Design of BLE Controlled Lighting System

1. INTRODUCTION

1.1 OBJECTIVES AND SCOPE OF THE WORK

In this project work I am implementing a smart switch to controll lighting system (RGB led) based on the Bluetooth Low Energy wireless protocol (BLE). The BLE divides the world into peripheral devices (sensors) and central devices(smartphone/laptop). Peripherals can either connect directly to a central device, or broadcast data to any device in range by sending out "advertising packets" once connected, the central device can get a list of services offered by the peripheral.

In this project work I created the BLE peripheral with a one LED custom service with two characteristics: a readable/writeable characteristic called "Switch," which is used to flip the switch on and off, and a second characteristic called "State," which can notify us of changes in the status of the switch. The application running on smartphone gets notifications regarding the status of light i.e. whether the led is on or off, and send a commands to Arduino UNO board over Bluetooth to toggle the LED, so it can update its local status and to change the colour, brightness of the RGB led. System having one push button for manual operation (on/off the light), and LED for indicate the RSSI of central device.

This proposed research work would be implemented using **Arduino** based embedded system design methodology, which includes Embedded hardware and firmware design modules. This project would be carried out with ATMEGA328P based Arduino UNO board and Aurduino driven Embedded EDA Tool kits.

Hardware requirements:

- ATmega328 Microcontroller based Arduino UNO Board
- BlueFruit BLE Module
- RGB Led
- USB Cable
- Bread Board
- Connecting wires

Software requirements:

- Embedded C
- codebender Compiler
- fritzing.0.9.3b.64.pc

2. HARDWARE IMPLEMENTATION OF DASH BOARD

2.1 DASH BOARD

The implementation of the Dash board can be divided in two parts.

- Hardware implementation
- Firmware implementation

The **Hardware implementation** deals in drawing the schematic on the plane paper according to the application, testing the schematic design over the breadboard using the various IC's to find if the design meets the objective, carrying out the PCB layout of the schematic tested on breadboard, finally preparing the board and testing the designed hardware.

The block diagram discusses about the required components of Dash board and working condition is explained using circuit diagram and system wiring diagram.

2.2 Block Diagram of UNO board

The block diagram of UNO board is as shown in Fig 2.1. It consists of microcontroller, BLE module, power supply unit, RGB-Led and push button (switch).

