LoRa 란?

[https://www.mokolora.com/ko/what-is-lora-iot/]

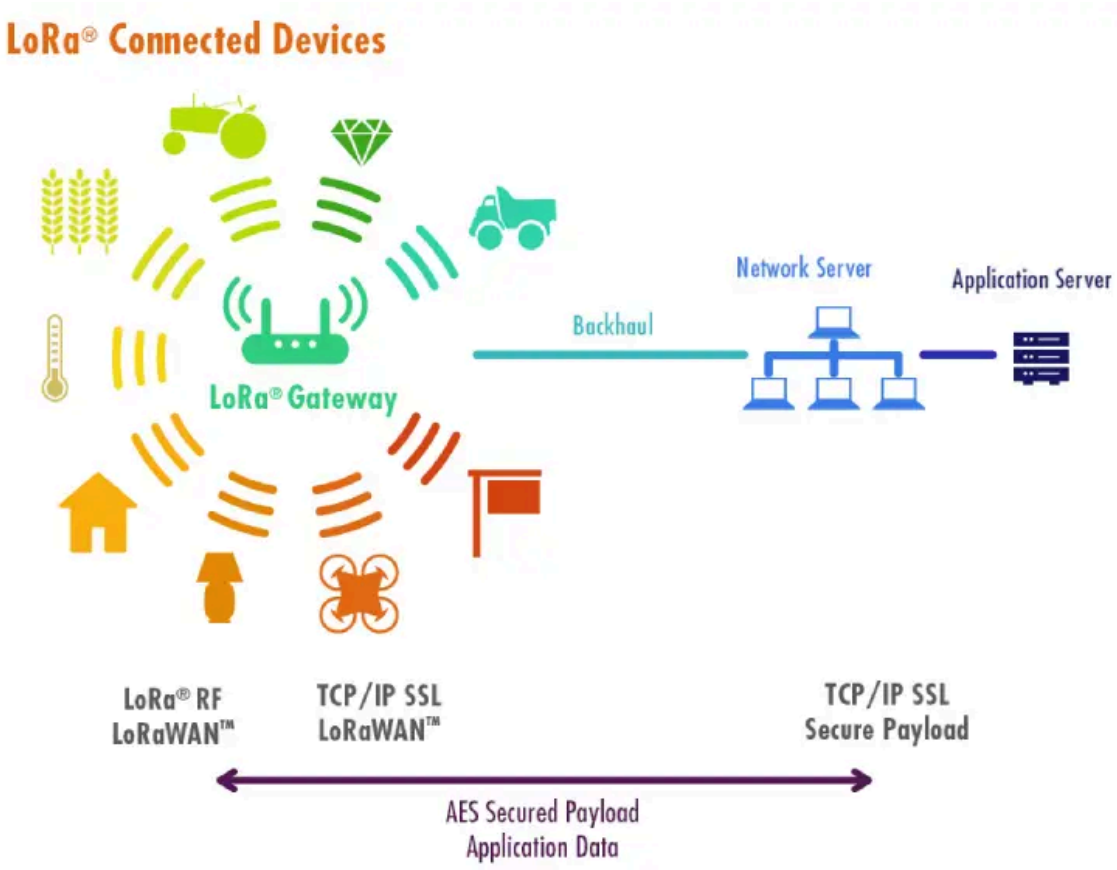
- LoRa는 저전력 WAN 전송 기술: LoRa는 Long-Range의 약자

  장거리는 또한 LoRa의 핵심 이점(15km 이상(Open-space)의 전송이 가능함.

- LPWAN 통신 기술 중 하나로, LoRa는 확산 스펙트럼(Spread-spectrum) 기술을 기반으로 장거리 무선 전송 방식
- LoRa는 `선형 주파수 변조 확산 스펙트럼`을 기반으로 하고 있으며,

