University of Colorado Boulder

TWO FACTOR AUTHENTICATION SYSTEM

BY

SIDDHANT JAJOO
SATYA MEHTA

UNDER THE GUIDANCE OF:

PROFESSOR RICHARD HEIDERBECHT,
UNIVERSITY OF COLORADO BOULDER
4/29/2019

GITHUB LINK: https://github.com/jajoosiddhant/Two-Factor-Authentication-System

TABLE OF CONTENTS

OVERVIEW

The aim of this project is to have a secure locking system to a highly confidential resource, for example: Bank Lockers. This is implemented in the form of **Two Factor Authentication System**. This is achieved with the help of a TIVA board and a BBG board. The TIVA board acts as the remote node and the BBG board acts as the control node. The control node is responsible for processing and making all the real-time decisions based on values/data received from the TIVA board. The TIVA + FreeRTOS is integrated with different sensors and input devices to obtain values as per the application. The control node and remote node will function in a closed loop. The Control node will drive the outputs of the Remote Node in the form of actuators. The communication between the two boards can be wired or wireless.
The Sensors integrated on the Remote Node are:

- Keypad.
- Fingerprint sensor-GT521F32.

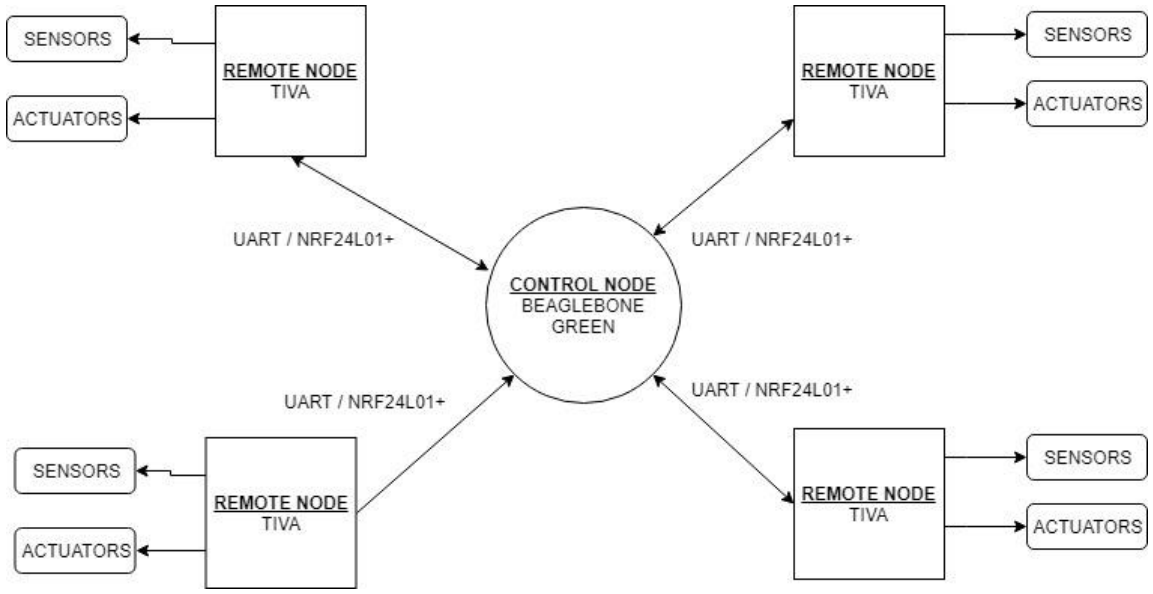
Actuators used on the Remote Node are:

- Buzzer
- LCD

The communication used between the boards is:

- NRF24L01+
- UART

Two communication methods have been used to provide fail-safe communication. If any one mode of communication fails, the system should switch to the other mode of communication and complete the required task.
The final generalized application would look like this:



We have designed and would be demonstrating the prototype of the above application in this report with a single Control Node and Remote Node. The remote node will wait for a valid fingerprint to be detected using interrupts. If a fingerprint is valid, it will send a packet to the remote node specifying the ID of the matched fingerprint otherwise it would send a "not matched" packet to the control node. Every Packet send will be responded with an acknowledgement by both the nodes. On receiving the packet specifying if the fingerprint is matched, the control node would respond with an OTP which would be required by the user to input via keypad. The messages for the steps to be executed would be continuously displayed on the LCD attached on the Remote Node. The entered OTP is verified with the OTP stored in the Control Node. If the OTP matches, then Access is granted otherwise buzzer is sound. All the data is logged into a text file on the Control Node. This includes all the messages of the Remote Node sent to the Control Node as well. In addition to this, A GUI has been designed on the Control Node having the following buttons:

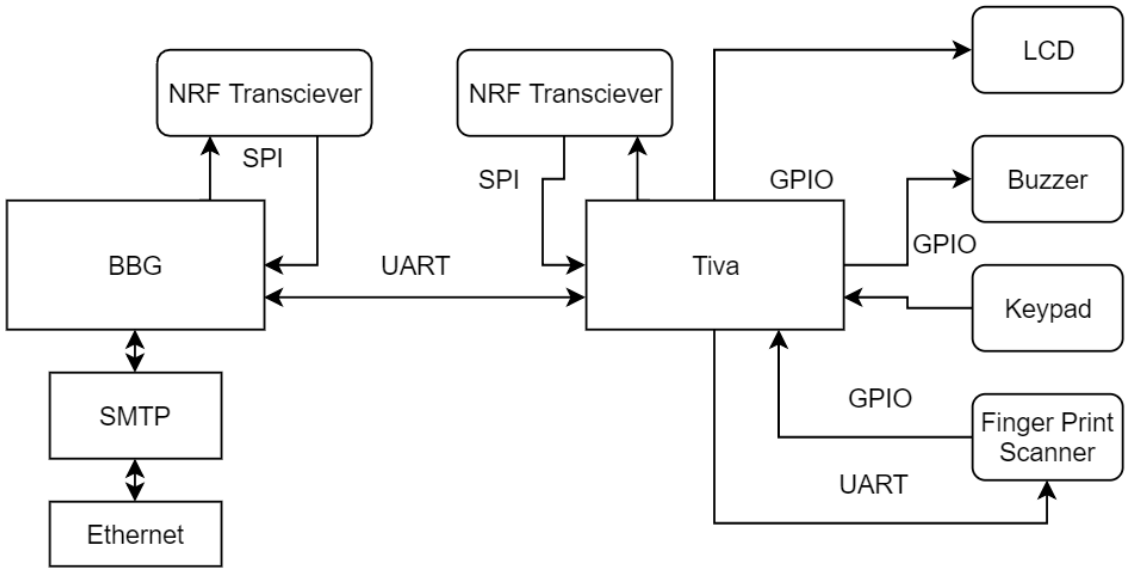
1. Buzzer off: Switches the Buzzer off.
2. Buzzer on: Switches the Buzzer on.
3. Delete all: Deletes all the registered fingerprints.
4. Add fingerprint: Starts the routine to add a fingerprint on the Remote Node.
5. Reset: Resets the Remote Node to the starting point.