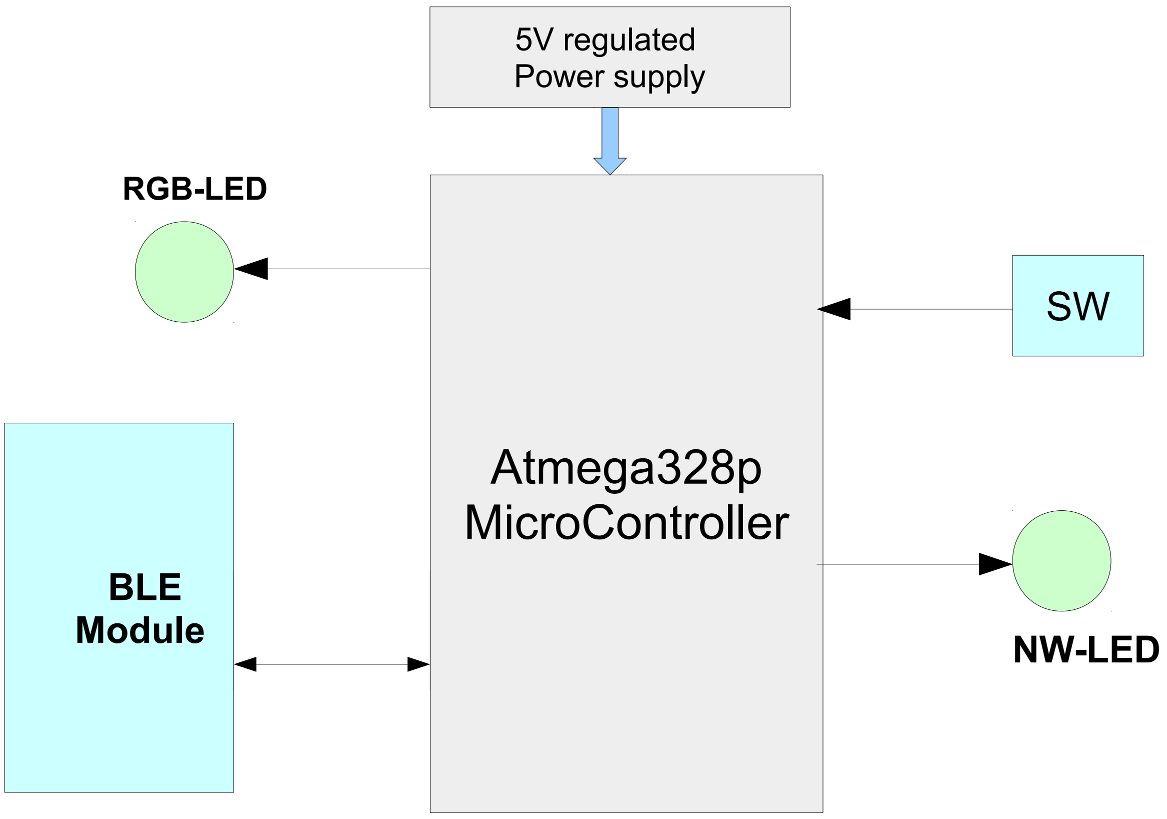


Fig.2.1 the Hardware diagram of the system

2.3 Circuit Diagram of Dash board : I have used the fritzing.0.9.3b.64.pc software for Schematic design. Fig 2.2 shows the circuit diagram of the Dash board.

