Lora Mesh

v1.0.0

Generated by Doxygen 1.10.0

1 Hierarchical Index 1

1.1 Class Hierarchy . . . 1

2 Class Index 3

2.1 Class List . . . 3

3 File Index 5

3.1 File List . . . 5

4 Class Documentation 7

4.1 LoRaMesh Class Reference . . . 7

4.1.1 Detailed Description . . . 8

4.1.2 Member Function Documentation . . . 8

4.1.2.1 createAckPacket() . . . 8

4.1.2.2 createMessagePacket() . . . 9

4.1.2.3 debug() . . . 9

4.1.2.4 debugPrint() . . . 10

4.1.2.5 debugPrintln() [1/2] . . . 10

4.1.2.6 debugPrintln() [2/2] . . . 10

4.1.2.7 getmyID() . . . 10

4.1.2.8 loraSetup() . . . 11

4.1.2.9 meshRead() . . . 11

4.1.2.10 meshSend() . . . 11

4.1.2.11 printRoutingTable() . . . 11

4.1.2.12 readAckCallback() . . . 12

4.1.2.13 readInt() . . . 12

4.1.2.14 readMsgCallback() . . . 12

4.1.2.15 routingPacket() . . . 12

4.1.2.16 updateRoutingTable() . . . 13

4.1.2.17 writeInt() . . . 13

4.1.3 Member Data Documentation . . . 13

4.1.3.1 callbackAckFunction . . . 13

4.1.3.2 callbackDataFunction . . . 13

4.1.3.3 debugging . . . 14

4.1.3.4 knownRoutes . . . 14

4.1.3.5 myID . . . 14

4.1.3.6 neighbours . . . 14

4.1.3.7 payload . . . 14

4.1.3.8 receivedMsg . . . 14

4.1.3.9 routingTable . . . 15

4.1.3.10 typeAckMessage . . . 15

4.1.3.11 typeDataMessage . . . 15

4.1.3.12 typeRoutingMessage . . . 15

4.2 LoRaMesh::packet Struct Reference 15
4.2.1 Detailed Description 16
4.2.2 Member Data Documentation 16
4.2.2.1 destination 16
4.2.2.2 msgId 16
4.2.2.3 nextHop 16
4.2.2.4 packetSize 16
4.2.2.5 payload 17
4.2.2.6 preamble 17
4.2.2.7 source 17
4.2.2.8 type 17

5 File Documentation **19**
5.1 LoRaMesh.cpp 19
5.2 LoRaMesh.h 23

Index **25**

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

LoRaClass
    LoRaMesh 7
LoRaMesh::packet 15

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

LoRaMesh
: Mesh network using LoRa communication 7

LoRaMesh::packet
: Structure representing a packet in the LoRaMesh network 15

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

LoRaMesh.cpp 19
LoRaMesh.h 23

Chapter 4

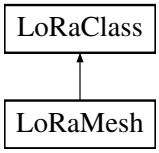
Class Documentation

4.1 LoRaMesh Class Reference

The LoRaMesh class represents a mesh network using LoRa communication.

`#include <LoRaMesh.h>`

Inheritance diagram for LoRaMesh:



Classes

- struct packet

  Structure representing a packet in the LoRaMesh network.

Public Member Functions

- int loraSetup (int myID, int ssPin, int rst, int dio0)
- unsigned short int readInt ()
- void writeInt (unsigned short int num)
- void debug (packet p)
- void meshRead ()
- packet createMessagePacket (String msg, int idToSend, int msgId, int packetType=0)
- packet createAckPacket (int sizeOfMsgToAck, int idToSend, int msgIdToAck)
- void meshSend (packet sendPacket)
- int getmyID ()
- void routingPacket ()
- void updateRoutingTable (int id, unsigned short int size)
- void printRoutingTable ()
- template<typename T >
  void debugPrint (T input)
- template<typename T >
  void debugPrintln (T input)
- void debugPrintln ()
- void readMsgCallback (std::function< void(int, int, String)> callbackData)
- void readAckCallback (std::function< void(int, int)> callbackAck)