CONCEPTS USED

This Project implements the following features and concepts:

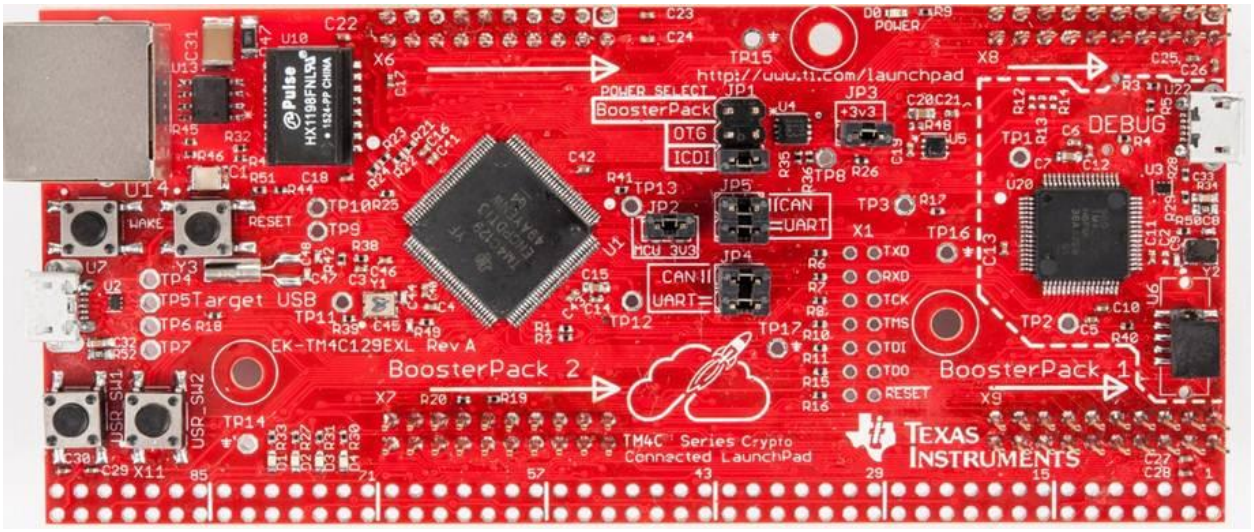
1. Multithreading using POSIX Threads.
2. Design Threads to operate in an event loop.
3. Error Checking and Handling.
4. Built in Self Tests.
5. Heartbeat Notification from all threads.
6. Implemented Log level – INFO, WARNING, ERROR, DEBUG.
7. POSIX Timers
8. POSIX Clocks
9. TIVA Hardware Timers.
10. Signal Handlers.
11. Synchronization Primitives.
12. Inter Process Communication (IPC) – Sockets and Message Queues.
13. Python scripts- SMTP lib and PyQT lib.

BLOCK DIAGRAM



BOARDS

TIVA C SERIES TM4C1294XL (FREERTOS)



[*]

The above image is the image for Tiva C Series TM4C1294XL used in this project as the Remote Node. This board runs on FreeRTOS. The board has inbuilt dedicated SPI, UART and I2C pins for use.

PINOUT

The following is the table for the different sensors and Actuators connecting to different GPIO pins for this project.

PORT AND PIN	FUNCTIONALITY
UART TO BBG	
PP0	RX
PP1	TX
KEYPAD	
PM0	COLUMN1 FROM RIGHT
PM1	COLUMN2 FROM RIGHT
PM2	COLUMN3 FROM RIGHT
PM6	COLUMN4 FROM RIGHT
PQ0	ROW 1 FROM TOP
PQ2	ROW 2 FROM TOP
PQ3	ROW 3 FROM TOP
PQ1	ROW 4 FROM TOP

BUZZER	
PG0 (PWM)	BUZZER (+) PIN
GND	BUZZER (-) PIN
LCD	
PK0	D4
PK1	D5
PK2	D6
PK3	D7
PK4	REGISTER SELECT(RS)
PK5	READ WRITE(1/0)
PK6	ENABLE
FINGERPRINT SENSOR (REFER FIGURE 1 AND 2 BELOW)	
PA7	TX (3.3V)
PA6	RX (3.3V)
GND	REFER FIG 1 BELOW
VCC (3.3V ~ 6V)	REFER FIG 1 BELOW
PL3	ICPCK
ICPDA	NOT USED.
GND	REFER FIG 2 BELOW
VCC (3.3V)	REFER FIG 2 BELOW
NRF24L01+ (REFER FIGURE 3 BELOW)	
VCC (3.3V)	REFER FIGURE 3 BELOW
GND	REFER FIGURE 3 BELOW
PC6	CSN
PC7	CE
PD3	SCLK
PD0	MISO
PD1	MOSI
PL2	IRQ