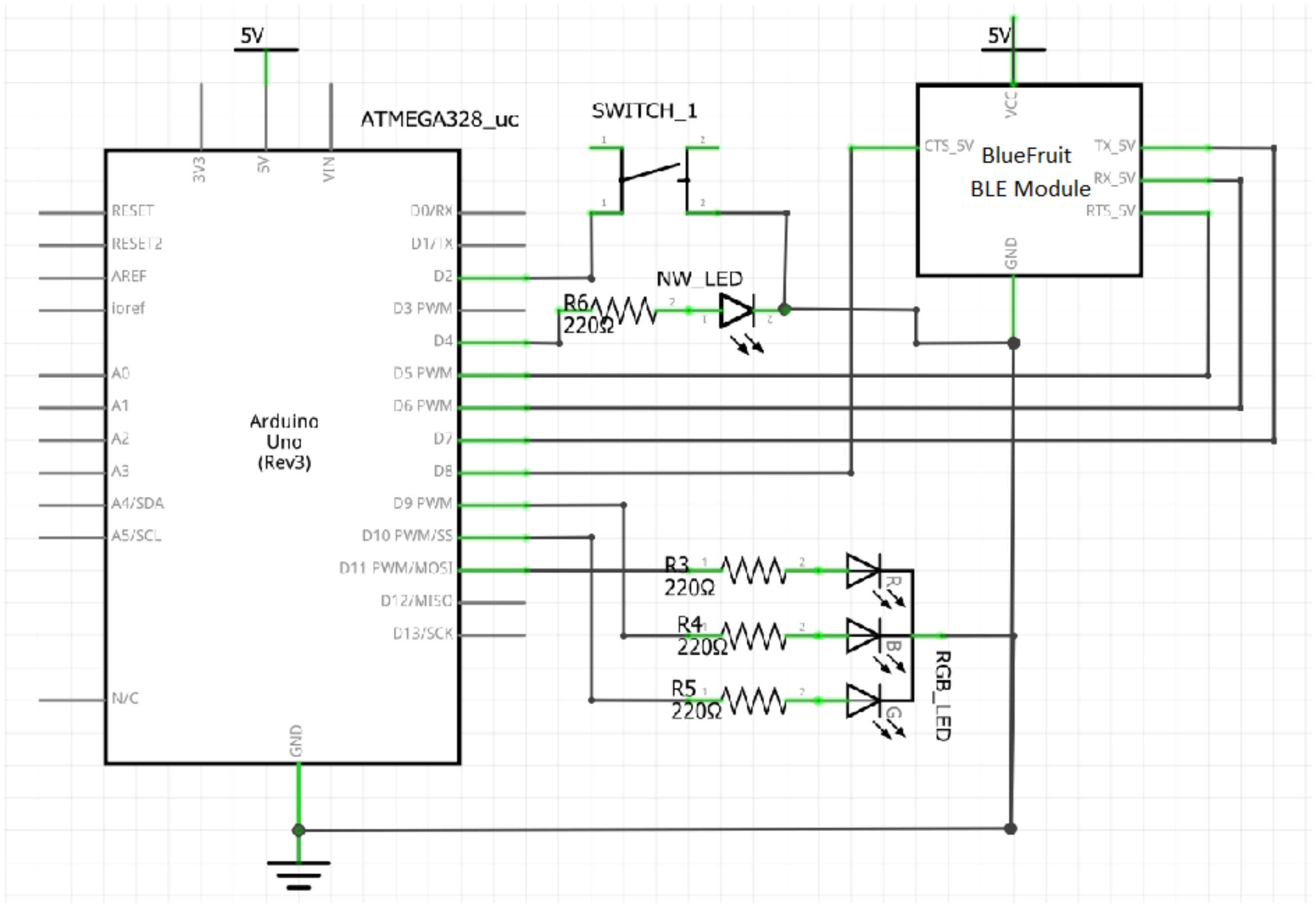


Fig 2.2: Circuit (Schematic) diagram of Dash board

2.4 Connections of Modules: PORT-D :

13	12	11	10	9	8	7	6	5	4	3	2	1
		R-Led	G-Led	B-Led	BLE_ CTS	BLE_ TXD	BLE_ RXD	BLE_ RTS	NW_ LED		SW-1	

Fig. 2.3: Port Map for BLE controlled Unit

1) RGB LED

Arduino pins 9, 10, 11 are connected to RGB LED +ve(anode) pins through 220E resister, and Arduino GND pin is connected to LED -ve(cathode) pin. Arduino pins 4 is connected to NWLED +ve(anode).

2) Push Button

One end of switch is connected to Arduino 2 pin and other end is connected to Arduino GND pin.

3) BlueFruit BLE module :

Arduino pins 5, 6, 7 and 8 are connected to RTS, RX, TX, and CTS pins of BlueFruit BLE module.

5. SOFTWARE IMPLEMENTATION OF DASH BOARD

5.1 SOFTWARE TOOLS

For programming the microcontroller, the Arduino project provides an **integrated development environment** (**IDE**) based on the Processing project, which includes support for the **C** and C++ programming languages, I used embedded C code is used for project implementation. The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform.

A program written with the IDE for Arduino is called a "sketch". A typical Arduino C/C++ sketch consist of two functions that are compiled and linked with a program stub main() into an executable cyclic executive program:

- Setup() : a function that runs once at the start of a program and that can initialize settings.
- Loop() : a function called repeatedly until the board powers off.

The system initialisation is done in **setup()** function :

Hardware Initialisation :

- Set PIN 2 as INPUT pin which is connected to push button named PWR_SWITCH and attached the OnOffSwitchPushed ISR to the gpio pin 2.
- Set pin 11, pin 10 and pin 9 are OUTPUT pins which are connected to the RGB led +ve(anode) pins 1, 3 and 4 through 220E resisters respectively. Pin 4 as OUTPUT this is connected RSSI indication (n/w status) led and set the pin LOW.
- Initialised the h/w serial port with 115200 baudrate for debugging.

BLE Module Initialisation :

