John Iddon
Physics 4990

LoRa Logs

The past few weeks prior to June 30:

- Transmit and receive tests have been successful.
- The 915MHz LoRa chips are legal in North America (unlike the 433 MHz and 868 MHz variants which would need one to be a HAM radio operator) and have been soldered to back plates that provide easy through-hole pins to mount with. The back plate labels are not correct for the LoRa modules as the back plates are originally intended for ESP8266 chips that have the same form factor as the LoRa modules. The chosen 915MHz LoRa chips are of the typical SX127X family present in almost all LoRa receivers.
- The HopeRFM chips appear to just be re-branded SX127X chips.
- The SX127X chips are 3.3V logic, and while they are not listed as 5V tolerant, they seemed to be in short tests (though I wouldn't want to push it past such and destroy good transceivers).
- Working with ESP8266 modules as the mircocontroller for interfacing with the SX127X chips proved to be more difficult than expected as one needs to use the “Arduino” pin numbering scheme (since we're working in the Arduino IDE) for the ESP8266, which is different from the board numbering, and the many other numbering schemes as well.

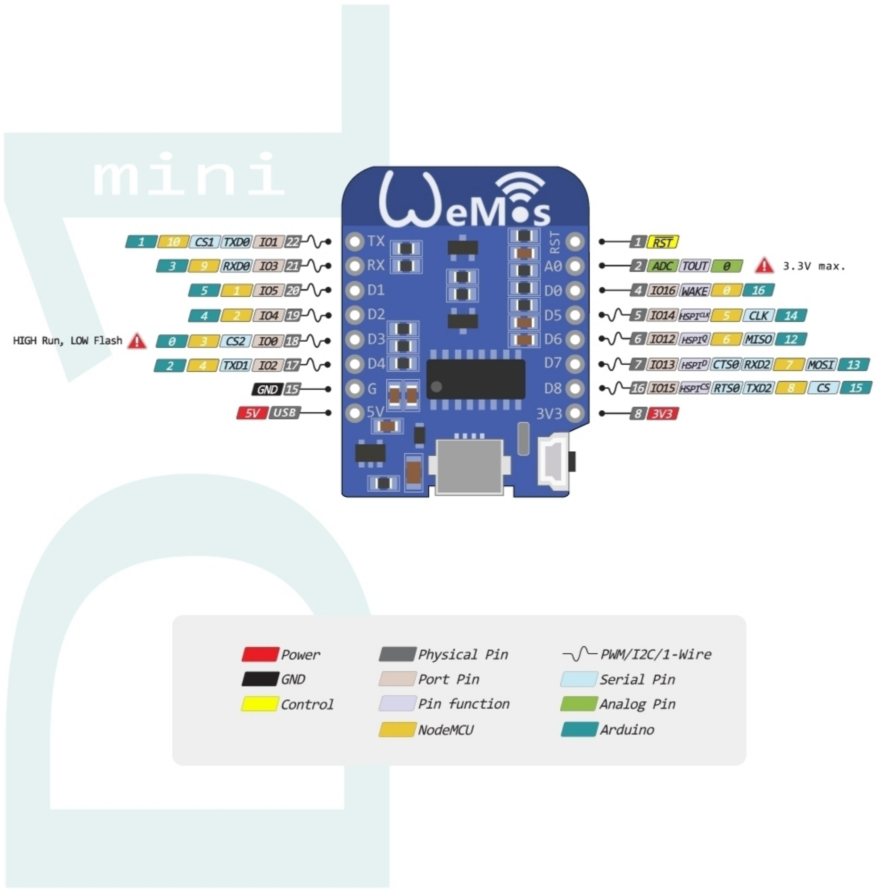


Figure 1: ESP8266 pin diagram with various layout numbering schemes. Image sourced from: https://escapequotes.net/esp8266-wemos-d1-mini-pins-and-diagram/

- The website template I created was used to make a website to hold the live LoRa data once sensors are constructed. The website is not yet live.

