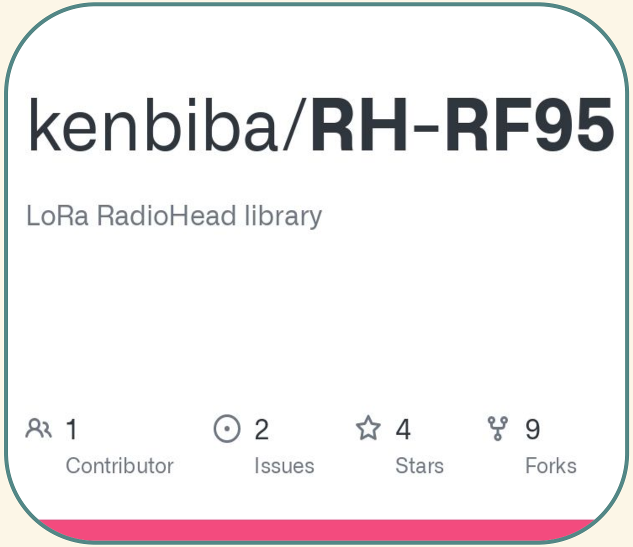
LoRa Documentation

Mihir, Brian, and Alex



kenbiba/**RH-RF95**

LoRa RadioHead library

1 Contributor

2 Issues

4 Stars

9 Forks

01

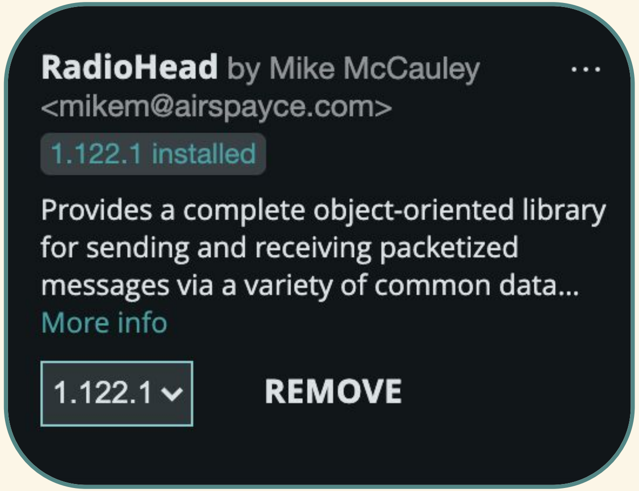
RH-RF95 Master Library

RH-RF95 Master

What we changed	Data Packet	Max Distance	Notes
Default Settings: one has the example server code, and the other has the example client code.	“Hello I’m from Lora!” and the second device responded: “And hello back to you!”	5 CM	No corruption, speeds of under 1 second!
Smaller Packet	“0” and the second device responded: “1”	5.5 CM	Practically no improvement at all, the 0.5cm is dismissible as tilts in the antenna, and other factors could have contributed to this.
rf95.setTxPower(23);	“0” and the second device responded: “1”	60 CM	This, once again, had sub-second speeds.
Larger Packet	“3.1415926535 8979323846 2643383279 5028841971 6939937510 5820974944 5923078164 0628620899 8628034825 3421170679”	24 CM	Here, we conclude that, yes, the packet size does affect the maximum distance of transfer. We also noticed a, proportionally, much higher delay—2 seconds.

RH-RF95 Master

What we changed	Data Packet	Max Distance	Notes
Switched to Arduino Mega	“3.1415926535 8979323846 2643383279 5028841971 6939937510 5820974944 5923078164 0628620899 8628034825 3421170679”	55 CM	The arduino Mega is able to use more power, which is why the distance increased. So far, power has been the most influential factor in term of range.
rf95.setModemConfig(0xc0);	“3.1415926535 8979323846 2643383279 5028841971 6939937510 5820974944 5923078164 0628620899 8628034825 3421170679”	75 CM	We changed the spreading factor.
Small packet	“0” and the second device responded: “1”	90 CM	We tested with a smaller packet to test if the SP was s fluke or not. It wasn't, changing the SP changes the distance.
rf95.setModemConfig(0x80);	“0” and the second device responded: “1”	100 CM	Multiple modem configs can be supported, we still have the old config. This one sets bandwidth to 500kHz.



RadioHead by Mike McCauley <mikem@airspayce.com>

1.122.1 installed

Provides a complete object-oriented library for sending and receiving packetized messages via a variety of common data...

More info

02

RadioHead Library

RadioHead - Lev Recommended we switch to this

What we changed	Data Packet	Max Distance	Notes
Default Settings: rf95_client uploaded to one, and rf95_server to the other.	"Hello I'm from Lora!" and the second device responded: "And hello back to you!"	5 CM	We are back to the beginning, but the previous testing indicates there are ways to improve this.
Smaller Packet	"0" and the second device responded: "1"	11 CM	Progress.

Added previous settings:
rf95.setModemConfig(0x80);
rf95.setModemConfig(0xc0);
rf95.setTxPower(23);
	"0" and the second device responded:
	"1"
		30 CM	Interesting... this is the same as the previous library trial, yet we are getting shorter distances...

Slightly adjusting the antenna	"0" and the second device responded: "1"	80 CM	It is super sensitive to antenna orientation, at this point.

RadioHead

What we changed	Data Packet	Max Distance	Notes
At this point we dived into more specific things 1. Long Range Mode versus FSK/OOK mode (It turns out it was already in Long Range Mode) 2. WaitPacketSent - getting more distance at the cost of time (This Worked and we only sacrificed 2-3s of time)			
We tinkered with the settings of the LoRa device it self, manually setting pins: Chip Select (10), Interrupt (2), and Reset (9) Pins. And set the frequency to 915 No longer does the server send *back* a different response. And we increased the wait time.	“Hello World!”	230 M!	This was not limited by distance this was limited by a **hill** in the way.
Different packet	A *thousand* “1”’s	130 M	Consistent corruption at 64th character onwards. This is due to the maximum amount of data that Arduino Serial can hold.

RadioHead

What we changed	Data Packet	Max Distance	Notes
Multiple Clients	30 “1”s - Client 1 30 “2”s - Client 2 30 “3”s - Client 3	N/A	Each message mixed together (lots of 1s 2s and 3s) We found out that to change it so each message was independent, each client had to have a different spreading factor.
Put each client on a different spreading factor	30 “1”s - Client 1 30 “2”s - Client 2 30 “3”s - Client 3	N/A	No interference, however it only can receive one at a time, and a lot of the time, the server does not receive any at all.

We concluded that the signals would interfere, no matter what, if they were all going at the same time. So we decided to set them to the same spreading factor, and just have them send messages individually. We did this by having the *clients* listening for a message first. This message would be a ClientID sent from the *server* arduino. Each individual client would then test their ID against the requested one, and if they match, then they will begin to send data. Ten seconds after the client stops sending data, the connection will time out on both ends, and the server can now connect to a new client. There was very little to test here, as all the added code was just 2 booleans and a few if statements.

Python

Along with the Arduino code, there is also some code that must uploaded to the Pi. And there is an additional file that must be running on the server side to capture all the data and to store it. This is some very simple python code. But it allows for the entire system to be controlled from it (you can input/type the client id into the serial monitor from the python terminal).

Final Testing

Environment	Max Distance	Notes
Low Interference No Obstruction	2 KM+	Zealandia Fence Line
High Interference High Obstruction	75 M	Through buildings on the massey campus
High Interference Medium Obstruction	230 M	Over the hill on Tasman Street
High Interference Low Obstruction	550 M	Along Tasman - Tory Street

Thanks!

This is the attribution slide for this slidesgo template

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

Please keep this slide for attribution