LoRa Setup Writeup

Henry Mitchell

I. Initial Testing

A. Introduction

The purpose of this experiment was to examine the direct wireless communication between two devices with SX1276RF1KAS Transceivers and Arduino Uno boards. For this experiment, no gateway was involved, so source code that allows direct communication between two nodes was used. This experiment implemented an approach using SX1276RF1KAS Transceivers and Arduino Uno boards. Communication was established between the two units before sensors were attached to the client unit, and the data from those sensors were sent to the server unit.

B. Materials

The following were used in developing this system:

- SX1276RF1KAS × 2
  - These are the LoRa units which transmit and receive the data from one Arduino to another
- 915MHz Antenna (yellow) × 2
  - These are the antennae which came with the LoRa units, and are plugged into the "HF" port on the LoRa units
- Arduino Uno board × 2
  - These are the micro-controllers which determine which data to collect and send
- USB 2.0 Type B cable × 2
  - These are to connect the Arduino units to a computer, in order to upload instructions to them
- Jumper Wires
  - These are to connect the Arduino and LoRa units, as well as the Arduino units and sensors
- Required Arduino libraries
  - These are the SPI library and RadioHead library.

The pin connections used in the setup of this system are shown in table I. A photograph of these to components

Purpose	LoRa	Arduino
Power supply	2 (VDD_RF)	3.3 V
	22 (VDD_ANA)	
	34 (VDD_FEM)	
Ground	32 (GND)	GND
SPI	1 (SCK)	D13
	3 (MOSI)	D11
	8 (MISO)	D12
	7 (NSS)	10
Digital I/O	12 (DIO0)	2
	5 (DIO1)	6
	17 (DIO2)	7
Reset	10 (NRESET)	8
RXTX	13 (RXTX)	3

TABLE I
PIN MAPPING FOR THE EXPERIMENT.

connected as described by the table is shown in fig. 1.

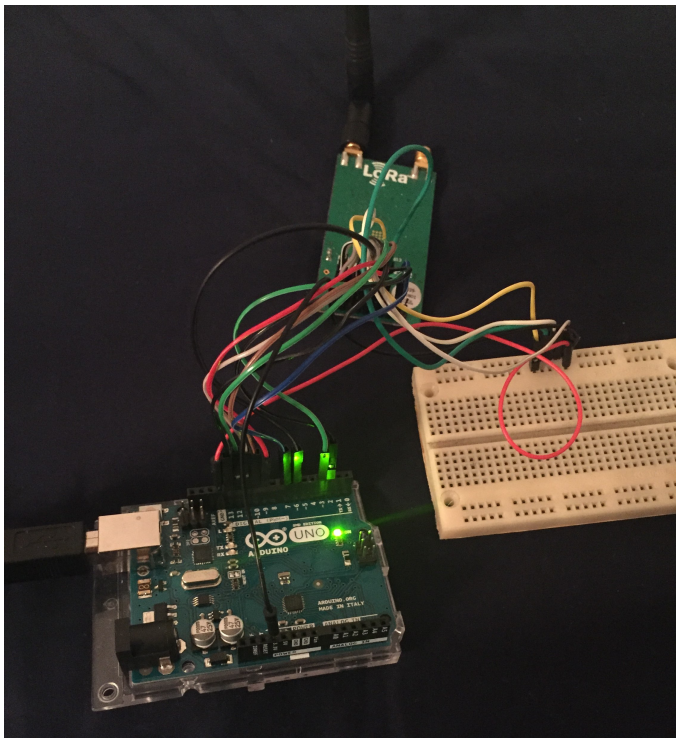

Fig. 1. A photograph of the components connected as described by the pin mapping.

C. Connecting Sensors

Before establishing communication between the two units, sensors must be connected. As things currently stand, the client code is set up for two sensors. One of these is to be connected to pin A0, and the other is to be connected to A1. In future versions of the software, the number of sensors will not be hard-coded, but will simply allow for a plug-and-play style of use. Additionally, any functions that will be used to process the raw data from the sensors must be edited (they are currently called `fix_temp` and `fix_sound`, and are in arduino/client/client.ino*). Sensors will also need to be connected to the Arduino's 3.3 V and ground pins.

D. Procedure

After all components have been connected as described, All that remains is to upload the source code. The unit with the sensors attached is the client unit, and will therefore have the client code (arduino/client/client.ino) uploaded to it. The other unit is the server unit, and will therefore have the server code (arduino/server/server.ino) uploaded to it. This is the unit for which it is useful to have the serial monitor open, in order to verify that it is receiving data from the client unit.

II. Gateway Communication

A. Introduction

The goal of this experiment was to establish communication between wireless transceivers and a wireless gateway. This experiment implemented an approach using SX1276RF1KAS Transceivers and Arduino Uno boards, as well as a Multi-Tech Systems MultiConnect® Conduit™ gateway. It connects the gateway and Arduino units to a network registered with The Things Network.