Public Attributes

- byte typeDataMessage = 0
- byte typeRoutingMessage = 1
- byte typeAckMessage = 6
- String receivedMsg
- char payload [244]
- bool debugging = true
- std::map< unsigned short int, std::pair< unsigned short int, unsigned short int > > routingTable
- std::map< unsigned short int, unsigned short int > neighbours
- std::vector< unsigned short int > knownRoutes

Private Attributes

- unsigned short int myID
- std::function< void(int, int, String)> callbackDataFunction
- std::function< void(int, int)> callbackAckFunction

4.1.1 Detailed Description

The LoRaMesh class represents a mesh network using LoRa communication.

This class extends the LoRaClass and provides additional functionality for creating and managing a mesh network. It includes methods for setting up the LoRa module, sending and receiving packets, managing routing tables, and debugging.

The LoRaMesh class also defines a packet structure for sending and receiving messages within the mesh network. It includes fields for the source, destination, next hop, packet size, message ID, type, preamble, and payload.

The class also provides callback functions for handling received data messages and acknowledgment messages.

Note

This class assumes the use of the LoRa library by Sandeep Mistry (https://github.com/sandeepmistry/arduino-LoRa) and requires the LoRa module to be properly configured.

Definition at line 24 of file LoRaMesh.h.

4.1.2 Member Function Documentation

4.1.2.1 createAckPacket()

```
LoRaMesh::packet LoRaMesh::createAckPacket (
            int sizeOfMsgToAck,
            int idToSend,
            int msgIdToAck )
```

A function to create an acknowledgment packet based on the message packet

Parameters

in	*sizeOfMsgToAck*	The size of the message to be acknowledged
in	*idToSend*	The recipient ID
in	*msgIdToAck*	The message ID to be acknowledged

Returns

A packet

Definition at line 477 of file LoRaMesh.cpp.

4.1.2.2 createMessagePacket()

```
LoRaMesh::packet LoRaMesh::createMessagePacket (
            String msg,
            int idToSend,
            int msgId,
            int packetType = 0 )
```

A function create a new packet

Parameters

in	*msg*	A string containing the message
in	*idToSend*	The recipient ID
in	*msgId*	The message ID
in	*packetType*	The type of the packet being sent

Returns

A packet

Definition at line 340 of file LoRaMesh.cpp.

4.1.2.3 debug()

```
void LoRaMesh::debug (
            packet p )
```

Used for debugging the packet received from the LoRa module and printing it to the serial monitor

Parameters

in	*p*	The packet to be debugged

Definition at line 153 of file LoRaMesh.cpp.

4.1.2.4 debugPrint()

```
template<typename T >
void LoRaMesh::debugPrint (
            T print )
```

A function to print debug messages to the serial monitor based on a global debugging flag

Parameters

in	*print*	The string to be printed

Definition at line 19 of file LoRaMesh.cpp.

4.1.2.5 debugPrintln() [1/2]

```
void LoRaMesh::debugPrintln ( )
```

A function to print debug messages to the serial monitor with new line and no input, aka. empty line based on a global debugging flag

Definition at line 44 of file LoRaMesh.cpp.

4.1.2.6 debugPrintln() [2/2]

```
template<typename T >
void LoRaMesh::debugPrintln (
            T print )
```

A function to print debug messages to the serial monitor with new line based on a global debugging flag

Parameters

in	*print*	The string to be printed

Definition at line 33 of file LoRaMesh.cpp.

4.1.2.7 getmyID()

```
int LoRaMesh::getmyID ( )
```

Getter for the ID of this node

Returns

The ID of this node

Definition at line 106 of file LoRaMesh.cpp.

4.1.2.8 loraSetup()

```
int LoRaMesh::loraSetup (
            int myID,
            int ssPin,
            int rst,
            int dio0 )
```

This function extends the default one from the LoRa library adding all of the necessary parameters for the LoRaMesh class and setting the necessary pins for the LoRa module to function

Parameters

in	*myID*	ID to be used by this node
in	*ssPin*	The slave select pin for the LoRa module
in	*rst*	The reset pin for the LoRa module
in	*dio0*	The interrupt pin for the LoRa module

Returns

1 if the LoRa module is successfully started, 0 otherwise

Definition at line 62 of file LoRaMesh.cpp.