June 30, 2019

- Power supply options:
  - One interesting method to me has been the idea of directly using power from a solar panel to do transmissions during the day from a remote sensor, that is to say, with no battery attached. I was hoping a typical USB car charger would work, and using two small panels in parallel I was able to run a large USB load (a USB high speed fan), though it dragged down the voltage significantly. It's perhaps possible then that with the most likely lower power consumption of the LoRa+microcontroller setup that it could be run directly from a solar panel without a battery in use, which would make it a very very long term sensor if it indeed will work.
  - I've checked out some of the 18650 Li-ion batteries I have, and two of them have built in protection circuits, or at least claim to have, which should be useful. I also have acquired 18650 battery holders and boost converters that can bring up the 18650's voltage to 5 volts, so they could be a useful option for powering sensors as well, and with a lithium cell charge controller circuit could easily be charged with a solar panel.
- Website:
  - LoRa website is now available live at https://johniddon.github.io/ . Bandwidth seems to be limited as large pictures take a long time to load, but that's perfectly fine for what I need for this website. Hosting is free too, you simply create a repository named username.github.io where username is your username, and then put your HTML files in the repository and boom, the website is live.
  - One issue with the website I ran into was that the repository needs to be public, as I had initially set it to private and it didn't work.
  - Another issue one may encounter is needing to wait a little time before it goes live and appears initially. It seems to work fine, it just takes patience when GitHub initially processes everything.

July 1, 2019

- Basic transmit and receive test:
  - For some reason my laptop has forgotten the boards configuration for the esp8266 based chips. Under preferences in the Arduino IDE add http://arduino.esp8266.com/stable/package_esp8266com_index.json in the "Additional Boards Manager URLs" section. Clicking OK, then go to Tools --> Board --> Board Manager. Install esp8266, specifically the one by ESP8266 Community.
  - I'm going to redo my code for the basic transmit and receive tests so that they're a touch more elegant.
  - Arduino IDE seems to have messed itself up as nothing will compile properly, not even the simplest of examples that I know should be fine and are designed to run on the selected boards, yet I still get errors along the lines of "Cannot compile for board Wemos D1". I'll reinstall tomorrow.

July 2, 2019

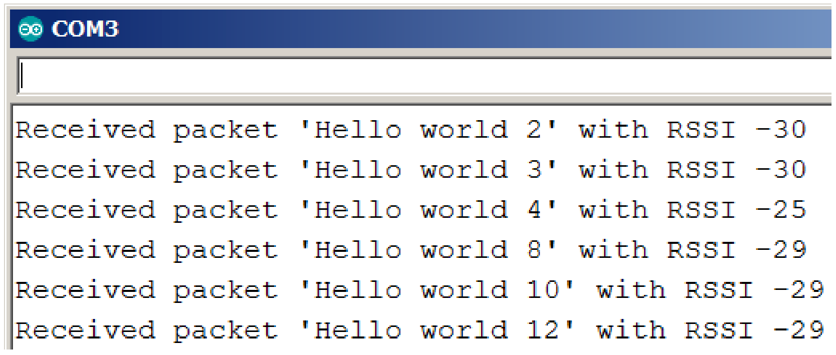
- Reinstalling Arduino IDE
    - Uninstalled existing Arduino IDE.
    - Manually deleted Arduino file in Documents under my user.
    - Reinstalled Arduino IDE.
        - Specifically I used the portable ZIP file version, so no “install” is required as it's just an executable.
    - Put in the additional boards manager as explained yesterday and install ESP8266 boards.
    - Reinstall LoRa library by Sandeep Mistry from the Libraries Manager.
    - Success! I can now compile example programs properly.
- Rebuilding my receive and transmit programs with thorough commenting.
    - Receive
        - Based upon the example receive program from the LoRa.h library, my receive program will print out received LoRa packets as well as the RSSI (Received Signal Strength Indicator) of that packet to the serial monitor.
        - In the box below is my code.

```
/*
John Iddon - LoRa Rx Basics.
Uses code from examples of the LoRa.h library by Sandeep Mistry.
*/
#include <ESP8266WiFi.h>
#include <SPI.h>
#include <LoRa.h> //Includes LoRa.h library by Sandeep Mistry

//pin definitions
#define ss 15 //defines slave select (SS) pin, AKA chip select.
#define rst 14 //defines reset pin.
#define dio0 5 //defines dio0 pin.

void setup()
{
  //open serial port.
  Serial.begin(115200);
  while (!Serial);
  Serial.println("LoRa Receiver");

  LoRa.setPins(ss, rst, dio0); //set pins


  //433E6 for Asia
  //866E6 for Europe
  //915E6 for North America
  while (!LoRa.begin(915E6)) //sets chip frequency. 915MHz is what's legal in NA without being a HAM.
  {
    Serial.println("b."); //test at this point in program to ensure functionality on serial monitor.
    delay(500);
  }
  // SyncWord helps keep other LoRa traffic out by ensuring only messages with the same SyncWord are output/used.
  // Ranges from 0-0xFF
  LoRa.setSyncWord(0xF3);
  Serial.println("LoRa Initializing OK!");
}


void loop()
{
  // Look to receive any packets if they're available.
  int packetSize = LoRa.parsePacket();
  if (packetSize)
  {
    // if packet has any size to it AKA exists.
    Serial.print("Received packet '");

    //Display the packet to serial monitor.
    while (LoRa.available())
    {
      String LoRaData = LoRa.readString();
      Serial.print(LoRaData);
    }
    // Print RSSI for packet to serial monitor.
    Serial.print("' with RSSI ");
    Serial.println(LoRa.packetRssi());
  }
}
```