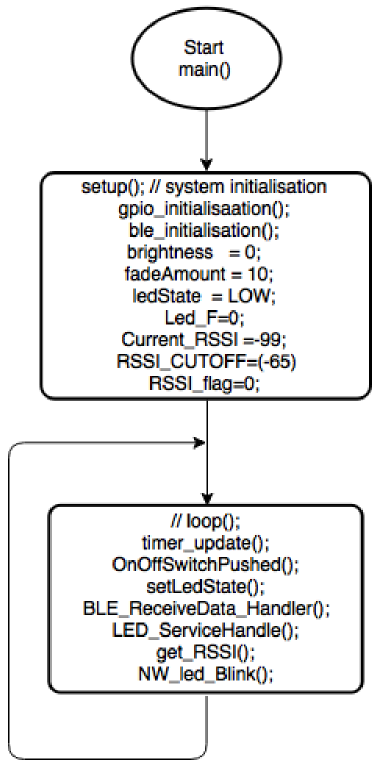
- Defined the pins 5, 6, 7 and 8 for software serial port for communicate with BLE module.
- Checked for the ble connection, if connected and if the FACTORYRESET_ENABLE is set performed the factory reset. Disabled the echo data from Bluefruit using ble.echo() function.
- Set the Device Name (for broadcast in the peripheral's advertising packet).
- Created LED custom service with two characteristics: create_LED_custom_GATT_Service()
    1. LED status : read/notify properties with UUID: 0x0002, called "State" which can notify us of changes in the status(0 : off; 1 : on) of the switch.
    2. LED control : write property with UUID: 0x0003, called "Switch" which is used to flip the switch on and off (0 : turn off; 1 : turn on).
- Wait for the central device to connect with the ble module. If connected send the LED activity command set module to command mode.

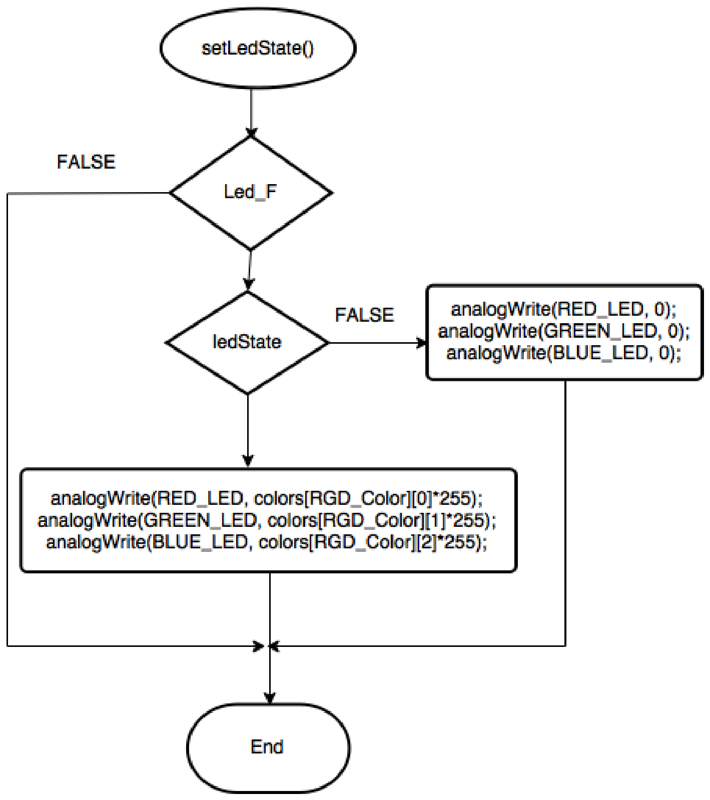
The working of circuit is implemented in the **loop()** function :

