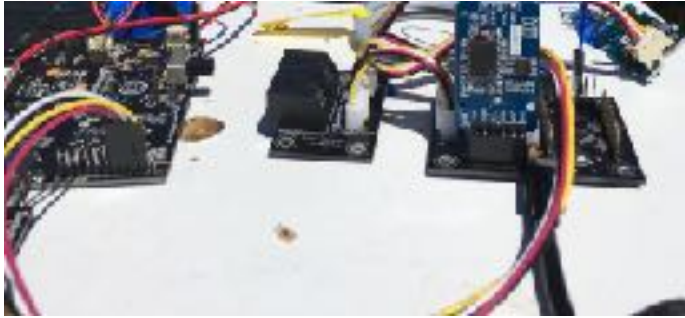


This is a Solar Powered Mini Pro LP Arduino based LoRa bi-directional radio link designed for hooking up the WeatherRack and AM2315 outside and then transmitting the data to a computer inside rather than using wires.



Features and Benefits:

- 6000 Meter range
- LoRa Standard
- Solar Powered System - no wires!
- Bi-directional link
- Grove connector based kit - no soldering
- Open source software included
- Supported by the GroveWeatherPi Raspberry Pi based Weather station
- Supported by the SwitchDoc Labs Raspberry Pi Data Logger software
- Supports both 3.3V and 5V I2C
- Quantity Discounts Available
- Immediate Availability

Introduction



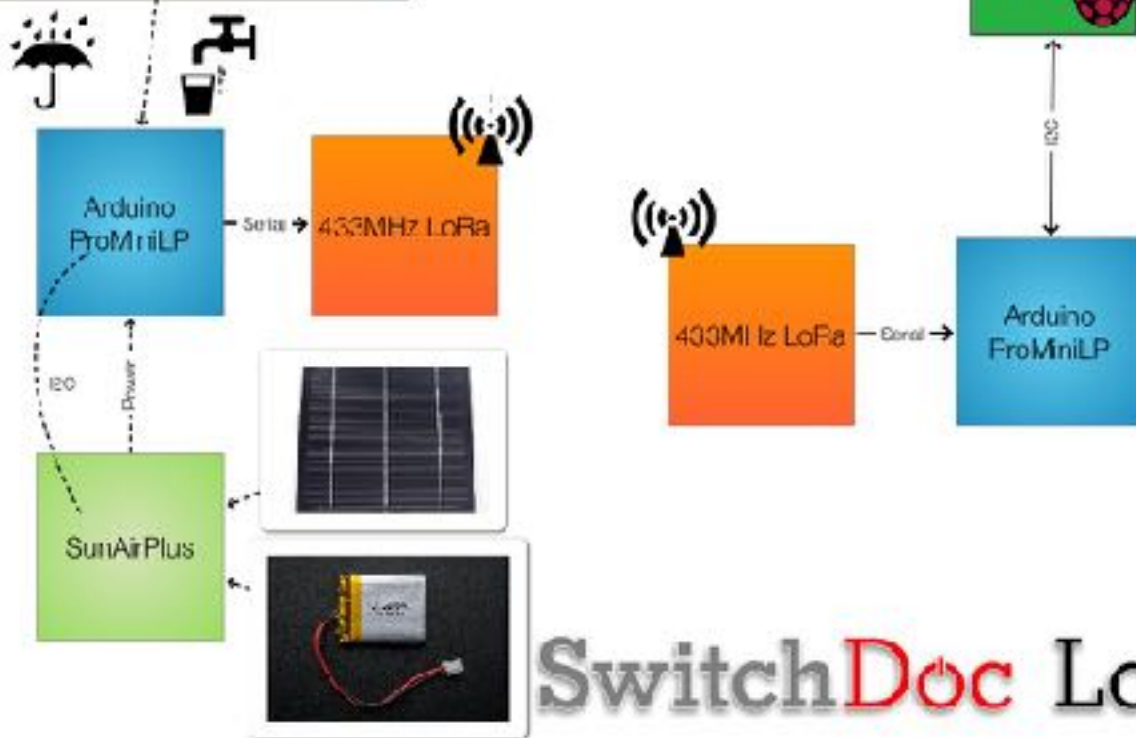
The issue is sometimes you don't want to run a wire all the way from the Weather Station to the wind and rain sensor. Using the Mini Pro LP, we built a WeatherRack reader and then we use a transmitter to send it back to the GroveWeatherPi station inside. We then added solar power to the system.