4.1.2.9 meshRead()

```
void LoRaMesh::meshRead ( )
```

A function to read and interpret the packet received from the LoRa module, this is where we use the callback functions to interpret the message

Definition at line 270 of file LoRaMesh.cpp.

4.1.2.10 meshSend()

```
void LoRaMesh::meshSend (
            packet sendPacket )
```

Function to send a packet to the LoRa module

Parameters

in	*sendPacket*	The packet to be sent

Definition at line 373 of file LoRaMesh.cpp.

4.1.2.11 printRoutingTable()

```
void LoRaMesh::printRoutingTable ( )
```

Used for printing the routing table to the serial monitor

Definition at line 191 of file LoRaMesh.cpp.

4.1.2.12 readAckCallback()

```
void LoRaMesh::readAckCallback (
            std::function< void(int, int)> callbackAck )
```

Register a callback function to be called when a acknowlegment is received so that any user can interpret the acknowlegment as they want

Parameters

callbackAck	The callback function to register.

Definition at line 95 of file LoRaMesh.cpp.

4.1.2.13 readInt()

```
unsigned short int LoRaMesh::readInt ( )
```

Used for reading an integer received from the LoRa module

Returns

The received integer

Definition at line 116 of file LoRaMesh.cpp.

4.1.2.14 readMsgCallback()

```
void LoRaMesh::readMsgCallback (
            std::function< void(int, int, String)> callbackData )
```

Register a callback function to be called when a message is received so that any user can interpret the message as they want

Parameters

in	*callbackData*	The callback function to register.

Definition at line 83 of file LoRaMesh.cpp.

4.1.2.15 routingPacket()

```
void LoRaMesh::routingPacket ( )
```

Sends a routing packet to the LoRa module. This function constructs a packet with routing information and sends it to the LoRa module. It updates the neighbor count and checks for any neighbors with a count above a threshold. If a neighbor's count exceeds the threshold, it removes the neighbor from the routing table and neighbor list. Finally, it sends the packet to the network.

Definition at line 406 of file LoRaMesh.cpp.

4.1.2.16 updateRoutingTable()

```
void LoRaMesh::updateRoutingTable (
            int id,
            unsigned short int size )
```

A function to update the routing table with a new node

Parameters

in	*id*	The ID of the node to be added to the routing table
in	*size*	The size of the packet received

Definition at line 214 of file LoRaMesh.cpp.

4.1.2.17 writeInt()

```
void LoRaMesh::writeInt (
            unsigned short int num )
```

Used for writing an integer to the LoRa module

Parameters

in	*num*	The integer to be sent

Definition at line 134 of file LoRaMesh.cpp.

4.1.3 Member Data Documentation

4.1.3.1 callbackAckFunction

```
std::function<void(int, int)> LoRaMesh::callbackAckFunction  [private]
```

Registered user function for handling acknolegments.

Definition at line 28 of file LoRaMesh.h.

4.1.3.2 callbackDataFunction

```
std::function<void(int, int, String)> LoRaMesh::callbackDataFunction  [private]
```

Registered user function for handling messages.

Definition at line 27 of file LoRaMesh.h.

4.1.3.3 debugging

```
bool LoRaMesh::debugging = true
```

Debugging flag.

Definition at line 48 of file LoRaMesh.h.

4.1.3.4 knownRoutes

```
std::vector<unsigned short int> LoRaMesh::knownRoutes
```

A table containing known routes by saving their nodeID.

Definition at line 51 of file LoRaMesh.h.

4.1.3.5 myID

```
unsigned short int LoRaMesh::myID  [private]
```

ID of this node.

Definition at line 26 of file LoRaMesh.h.

4.1.3.6 neighbours

```
std::map<unsigned short int,unsigned short int> LoRaMesh::neighbours
```

Neighbour table containing nodeID and hopCount.

Definition at line 50 of file LoRaMesh.h.