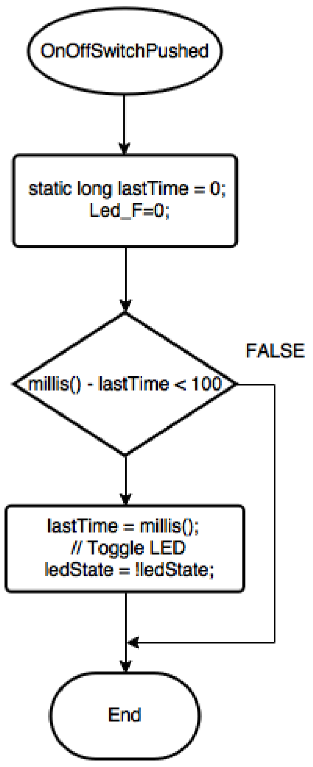
- setLedState() : Checks the brightness flag(Led_F), if set exit from the function else check the led status flag (ledState) if set generates the analog voltages on pins 11, 10 and 9 conrresponding to the colors[21]*255 value this valur is set by Color_Change(flag) function (flag=1: Increment; flag=0: Decrement order). If ledState is Low, led is off.
- OnOffSwitchPushed(): If we press the button this ISR function toggles the ledState flag(0/ 1) .
- Brightness_Change(flag): Checks the led status flag (ledState), if ledState=0, exit from the function else increase or decrease the led brightness(flag=1: Increment; flag=0: Decrement).
- get_RSSI(): Checks the ble connection if not connected sets the RSSI_flag=0, Current_RSSI= -99; and retur from the function, if connected gets the rssi and saves it in Current_RSSI variable and calls the NW_Blink() function which is used to indicate the RSSI of conneced device.
- NW_Blink() : Checks the whether the Current_RSSI is less than RSSI_CUTOFF or not, if less set the RSSI_flag =0, and blinks the RSSI_LED multiple times, else set the RSSI_flag=1, and blinks the RSSI_LED single time i.e. the signal is sufficient.
- Read_Write_GATTcharacteristic(): Used to read the characteristic values and write the characteristic values to the ble module. This function takes i_characteristic and i_rw(0:read, 1:write) as input parameters and o_vlaue as in/out parameter.
- LED_ServiceHandle(): Checks the RSSI_flag, if RSSI_flag=0, exit from the function, else send the notification to the connected device regarding the led status(0 : off; 1 : on) and get the "Switch" status, which is used to update the ledState flag (0 : turn off; 1 : turn on).
- BLE_ReceiveData_Handler(): Checks the RSSI_flag, if RSSI_flag=0, exit from the function, else checks whether any data is received from central device if received performs the correspoding functionalities i.e flip the led on/off, change the colour and change brightness.

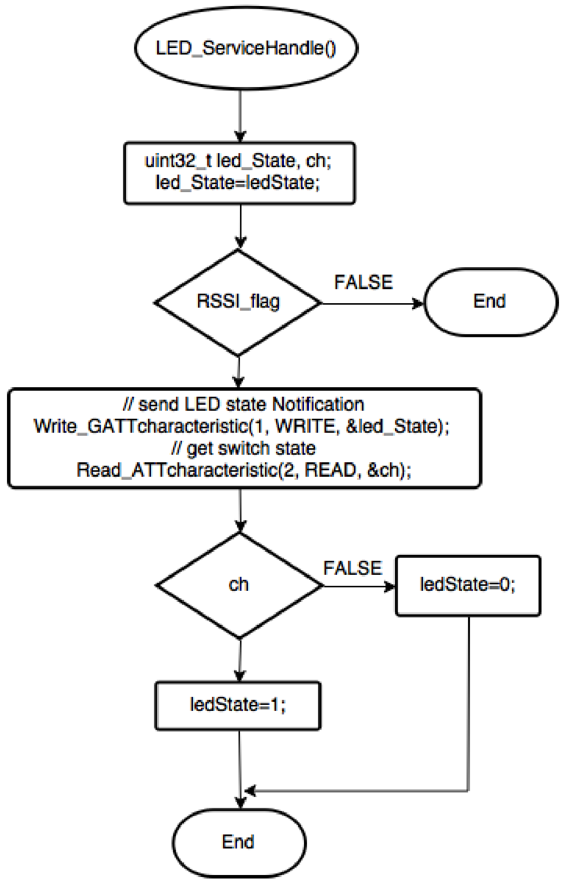
Testing Bluetooth Services:

- Compiled and uploaded the sketch to the UNO board.
- Now we can turn the LED on and off by using the button. Open the Serial Console to see the debug messages when we push the button to turn the LED on/off.
- By running 'nRF connect' app on smartphone we can get notifications regarding the status of light i.e. whether the led is on or off, and send a commands to Arduino board over Bluetooth to toggle the LED.
- I created 5 buttons(SW:On/off; LF/RT: change color; UP/DN: change brightness) in nRF toolbox app (UART) to control LED. Using this we can turn the LED on/off, change the colour, and chane the brightness of the RGB led.