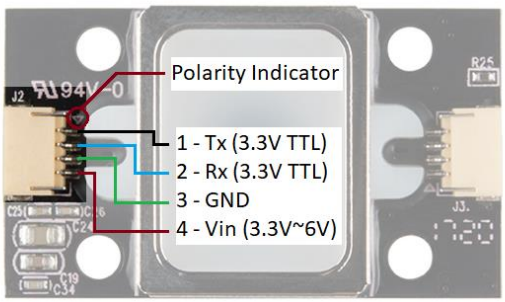


FIGURE 1[*]

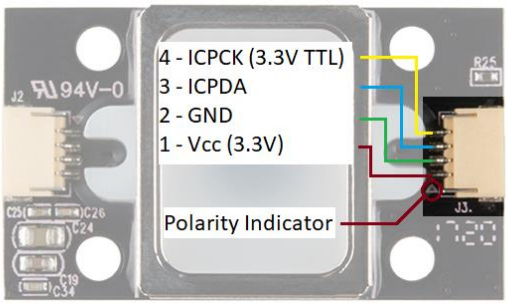


FIGURE 2[*]

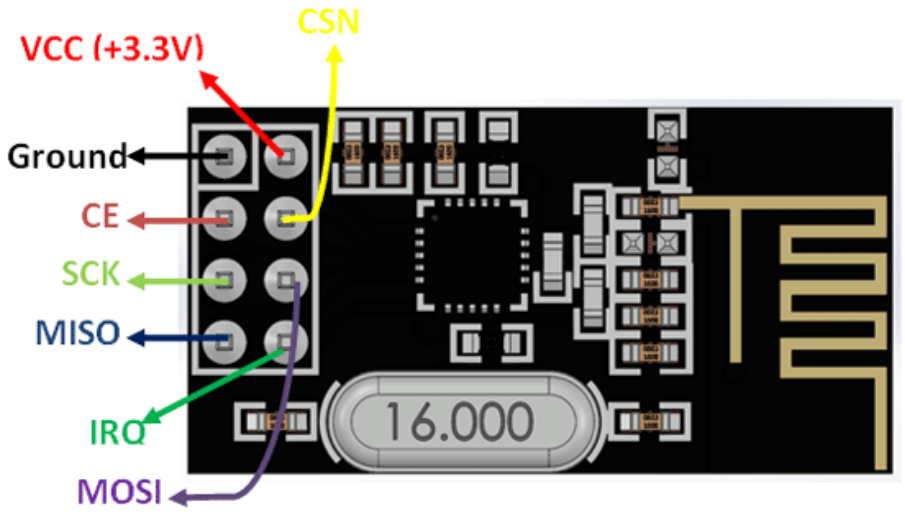


FIGURE 3[*]

BEAGLEBONE GREEN (LINUX)

[*]

The above image is the image for Beaglebone Green used in this project as the Control Node. This board runs on linux. The stock image which is called as debian is used in this project. The board has inbuilt dedicated SPI, UART and I2C pins for use. All the logging functionality is done on the BBG.

PINOUT

PINS (P9 HEADER)	FUNCTIONALITY
1	GND
2	GND
3	VCC (3.3 V)
11	RX (UART FOR TIVA)
12 (GPIO 60)	CS (NRF24L01+)
13	TX (UART FOR TIVA)
14 (GPIO 50)	IRQ (NRF24L01+)
16 (GPIO 51)	CE (NRF24L01+)
18	MOSI (NRF24L01+)
21	MISO (NRF24L01+)
22	SCLK (NRF24L01+)

SENSORS

Fingerprint Sensor (GT-521F32)

The fingerprint scanner has the ability to:

- Enroll a Fingerprint
- Identify a Fingerprint
- Capable of 360° Recognition

The Fingerprint Sensor can store upto 300 fingerprints in its database. All the communication to read and write its registers and store a valid fingerprint in its database is achieved using UART. There are provisions to even delete specified or all stored fingerprints. The fingerprint sensor has an ICPCK pin which transitions from low to high when the finger is pressed on the scanner and a high to low transition when the finger is lifted from the scanner. If the finger is continuously pressed against the scanner the output on ICPCK pin stays high. The ICPCK pin stays low when no finger is pressed against the scanner.

Behavior	Status
Just touch frame	ICPCK=> "L"-->"H"
No touch frame	ICPCK=> "L"-->"L"
Keep touch frame	ICPCK=> "H"-->"H"
Taking off finger on frame	ICPCK=> "H"-->"L"

Status of ICPCK pin

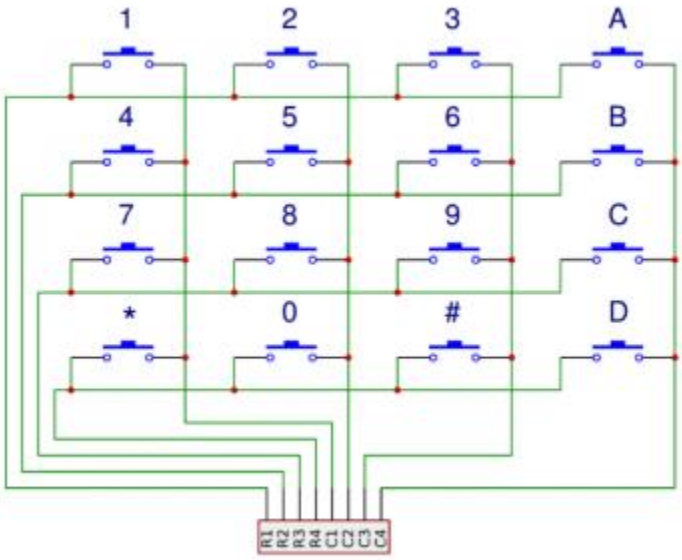
This output would be used as a GPIO interrupt and further actions would be taken based on the interrupt.

4X4 Matrix Keypad

[*]

A 4x4 Matrix keypad is used in order to enter the OTP received by the user from the BBG board. The output pins of the keypad would be interfaced to the GPIO pins of the TIVA board. The GPIO configuration is interrupt based and when a key is pressed interrupt will be generated to detect which key was pressed. Problems such as debouncing are taken care of in the code. The system would be coded in such a way that the keypad would input the values only when an OTP is asked to the user. Pressing the keypad buttons at any other time would yield nothing but a display message on the LCD to follow next steps.