  신호 스펙트럼 변조의 낮은 전력 소모 특성을 유지할 뿐만 아니라 통신 거리를 대폭 증가시킴

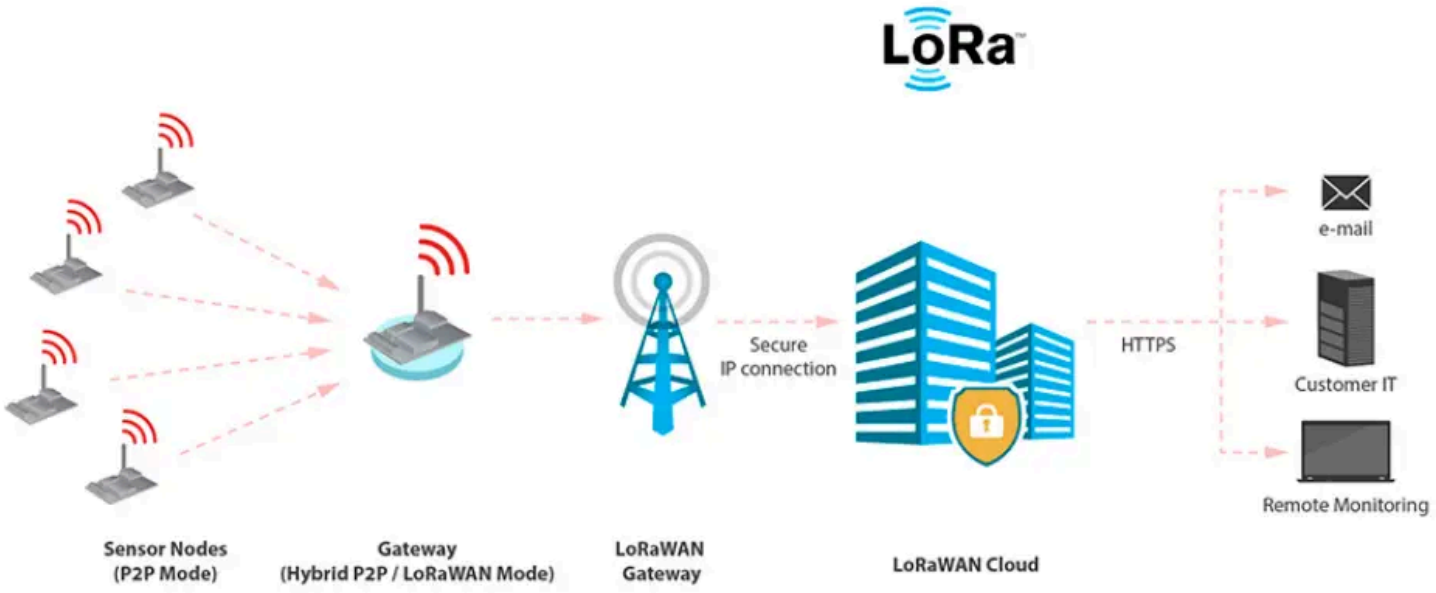
- 높은 감도와 강력한 신호 대 잡음비를 가지고 있음.



사물 인터넷 소개

- 사물 인터넷 : 사물을 네트워크에 연결하는 장치 및 기술을 의미.
- 사물 자체는 환경의 변화를 감지할 수 없으나, 다양한 IoT 센서 디바이스를 설치해야 함.
- 정보 센서와 같은, 레이저 스캐너, 무선 주파수 식별 장치, 온도 및 습도 센서, 등. 이러한 데이터가 수집된 후, 처리를 위해 클라우드 서버 또는 센터로 전송되며, 서버가 데이터를 처리한 후, 정보를 기반으로 피드백을 제공함. 사물은 센서를 통해 객체 및 서버에 연결되며, 네트워크 장치, 지능형 인식을 실현하기 위한 게이트웨이, 객체 및 프로세스의 인식 및 관리. 사물 인터넷은 인터넷과 전통적인 통신 네트워크를 기반으로 하는 정보 매체을 의미함.

LoRa IoT



- LoRa IoT는 LoRa 모듈을 통해 사물을 네트워크에 연결하는 것 (게이트웨이, 및 기타 장치).
- LoRa IoT는 LoRa 시스템을 통해 각종 센서 및 센서와 같은 최종 장치에서 수집한 다양한 필수 정보를 네트워크 노드 및 서버로 전송.
- 단말 장치는 서버 또는 다른 단말 장치가 보낸 정보에 따라 반응할 수도 있으며, LoRa IoT 시스템의 연결은 양방향임.

LoRa IoT 시스템의 장단점

이점: LoRa IoT 네트워크는 전송 거리가 긴 특성을 가지고 있음. 낮은 작동 전력 소비, 많은 네트워크 노드, 강력한 간섭 방지 능력, 저렴한 비용.