4.1.3.7 payload

```
char LoRaMesh::payload[244]
```

Packet data.

Definition at line 47 of file LoRaMesh.h.

4.1.3.8 receivedMsg

```
String LoRaMesh::receivedMsg
```

Received message.

Definition at line 46 of file LoRaMesh.h.

4.1.3.9 routingTable

```
std::map<unsigned short int,std::pair<unsigned short int,unsigned short int> > LoRaMesh↩
::routingTable
```

Routing table containing nodeID, nextHop, and hopCount.

Definition at line 49 of file LoRaMesh.h.

4.1.3.10 typeAckMessage

```
byte LoRaMesh::typeAckMessage = 6
```

Acknowledgment type packet.

Definition at line 32 of file LoRaMesh.h.

4.1.3.11 typeDataMessage

```
byte LoRaMesh::typeDataMessage = 0
```

Data type packet.

Definition at line 30 of file LoRaMesh.h.

4.1.3.12 typeRoutingMessage

```
byte LoRaMesh::typeRoutingMessage = 1
```

Routing type packet.

Definition at line 31 of file LoRaMesh.h.

The documentation for this class was generated from the following files:

- LoRaMesh.h
- LoRaMesh.cpp

4.2 LoRaMesh::packet Struct Reference

Structure representing a packet in the LoRaMesh network.

```
#include <LoRaMesh.h>
```

Public Attributes

- unsigned short int source
- unsigned short int destination
- unsigned short int nextHop
- unsigned short int packetSize
- unsigned short int msgId = 2
- byte type
- byte preamble
- char payload [244]

4.2.1 Detailed Description

Structure representing a packet in the LoRaMesh network.

Definition at line 36 of file LoRaMesh.h.

4.2.2 Member Data Documentation

4.2.2.1 destination

```
unsigned short int LoRaMesh::packet::destination
```

Receiver nodeID.

Definition at line 38 of file LoRaMesh.h.

4.2.2.2 msgId

```
unsigned short int LoRaMesh::packet::msgId = 2
```

Message ID.

Definition at line 41 of file LoRaMesh.h.

4.2.2.3 nextHop

```
unsigned short int LoRaMesh::packet::nextHop
```

Next hop nodeID.

Definition at line 39 of file LoRaMesh.h.

4.2.2.4 packetSize

```
unsigned short int LoRaMesh::packet::packetSize
```

Size of the packet.

Definition at line 40 of file LoRaMesh.h.

4.2.2.5 payload

```
char LoRaMesh::packet::payload[244]
```

Packet data.

Definition at line 44 of file LoRaMesh.h.

4.2.2.6 preamble

```
byte LoRaMesh::packet::preamble
```

Preamble.

Definition at line 43 of file LoRaMesh.h.

4.2.2.7 source

```
unsigned short int LoRaMesh::packet::source
```

Sender nodeID.

Definition at line 37 of file LoRaMesh.h.

4.2.2.8 type

```
byte LoRaMesh::packet::type
```

Type of the packet.

Definition at line 42 of file LoRaMesh.h.

The documentation for this struct was generated from the following file:

- LoRaMesh.h

Chapter 5

File Documentation

5.1 LoRaMesh.cpp

```
00001 #include "LoRaMesh.h"
00002 #include <SPI.h>
00003 #include <LoRa.h>
00004 #include <iterator>
00005 #include <vector>
00006 #include <algorithm>
00007 #include <map>
00008 #include <utility>
00009 #include <stdio.h>
00010
00018 template <typename T>
00019 void LoRaMesh::debugPrint(T print)
00020 {
00021     if (debugging)
00022         Serial.print(print);
00023 }
00024
00032 template <typename T>
00033 void LoRaMesh::debugPrintln(T print)
00034 {
00035     if (debugging)
00036         Serial.println(print);
00037 }
00038
00044 void LoRaMesh::debugPrintln()
00045 {
00046     if (debugging)
00047         Serial.println();
00048 }
00049
00062 int LoRaMesh::loraSetup(int myID, int ssPin, int rst, int dio0)
00063 {
00064     debugPrint("loraSetup(");
00065     debugPrint(myID);
00066     debugPrintln(")");
00067
00068     this->myID = myID;
00069     setPins(ssPin, rst, dio0);
00070     if (!begin(868E6)) return 0;
00071     routingTable[this->myID] = {this->myID, 0};
00072     knownRoutes.push_back(this->myID);
00073     return 1;
00074 }
00075
00083 void LoRaMesh::readMsgCallback(std::function<void(int, int, String)> callbackData) //
00084 {
00085     callbackDataFunction = callbackData;
00086     debugPrintln("readMsgCallback");
00087 }
00088
00095 void LoRaMesh::readAckCallback(std::function<void(int, int)> callbackAck)
00096 {
00097     callbackAckFunction = callbackAck;
00098     debugPrintln("readAckCallback");
00099 }
00100
00106 int LoRaMesh::getmyID()
00107 {
```

```
00108     return myID;
00109 }
00110
00116 unsigned short int LoRaMesh::readInt()
00117 {
00118     debugPrint("readInt(");
00119     debugPrint("");
00120     debugPrintln(")");
00121
00122     byte low = (byte)read();
00123     byte high = (byte)read();
00124     delay(10);
00125     return (high << 8 | low);
00126 }
00127
00134 void LoRaMesh::writeInt(unsigned short int num)
00135 {
00136     debugPrint("writeInt(");
00137     debugPrint(num);
00138     debugPrintln(")");
00139
00140     byte high = highByte(num);
00141     byte low = lowByte(num);
00142     write(low);
00143     write(high);
00144 }
00145
00153 void LoRaMesh::debug(packet p)
00154 {
00155     debugPrint("packet(");
00156     debugPrint("p");
00157     debugPrint(") ");
00158     debugPrint("debugging = ");
00159     debugPrintln(debugging);
00160
00161     if (!debugging)
00162     {
00163         return;
00164     }
00165     debugPrint("Preamble: ");
00166     debugPrintln(p.preamble);
00167     debugPrint("Destination: ");
00168     debugPrintln(p.destination);
00169     debugPrint("Source: ");
00170     debugPrintln(p.source);
00171     debugPrint("Next Hop: ");
00172     debugPrintln(p.nextHop);
00173     debugPrint("Type: ");
00174     debugPrintln(p.type);
00175     debugPrint("Packet Size: ");
00176     debugPrintln(p.packetSize);
00177     debugPrint("Payload: ");
00178     for (int i = 0; i < p.packetSize; ++i)
00179     {
00180         debugPrint(p.payload[i]);
00181     }
00182     debugPrint("\nMsgId: ");
00183     debugPrintln(p.msgId);
00184     debugPrintln();
00185 }
00186
00191 void LoRaMesh::printRoutingTable()
00192 {
00193     debugPrint("printRoutingTable(");
00194     debugPrint("");
00195     debugPrintln(")");
00196
00197     for (const auto &x : routingTable)
00198     {
00199         Serial.print(x.first);
00200         Serial.print(" ");
00201         Serial.print(x.second.first);
00202         Serial.print(" ");
00203         Serial.println(x.second.second);
00204     }
00205 }
00206
00214 void LoRaMesh::updateRoutingTable(int id, unsigned short int size)
00215 {
00216     debugPrint("updateRoutingTable");
00217     debugPrint(id);
00218     debugPrint(", ");
00219     debugPrint(size);
00220     debugPrintln(")");
00221
00222     std::vector<unsigned short int> vNew;
00223     for (int i = 0; i < size / 3; ++i)
```

```
00224     {
00225         int nextHop = readInt();
00226         int nodeID = readInt();
00227         int newRssi = readInt();
00228         vNew.push_back(nodeID);
00229         debugPrint(nextHop);
00230         debugPrint(" ");
00231         debugPrint(nodeID);
00232         debugPrint(" ");
00233         debugPrint(newRssi);
00234         debugPrintln();
00235 
00236         if (routingTable.find(nodeID) == routingTable.end())
00237             routingTable[nodeID] = {id, newRssi};
00238         else
00239         {
00240             if (routingTable[nodeID].second > newRssi)
00241                 routingTable[nodeID] = {id, newRssi};
00242         }
00243     }
00244 
00245     for (const auto &x : routingTable)
00246     {
00247         if (x.second.first == id)
00248         {
00249             if (std::find(vNew.begin(), vNew.end(), x.first) == vNew.end())
00250             {
00251             }
00252             // routingTable.erase(x.first);
00253         }
00254     }
00255 
00256     if (neighbours.find(id) == neighbours.end())
00257         neighbours.insert({id, 0});
00258     else
00259         neighbours[id] = 0;
00260 
00261     if (debugging) printRoutingTable();
00262 }
00263 
00270 void LoRaMesh::meshRead()
00271 {
00272     debugPrint("meshRead(");
00273     debugPrint("");
00274     debugPrintln(")");
00275 
00276     packet receivePacket;
00277     receivePacket.preamble = (byte)read();
00278     if (receivePacket.preamble != 170)
00279         return;
00280 
00281     receivePacket.destination = readInt();
00282     receivePacket.source = readInt();
00283     receivePacket.nextHop = readInt();
00284     receivePacket.type = (byte)read();
00285     receivePacket.packetSize = readInt();
00286 
00287     if (receivePacket.type == typeRoutingMessage)
00288     {
