Dear Sir, Greetings!
By profession, I am a teacher; teaching in the School of Computing of Trident Academy of Technology.
I am facing problem as described below in detail.
Kindly extend your help please
With sincere regards, I am eagerly looking forward to your valuable response and constructive suggestions please,
A.K.Samal

Configuration

```
Hardware:    ESP8266 12E NodeMCU Amica
Software:    Arduino IDE v1.8.19
Library:     Mike McCauley AirSpayce RadioHead
             Packet Radio library for embedded microprocessors
             Version: 1.130
             Library Module: RH_RF95.h, RHRouter.h, RHMesh.h
LoRa Module: SX1278, RFM98, 433 MHz, HopeRF
             SX1276, RFM95, 866 MHz, HopeRF
             SX1278, Ra-02, 433 MHz, AI Thinker
             SX1262, 866 MHz, Waveshare
```

NodeMCU to RFM95 Interface

```
LoRa RFM95    ESP8266 NodeMCU
  3.3V-----------3.3V
  Gnd------------Gnd
  RST------------RST
  En/Nss---------GPIO2/D4
  MOSI-----------GPIO13/D7
  MISO-----------GPIO12/D6
  SCK------------GPIO14/D5
  G0/DIO0--------GPIO15/D8
```

This mode of interface is already tested with different programs using HopeRF's SX1278 RFM98 (433 MHz) and SX1276 RFM95 (866 MHz) and Waveshare's SX1262 (866 MHz)