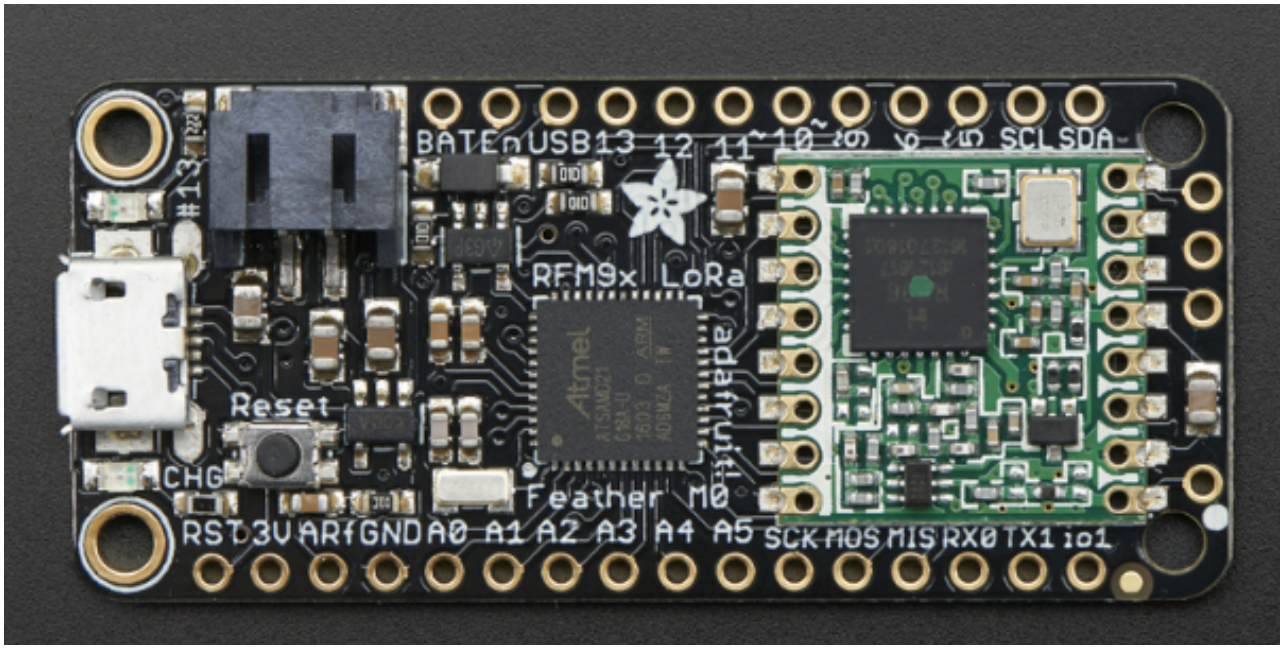
- 긴 전송 거리: 감도 -148dBm, 최대 통신 거리 15 킬로미터
- 낮은 작동 에너지 소비: 알로하 방식은 데이터가 있을 때만 연결, 배터리는 몇 년 동안 작동할 수 있음.
- 다중 네트워크 노드: 유연한 네트워킹 모드에서, 여러 노드를 연결할 수 있음
- 강력한 간섭 방지: 프로토콜에는 LBT의 기능이 있음며, 알로하 방식을 기반으로, 자동 주파수 점프 및 속도 적응 기능 포함
- 저렴한 비용: 무면허 스펙트럼, 낮은 노드/단말 비용

단점: LoRa IoT 네트워크에는 많은 장점이 있으나 또한 특정 단점이 있음.

- 스펙트럼 간섭: LoRa의 지속적인 개발로, 로라 장비, 네트워크 구축이 계속해서 증가하고 있어 특정 스펙트럼 간섭이 서로간에 발생할 가능성이 높아짐.
- 새로운 네트워크 구축 필요: LoRa 배포 프로세스 중, 사용자는 자신의 네트워크를 구축해야 하는 경우가 발생함.
- 작은 페이로드: LoRa 전송 데이터의 페이로드는 상대적으로 작고 바이트 제한이 있음.


MOKO SMART
IOT TECH EXPO
IOT TECH EXPO 2023
Booth No.: 209
Date: September 26-27, 2023
Location: RAI Amsterdam Convention Centre, Amsterdam, Netherlands

Adafruit Feather M0 Radio with LoRa Radio Module



- At the Feather M0's heart is an ATSAMD21G18 ARM Cortex M0 processor, clocked at 48 MHz and at 3.3V logic, the same one used in the new Arduino Zero.
- This chip has a whopping 256K of FLASH (8x more than the Atmega328 or 32u4) and 32K of RAM (16x as much)!
- This chip comes with built in USB so it has USB-to-Serial program & debug capability built in with no need for an FTDI-like chip.

