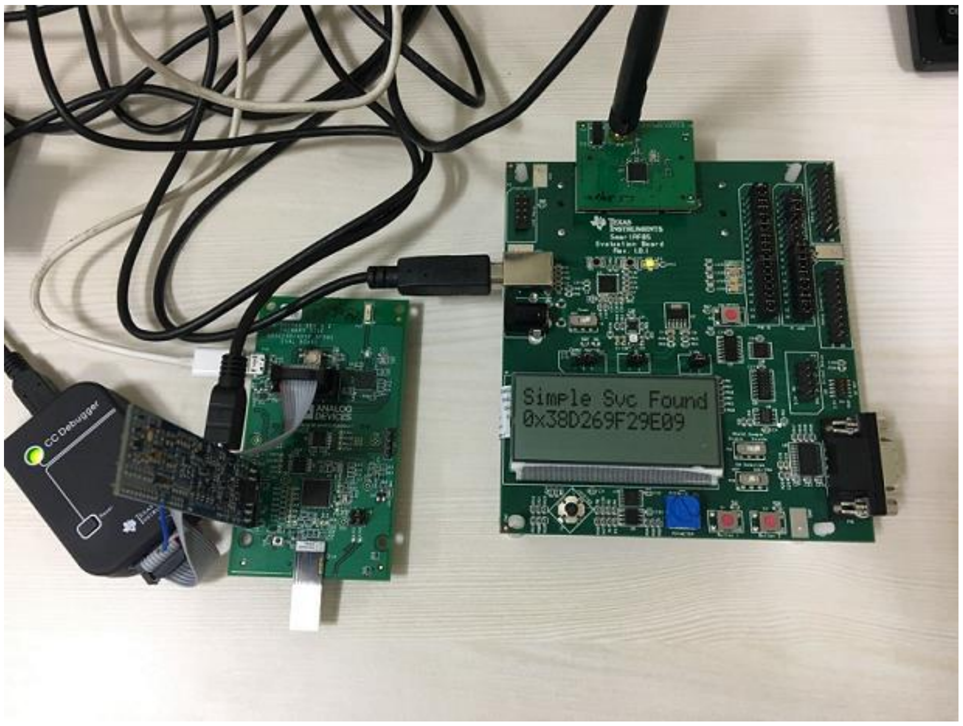


Figure 3.12: BLE central received handle addresses from BLE peripheral

GATT client writes data into the GATT server using the controller characteristic and the reads the data using the reporter characteristic. The BLE communication using GATT profile is shown in Figure 3.13

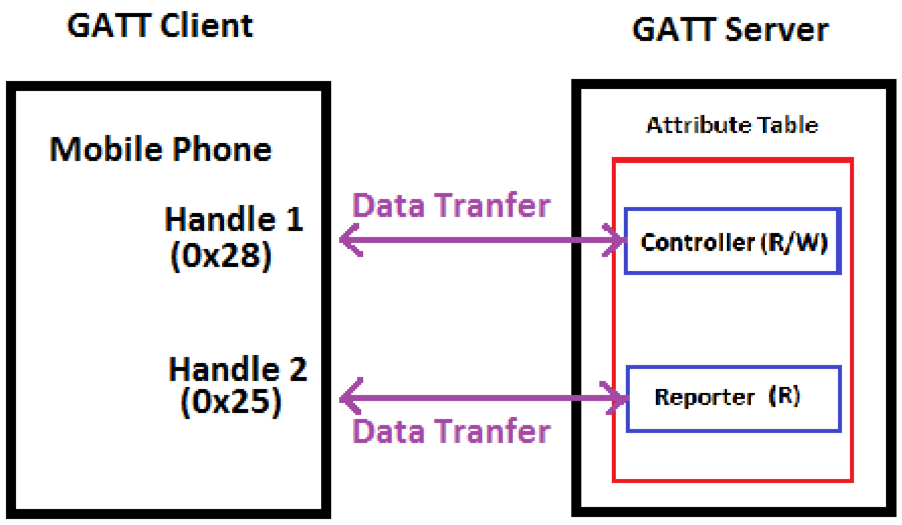


Figure 3.13: GATT client and GATT server data transfer

## 3.6 Working of SCiO

Once the BLE connection is made between the BLE central and the BLE peripheral then the GATT client (BLE central) sends the command to the GATT server (BLE peripheral) to capture the infrared image.

Since the commands are sent from GATT client to GATT server using characteristics, the command size is usually 16 bytes. Each command consists of

a. Message prefix
b. Command
c. Data length
d. Data.

The message prefix is used to check for the authenticity of the command received, followed by the actual command sent from the GATT client, then any data that is sent along with the command.