The Schematic for a 4x4 keypad shows how the rows and columns are connected:



The above image shows accurately how the connections have been made.

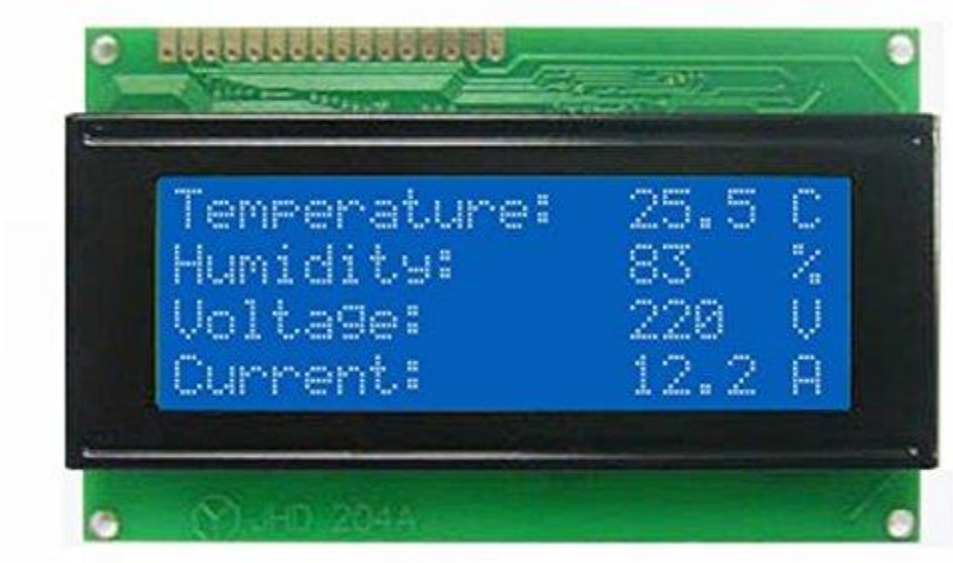
ACTUATORS

Buzzer

[*]

The buzzer will be interfaced to the PWM pin of TIVA. The buzzing sound can be controlled using PWM. The buzzer will be used to indicate that an unauthenticated person is trying to access the particular locked resource.

LCD



The LCD Hitachi HD44780 is used to display messages for the user. The LCD will display the instructions for the user to follow in order to unlock the locker.

COMMUNICAION INTERFACE

UART

UART would be used to initialize the fingerprint sensor, get data from fingerprint sensor and read the fingerprint from the sensor. In addition to this UART would also be used to communicate between the remote node (TIVA) and the control node (BBG). The idea behind using the UART is to provide a fail-safe system.

PACKET STRUCTURE

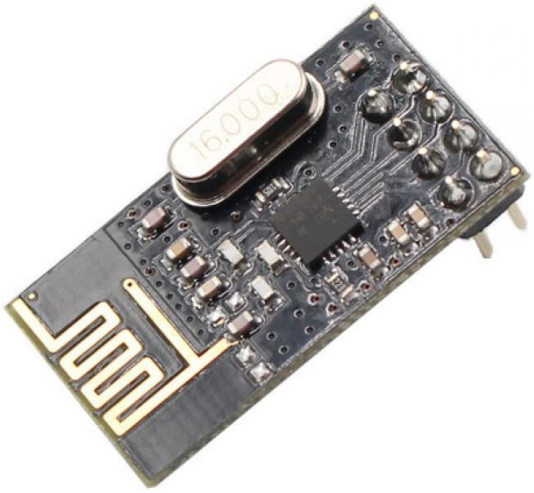
PREAMBLE (1 BYTE)	PACKET ID (1BYTE)	PAYLOAD SIZE (2 BYTES)	PAYLOAD MAX (50 BYTES)	ACK (1 BYTE)	CRC (2 BYTES)	POSTAMBLE (1 BYTE)

Above is the Packet structure which is used for transferring data between TIVA and BBG.
The following packet structure is stored in a structure and the whole structure is transferred between the Beaglebone and the TIVA.

The following are the fields and its description:

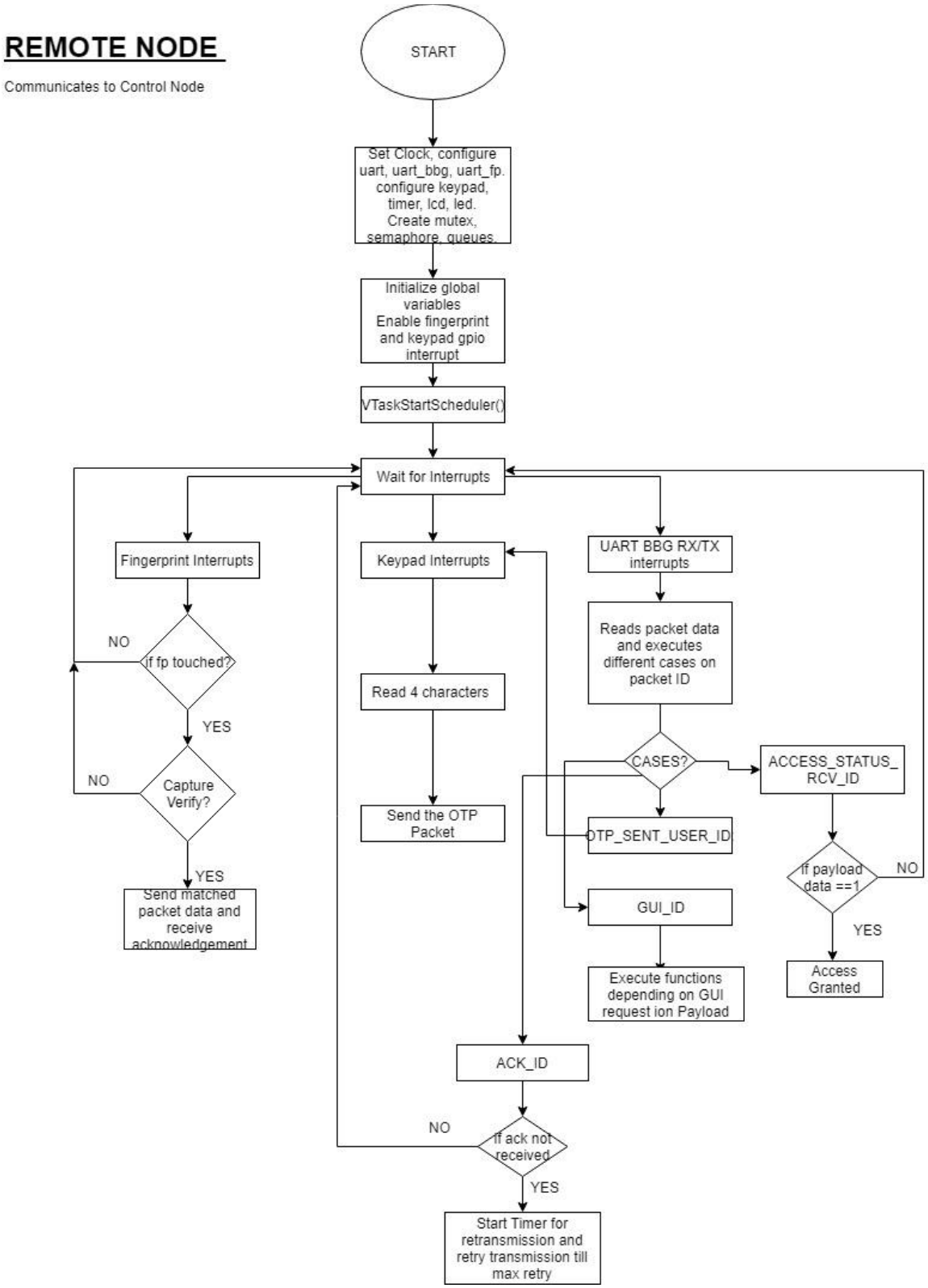
1. PREAMBLE: The preamble is used to signify the starting point of the Packet and can be used for synchronization. The value is 0xAB.
2. PACKET ID: This field specifies the event that needs to be triggered on the remote node and the control node. Further action/steps are taken depending on the Packet Id received.
3. PAYLOAD SIZE: This field consists of the size of the payload that is being transferred or received.
4. PAYLOAD: This field consists of data to be sent or received. Its maximum value can be as much as the value that can be stored in the PAYLOAD SIZE field. The payload is currently initialized to support up to 50 Bytes.
5. ACK: This field signifies if an acknowledgement is required in response to the packet send. If ACK is 0, receiver does not respond with an acknowledgement and if ACK is 1, receiver responds with an acknowledgement.
6. CRC: This field consists of CRC value calculated by using a checksum algorithm. The packet is discarded if this value does not match with the value calculated by the receiver.
7. POSTAMBLE: This marks the end of the packet. The value is 0xBA.