Error Report
Facing runtime error as presented below

Program
```
#include <SPI.h>
#include <Wire.h>
#include <Arduino.h>
#include <RHRouter.h>
#include <RHMesh.h>
#include <RH_RF95.h>

#define RFM95_CS  2    // NSS
#define RFM95_INT 15     // DIO0
#define RF95_FREQ 866.0
#define RH_HAVE_SERIAL
#define N_NODES 4 // Number of Nodes in the Mesh Network
#define RH_MESH_MAX_MESSAGE_LEN 50

uint8_t nodeId;
uint8_t routes[N_NODES]; // full routing table for mesh
int16_t rssi[N_NODES]; // signal strength info

// Singleton instance of the radio driver
RH_RF95 rf95(RFM95_CS, RFM95_INT);

// Class to manage message delivery and receipt, using the driver declared above
RHMesh *manager;

// message buffer
char buf[RH_MESH_MAX_MESSAGE_LEN];

void setup()
{
  randomSeed(analogRead(0));

  for(uint8_t n=1; n<=N_NODES; n++)
  {
    routes[n-1] = 0;
    rssi[n-1] = 0;
  }
```

```
  Serial.begin(9600);
  while (!Serial) ; // Wait for serial port to be available

  nodeId = 2;
  if (nodeId > 10)
  {
    Serial.print(F("Invalid nodeId: "));
    Serial.println(nodeId);
    nodeId = 1;
  }
  Serial.print(F("initializing node "));

  manager = new RHMesh(rf95, nodeId);

  if (!manager->init())
  {
    Serial.println(F("init failed"));
  }
  else
  {
    Serial.println("done");
  }
  rf95.setTxPower(23, false);
  rf95.setFrequency(RF95_FREQ);
  rf95.setCADTimeout(500);

  // Possible configurations:
  // Bw125Cr45Sf128 (the chip default)
  // Bw500Cr45Sf128
  // Bw31_25Cr48Sf512
  // Bw125Cr48Sf4096

  // long range configuration requires for on-air time
  boolean longRange = false;
  if (longRange)
  {
    RH_RF95::ModemConfig modem_config = {
      0x78, // Reg 0x1D: BW=125kHz, Coding=4/8, Header=explicit
      0xC4, // Reg 0x1E: Spread=4096chips/symbol, CRC=enable
      0x08  // Reg 0x26: LowDataRate=On, Agc=Off.  0x0C is LowDataRate=ON, ACG=ON
    };
    rf95.setModemRegisters(&modem_config);
    if (!rf95.setModemConfig(RH_RF95::Bw125Cr48Sf4096))
    {
      Serial.println(F("set config failed"));
    }
  }
  Serial.println("RF95 ready");
}

const __FlashStringHelper* getErrorString(uint8_t error)
{
  switch(error) {
    case 1: return F("invalid length");
    break;
    case 2: return F("no route");
    break;
    case 3: return F("timeout");
    break;
    case 4: return F("no reply");
    break;
    case 5: return F("unable to deliver");
    break;
  }
  return F("unknown");
}

void updateRoutingTable()
{
  for(uint8_t n=1; n<=N_NODES; n++)
  {
    RHRouter::RoutingTableEntry *route = manager->getRouteTo(n);
    if (n == nodeId)
    {
      Serial.println(F("Test Point->A"));
      routes[n-1] = 255; // self
    }
    else
    {
      Serial.println(F("Test Point->B"));
      routes[n-1] = route->next_hop;
```

```
      Serial.println(F("Test Point->C"));
      if (routes[n-1] == 0)
      {
        // if we have no route to the node, reset the received signal strength
        rssi[n-1] = 0;
      }
    }
  }
}

// Create a JSON string with the routing info to each node
void getRouteInfoString(char *p, size_t len)
{
  p[0] = '\0';
  strcat(p, "[");
  for(uint8_t n=1;n<=N_NODES;n++)
  {
    strcat(p, "{\"n\":");
    sprintf(p+strlen(p), "%d", routes[n-1]);
    strcat(p, ",");
    strcat(p, "\"r\":");
    sprintf(p+strlen(p), "%d", rssi[n-1]);
    strcat(p, "}");
    if (n<N_NODES) {
      strcat(p, ",");
    }
  }
  strcat(p, "]");
}

void printNodeInfo(uint8_t node, char *s)
{
  Serial.print(F("node: "));
  Serial.print(F("{"));
  Serial.print(F("\""));
  Serial.print(node);
  Serial.print(F("\""));
  Serial.print(F(": "));
  Serial.print(s);
  Serial.println(F("}"));
}

void loop()
{
  //Serial.println(F("Test Point->A"));
  for(uint8_t n=1;n<=N_NODES;n++)
  {
    if (n == nodeId) continue; // self

    updateRoutingTable();
    getRouteInfoString(buf, RH_MESH_MAX_MESSAGE_LEN);

    Serial.print(F("->"));
    Serial.print(n);
    Serial.print(F(" :"));
    Serial.print(buf);

    // send an acknowledged message to the target node
    uint8_t error = manager->sendtoWait((uint8_t *)buf, strlen(buf), n);
    if (error != RH_ROUTER_ERROR_NONE)
    {
      Serial.println();
      Serial.print(F(" ! "));
      Serial.println(getErrorString(error));
    }
    else
    {
      Serial.println(F(" OK"));
      // we received an acknowledgement from the next hop for the node we tried to send to.
      RHRouter::RoutingTableEntry *route = manager->getRouteTo(n);
      if (route->next_hop != 0)
      {
        rssi[route->next_hop-1] = rf95.lastRssi();
      }
    }
    if (nodeId == 1) printNodeInfo(nodeId, buf); // debugging

    // listen for incoming messages. Wait a random amount of time before we transmit
    // again to the next node
    unsigned long nextTransmit = millis() + random(3000, 5000);
    while (nextTransmit > millis())
    {
```

```
      int waitTime = nextTransmit - millis();
      uint8_t len = sizeof(buf);
      uint8_t from;
      if (manager->recvfromAckTimeout((uint8_t *)buf, &len, waitTime, &from))
      {
        buf[len] = '\0'; // null terminate string
        Serial.print(from);
        Serial.print(F("->"));
        Serial.print(F(" :"));
        Serial.println(buf);
        if (nodeId == 1) printNodeInfo(from, buf); // debugging
        // we received data from node 'from', but it may have actually come from an intermediate node
        RHRouter::RoutingTableEntry *route = manager->getRouteTo(from);
        if (route->next_hop != 0)
        {
          rssi[route->next_hop-1] = rf95.lastRssi();
        }
      }
    }
  }
}
```