00289         debugPrintln("typeRoutingMessage Receved!");
00290         updateRoutingTable(receivePacket.source, receivePacket.packetSize);
00291         return;
00292     }
00293 
00294     for (int i = 0; i < (int)receivePacket.packetSize; ++i)
00295     {
00296         receivePacket.payload[i] = (char)(((byte)read()) - 1);
00297         delay(10);
00298     }
00299     receivePacket.msgId = readInt();
00300     receivePacket.payload[receivePacket.packetSize] = '\0';
00301     receivedMsg = String(receivePacket.payload);
00302     debug(receivePacket);
00303 
00304     if (receivePacket.destination == myID)
00305     {
00306         if (receivePacket.type == typeAckMessage)
00307         {
00308             debugPrintln("typeAckMessage Receved!");
00309             callbackAckFunction(receivePacket.msgId, receivePacket.source);
00310             return;
00311         }
00312         if (receivePacket.type == typeDataMessage)
00313         {
00314             debugPrintln("typeDataMessage Receved!");
00315             callbackDataFunction(receivePacket.source, receivePacket.msgId, receivedMsg);
00316             delay(500);
```

```
00317             meshSend(createAckPacket(receivePacket.packetSize, receivePacket.source,
      receivePacket.msgId));
00318         }
00319     }
00320     if (receivePacket.nextHop != myID)
00321         return;
00322     if (receivePacket.type == typeDataMessage)
00323     {
00324         receivePacket.nextHop = routingTable[receivePacket.destination].first;
00325         delay(2500);
00326         meshSend(receivePacket);
00327     }
00328     return;
00329 }
00330
00340 LoRaMesh::packet LoRaMesh::createMessagePacket(String msg, int idToSend, int msgId, int packetType)
00341 {
00342     debugPrint("createMessagePacket(");
00343     debugPrint(msg);
00344     debugPrint(", ");
00345     debugPrint(idToSend);
00346     debugPrint(", ");
00347     debugPrint(msgId);
00348     debugPrintln(")");
00349
00350     packet Packet;
00351     Packet.preamble = (byte)170;
00352     Packet.destination = idToSend;
00353     Packet.source = myID;
00354     Packet.nextHop = routingTable[Packet.destination].first;
00355     Packet.type = (byte)packetType;
00356     Packet.packetSize = msg.length();
00357     msg.toCharArray(Packet.payload, 244);
00358     debugPrint("payload = ");
00359     for (int i = 0; i < Packet.packetSize; ++i)
00360         debugPrint(Packet.payload[i]);
00361     Packet.msgId = msgId;
00362     if (debugging)
00363         Serial.printf("Packet.msgId = %d\n", Packet.msgId);
00364     return Packet;
00365 }
00366
00373 void LoRaMesh::meshSend(packet sendPacket)
00374 {
00375     debugPrint("meshSend(");
00376     debug(sendPacket);
00377     debugPrintln(")");
00378
00379     beginPacket();
00380     write(sendPacket.preamble);
00381     writeInt(sendPacket.destination);
00382     writeInt(sendPacket.source);
00383     writeInt(sendPacket.nextHop);
00384     write(sendPacket.type);
00385     writeInt(sendPacket.packetSize);
00386     for (int i = 0; i < sendPacket.packetSize; ++i)
00387     {
00388         write((byte)((int)sendPacket.payload[i] + 1));
00389     }
00390     writeInt(sendPacket.msgId);
00391     debugPrint("endPacket() = ");
00392     debugPrintln(endPacket(1));
00393     return;
00394 }
00395
00396
00406 void LoRaMesh::routingPacket() //Bivski helloPacket
00407 {
00408     packet sendPacket;
00409     sendPacket.preamble = (byte)170;
00410     sendPacket.destination = 0;
00411     sendPacket.source = myID;
00412     sendPacket.nextHop = 0; // 0 je broadcast
00413     sendPacket.type = (byte)typeRoutingMessage;
00414     sendPacket.packetSize = knownRoutes.size() * 3;
00415     sendPacket.payload[0] = '\0';
00416     debugPrint("helloPacket(");
00417     debug(sendPacket);
00418     debugPrintln(")");
00419
00420     for (const auto &x : neighbours)
00421     {
00422         neighbours[x.first]++;
00423     }
00424
00425     debugPrint("neighbours = ");
00426     for (const auto &x : neighbours)
```

```
00427     {
00428         debugPrint("(");
00429         debugPrint(x.first);
00430         debugPrint(", ");
00431         debugPrint(x.second);
00432         debugPrint("), ");
00433         if (x.second >= 4)
00434         {
00435             for (const auto &y : routingTable)
00436                 if (y.second.first == x.first)
00437                     routingTable.erase(y.first);
00438             neighbours.erase(x.first);
00439             if (debugging) printRoutingTable();
00440         }
00441     }
00442     debugPrintln();
00443 
00444     beginPacket();
00445     write(sendPacket.preamble);
00446     writeInt(sendPacket.destination);
00447     writeInt(sendPacket.source);
00448     writeInt(sendPacket.nextHop);
00449     write(sendPacket.type);
00450     writeInt(sendPacket.packetSize);
00451     debugPrintln("knownRoutes = (");
00452 
00453     for (const auto &x : routingTable)
00454     {
00455         debugPrint(x.first);
00456         debugPrint(", ");
00457         writeInt(x.first);
00458         writeInt(x.second.first);
00459         writeInt(x.second.second);
00460     }
00461     debugPrintln(")");
00462     debugPrint("endPacket() = ");
00463     debugPrintln(endPacket(1));
00464     return;
00465 }
00466 
00477 LoRaMesh::packet LoRaMesh::createAckPacket(int sizeOfMsgToAck, int idToSend, int msgIdToAck)
00478 {
00479     debugPrint("createAckPacket(");
00480     debugPrint(sizeOfMsgToAck);
00481     debugPrint(", ");
00482     debugPrint(idToSend);
00483     debugPrint(", ");
00484     debugPrint(msgIdToAck);
00485     debugPrintln(")");
00486 
00487     return createMessagePacket(String(sizeOfMsgToAck), idToSend, msgIdToAck, typeAckMessage);
00488 }
```