Here's some handy specs! Like all Feather M0's you get:

- Measures 2.0" x 0.9" x 0.3" (51mm x 23mm x 8mm) without headers soldered in
- Light as a (large?) feather - 5.8 grams
- ATSAMD21G18 @ 48MHz with 3.3V logic/power
- No EEPROM
- 3.3V regulator with 500mA peak current output
- USB native support, comes with USB bootloader and serial port debugging
- You also get tons of pins - 20 GPIO pins
- Hardware Serial, hardware I2C, hardware SPI support
- 8 x PWM pins
- 10 x analog inputs
- 1 x analog output
- Built in 100mA lipoly charger with charging status indicator LED
- Pin #13 red LED for general purpose blinking
- Power/enable pin

- 4 mounting holes
- Reset button

This **Feather M0 LoRa Radio** uses the extra space left over to add an RFM9x LoRa 868/915 MHz radio module.

These radios are not good for transmitting audio or video, but they do work quite well for small data packet transmission when you need more range than 2.4 GHz (BT, BLE, WiFi, ZigBee).

- SX1276 LoRa® based module with SPI interface
- Packet radio with ready-to-go Arduino libraries
- Uses the license-free ISM bands (ITU "Europe" @ 433MHz and ITU "Americas" @ 900MHz)
- +5 to +20 dBm up to 100 mW Power Output Capability (power output selectable in software)
- ~300uA during full sleep, ~120mA peak during +20dBm transmit, ~40mA during active radio listening.
- Simple wire antenna or spot for uFL connector

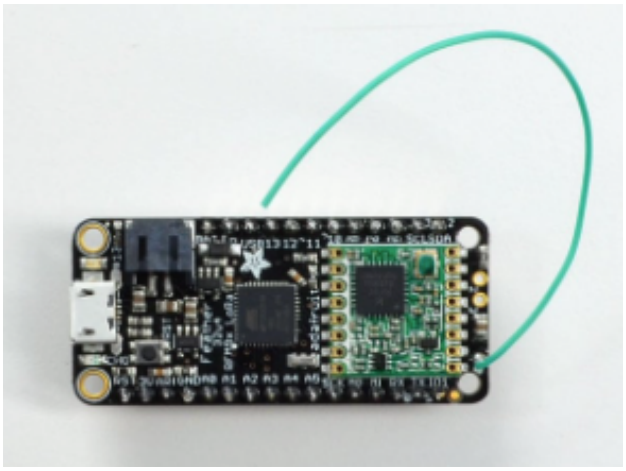
> Our initial tests with default library settings: over 1.2mi/2Km line-of-sight with wire quarter-wave antennas.

Antenna Options

Wire Antenna

A wire antenna, aka "`quarter wave whip antenna`" is low cost and works very well!

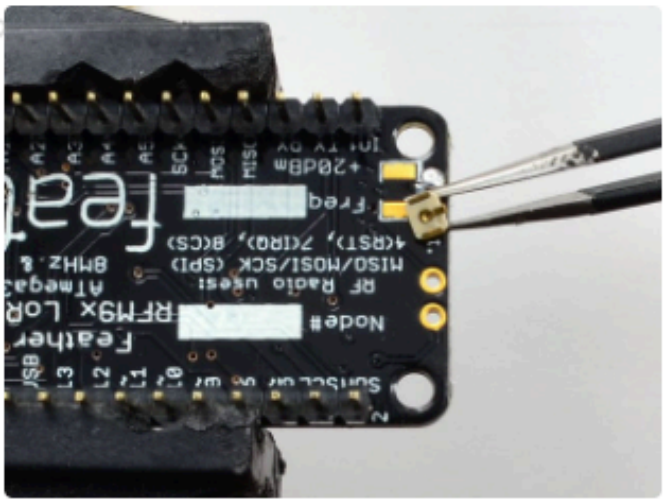
You just have to cut the wire down to the right length.



