

Tutorial #1: Getting Started with the LoRa

1 INTRODUCTION

This is a *Getting Started* tutorial, that provides the reader with the most basic concepts required to get a simple project working using LoRa transceivers. This is as basic of a tutorial as we could put together, requiring the least amount of hardware and coding. Many advanced concepts of RF communication as they relate to LoRa are not covered. In this tutorial, two LoRa transceivers are each connected to a Pro Mini microcontroller. One LoRa is designated as the “sender”, while the other as the “receiver”. The sender increments an integer value with each Arduino loop iteration and the receiver prints out the corresponding value.

The *Serial Peripheral Interface (SPI)* protocol is used to communicate between the LoRas and the Pro Minis. This configuration requires four connections between these two devices. In addition to the four SPI connections, there are two more data signals, and the two power connections. Please ensure that you are using a 3.3V Pro Mini and a 3.3V LoRa transceiver module. Note that the SX1276 LoRa module (and most LoRa modules) operate at 3.3V because the LoRa chips operates at 3.3V, and the whole point of LoRa is low-power and long-range. If both devices work on 3.3V logic the SPI and logic signals can be connected directly between the two devices. If the logic levels of the devices are not the same, a bi-directional logic level shifter that will handle the data lines at the SPI data rate will be required between the devices. This tutorial does not cover wiring that includes the logic level shifter.

2 REQUIRED HARDWARE

The following hardware is the minimum required to get two LoRa modules talking to each other:

- 1x *PTSolns LoRa 915MHz SX1276 Long Range Transceiver Prototyping Module*
 - This package includes two sets of LoRa transceivers and PTSolns LoRa Keys. **Soldering is required.**
 - The LoRa modules included in this package works in the North American 915MHz band.
 - This package also includes two logic level shifters, voltage regulators, and smoothing capacitors. In this tutorial these items are not required but will become useful in Tutorial #2.
- 2x Pro Mini microcontroller
 - These are common microcontrollers such as the one shown in Fig. 1. There are different versions available, with slightly different pinouts, but any of them should work.
 - In this tutorial we are using a **328P, 5V, 16MHz** Pro Mini, not the 3.3V. The ATmega328P can be powered with voltages between 1.8V to 5.5V. **However**, supplying 3.3V and continuing to run the microcontroller at 16MHz is outside the ATmega328P recommended range of operation. From our testing we have observed no issues for the application of this tutorial. Therefore, use either the 3.3V or the 5V Pro Mini.
 - NOTE: If you use the 5V, 16MHz version of the Pro Mini make sure you run it at 3.3V, or use logic level shifters between the Pro Mini and the LoRa module.

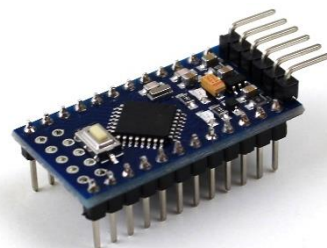


Figure 1: 328P, 5V, 16MHz Pro Mini

- 2x FTDI USB to TTL serial converter module (and corresponding USB cable)
 - o There are many such modules available, with different USB port sizes, different ICs (FT232RL, CH340G, etc). Some require the installation of a special driver, which is commonly provided by the distributor of the module. Any of these work, but we have a preference for the common red FTDI module as shown in Fig. 2. **Ensure that the 2-pin jumper is set to 3.3V.**

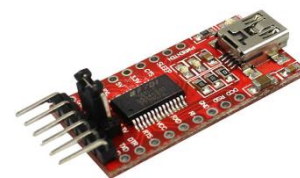


Figure 2: FTDI USB to TTL Serial Converter Module

- 2x standard breadboards
- Several preformed jumper wires for breadboards, or dupont wires
- Benchtop power supply, or equivalent

3 REQUIRED SOFTWARE

The Arduino IDE software is required. This can be downloaded freely. Once this software is installed, there are some sub-steps the user may, or may not, have to do in order to get the software setup correct. These steps are as follows:

- 1) Depending on which FTDI USB to TTL serial converter module is used, a special driver, such as the CH340 driver, might have to be installed. This is a well-documented driver for which there are many tutorials available. The tutorial we recommend is called “*Tutorial: First time using the Nano Microcontroller*” by PTSolns. However, if the module as shown in Fig. 2 is used, then it is likely no special driver will be required.
- 2) The script shown in Section 5 below requires the LoRa.h library to be installed within the IDE software. Open Tools\Manage Libraries ... and search for “LoRa” by Sandeep Mistry.