The BLE peripheral after the connection is established executes a state machine by switching to the corresponding command sent by the GATT client. When the GATT server receives the capture image command from the GATT client as shown in Figure 3.14, it sends the command to the DSP micro controller through UART communication. The UART settings that are used are baud rate of 115200 bits/second, 8 data bits, 1 stop bit and no parity.

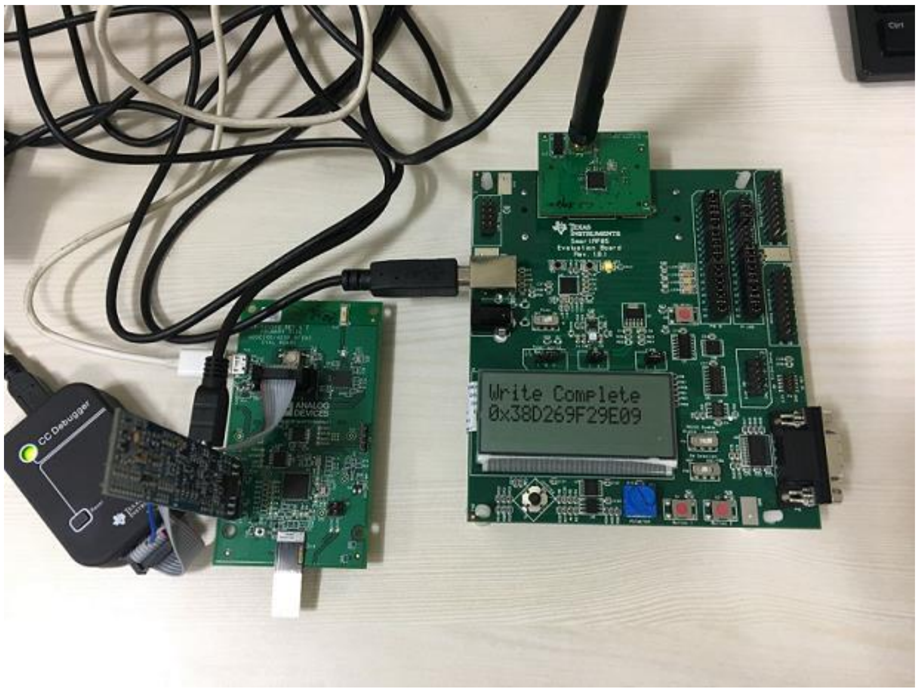


Figure 3.14: BLE central sending commands to BLE peripheral

The DSP Micro Controller receives the command from the BLE peripheral and sends the command to the Image Sensor (ADSC100) through I2C Communication.

The ADSC100 image sensor has a LED driver and a spectrometer, which performs NIR spectroscopy by projecting NIR waves on to the object that is being scanned, the object absorbs some of the waves and reflects some of the waves back depending on its chemical compositions and properties, the spectrometer captures these reflected waves, and sends the image data to the DSP micro controller.

## 3.7 Image processing

The image processing algorithm has been developed by Analog Devices and the image sensor ADSC100 is designed such that it captures the infrared image of an object and it divides the image into 6 channels. The image captured by ADSC100 is of size 640x480 pixels. Each pixel represents a data of 2 bytes which equals the total image size to 600 KB of data.

The image is divided into 6 channels, each channel representing a small rectangular part of the image as shown in Figure 3.15.

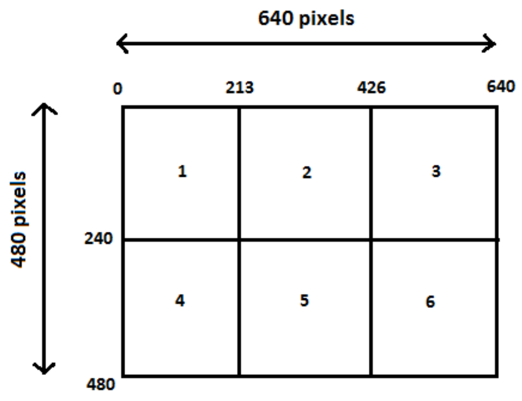


Figure 3.15: Image channels

The center of each channel is known as the channel center. With the channel center as the center, 69 rings of increasing radius are drawn until we reach the boundaries of the channel. A particular ring is selected and the intensity of all the pixels that are present inside the ring is summed up and stored. This process is repeated for all the rings in the channel. This yields 69 blocks of data corresponding to the 69 rings considered inside a channel.

This process is repeated for all the 6 channels and finally we have data for 6 channels, each channel having 69 blocks of data.

But the ADSP-BF592 has a data RAM of 32 KB. So, at a time only 5 rows of image is collected from the image sensor into the DSP micro controller through the PPI Interface and the dead pixel algorithm is applied to determine if any dead pixels are present in the image rows and then image processing algorithm is applied on it. The above process is repeated until the whole image is processed. The processed data of the complete image occupies a memory of 1.6 KB.

