Find It!

Using 2.4 GHz to determine distance and direction to your lost item

By **Don Powrie** (USA)

Of the numerous technologies currently available to help you locate a lost item, most have the same limitations: they are only useful if the items are within Bluetooth's 30- to 100-foot (10 - 30 m) distance limit, or they require you to purchase a cellular modem and pay a monthly service fee so that you can send GPS data across the cellular network. This article will show you how to locate items that are up to miles away (without a cell phone, cellular network or GPS receiver) — not only showing you the distance, but also the direction to that item!

Applications for this type of locator are probably endless, but I think the ultimate would be a dog locator. A good friend of mine had a beautiful auburn Doberman that loved to chase rabbits through our suburban neighbourhood. Apollo was very large and very strong; and unless you had a tight grip on the leash at the moment he spotted a rabbit, he would be gone — out of sight — in an instant. Whenever this happened after the sun went down, it could take hours of searching the neighbourhood through a thousand potential hiding places between houses and in the adjacent alleys.

SX1280, 2.4 GHz and ranging

While it's true that 2.4 GHz has inherent limitations such as its inability to easily penetrate walls and other solid objects, it does have the advantage of being approved for license-free use worldwide. Going a step further, Semtech™ added LoRa® modulation to their SX1280 transceiver IC for applications that require greater link budget with a resulting improved range. There is a lot of information about LoRa modulation available on the web, so we need not go into it here. But, suffice it to say, the receive sensitivity for LoRa mode in the SX1280 is significantly improved as compared to most, if not all, non-LoRa 2.4-GHz transceivers. Granted, this improvement in sensitivity is derived via LoRa "spreading" within the digital realm, but the net effect is the same — vastly improved TX/RX distance between transceivers.

Figure 1. DLP-RFS1280 2.4-GHz LoRa Transceiver.

Long-range LoRa mode notwithstanding, I think the feature that sets the SX1280 apart from the rest is its industry-unique, built-in Ranging feature. Ranging works by sharing an ID between two SX1280-based transceivers by declaring one the Master, and then bouncing data packets off a Slave transceiver in LoRa mode and measuring the time of flight of those packets in order to calculate distance between the two. This may be an oversimplification, but the primary takeaway is the understanding that this feature is possible due to the high frequency clock in the Master (2.4 GHz) driving a 24-bit counter that measures the round-trip time of the data packet and the Slave's ability to respond to these "ranging" packets without host microcontroller intervention. Once placed in Ranging mode, an SX1280-based Slave transceiver will receive and retransmit ranging packets purely from silicon, thereby providing the fastest possible turnaround time for the packet.

For my locator device, I selected DLP Design's DLP-RFS1280 pre-certified module (**Figure 1**) due in large part to its on-board chip antenna. If this locating device performed as well as I had envisioned, I would want as small an antenna as possible to keep the overall device compact.

Pairing

For any two RF transceivers to work together exclusively (ignoring other transceivers within range), they must share their IDs with each other. This process is called pairing. In this application, the host microcontroller is the STMicroelectronics™ STM32L073 32-bit device, implemented on a Nucleo microcontroller development board. Each STM32L073 has a unique ID in silicon that can be read from address 0x1FF80050. Of the 32 bits read from this address, I discarded 16, keeping the other 16 for the ID that can be sent to another transceiver.

To perform the pairing, a standardised set of RF parameters (TX/RX frequency, modulation type, bandwidth, etc.) are selected, and the ID is transmitted after

any time the microcontroller is powered up or reset. To pair two transceivers, the transceiver receiving the ID is placed in a wait mode with the standard parameters selected, and the other transceiver is simply reset. This process is repeated for the other transceiver, and the pairing process is complete. (Any non-broadcast packets transmitted from this point forward must have the destination transceiver's ID in the payload or the packet is ignored.)

Time of flight

In order to measure the time of flight of an RF packet travelling at nearly the speed of light, you need a very high-resolution counter. The use of a 2.4-GHz clock in the SX1280 incrementing a 24-bit counter that ticks until the master receives the Ranging reply yields a timing resolution of about 400 picoseconds. Considering that the speed of light is about one foot per nanosecond, this yields a distance resolution of about 6 inches (15 cms) — a perfect system for measuring the round-trip time of Ranging packets and determining the distance between transceivers.

One of the hurdles in measuring the time of flight of RF packets is the effect of multipath or reflections. A single packet arriving at the receiver may have taken more than one path to get there, bouncing off objects along the way. For this reason, the RF carrier frequency is changed for every Ranging packet sent for a total of 40 channels between 2.402 and 2.48 GHz, and the process is repeated several times. The resulting time of flight for all of these packets is accumulated and used to compute the mean value so that the estimated distance between transceivers can be calculated.

System calibration

A detailed description of how one goes about calibrating this Ranging system, whereby the hardware implementation is characterised so that the desired accuracy can be achieved, is beyond the scope of this article. However, one way to go about it is to measure the time of flight of a standard distance between transceivers with no possibility of multipath signals (e.g. when using a 100-foot (30-m) coax cable), and base calculated distances on this standard in conjunction with the known speed of light. Another is to collect real-world data from outdoor, line-of-sight environments across a large number of distances ranging from 2 feet to 2 miles (60 cm to 3.2 km), and then use curve fitting to create polynomial equations based upon this data to calculate the distance. I used the latter of these two methods to calibrate my system, and **Figures 2 and 3** show our data-collection setup.