*All file paths are relative to the main git repository.

B. Materials

The following were used in developing this system:

- SX1276RF1KAS × 1
  - This is the LoRa unit which transmits the data from the Arduino to the gateway
- 915MHz Antenna (yellow) × 1
  - This is the antenna which came with the LoRa units, and is plugged into the "HF" port on the LoRa unit
- Arduino Uno board × 1
  - This is the micro-controller which determines which data to collect and send
- USB 2.0 Type B cable × 1
  - This is to connect the Arduino unit to a computer, in order to upload instructions to them
- Jumper Wires
  - These are to connect the Arduino and LoRa unit
- Multi-Tech Systems MultiConnect® Conduit™ (MTCDT-LAT1-247A)
  - This is the gateway to connect the LoRa network to the cloud
- Required Arduino libraries
  - These are the SPI library and LMIC.

We are using the same pin mapping as in the previous section, shown in table I.

C. Caveats

As things currently stand, the gateway can not be connected to a secured network. While the network can have a WPA2 password, it can not have a firewall or any additional security measures. You can test your internet connection from the gateway by `ping`ing Google's `8.8.8.8` server. If you see results like those shown in fig. 2, then your network is sufficiently exposed.

```
admin@mtcdt:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=56 time=26.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=56 time=24.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=56 time=38.3 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=56 time=22.7 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 22.744/28.022/38.344/6.091 ms
```

Fig. 2. The expected output of a `ping 8.8.8.8` from the gateway.

Additionally, as of the time of this writing, the `config.h` file for the LMIC library is not configured for this use. The necessary changes are as follows:

- Comment the line that says **`#define`** `CFG_eu868 1` and un-comment the line that says **`#define`** `CFG_us915 1`
- Un-comment the line that says **`#define`** `DISABLE_INVERT_IQ_ON_RX`

D. Procedure

In order to establish communication between the Arduino and the gateway, we need to prepare both components. To prepare the gateway to act as an interface between the Arduino and the cloud, I registered it with The Things Network and followed the instructions provided by The Things Network.

Once all of the components are registered with The Things Network, we can set up our Arduino. After logging into The Things Network's website, we need to navigate to the "Applications" pane. This will allow us to get the necessary information to allow our Arduinos to join the LoRa network. We need a device EUI, an application

EUI, and an application key in order for the Arduino to join the network. These are available on the console, as shown in fig. 3*.

Fig. 3. The device overview from The Things Network console.

These get put into the over-the-air-activation (OTAA) script at arduino/ttn_otaa/ttn_otaa.ino as shown in fig. 4. Once these edits have been made, the script is uploaded to the Arduino, which tries to join the network established

```
41 static const u1_t PROGMEM APPEUI[8]={ 0xBC, 0x78, 0x00, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 };
42 void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}
43
44 // This should also be in little endian format, see above.
45 static const u1_t PROGMEM DEVEUI[8]={ 0xBC, 0xFB, 0xE1, 0x3B, 0x31, 0x75, 0x91, 0x00 };
46 void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}
47
48 // This key should be in big endian format (or, since it is not really a
49 // number but a block of memory, endianness does not really apply). In
50 // practice, a key taken from ttnctl can be copied as-is.
51 // The key shown here is the semtech default key.
52 static const u1_t PROGMEM APPKEY[16]={ [redacted] };
53 void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}
```

Fig. 4. The placement of the information from the The Things Network console.

by the gateway.

E. Results

1) First Iteration: As things currently stand, the gateway is able to receive transmissions from the Arduino, but the Arduino is unable to receive the acknowledge that the gateway sends upon OTAA. However, the gateway is receiving data from the Arduino, and forwarding said data to The Things Network, where it is accessible. We can see the Arduino's attempts at establishing a connection in fig. 5.

However, we can see that the gateway is receiving the data that the Arduino is transmitting, because the data are appearing on The Thing Network's console. The gateway's input and output are shown together in fig. 6, while only the data received from the Arduino are shown in fig. 7.

2) Second Iteration: After much troubleshooting, the Arduino was able to receive the acknowledge from the gateway. This was achieved by changing the tolerance on the Arduino's clock, adding

```
1 LMIC_setClockError(MAX_CLOCK_ERROR * 1 / 100);
```

to the OTAA script's `setup` function. This allowed for a stable connection.

The issue with this connection is that the range is extremely limited, with a maximum of approximately 20 m.

*The app key has been blurred out, as it should be kept secure to prevent new devices from being added to the network without administrator permission.

```
Starting
Packet queued
153: EV_JOINING
3806440: EV_JOIN_FAILED
4664522: EV_JOIN_FAILED
```

Fig. 5. The Arduino's attempts to establish a connection in the network. Because it is not receiving the gateway's ACK signal, it is viewing its attempts as failures.