    - The receiver code works as desired when using an example transmitter.
  - Transmit
    - Also based on the example from the LoRa.h library. Useful in that it will count upward so that one can ensure all of the packets are being received, something that will be especially useful when doing range tests to see when we start to miss certain packets.
    - The following code is used:

```
#include <ESP8266WiFi.h>
#include <SPI.h>
#include <LoRa.h>

#define ss 15 //defines slave select (SS) pin, AKA chip select.
#define rst 14 //defines reset pin.
#define dio0 5 //defines dio0 pin.


int counter = 0; //variable to be incremented for each packet sent

void setup() {
  Serial.begin(115200); //Open serial port.
  while (!Serial);

  Serial.println("LoRa Sender");

  LoRa.setPins(ss, rst, dio0); //set pins


  if (!LoRa.begin(915E6)) { //Sets LoRa chip to 915MHz.
    Serial.println("Starting LoRa failed!");//Prints if initialization of chip fails.
    while (1);
  }
}

void loop() {
  Serial.print("Sending packet: ");
  Serial.println(counter);

  // send packet
  LoRa.beginPacket(); //starts packet.
  LoRa.print("Hello world "); //Adds string to packet.
  LoRa.print(counter);//Appends counter variable value to packet.
  LoRa.endPacket(); //Ends the packet and sends it over LoRa.

  counter++; //increments the counter.

  delay(5000); //Limits transmitter to broadcast once every 5 seconds.
}
```

    - Interestingly, now when using the numbered sends, I see that occasionally we're missing a few transmissions, as shown in this serial monitor screen shot in Figure 2. The true question is however, is the issue with the transmitter or the receiver.

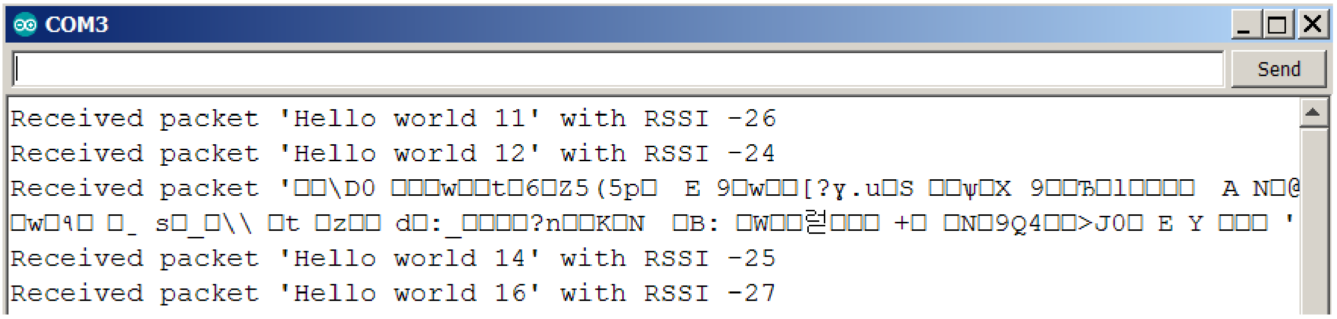


```
Received packet 'Hello world 2' with RSSI -30
Received packet 'Hello world 3' with RSSI -30
Received packet 'Hello world 4' with RSSI -25
Received packet 'Hello world 8' with RSSI -29
Received packet 'Hello world 10' with RSSI -29
Received packet 'Hello world 12' with RSSI -29
```

Figure 2: Screenshot of the serial monitor output from the LoRa receiver. It can be seen in the counting numbers how certain transmissions are being lost.