4 SETTING UP THE HARDWARE

Before setting up the sender and receiver circuits, the LoRa modules and the corresponding LoRa keys must be soldered together (twice – one for the sender side and one for the receiver side). The final product should look like what is shown in Fig. 3.

The sender and receiver circuits are identical in terms of hardware and connections. The circuit schematic shown in Fig. 4 should be used as a reference to assemble the circuits on the breadboards. Setting up circuits on breadboards can be tricky. The use of a multimeter is recommended to ensure good electrical connections are made. The user may wish to use something more permanent than a temporary breadboard. For this we recommend the *PTSolns Proto-Half* boards.

These have a similar layout to a standard breadboard but allow for the components to be soldered to make a permanent product. The hardware connections between the three main devices are outlined in Table 1 & Table 2.

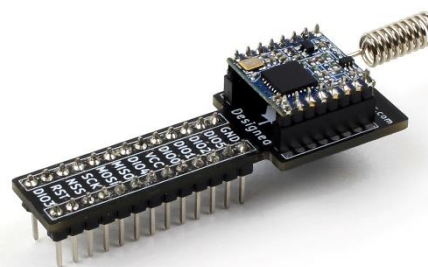


Figure 3: LoRa module and LoRa key fully assembled.

Table 1: Hardware Connection for FTDI USB to TTL Serial Converter Module to Pro Mini

FTDI USB to TTL Serial Converter	Pro Mini
GND	GND
VCC ¹	VCC
Rx	Tx
Tx	Rx
DTR	DTR

¹ Ensure the 2-Pin jumper is on the 3.3V option.

Table 2: Hardware Connection for FTDI USB to TTL Serial Converter Module to Pro Mini

Pro Mini	LoRa
GND	GND
VCC	VCC
2	DIO2
9	RST
10	NSS
11	MOSI
12	MISO
13	SCK

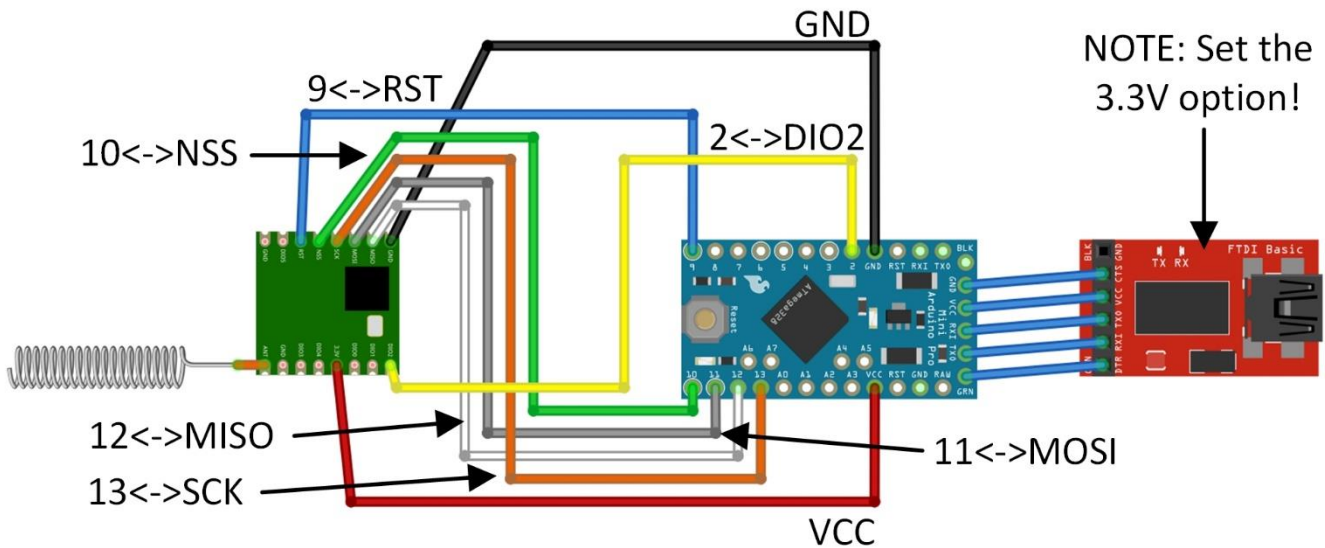


Figure 4: Wiring Schematic (ESP Master Key and Breadboard not shown).