NRF24L01+

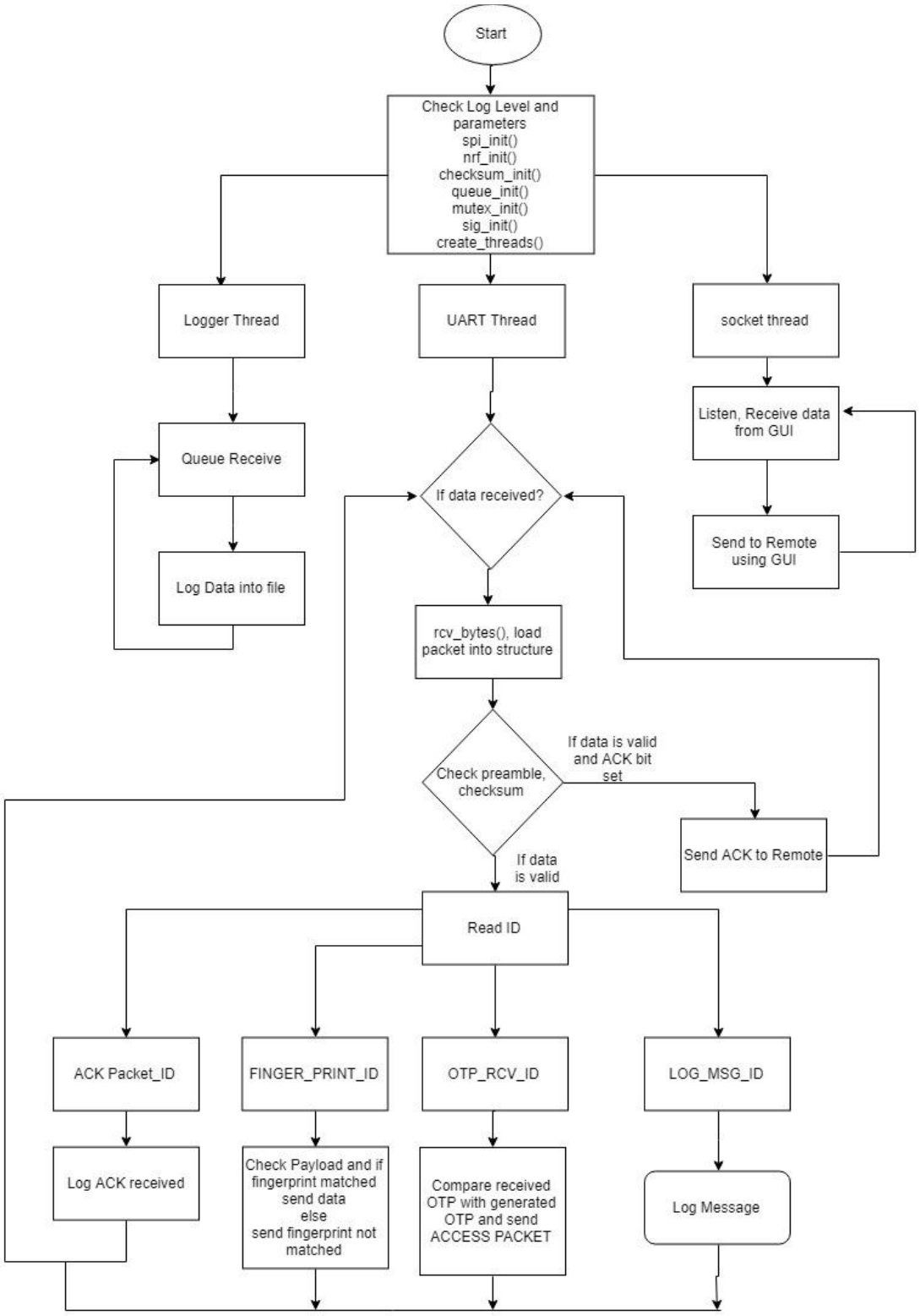


The NRF24L01+ module will be used to communicate wirelessly through RF. The transceivers are connected to the modules using SPI interface. The choice for NRF24L01+ module as the communication medium was to have a wireless communication between the remote node and control node. Having wireless communication would help in system maintenance.

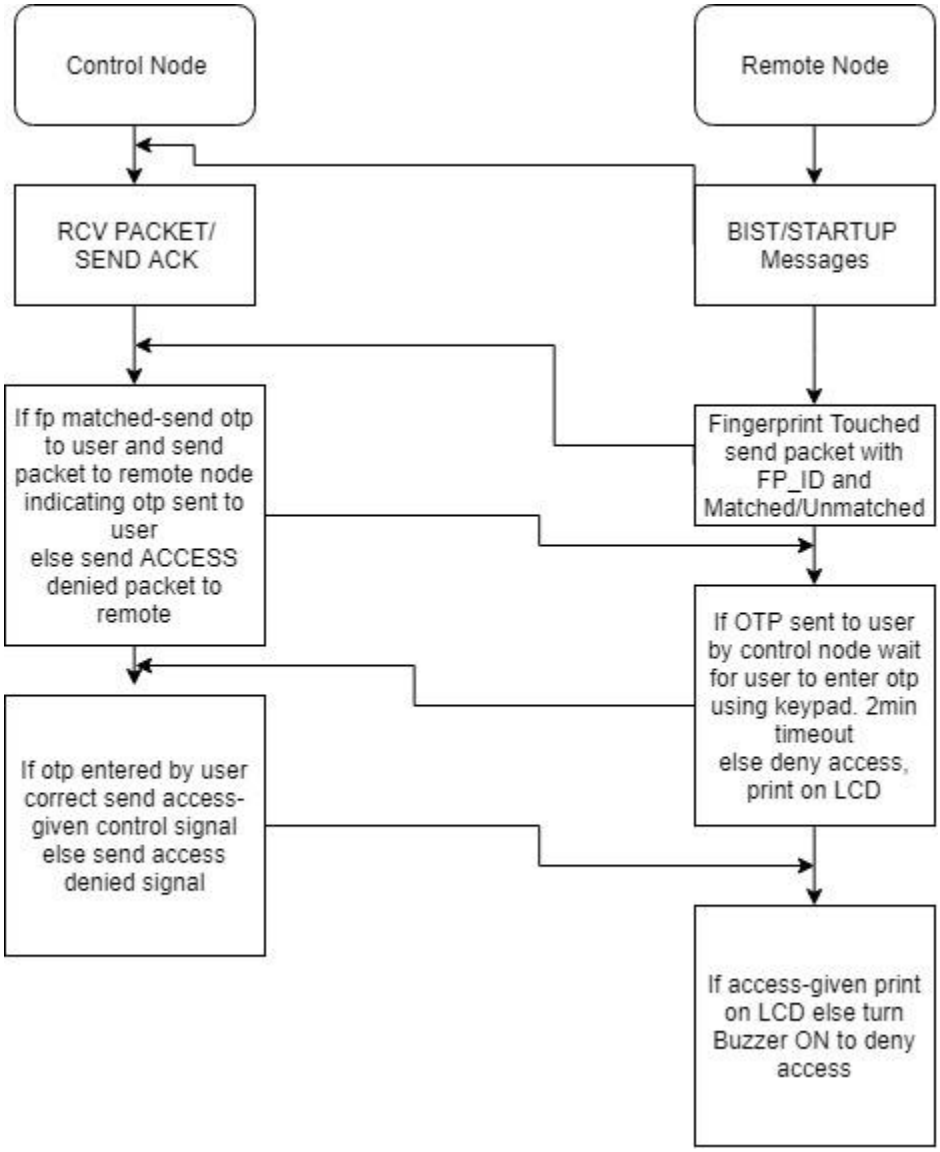
REMOTE NODE FLOWCHART



REMOTE NODE FLOWCHART



BASIC COMMUNICATION FLOW



MAJOR REQUIREMENTS

- Fingerprint scanned by authenticated person should be able to have access to the system.
- Fingerprint scanned by unauthenticated person should not have access to the system.
- Keypad as an input device to enter the OTP.
- Buzzer signal from the control node should ring when the access is made by any unauthorized person (Actuator).
- OTP calculation and verification to be done on the control node.
- LCD displays all the required messages for the user.
- GUI can control Buzzer and Finger Print Sensor functionalities and door access.
- System will support degraded functionality in absence of any sensor.
- Logging functionality on the Control Node including all the text messages received from the Remote Node as well

SCREENSHOTS: -

BBG:

```
[1556582892 sec, 499030572 nsec] BIST: Queue initialization successful.

[1556582892 sec, 499620577 nsec] Signal initialization successful.

[1556582892 sec, 500066341 nsec] BIST: Mutex initialization successful.

[1556582892 sec, 500848167 nsec] BIST: Thread Creation successful.

[1556582892 sec, 500867249 nsec] Reached main while loop.

[1556582892 sec, 503019906 nsec] Entered Socket Thread
[1556582892 sec, 504318699 nsec] Entered Logger Thread.

[1556582892 sec, 509010655 nsec] Entered NRF Thread.
```