Start
main()
setup(); // system initialisation
gpio_initialisaation();
ble_initialisation();
brightness   = 0;
fadeAmount = 10;
ledState = LOW;
Led_F=0;
Current_RSSI =-99;
RSSI_CUTOFF=(-65)
RSSI_flag=0;
// loop();
timer_update();
OnOffSwitchPushed();
setLedState();
BLE_ReceiveData_Handler();
LED_ServiceHandle();
get_RSSI();
NW_led_Blink();


setLedState()
FALSE
Led_F
ledState
FALSE
analogWrite(RED_LED, 0);
analogWrite(GREEN_LED, 0);
analogWrite(BLUE_LED, 0);
analogWrite(RED_LED, colors[RGD_Color][0]*255);
analogWrite(GREEN_LED, colors[RGD_Color][1]*255);
analogWrite(BLUE_LED, colors[RGD_Color][2]*255);
End


OnOffSwitchPushed
static long lastTime = 0;
Led_F=0;
millis() - lastTime < 100
FALSE
lastTime = millis();
// Toggle LED
ledState = !ledState;
End


LED_ServiceHandle()
uint32_t led_State, ch;
led_State=ledState;
RSSI_flag
FALSE
End
// send LED state Notification
Write_GATTcharacteristic(1, WRITE, &led_State);
// get switch state
Read_ATTcharacteristic(2, READ, &ch);
ch
FALSE
ledState=0;
ledState=1;
End