5 SETTING UP THE SOFTWARE

The scripts for the sender and receiver in this tutorial have been reduced to the bare minimum. This is to keep the coding as simple as possible and get started with the LoRa as quickly as possible. However, the user is reminded that many of the details relating to the operation of the LoRa are omitted.

As mentioned above, in this tutorial we are using a 328P, 5V, 16MHz Pro Mini, but running it at 3.3V. As the IDE software does not care what voltage is being used, we can safely select the 5V option in the software. However, **please remember to set the 2-Pin jumper to 3.3V in the hardware**. Within the IDE software check the settings as shown in Fig. 5.

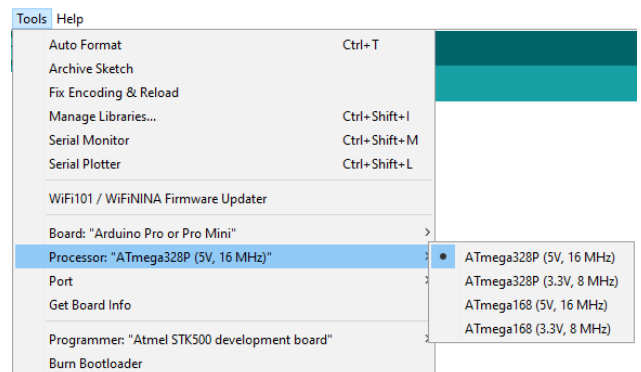


Figure 5: Configuring the IDE software for the Pro Mini.

The sender and receiver scripts are outlined below. Both the sender and receiver parts are contained in the same script. When uploading the sender side simply uncomment the line `#define COMPILE_TX`, and comment out the line `#define COMPILE_RX` and vice versa when uploading the receiver side.

```

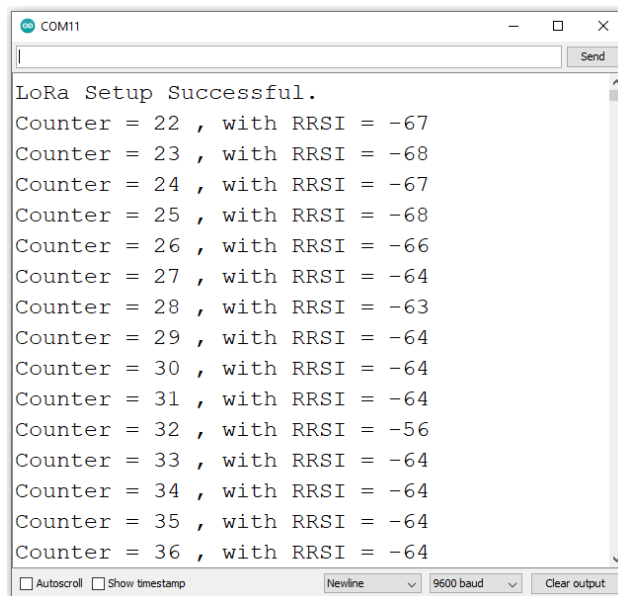
1 // Tutorial #1 - Getting Started with the LoRa
2 // Last update: Feb 13, 2022
3
4 // WIRING:
5 // Pro Mini ..... LoRa
6 // D2 ..... DIO2
7 // D9 ..... RST
8 // D10 ..... NSS
9 // D11 ..... MOSI
10 // D12 ..... MISO
11 // D13 ..... SCK
12
13 // INSTRUCTIONS:
14 // Loading receiver -> Comment COMPILE_TX & uncomment COMPILE_RX
15 // Loading sender -> Uncomment COMPILE_TX & comment COMPILE_RX
16 //#define COMPILE_TX
17 #define COMPILE_RX
18
19 #define freq_US 915E6 // In Canada 915MHz is used
20 #define SS 10 // default = 10. NSS on LoRa
21 #define RST 9 // default = 9, -1 = disable (connect RST pin of MCU with RST of LoRa)
22 #define DIO0 2 // default = 2. Most be interrupt capable.
23 #include <LoRa.h>
24 #include <SPI.h>
25
26 int counter = 0;
27
28 void setup(){
29   Serial.begin(9600);
30   while (!Serial);
31   LoRa.setPins(SS, RST, DIO0);
32   while (!LoRa.begin(915E6)){
33     Serial.println(".");
34     delay(50);
35   }
36   Serial.println("LoRa Setup Successful.");
37 }
38
39 #ifdef COMPILE_TX
40 void loop() {
41   Serial.print("Counter = ");
42   Serial.println(counter);
43   LoRa.beginPacket();
44   LoRa.print(counter);
45   LoRa.endPacket();
46   counter++;
47   delay(2000);
48 }
49 #endif
50
51 #ifdef COMPILE_RX
52 void loop(){
53   int packetSize = LoRa.parsePacket();
54   if (packetSize){
55     Serial.print("Counter = ");
56     while (LoRa.available()){
57       String data = LoRa.readString();
58       Serial.print(data);
59     }
60     Serial.print(" , with RRSI = ");
61     Serial.println(LoRa.packetRssi());
62   }
63 }
64 #endif