The Solar WXLink LoRa product kit contains two Mini Pro LP Arduino boards, a WXLinkWR Weather Rack Interface board, two LoRa transceivers, SunAirPlus Solar Power Controller and Data Collector, a 330mA 6V Solar Cell and one DS3231 RTC Board.

Theory of Operation



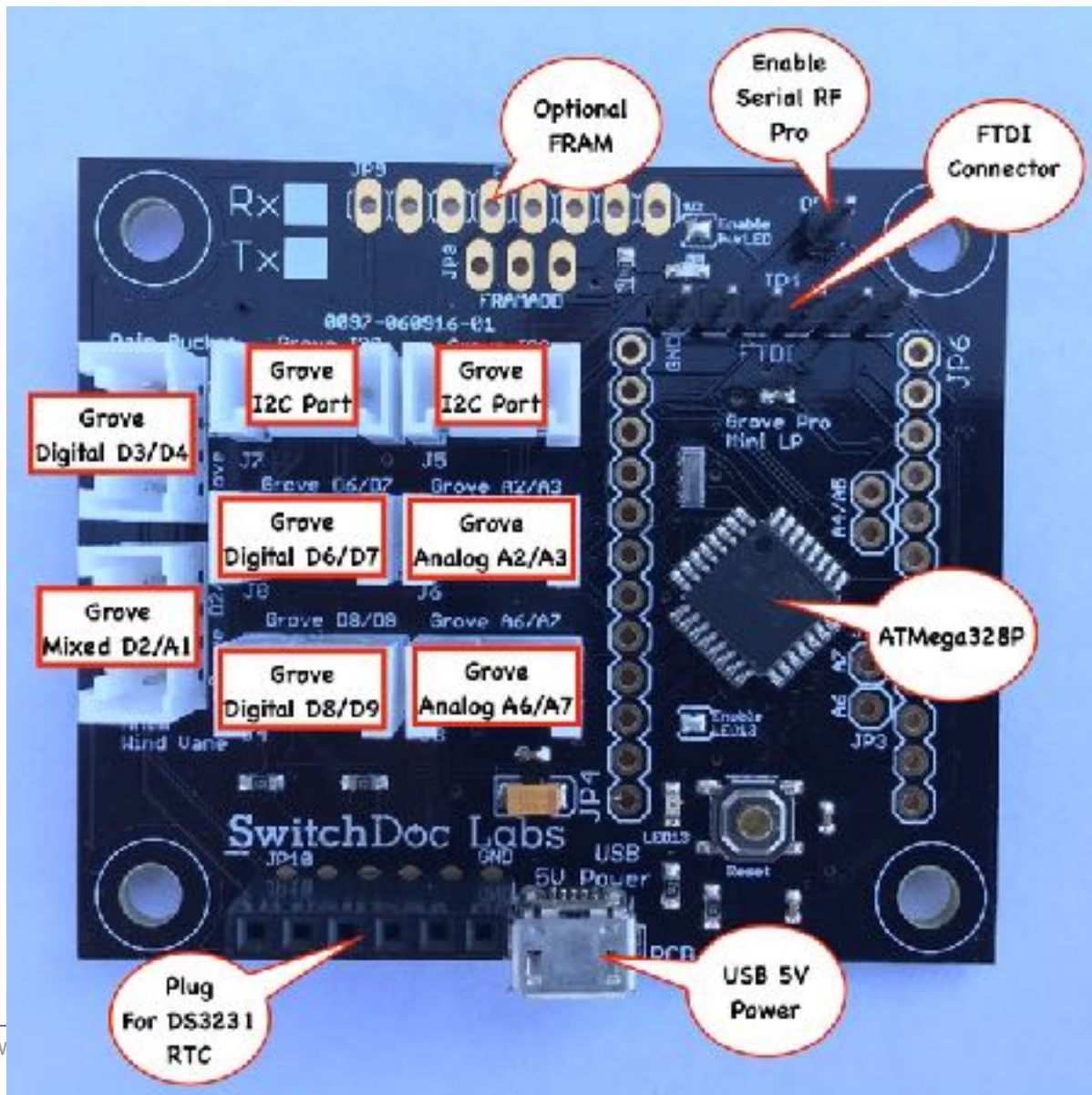
Solar WXLINK LoRa Wireless Transmit / Receive System