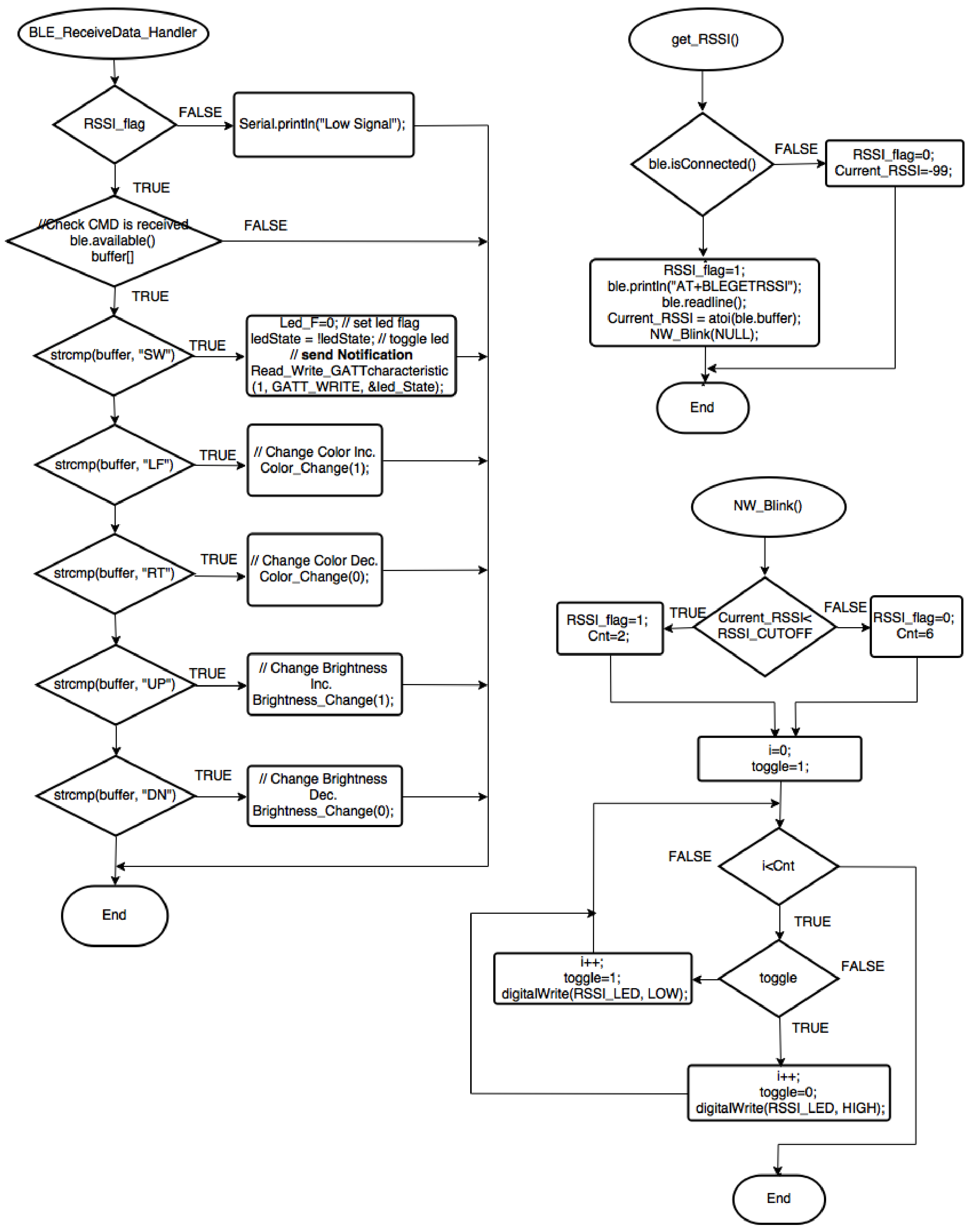
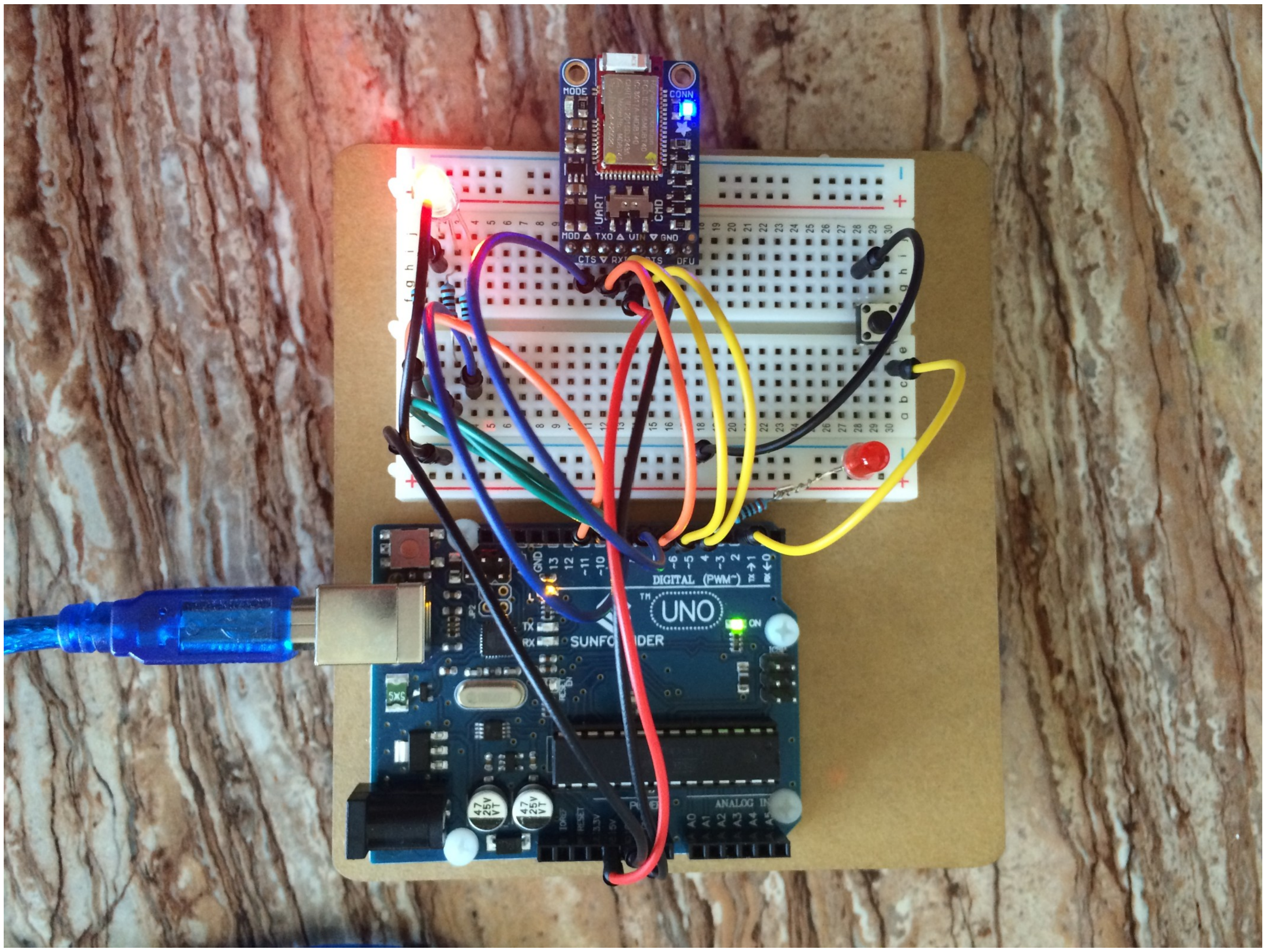