Figure 1-Initialization

```
[1556583865 sec, 580311900 nsec] [REMOTE_NODE]: ACK Packet received

[1556583865 sec, 769906197 nsec] [REMOTE_NODE]: Buzzer on
[1556583868 sec, 815069299 nsec] Connected to remote Host

[1556583868 sec, 816110653 nsec] GUI Request received

[1556583868 sec, 817728722 nsec] Retry timer started
[1556583868 sec, 817758886 nsec] Retry Timer started
[1556583868 sec, 817766636 nsec] Buzzer OFF Request received from GUI
[1556583868 sec, 823836561 nsec] [REMOTE_NODE]: ACK Packet received

[1556583869 sec, 33951781 nsec] [REMOTE_NODE]: Buzzer off
[1556583876 sec, 982435907 nsec] [REMOTE_NODE]: Verifying Fingerprint
```

Figure 2-Remote Node Logs

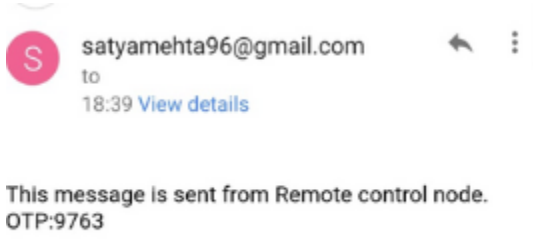


Figure 3-Otp sent to user

```
Postamble ba
CRC 0:
[1556585486 sec, 396659193 nsec] Retry timer started
[1556585486 sec, 396720064 nsec] Retry Timer started
[1556585486 sec, 396730147 nsec] Sent OTP sent packet to the remote node
[1556585486 sec, 402049062 nsec] [REMOTE NODE]: ACK Packet received
```

Figure 4- Retry Timer starts and sends message if ACK not received

```
[1556584964 sec, 120890340 nsec] OTP incorrect

[1556584964 sec, 121983271 nsec] Retry timer started
[1556584964 sec, 122009353 nsec] Retry Timer started
[1556584964 sec, 125828405 nsec] [REMOTE_NODE]: ACK Packet received

[1556584964 sec, 773974548 nsec] [REMOTE_NODE]: Access Denied, One more try left
[1556584971 sec, 461768062 nsec] [REMOTE_NODE]: Verifying Fingerprint
[1556584971 sec, 465150684 nsec] Siddhant wants access

[1556584973 sec, 399836965 nsec] Sent OTP to the Siddhant

[1556584973 sec, 400090490 nsec] Retry timer started
[1556584973 sec, 400119530 nsec] Retry Timer started
[1556584973 sec, 400135321 nsec] Sent OTP sent packet to the remote node
[1556584973 sec, 407917749 nsec] [REMOTE_NODE]: ACK Packet received

[1556584987 sec, 555606294 nsec] OTP incorrect

[1556584987 sec, 556926045 nsec] Retry timer started
[1556584987 sec, 556952126 nsec] Retry Timer started
[1556584987 sec, 561093492 nsec] [REMOTE_NODE]: ACK Packet received

[1556584987 sec, 830117116 nsec] [REMOTE_NODE]: Access Denied
[1556585001 sec, 931091517 nsec] Connected to remote Host
```

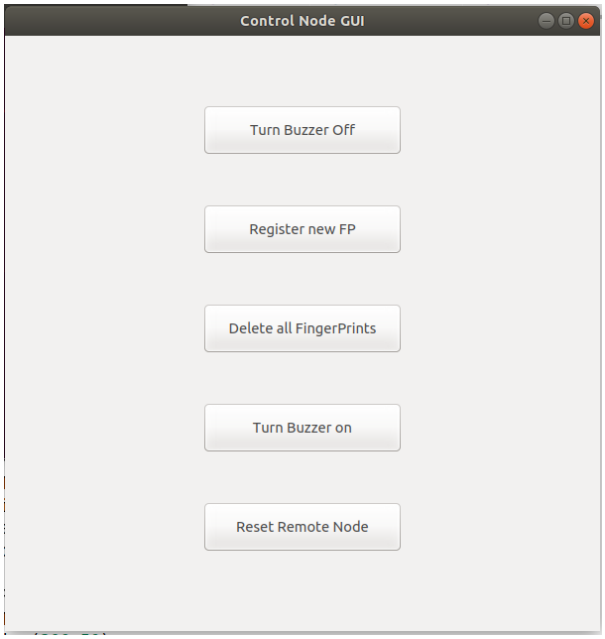
Figure 5-Two Tries given to the user if entered incorrect otp

ADDITIONAL FEATURES

GRAPHICAL USER INTERFACE(GUI) ON THE CONTROL NODE (BBG)

A GUI has been designed on the Control Node having the following buttons:

1. Buzzer off: Switches the Buzzer off.
2. Buzzer on: Switches the Buzzer on.
3. Delete all: Deletes all the registered fingerprints.
4. Add fingerprint: Starts the routine to add a fingerprint on the Remote Node.
5. Reset: Resets the Remote Node to the starting point.
6. Give Access: Allows access to the user.



This would result in ease of adding and deleting fingerprints in the database without needing to change any code.

STARTUP/BUILT-IN SELF TESTS

Before the application begins operation, some startup tests are run at the beginning to verify that the hardware and software are in working order. These tests consist of:

BEAGLEBONE GREEN

- Communication to the threads to make sure they have all started and up and running.
- Sending log messages to the logger task when individual BIST tests have finished along with an indicator if the hardware is connected and in working order. If something does not startup correctly, an error statement is logged and the USR Led is turned on.
- Queues and Signal Initializations.

TIVA TM4C1294XL

- Communication with the Fingerprint sensor GT-521F32 that can confirm that UART works and that the hardware is functioning.
- Sending log messages to the BBG when individual BIST tests have finished along with an indicator if the hardware is **connected** and in working order. If something does not startup correctly, an error statement is logged and the USR Led is turned on.

