```
#include<TonePlayer.h>
#include <timer.h>
#include <SPI.h>
#include<nRF24L01.h>
#include <RF24.h>
//
TonePlayer tone1 (TCCR1A, TCCR1B, OCR1AH, OCR1AL,
TCNT1H, TCNT1L);

#define CE_PIN   7
#define CSN_PIN 8

unsigned long counterStart = 0;
unsigned long currentMillis;
unsigned long singalComein;
bool counterRunning = false;
const int ledPin =
10;
  // LED - digital 10
const int buzzer = 9;
boolean ledState =
LOW;                                  // ledState to
store the state of LED
const int red_light_pin= 4;
const int green_light_pin = 5;
const int blue_light_pin = 6;

const byte address[6] = "00001";

char dataReceived[32] = "";
char result[32];
const char sender[] = "singal comeIn";
```

```
const char starter[] = "count start";
const char beginer[] = "counters Begin";
const char restarter[] = "counter Restart";
const char reBeginer[] = "counters ReBegin";
const char snoozer[] = "snooze";

//char dataReceived[10]; // this must match data
received from sender
int ackData[2] = {109, -4000}; // the two values to
be sent to the sender
char repData[] = "Times up";
char repData2[] = "Times up Again";
char repData3[] = "counter Rebegin";
char nolData[] = "singal respose";

unsigned long totalTimer = 60000;
unsigned long stage_01 = 40000;
unsigned long stage_02 = 55000;

RF24 radio(CE_PIN, CSN_PIN); // CE, CSN



void setup(){

Serial.begin(9600);
Serial.println("Receiver Starting");

radio.begin();
radio.setDataRate( RF24_250KBPS );
radio.openReadingPipe(1, address);
radio.enableAckPayload();
radio.startListening();
```

```
radio.writeAckPayload(1, &ackData,
sizeof(ackData)); // pre-load data


pinMode(ledPin,OUTPUT);
// define LED as output signal
pinMode(buzzer,OUTPUT);
pinMode(red_light_pin, OUTPUT);
pinMode(green_light_pin, OUTPUT);
pinMode(blue_light_pin, OUTPUT);
//counterRunning = true;
digitalWrite(ledPin,ledState);
}

void loop() {
  if(counterRunning && (millis() - counterStart >=
0) && (millis() - counterStart < stage_01)){
    RGB_color(0,255,0);
     }
  if(counterRunning && (millis() - counterStart >=
stage_01) && (millis() - counterStart < stage_02)){
    RGB_color(255,255,0);
     }
  if(counterRunning && (millis() - counterStart >=
stage_02) && (millis() - counterStart <
totalTimer)){
    RGB_color(255,0,0);
     }

  if (radio.available()) {
   radio.read(&dataReceived, sizeof(dataReceived));
    strcpy(result, dataReceived);
```

```
    singalComein  = millis();
    Serial.print("singal recieved: ");
    Serial.println(result);
    Serial.println(singalComein);
    radio.writeAckPayload(1, &nolData,
sizeof(nolData));
    Serial.print(" nolData sent ");
    Serial.println(nolData);


  //once receive code from power button, the state
of LED is changed from HIGH to LOW or from LOW to
HIGH.
  if((counterRunning == true) && (strcmp(result,
sender) == 0) && (millis() - counterStart >=
totalTimer)){
    ledState = HIGH;
    digitalWrite(ledPin,ledState);
    tone1.tone(3000);
    currentMillis = millis();
    for(int i = 0; i < 100 ; i++){
      radio.writeAckPayload(1, &repData,
sizeof(repData)); // load the payload for the next
time
      Serial.print(" repData sent ");
      Serial.println(repData);
       }
    counterRunning = false;
    }

  if(counterRunning == false && strcmp(result,
starter) == 0){
```

```
      counterRunning = true;
      counterStart = millis();
     }
  if(counterRunning == false && strcmp(result,
reBeginer) == 0){
      for(int i = 0; i < 100 ; i++){
       radio.writeAckPayload(1, &repData3,
sizeof(repData3)); // load the payload for the next
time
       Serial.print(" repData sent ");
       Serial.println(repData3);
       radio.read(&dataReceived,
sizeof(dataReceived));
       strcpy(result, dataReceived);
       }
     }
  if(strcmp(result, restarter) == 0){
    ledState = HIGH;
   digitalWrite(ledPin,ledState);
    tone1.tone(3000);
    currentMillis = millis();
    counterRunning = false;
    for(int i = 0; i < 100 ; i++){
   radio.writeAckPayload(1, &repData2,
sizeof(repData2)); // load the payload for the next
time
    Serial.print(" repData sent ");
   Serial.println(repData2);
    }



    }
```

```
  }
 if(senderOutOfRange(singalComein) &&
counterRunning == false){
    ledState =
LOW;                                    //reverse
   digitalWrite(ledPin,
ledState);                          //change the state
of LED
    tone1.noTone();
    counterStart = millis();
     }
  if((strcmp(result, snoozer) == 0) &&
counterRunning == false){
    ledState =
LOW;                                    //reverse
   digitalWrite(ledPin,
ledState);                          //change the state
of LED
    tone1.noTone();
     }


}
void RGB_color(int red_light_value, int
green_light_value, int blue_light_value)
 {
 analogWrite(red_light_pin, red_light_value);
 analogWrite(green_light_pin, green_light_value);
 analogWrite(blue_light_pin, blue_light_value);
}
```

```
bool senderOutOfRange(unsigned long singalComein){
  if(millis() - singalComein >= 5000){
    return true;
    }else{
      return false;
       }
}
void showData() {
  Serial.print("Data received ");
 Serial.println(dataReceived);
  Serial.print(" ackPayload sent ");
 Serial.print(ackData[0]);
  Serial.print(", ");
 Serial.println(ackData[1]);
}
```