5.2 LoRaMesh.h

```
00001 #ifndef LORAMESH_H
00002 #define LORAMESH_H
00003 
00004 #include <SPI.h>
00005 #include <LoRa.h>
00006 #include <map>
00007 #include <utility>
00008 #include <vector>
00009 
00024 class LoRaMesh : public LoRaClass{
00025 private:
00026   unsigned short int myID;
00027   std::function<void(int, int, String)> callbackDataFunction;
00028   std::function<void(int, int)> callbackAckFunction;
00029 public:
00030     byte typeDataMessage = 0,
00031     typeRoutingMessage = 1,
00032     typeAckMessage = 6;
00036     struct packet{
00037       unsigned short int source;
00038       unsigned short int destination;
00039       unsigned short int nextHop;
00040       unsigned short int packetSize;
00041       unsigned short int msgId = 2;
00042       byte type;
00043       byte preamble;
00044       char payload[244];
00045     };
```

```
00046     String receivedMsg;
00047     char payload[244];
00048     bool debugging = true;
00049     std::map<unsigned short int,std::pair<unsigned short int,unsigned short int»routingTable;
00050     std::map<unsigned short int,unsigned short int>neighbours;
00051     std::vector<unsigned short int>knownRoutes;
00052     int loraSetup(int myID, int ssPin, int rst, int dio0);
00053     unsigned short int readInt();
00054     void writeInt(unsigned short int num);
00055     void debug(packet p);
00056     void meshRead();
00057     packet createMessagePacket(String msg, int idToSend, int msgId, int packetType = 0);
00058     packet createAckPacket(int sizeOfMsgToAck, int idToSend, int msgIdToAck);
00059     void meshSend(packet sendPacket);
00060     int getmyID();
00061     void routingPacket();
00062     void updateRoutingTable(int id, unsigned short int size);
00063     void printRoutingTable();
00064     template <typename T> void debugPrint(T input);
00065     template <typename T> void debugPrintln(T input);
00066     void debugPrintln();
00067     void readMsgCallback(std::function<void (int, int, String)> callbackData);
00068     void readAckCallback(std::function<void (int, int)> callbackAck);
00069 };
00070
00071 #endif
```