Fig. Flow Chart for Communication Unit

6. Appendix

- **Sketch URL :** https://codebender.cc/sketch:**357793**
- Youtube video clip : **https://youtu.be/Nw2sT6Ii_4Y**

Hardware setup :



Serial console output of One Complete session Run which demonstrate all the features of system :

```
Warning! Because of a known bug on Chrome, you should press the disconnect button before physically disconnecting
your device.Or else the connection with the serial monitor will stay open until you restart your browser.

***** Initialising BLE Module ***
ATZ
<- OK
ATE=0
<- OK
Setting Dev Name:
AT+GAPDEVNAME=NAGARAJU
<- OK
Requesting BLE info:
----------------
BLEFRIEND32
nRF51822 QFACA10
CFCDB6766551E03C
0.6.7
0.6.7
Sep 17 2015
S110 8.0.0, 0.2
----------------
Creating GATT Service
AT+GATTCLEAR
<- OK
AT+GATTADDSERVICE=UUID128=cf-4d-de-38-5b-65-11-e6-8b-77-86-f3-0c-a8-93-d3
<- 1
OK
AT+GATTADDCHAR=UUID=0x0003,PROPERTIES=0x12,MIN_LEN=1,VALUE=2
<- 1
OK
AT+GATTADDCHAR=UUID=0x0002,PROPERTIES=0x08,MIN_LEN=1,VALUE=0
<- 2
OK
AT+GATTLIST
<- ID=01,UUID=0xDE38,UUID128=CF-4D-DE-38-5B-65-11-E6-8B-77-86-F3-0C-A8-93-D3
  ID=01,UUID=0x0003,PROPERTIES=0x12,MIN_LEN=1,MAX_LEN=1,VALUE=0x02
  ID=02,UUID=0x0002,PROPERTIES=0x08,MIN_LEN=1,MAX_LEN=1,VALUE=0x00
OK
ATZ
<- OK
Waiting for connection!
Toggle LED
Toggle LED
Toggle LED
Toggle LED
Change LED activity MODE
Switching to CMD mode!
*** Initialising BLE OK! ***

Controlling led:1
Controlling led:0
Controlling led:1
Controlling led:0
Controlling led:1
Controlling led:0
 Change Color Dec.
Power ON LED
 Change Color Dec.
Power ON LED
Controlling led:1
 Change Color Inc.
 Change Color Inc.
 Change Color Inc.
 Change Color Inc.
 Change Color Inc.
 Change Color Inc.
 Change Color Inc.
 Change Color Dec.
 Change Color Dec.
 Change Color Dec.
 Change Color Dec.
 Change Color Dec.
 Change Color Dec.
 Change Brightness Inc. : 0
 Change Brightness Inc. : 60
 Change Brightness Inc. : 70
 Change Brightness Inc. : 80
 Change Brightness Inc. : 90
 Change Brightness Inc. : 100
 Change Brightness Inc. : 110
 Change Brightness Inc. : 120
 Change Brightness Dec. : 110
 Change Brightness Dec. : 100
 Change Brightness Dec. : 90
 Change Brightness Dec. : 80
 Change Brightness Dec. : 70
 Change Brightness Dec. : 60
 Change Brightness Dec. : 50
Controlling led:0
Controlling led:1
Controlling led:0
Controlling led:1
Controlling led:0
Controlling led:1
Toggle LED
Toggle LED
Toggle LED
Toggle LED
Toggle LED
Toggle LED
Toggle LED
Toggle LED
Toggle LED
```