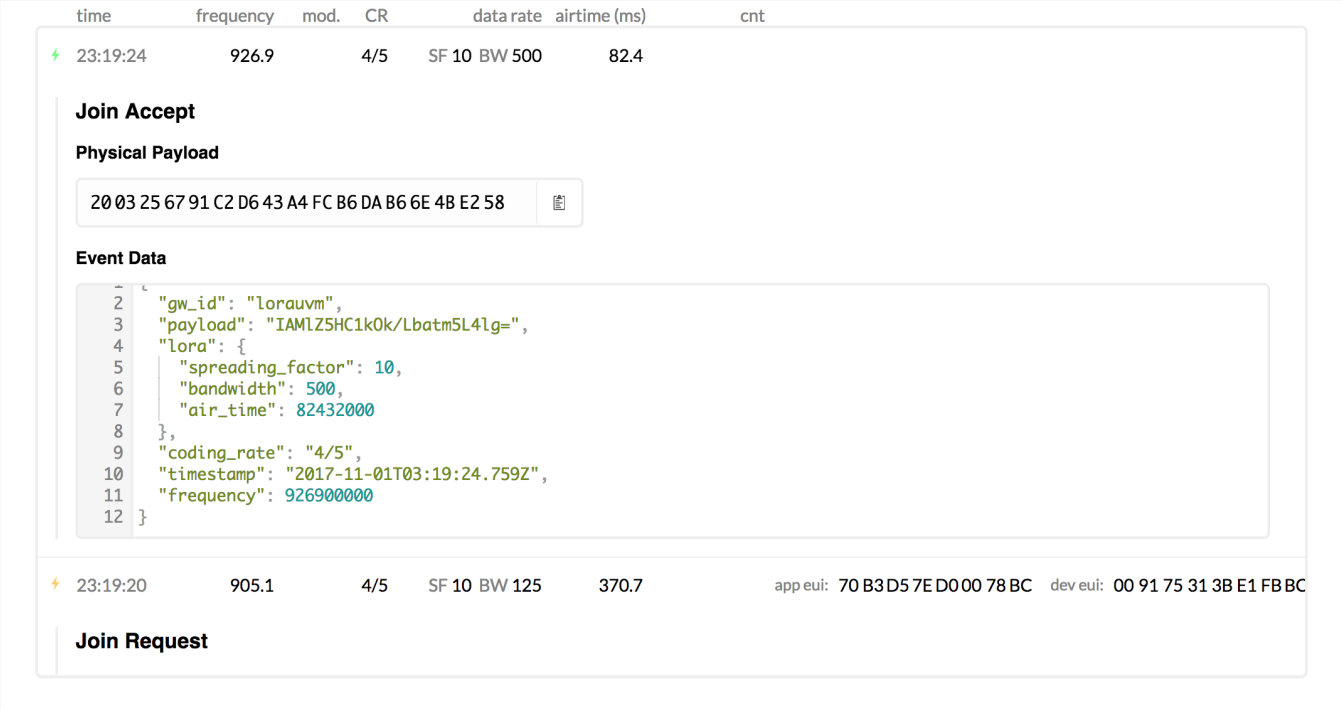

Fig. 6. The data received and transmitted by the gateway, as they appear on The Things Network's console.

F. Discussion

The next step would be to find ways to extend the range of communication between the Arduino and the gateway.

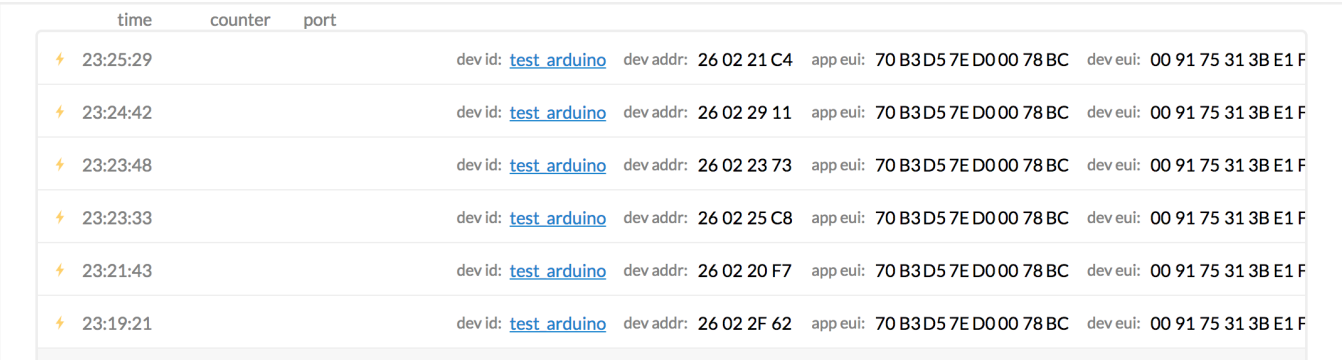

Fig. 7. The data transmitted by the Arduino, as they appear on The Things Network's console.