Index

callbackAckFunction
    LoRaMesh, 13
callbackDataFunction
    LoRaMesh, 13
createAckPacket
    LoRaMesh, 8
createMessagePacket
    LoRaMesh, 9

debug
    LoRaMesh, 9
debugging
    LoRaMesh, 13
debugPrint
    LoRaMesh, 9
debugPrintln
    LoRaMesh, 10
destination
    LoRaMesh::packet, 16

getmyID
    LoRaMesh, 10

knownRoutes
    LoRaMesh, 14

LoRaMesh, 7
    callbackAckFunction, 13
    callbackDataFunction, 13
    createAckPacket, 8
    createMessagePacket, 9
    debug, 9
    debugging, 13
    debugPrint, 9
    debugPrintln, 10
    getmyID, 10
    knownRoutes, 14
    loraSetup, 10
    meshRead, 11
    meshSend, 11
    myID, 14
    neighbours, 14
    payload, 14
    printRoutingTable, 11
    readAckCallback, 12
    readInt, 12
    readMsgCallback, 12
    receivedMsg, 14
    routingPacket, 12
    routingTable, 14
    typeAckMessage, 15
    typeDataMessage, 15
    typeRoutingMessage, 15
    updateRoutingTable, 13
    writeInt, 13
LoRaMesh::packet, 15
    destination, 16
    msgId, 16
    nextHop, 16
    packetSize, 16
    payload, 16
    preamble, 17
    source, 17
    type, 17
loraSetup
    LoRaMesh, 10

meshRead
    LoRaMesh, 11
meshSend
    LoRaMesh, 11
msgId
    LoRaMesh::packet, 16
myID
    LoRaMesh, 14

neighbours
    LoRaMesh, 14
nextHop
    LoRaMesh::packet, 16

packetSize
    LoRaMesh::packet, 16
payload
    LoRaMesh, 14
    LoRaMesh::packet, 16
preamble
    LoRaMesh::packet, 17
printRoutingTable
    LoRaMesh, 11

readAckCallback
    LoRaMesh, 12
readInt
    LoRaMesh, 12
readMsgCallback
    LoRaMesh, 12
receivedMsg
    LoRaMesh, 14
routingPacket

LoRaMesh, 12
routingTable
LoRaMesh, 14

source
LoRaMesh::packet, 17

type
LoRaMesh::packet, 17
typeAckMessage
LoRaMesh, 15
typeDataMessage
LoRaMesh, 15
typeRoutingMessage
LoRaMesh, 15

updateRoutingTable
LoRaMesh, 13

writeInt
LoRaMesh, 13