CHECKSUM ON TIVA AND BBG

The checksum field was included so that the packet received can be checked for any errors and if any errors are received, acknowledgement is not received which will lead to the transmitter sending the packet again.

LOG LEVELS AND HIGHER PRIORITIES TO ERRORS ON BEAGLEBONE GREEN

Different log levels have been implemented in this application on the Control Node(BBG). Different bits in uint8_t represent different log levels. The different log levels are as follows:

1. INFO (bit 1) – This level only displays the temperature and light readings.
2. WARNING (bit 2) – All the stuff that does not significantly affect the system and can be let as a warning to the user is displayed in this log level.
3. ERRORS (bit 3) – Only the errors are reported in the text file in this log level.
4. DEBUG (bit 4) – All the debug messages and every other message is printed in the text file in this particular log level.

The log level is implemented by assigning a log level to every printf. The log level is accepted as a command line argument. Only the accepted command line log level messages are displayed.

In order to notify the users of errors as quickly as possible, the errors are given the highest priority. The APIs are made in order to support modularity so that if required to interface any additional sensors in future, such features can be useful.

SIGNAL HANDLER ON BEAGLEBONE GREEN

A signal handler is implemented in order to gracefully close and exit the program. All the objects, file descriptors, message queues, threads created and any memory that was allocated at the beginning or during the program execution is freed or destroyed on the reception of the signal INT (CTRL + C).

FAULT TOLERANCE AND FAIL SAFE METHODS

- If fingerprint sensor fails, then the control node can control the access service using GUI.
- If communication fails, then the ack packets will not be received and therefore sending node will retry sending packets. If the retries exceed the max retries then both the systems will reset and try to setup the communication link
- If the fingerprint Sensor fails, there is a provision to the user to input a secret passcode only known to him. Upon validation of this passcode an OTP would be generated and sent to the user via email. Access would be granted on successful verification of the OTP.
- If the system is not recoverable by itself, the system could be reset by clicking on the Reset button in the GUI. In addition to this, turning the buzzer off is also achieved using the GUI if the OTP is entered is incorrect.

VERIFICATION PLAN

TASK	EXPECTED DATE	Completion Date
NRF Module- Interfacing	4/16/2019	04/18/2019
Keypad, LCD	4/19/2019	4/19/2019
NRF Communication	4/22/2019	SPI driver written
Buzzer, UART	4/24/2019	4/25/2019
BBG, TIVA Integration	4/26/2019	4/27/2019
Fault Management	4/27/2019	4/28/2019
BIST,Logging functionality	4/28/2019	4/29/2019
Testing of final application	4/29/2019	4/29/2019

TEST PLAN AND RESULTS

Tests	Result	Validation
Fingerprint sensor user registration and database	Pass	Retrieved count and id from the fingerprint after successful registration
Fingerprint sensor delete database	Pass	Retrieved count from fingerprint after deletion
NRF SPI Interface (Read/Write register values	Pass	Wrote registers and read back to verify
NRF Communication	Fail	SPI driver working but not able to receive packets on both ends.
UART communication (Interrupt based on Tiva)	Pass	TX/RX interrupts generated and verified by sending, receiving data
Print data on LCD	Pass	LCD was able to print data we write
Buzzer interfacing (PWM)	Pass	Buzzer rings on enabling PWM channel
Keypad interfacing (using GPIO interrupts and avoid debouncing)	Pass	GPIO interrupts were generated on pressing keys on keypad
Python script which takes command line argument and sends an email	Pass	Executed python script using system() and validated by checking the email
Authenticated person can access the system	Pass	If registered user places his finger lcd prints access granted
Unauthenticated person is not able to access the system and buzzer rings.	Pass	If unregistered user places his finger on scanner buzzer rings.
GUI able to stop/start the buzzer	Pass	Button press on gui stops/starts the buzzer
GUI able to allow access to any user	Pass	LCD prints access granted when GUI button pressed
GUI able to register new fingerprint, delete all fingerprints	Pass	User is able to register its fingerprint when pressed button n GUI
GUI able to reset the remote node	Pass with exception	Initializations are done again but the old thread is not cancelled and new one is not created.
Two attempts to given to the user if OTP entered wrong	Pass	Entered wrong OTP in keypad for the first try and then entered correct OTP. LCD displays all the required messages for the user.

CHALLENGES FACED

- The scope of the project was so vast that it required a lot of thinking to decide a topic that could contribute to a real life application. It taught us to not be over ambitious and submit a decent and working project than a partial working project in the specified time frame along with good documentation.
- Creating a Packet structure to communicate between the TIVA board and Beaglebone board was quite challenging. Care had to be taken that the payload sent by the transmitter was received by the receiver and the payload received was accurate. This was achieved using the ACK and CRC fields.
- The most challenging part was to interface NRF24L01+ with the TIVA and Beaglebone Green using SPI. We did manage to complete read/write registers of the NRF module using SPI but communication between the two modules is still incomplete. We think there are some timing constraints and synchronization requirements that we have failed to understand which have led to our module not communicating.
- UART and SPI were initially not working on BBG due to the cape and overlay issues. The default pin configurations on Beaglebone are GPIO rather than any communication peripheral. Hence, we were required to make some additions in /boot/uEnv.txt. Also, the pin functionality can be selected using config-pin command. We made a bash script which is required to be executed at every reboot.
- Implementing Fingerprint Sensor using UART was a little difficult. While implementing UART, the data was being loop-backed and we were not able to send data to the receiver. This problem was solved by having a common ground for both the TIVA and BBG boards.
- During socket communication between python-based GUI and control node process the GUI was only able to send data to the control node once. This issue was mainly due to closing the socket and trying to access it again. The problem was solved by creating new instances every time we want to send the data. The old instances were getting deleted once we close the socket.

REFERENCES

- https://infocenter.nordicsemi.com/pdf/nRF24L01P_PS_v1.0.pdf
- http://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/
- https://www.sparkfun.com/products/14518
- https://barrgroup.com/Embedded-Systems/How-To/CRC-Calculation-C-Code