Figure 2. Short-range (2 – 900 ft) (0.6 – 300 m) calibration data collection.

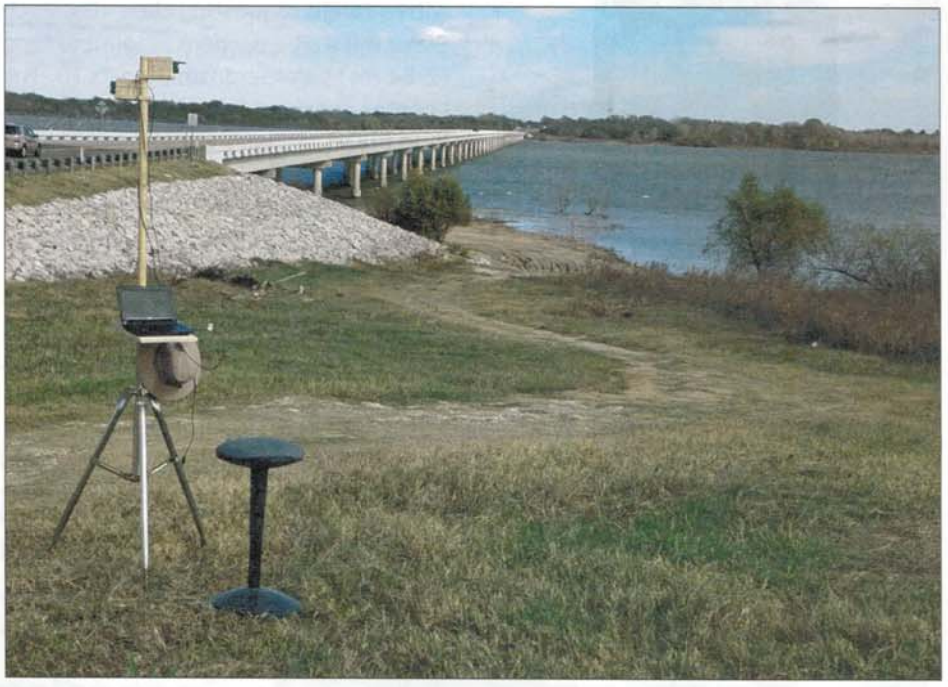

Figure 3. Long-range calibration data collection.

For distances out to 2+ miles (3+ km), we visited an area lake where we could be guaranteed good line of sight across long distances, see Figure 3.

Direction locator

Knowing the distance from your position to another transceiver is a good start towards finding your lost item, but in which direction should you begin walking? My solution was to create an electronic locator of sorts (**Figure 4**) that uses a high-gain Yagi antenna [1]. I found this antenna design on the web, and modified it for slightly higher directionality by adding a couple of directors. The idea for determining direction is to use the RSSI (receive signal strength) value as reported by the Slave transceiver. In this scenario, the Locator sends a LoRa packet with Spreading Factor 12 to the Slave that measures the RSSI level of that packet and sends this data back to the Locator. Since the Locator is transmitting using a directional antenna, the Slave receives more signal (higher RSSI) when the Locator's antenna is pointed directly at the Slave. The intensity of this RSSI signal is then reported by the Locator both as an audible tone and using visual indicators on an LCD display to indicate direction to the target.

To operate the Locator, simply select a Slave ID from the list of learned IDs (see below), and horizontally sweep the Yagi antenna slowly in all directions around your body while listening to the tone. As the antenna points more directly at the target, the pitch of the tone will increase. Once the direction is known, set the switch to Range to get a read on the distance and start walking. The distance as you approach your target will be reported in real time on the display.

Figure 4. Locator with Yagi antenna.

Learn mode

Instead of having one transceiver paired exclusively with another, I designed the Locator to be able to learn the IDs of up to 50 transceivers. To enter Learn mode, press and hold the Up button, and power up the Locator by flipping the power switch on the bottom of the board. The Locator will then broadcast a packet that is received by all Slave transceivers within range requesting their IDs. Each Slave will wait a random amount of time (2-64 ms) before transmitting its ID. The Locator receives and stores these IDs in EEPROM memory so that one can be selected for the Search process, and then sends a packet to each new ID instructing it to remain quiet (i.e. not respond to the next Learn request).

To select a transceiver to locate, simply use the Up and Down buttons to scroll through and select a Slave transceiver ID. The STM32L073's built-in EEPROM memory is also used to store the selected Slave's transceiver ID, so it is used by default on the next power-up.

Slave transceivers

For the Slave transceivers, I used the DLP Design DLP-RFS1280ACT (**Figure 5**) since its firmware is nearly identical to that which is utilised in the Locator device.