There are eight main parts in this kit:

- Two Grove Pro Mini LP Arduino Boards
- Two 433MHz LoRa Transceivers
- Grove WXLINKWR
- DS3231 RTC
- 330mA/6V Solar Cell
- SunAirPlus Solar Power Controller and Data Collector

Grove Pro Mini LP Board



The Arduino Pro Mini is built on an ATmega328P that contains 32kB Flash, 1kB EEPROM, and Internal SRAM of 2kB. Not a ton of memory but enough for what we are doing. It also contains a 4 channel 10 bit ADC (Analog to Digital Converter) and has 8bit PWM capability. You have 12 digital GPIO pins available and a hardware serial port. The unit is programmed using an FTDI cable rather than a USB cable. The USB cable is on the Grove ProMini LP strictly for a power supply connection. You can see all of the pinouts of the Grove Pro Mini below and in the grove Pro Mini LP specification available on switchdoc.com.

DS3231/EEPROM Real Time Clock

The SwitchDoc Labs DS3231/EEPROM combination is included with the WXLINK kit. It is plugged in by the user into JP10 of the TX Mini Pro LP Board, taking care to having the battery facing the top of the board.

The DS3231 is a low-cost, extremely accurate I²C real-time clock (RTC) with an integrated temperature-compensated crystal oscillator (TCXO) and crystal. The device incorporates a battery input, and maintains accurate timekeeping when main power to the device is interrupted. The integration of the crystal resonator enhances the long-term accuracy of the device.

The RTC maintains seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. Two programmable time-of-day alarms and a programmable square-wave output are provided. Address and data are transferred serially through the I²C bidirectional bus.

A precision temperature-compensated voltage reference and comparator circuit monitors the status of VDD to detect power failures, to provide a reset output, and to automatically switch to the backup supply (battery included) when necessary. Additionally, the RST pin is monitored as a pushbutton input for generating a μ P reset.

The specifications for this device is on the SwitchDoc Labs Weather Board product page.

AT24C32 32KB EEPROM

The AT24C32 provides 32,768 bits of serial electrically erasable and programmable read only memory (EEPROM) organized as 4096 words of 8 bits each. The EEPROM drivers are included in the SwitchDoc driver software.

Grove 433MHz LoRa Transceiver

The main functional module in the LoRa Radio 433MHz is RFM98, which is a transceiver features the LoRa long range modem that provides ultra-long range spread spectrum communication and high interference

immunity whilst mini-missing current consumption. The CPU on board the LoRa Radio 433MHz is an ATmega168,

There is also an integrated a s wire antenna to receive and transmit the signal, if the signal is too weak to receive,, you can use the MHF connector next to the wire antenna.

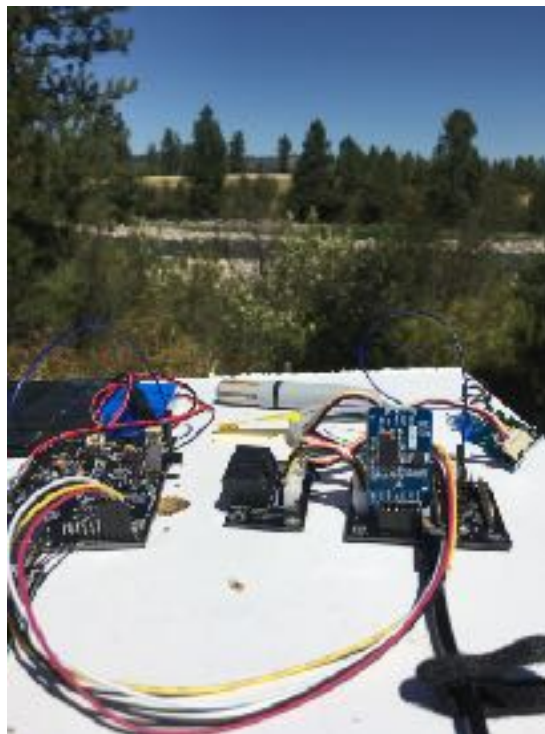
This is the 433MHz version/

Note

- Keep the antenna vertical to the board and as straight as possible
- Avoid having any big metal object near the antenna

Features

- Uses RFM95 module based on SX1276 LoRa®
- ~28mA(Avg) @+20dBm continuous transmit
- ~8.4mA(Avg)@standby mode
- ~20mA(Avg) @receive mode, BW-500kHz
- Simple wire antenna or MHF Connector for external high gain antenna
- +20dBm - 100 mW Power Output Capability



You can set the UART baud rate, frequency, output power, data rate, frequency deviation, receiving bandwidth parameters, etc. It comes pre-programmed for 433MHz and 9600 baud.