```

6 RESULTS AND CONCLUSION

With both the sender and the receiver scripts successfully uploaded, the communication between the LoRa modules should be established. To see the results, the receiver microcontroller should be plugged in to the computer running the Arduino IDE via the USB, and the IDE Monitor opened (“Tools\Serial Monitor”). The Serial.print commands will write the output to the IDE Monitor. If all is successful the monitor should print out an integer that increments every two seconds, similar to what is shown in Figure 6.

The Received Signal Strength Indicator (RSSI) value gives an indication on the received signal strength. The more negative the RSSI, is the weaker the communication link between the sender and receiver. In Figure 6 the RSSI is around -65dBm. In terms of LoRa communication, an RSSI of -30dBm is considered too strong, -50dBm is very strong, and an RSSI of -145 is considered weak. If the LoRa radios are too close together, and the signal is too strong, the transceivers may be swamped, and some packets may be missed. If the RSSI is above -45dBm and it appears that packets are being missed, then simply move the radios about a meter apart (or until the RSSI is below -45dBm).



```
COM11
LoRa Setup Successful.
Counter = 22 , with RSSI = -67
Counter = 23 , with RSSI = -68
Counter = 24 , with RSSI = -67
Counter = 25 , with RSSI = -68
Counter = 26 , with RSSI = -66
Counter = 27 , with RSSI = -64
Counter = 28 , with RSSI = -63
Counter = 29 , with RSSI = -64
Counter = 30 , with RSSI = -64
Counter = 31 , with RSSI = -64
Counter = 32 , with RSSI = -56
Counter = 33 , with RSSI = -64
Counter = 34 , with RSSI = -64
Counter = 35 , with RSSI = -64
Counter = 36 , with RSSI = -64
```

Figure 6: Serial Monitor output of the Rx side.

7 TROUBLESHOOTING

“RF is hard.” – D. Cornish

The following are suggestions for troubleshooting. There are several things that can go wrong, or might prevent the project from working as intended. This is true for any electronic DIY project, but particularly so when dealing with RF devices.

If code is not uploading

1. Try changing the USB cable.
2. Ensure proper port within IDE is selected.
3. Try loading a generic example (e.g. “Hello World”) that came with IDE installation. Ensure that the generic example is working before moving on. For this, disconnect all LoRa connections, so that you can ensure the programmer and microcontroller has not defective.
4. In the IDE software, under File\Preferences enable “Show verbose output during:” on “compilation” and “upload”. Read the output upon trying to upload the script. You can copy and paste parts of the output and search online in forums. More than likely the issue has been covered extensively online.

If hardware is not working

1. Using a multimeter on DC voltage setting, ensure that VCC to GND reads approximately 3.3V at the FTDI USB to TTL Serial Converter module, the Pro Mini, and the LoRa.
2. Disconnect power. Using a multimeter on conductivity setting, check all connections between all three components. Using the probe try to touch the pins directly on each of the modules. In other words, when checking conductivity place the probes directly on the modules, and hence including the breadboard and the LoRa Key. While doing this step also ensure that all connections as per Table 1 and Table 2.
3. While working only on one side (either the sender or the receiver side), try exchanging out one part at a time, while keeping the rest of the setup unchanged. E.g. change the FTDI USB to TTL Serial Converter module with the spare one. Test again. If fails, change out the Pro Mini. Repeat.
4. Throughout the script place output snippets to help debug. Something similar to: `Serial.println("goto x");` where x is a number you change so you know what code has been excluded.