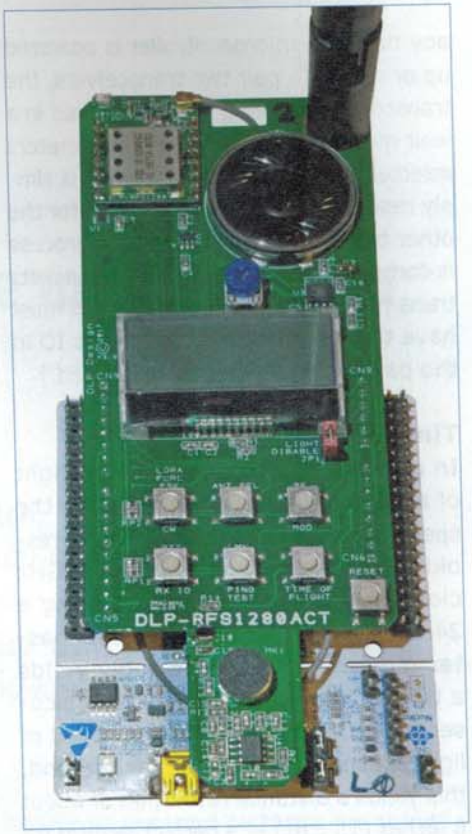

Figure 5. DLP-RFS1280ACT used as a Slave transceiver.

The project source code

For this project I started with C++ demo source code from Semtech for the SX1280 Demonstration Platform found on the Mbed website. (Most of the heavy lifting in the source code development was done by the engineers at Semtech using Mbed libraries.) All I had to do was convert it over from C++ to straight C code since I'm much more comfortable with the C programming language for microcontroller firmware development. Making this project even more appealing was the availability of a free C compiler from Keil for the STM32F0 and STM32L0 devices. Their MDK compiler [2] provides the ability to set breakpoints, single step through code while watching variables, etc. While Mbed is great for bringing up a new project quickly due to its built-in libraries, when you get into serious firmware development, a good debugger is indispensable.

The next step was to select a different STM32 microcontroller. The device used by Semtech for their demo hardware was the STM32L476 by way of the NUCLEO-L476RG. This is a very powerful micro with a ton of flash program memory (1 MB), but it's also a bit power hungry.

This is a great device for use in development tools when you don't know what you are going to need down the road and you want to make sure you don't run out of horsepower or memory. In this case, I chose to go with the STM32L073 to save power (which is appropriate for devices like a battery-driven pet collar), to save money and because Mbed supports the NUCLEO-L073RZ.

I created a nearly empty shell program (keeping the SPI interface code and a few other items) on the Mbed website, and immediately exported it to MDK. Doing so brought along the entire Mbed library, so I could take advantage of other handy Mbed features if needed. From there, the bulk of the code conversion to straight C was fairly simple. Again, most of the code for working with the SX1280 was already done for me, so the overall process only took a few days. The resulting C source code is fairly easy to follow and understand, and is available for download from the DLP Design website as well as the *ElektorLabs Magazine* web page supporting this publication [3].

Parting comments

By using a Yagi antenna and LoRa modulation, I was able to achieve very positive results with my locator system. I successfully located other transceivers through houses in our neighbourhood and between several sets of locations within our home.

Expanding upon the concept, I took the system to three different, large retail establishments to see how it would perform in environments with lots of metal shelving, customers, etc. (I'll call these three locations Store L, Store T and Store W.) In each of the three stores, I placed a Slave transceiver in one corner and took the Locator to the exact opposite corner of the store. In Store L, I was able to easily ping the Slave transceiver with no dropped packets at a reported distance of 505 feet (168 m). The problem was determining an exact direction to the Slave. Regardless of where I pointed the Locator, the Slave always reported approximately the same RSSI value. I was able to improve upon this result by moving to the centre of the store where the RSSI values were higher when pointing at the Slave. So, in this test environment, the Locator could immediately report the presence of the Slave target and the distance; but in order to determine the direction, I had to move around a bit. In Store T, I once again placed a Slave transceiver in one corner of the store. This time I was not able to ping the Slave from the opposite corner of the store at a distance of ~575 feet (~192 m). (It wasn't until I positioned myself in the centre of the store that I was able to get a ping reply. From there I could get good direction readings as well. Store W gave the same results as Store L with a corner-to-corner distance of 505 feet (168 m). From this I concluded that so long as I'm willing to walk around a bit, the Locator works well within a large retail environment, especially if I start at the centre of the store. As expected, the best results are achieved when used outdoor using line of sight. There I was able to determine distance and direction to Slave transceivers located up to ~2 miles (~3.2 km) away. In the end, while this system might not curb Apollo's yearning to catch that rabbit, it would certainly help my friend find his pet in the wee hours of the morning! ⏮

180175-01

References and Web Links

[1] Application Note DN034 — SWRA350, 2.4 GHz YAGI PCB Antenna, By Richard Wallace & Steve Dunbar
www.ti.com/general/docs/litabsmultiplefilelist.tsp?literatureNumber=swra350

[2] ARM KEIL Compiler, www2.keil.com/stmicroelectronics-stm32/mdk

[3] Source code files for project: www.elektormagazine.com/180175-01