Compilation Output:

```
F:\00-AKSam-MQTT-InfluxDB-Grafana-ESP-RPi\Mesh_Dev-18-06-2024\00-Nootropic-LoRa-Mesh-Network\LoRaMesh\LoRaMesh.ino:13:
warning: "RH_MESH_MAX_MESSAGE_LEN" redefined
   13 | #define RH_MESH_MAX_MESSAGE_LEN 50
      |
In file included from F:\00-AKSam-MQTT-InfluxDB-Grafana-ESP-RPi\Mesh_Dev-18-06-2024\00-Nootropic-LoRa-Mesh-
Network\LoRaMesh\LoRaMesh.ino:5:
C:\Users\Abhaya Kumar\Documents\Arduino\libraries\RadioHead/RHMesh.h:120: note: this is the location of the previous
definition
  120 |     #define RH_MESH_MAX_MESSAGE_LEN (RH_ROUTER_MAX_MESSAGE_LEN - sizeof(RHMesh::MeshMessageHeader))
      |
. Variables and constants in RAM (global, static), used 29924 / 80192 bytes (37%)
╠   SEGMENT  BYTES    DESCRIPTION
╠══ DATA     1496     initialized variables
╠══ RODATA   1012     constants
╚══ BSS      27416    zeroed variables
. Instruction RAM (IRAM_ATTR, ICACHE_RAM_ATTR), used 61095 / 65536 bytes (93%)
╠   SEGMENT  BYTES    DESCRIPTION
╠══ ICACHE   32768    reserved space for flash instruction cache
╚══ IRAM     28327    code in IRAM
. Code in flash (default, ICACHE_FLASH_ATTR), used 248212 / 1048576 bytes (23%)
╠   SEGMENT  BYTES    DESCRIPTION
╚══ IROM     248212   code in flash
esptool.py v3.0
Serial port COM23
Connecting....
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: b4:8a:0a:f2:63:fd
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 283200 bytes to 207351...
Writing at 0x00000000... (7 %)
Writing at 0x00004000... (15 %)
Writing at 0x00008000... (23 %)
Writing at 0x0000c000... (30 %)
Writing at 0x00010000... (38 %)
Writing at 0x00014000... (46 %)
Writing at 0x00018000... (53 %)
Writing at 0x0001c000... (61 %)
Writing at 0x00020000... (69 %)
Writing at 0x00024000... (76 %)
Writing at 0x00028000... (84 %)
Writing at 0x0002c000... (92 %)
Writing at 0x00030000... (100 %)
Wrote 283200 bytes (207351 compressed) at 0x00000000 in 18.5 seconds (effective 122.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Serial Terminal Execution Output

-

```
ʕʔ▯ʔrʕzʔʔʔ▯ʔBʕinitializing node done
RF95 ready
Test Point->B

--------------- CUT HERE FOR EXCEPTION DECODER ---------------

Exception (28):
epc1=0x40201191 epc2=0x00000000 epc3=0x00000000 excvaddr=0x00000001 depc=0x00000000

>>>stack>>>

ctx: cont
sp: 3ffffdf0 end: 3fffffd0 offset: 0150
3fffff40:  00000000 00000001 00000002 401003fc
3fffff50:  000f4240 00000083 3ffee570 3ffeeaa4
3fffff60:  00000001 3ffee538 3ffeea78 4020134e
3fffff70:  4010012e 01a59480 3ffee570 40204da0
3fffff80:  3ffe87fa ffffffc 3ffee570 40202d98
3fffff90:  402033ec 3ffe87f8 3ffeea20 3ffeeaa4
3fffffa0:  3fffdad0 0000000a 3ffeea20 3ffeeaa4
3fffffb0:  3fffdad0 00000000 3ffeea78 40203e3c
3fffffc0:  feefeffe feefeffe 3fffdab0 40100f39
<<<stack<<<

--------------- CUT HERE FOR EXCEPTION DECODER ---------------
```

Espressif Exception Decoder Output

Exception 28: LoadProhibited: A load referenced a page mapped with an attribute that does not permit loads
PC: 0x40201191
EXCVADDR: 0x00000001

Decoding stack results
0x401003fc: **__digitalWrite(uint8_t, uint8_t)** at C:\Users\Abhaya Kumar\AppData\Local\Arduino15\packages\esp8266\hardware\esp8266\3.1.2\cores\esp8266**core_esp8266_wiring_digital.cpp** line **87**
0x4020134e: **loop()** at F:\00-AKSam-MQTT-InfluxDB-Grafana-ESP-RPi\Mesh_Dev-18-06-2024\00-Nootropic-LoRa-Mesh-Network\LoRaMesh/**LoRaMesh.ino** line **169**
0x4010012e: **RHSPIDriver::spiWrite(unsigned char, unsigned char)** at C:\Users\Abhaya Kumar\Documents\Arduino\libraries\RadioHead**RHSPIDriver.cpp** line **69**
0x40204da0: **uart_write(uart_t*, char const*, size_t)** at C:\Users\Abhaya Kumar\AppData\Local\Arduino15\packages\esp8266\hardware\esp8266\3.1.2\cores\esp8266**uart.cpp** line **547**
0x40202d98: **RH_RF95::setFrequency(float)** at C:\Users\Abhaya Kumar\Documents\Arduino\libraries\RadioHead**RH_RF95.cpp** line **396**
0x402033ec: **HardwareSerial::write(unsigned char const*, unsigned int)** at C:\Users\Abhaya Kumar\AppData\Local\Arduino15\packages\esp8266\hardware\esp8266\3.1.2\cores\esp8266/**HardwareSerial.h** line **193**
0x40203e3c: **loop_wrapper()** at C:\Users\Abhaya Kumar\AppData\Local\Arduino15\packages\esp8266\hardware\esp8266\3.1.2\cores\esp8266**core_esp8266_main.cpp** line **258**