## 3.8 AES encryption and Authentication

The processed image data is encrypted by AES algorithm using Cipher clock chaining (CBC) mode [9], 128 bit encryption key. Initially a random number is generated and secure hash (SHA1) algorithm [9] is applied on it to generate a 20 byte SHA1 hash value which is used as the initial vector in CBC mode of AES encryption.

After the data is encrypted using the 20 byte initial vector, the data is authenticated using Hash based message authentication code (HMAC) algorithm [9]. The HMAC algorithm is applied on the data to generate a 20 byte HMAC hash value.

After the completion of these steps, the encrypted image data along with the initial vectors and the HMAC hash value of size around 1.6 KB is sent to the BLE peripheral using UART communication.

The encrypted data of size 1.6 KB is sent from the BLE peripheral (GATT server) to the BLE central (GATT client) as shown in Figure 3.15 with the help of reporter characteristic.

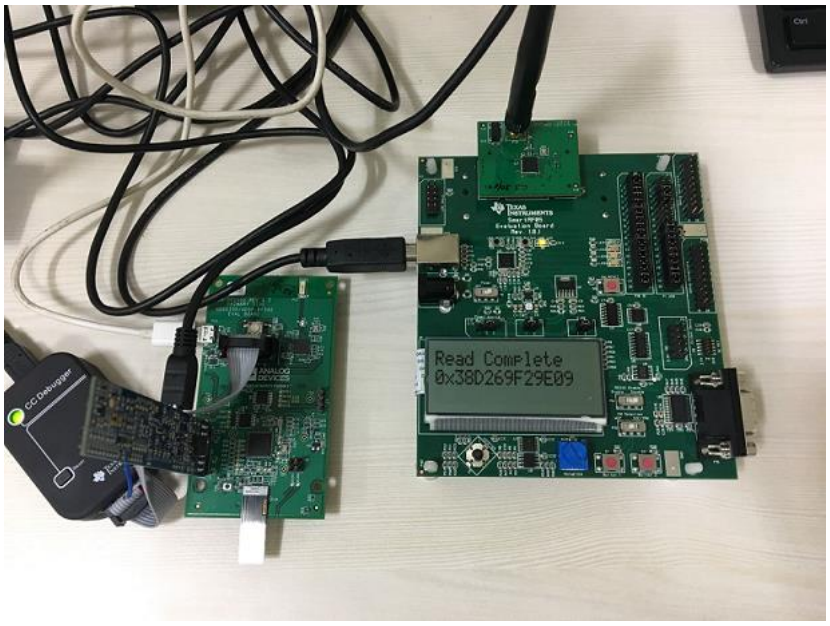


Figure 3.16: BLE central received the processed image data

# Chapter 4

## Future Work and Conclusion

In this chapter, we will discuss about the future works of the projects.

In my project I have worked until the encrypted processed image data reached BLE central. After the data is received by BLE central, the data has to be sent to the cloud via Wi-Fi and it has to be compared with the database that is present in the cloud. The cloud sends back the corresponding information about the scanned object to the phone app.

BLE central that is used in the setup has to be replaced by the mobile phone. So, the corresponding phone app has to developed, with appropriate GAP, GATT profiles to connect with the BLE peripheral.

The SCiO module has to be interfaced with smart phone case and the working of the SCiO should be controlled from the moduware phone app.

## References

[1] Moduware, technological partner of Analog Devices in developing Smart phone cases- https://moduware.com/

[2] Consumer Physics, technological partner of Analog Devices in developing SCiO Module - https://www.consumerphysics.com/

[3] Datasheet of ADuCM3029 MCU - http://www.analog.com/media/en/technical-documentation/data-sheets/ADuCM3027_3029.pdf

[4] Schematics of ADICUP3029 - https://wiki.analog.com/resources/eval/user-guides/eval-adicup3029/hardware/adicup3029

[5] Hardware reference manual of ADSP-BF592 - http://www.analog.com/media/en/dsp-documentation/processor-manuals/ADSP-BF59x_hwr_rev1.2.pdf

[6] Datasheet of ADSP-BF592 - http://www.analog.com/media/en/technical-documentation/data-sheets/ADSP-BF592.pdf

[7] Smart RF Board BLE stack - http://www.ti.com/tool/BLE-STACK

[8] Analog Devices, image sensor ADSC100 - http://investor.analog.com/releasedetail.cfm?releaseid=955544

[9] Cryptography and Network Security by William Stallings- http://www.inf.ufsc.br/~bosco.sobral/ensino/ine5680/material-cripto-seg/2014-1/Stallings/Stallings_Cryptography_and_Network_Security.pdf