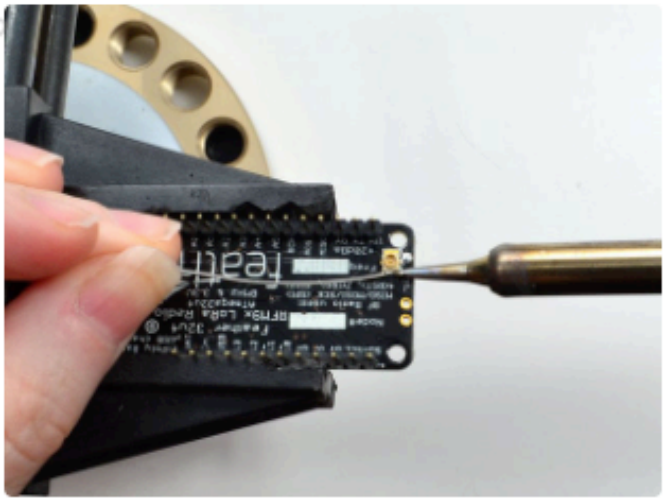
Cut a stranded or solid core wire to the proper length for the module/frequency:

- **433 MHz** - 6.5 inches, or 16.5 cm
- **868 MHz** - 3.25 inches or 8.2 cm
- **915 MHz** - 3 inches or 7.8 cm

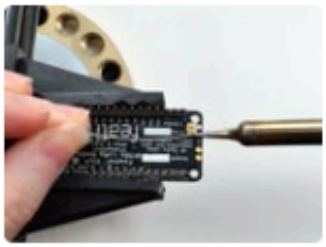
uFL Antenna

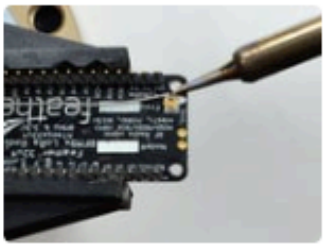


Check the bottom of the uFL connector, note that there's two large side pads (ground) and a little inlet pad. The other small pad is not used!

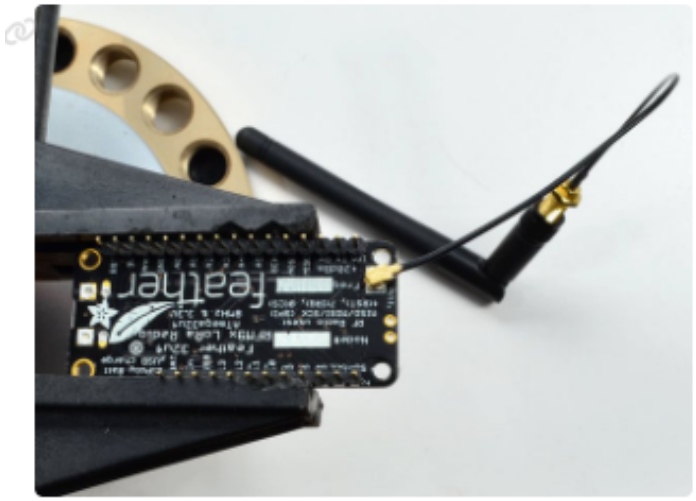


Solder all three pads to the bottom of the Feather





For all radio-capable Feather boards:



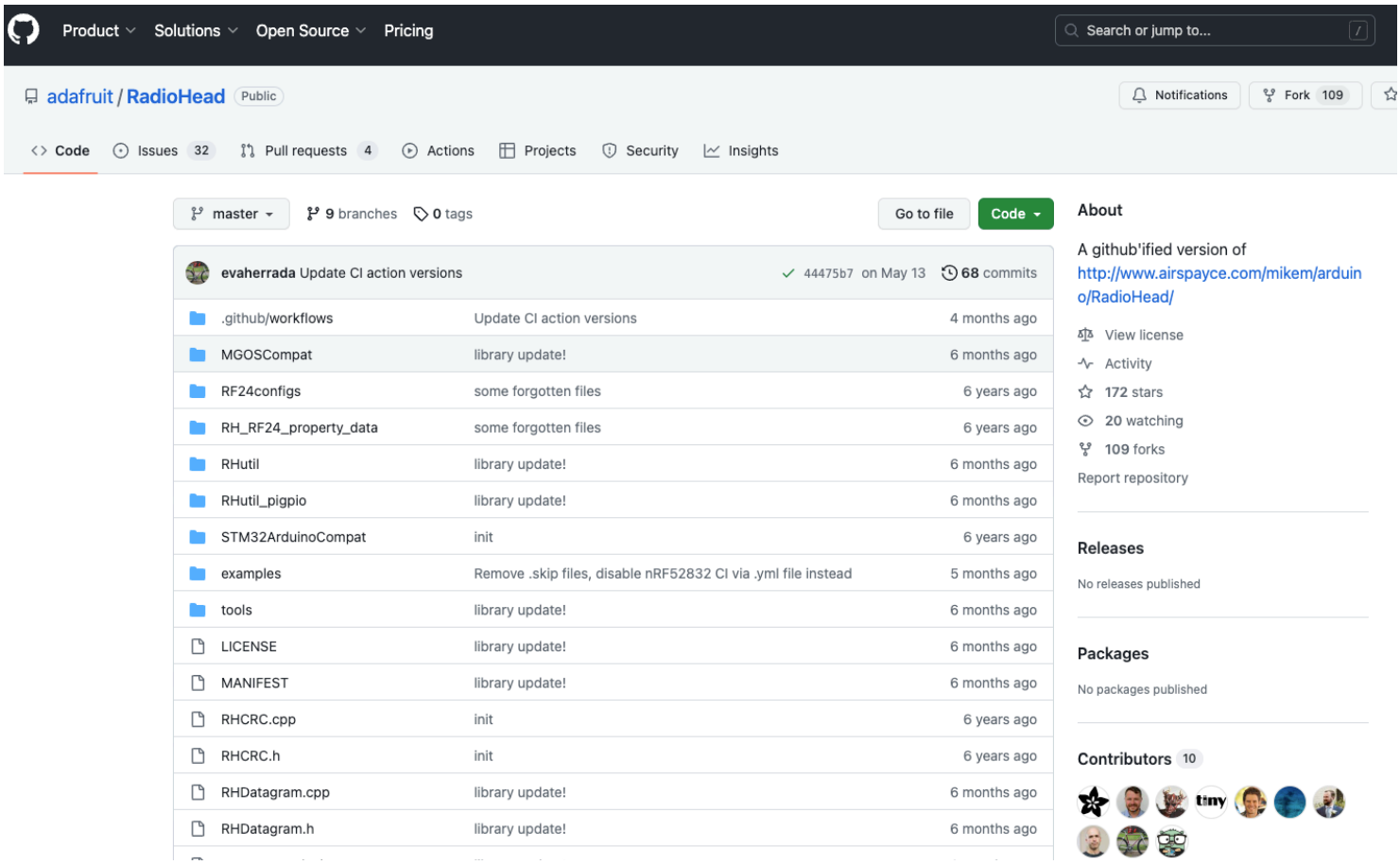
Once done attach your uFL adapter and antenna!

RadioHead RFM9x Library example

To begin talking to the radio, you will need to download the RadioHead library. You can do that by visiting the GitHub repo and manually downloading or, easier, just click the button below to download as a zip file:

Download RadioHead Library

Note that while all the code in the examples below are based on this version, you can visit the RadioHead documentation page to get the most recent version which may have bug-fixes or more functionality



[https://github.com/adafruit/RadioHead]

RadioHead Packet Radio library for embedded microprocessors

[https://www.airspayce.com/mikem/arduino/RadioHead/]

RadioHead

RadioHead Packet Radio library for embedded microprocessors

This is the RadioHead Packet Radio library for embedded microprocessors. It provides a complete object-oriented library for sending and receiving packetized messages via a variety c common data radios and other transports on a range of embedded microprocessors.

The version of the package that this documentation refers to can be downloaded from http://www.airspayce.com/mikem/arduino/RadioHead/RadioHead-1.125.zip You can find the late version of the documentation at http://www.airspayce.com/mikem/arduino/RadioHead

You can also find online help and discussion at http://groups.google.com/group/radiohead-arduino Please use that group for all questions and discussions on this topic. Do not contact author directly, unless it is to discuss commercial licensing. Before asking a question or reporting a bug, please read

- http://en.wikipedia.org/wiki/Wikipedia:Reference_desk/How_to_ask_a_software_question
- http://www.catb.org/esr/faqs/smart-questions.html
- http://www.chiark.greenend.org.uk/~shgtatham/bugs.html

Caution: Developing this type of software and using data radios successfully is challenging and requires a substantial knowledge base in software and radio and data transmission technologies and theory. It may not be an appropriate project for beginners. If you are a beginner, you will need to spend some time gaining knowledge in these areas first.

Overview

RadioHead consists of 2 main sets of classes: Drivers and Managers.

- Drivers provide low level access to a range of different packet radios and other packetized message transports.
- Managers provide high level message sending and receiving facilities for a range of different requirements.

Every RadioHead program will have an instance of a Driver to provide access to the data radio or transport, and usually a Manager that uses that driver to send and receive messages the application. The programmer is required to instantiate a Driver and a Manager, and to initialise the Manager. Thereafter the facilities of the Manager can be used to send and recei messages.