    - Part of me also is leery of the serial monitor, so let's slow that down a touch first to 9600 baud (changed in program line Serial.begin(9600);). At this lower baud rate of the serial port on the receiver module I miss messages between every received one, which leads me to believe perhaps I'm overwhelming the serial port.

- Let's set it the serial port on the receiver back to 115200 baud, as at 115200 baud the serial port of the transmitter never misses any of the prints of the counter, leading me to believe it's either the receiver or it's something to do with the LoRa modules themselves getting overwhelmed at transmitting at such a fast pace perhaps? Let's try slowing down the LoRa transmissions on the transmitter to once every 10 seconds to start off.
- With the transmitter transmitting once every 10 seconds we're still missing transmissions, at once case we even got a string of garbage received, as shown in Figure 3. I wonder if we're perhaps dealing with some sort of interference source.



```
Received packet 'Hello world 11' with RSSI -26
Received packet 'Hello world 12' with RSSI -24
Received packet '□□\D0 □□□w□□t□6□Z5(5p□  E 9□w□□[?ɣ.u□S □□ψ□X 9□□Ђ□1□□□□  A N□@
□w□٩□ □_ s□_□\\ □t □z□□ d□:_□□□□?n□□K□N  □B: □W□□럴□□□ +□ □N□9Q4□□>J0□ E Y □□□ '
Received packet 'Hello world 14' with RSSI -25
Received packet 'Hello world 16' with RSSI -27
```

Figure 3: Serial monitor output with transmitter sending once every 10 seconds. Transmissions are still being missed by the receiver, and one transmission received even was a jumbled mess, perhaps due to interference.

- Sometimes we're missing large gaps of transmissions, and other times we get several in a row, so the issue seems to be inconsistent, which is something interference also likes to be. Let's try changing the sync word on both the transmitter and receiver and see if that helps or not.
- Changing the sync word on only the receiver still let it receive some of the messages from the transmitter, which makes me question if the sync word portion of the library is functional or not.
- It appears the joke is on me, so to speak, as I didn't have a sync word set in the transmitter. Now the receiver seems to be getting every sent packet. I'm glad this was a simple mistake and not a larger issue at play.
- Occasionally there is still a missed or jumbled packet, but only 1, never the several in a row I was seeing before. This seems acceptable for now, and is something I can easily overcome with a simple call back in future implementations with a confirmation of a received packet. I've received over 100 packets with only 3 errors, that's probably more reliable than most of our internet connections to remote servers, so I'll happily call this a success for now. Below is the fixed transmitter code.

```
/*
 * Code by John Iddon
 * Includes code from examples in the LoRa.h library from Sandeep Mistry
 */

#include <ESP8266WiFi.h>
#include <SPI.h>
#include <LoRa.h>

#define ss 15 //defines slave select (SS) pin, AKA chip select.
#define rst 14 //defines reset pin.
#define dio0 5 //defines dio0 pin.


int counter = 0; //variable to be incremented for each packet sent

void setup() {
  Serial.begin(115200); //Open serial port.
  while (!Serial);

  Serial.println("LoRa Sender");

  LoRa.setPins(ss, rst, dio0); //set pins


  if (!LoRa.begin(915E6)) { //Sets LoRa chip to 915MHz.
    Serial.println("Starting LoRa failed!");//Prints if initialization of chip fails.
    while (1);
  }
  // SyncWord helps keep other LoRa traffic out by ensuring only messages with the same SyncWord are output/used.
  // Ranges from 0-0xFF
  LoRa.setSyncWord(0xF3);
}

void loop() {
  Serial.print("Sending packet: ");
  Serial.println(counter);

  // send packet
  LoRa.beginPacket(); //starts packet.
  LoRa.print("Hello world "); //Adds string to packet.
  LoRa.print(counter);//Appends counter variable value to packet.
  LoRa.endPacket(); //Ends the packet and sends it over LoRa.

  counter++; //increments the counter.

  delay(5000); //Limits transmitter to broadcast once every 5 seconds.
}
```

July 3, 2019

- Website
  - Website has been updated, making the link bar more compatible with smaller screens, adding sample graphs from my solar voltage sensors to the live data page, and I've also added information to the Guides + Build Log page. This build log is now also attached to the website as a PDF that opens in a new tab for the viewer when they click the link.
  - QR codes successfully can redirect a phone to the website, which will be handy for marking the sensors with. I'll do up some high resolution ones soon, and will perhaps see if I can cut them out of vinyl accurately enough to be scannable as that would be useful for marking the outsides of sensor containers.