The 433Mhz LoRa board is rated for 6000 meters in free air. Free air means with no obstruction and line of sight. Anything in your way reduces the received power and reduces the range. To test the range, We took the solar powered transmitter system on a walk down to the Spokane River. We ran two tests. One with the receiver behind three interior walls and one exterior wall and then repeated the test with the receiver only behind one interior wall. We specifically looked for continuous data flow. We found we could go about 5% or 10% further and still get a packet now and again.



Table 1 - Tested Transmission Distance

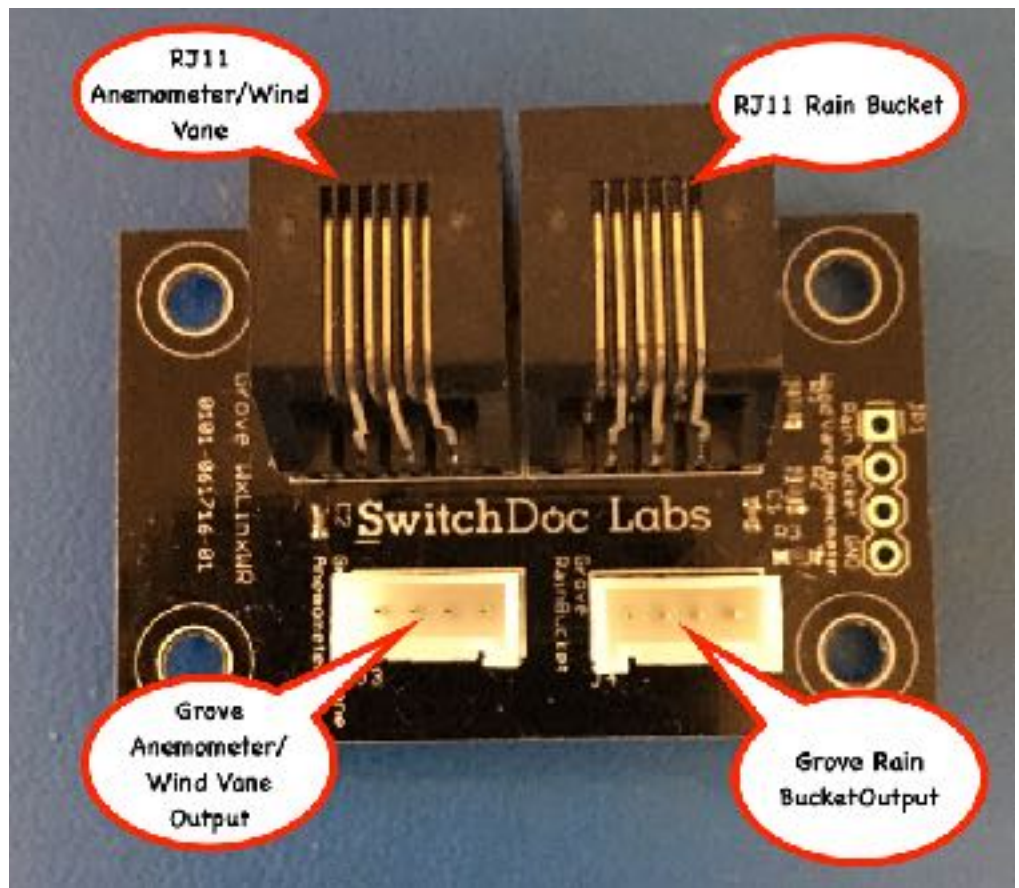
Receiver Condition	Maximum Transmitter Distance
Behind 3 Interior / 1 Exterior Wall	220 meters / 720 feet
Behind 1 Exterior Wall	438 meters / 1437 feet
Yagi Antenna on Transmitter	1600 meters / 5200 feet

Grove WXLINKWR

This board is included to provide a plug and play interface to the SwitchDoc Labs WeatherRack set of weather sensors. To the left is a picture of the WeatherRack connected up to a box containing the solar version of the WXLINK. Totally wireless.

The GroveWXLINKWR has four ports. These are:

- RJ11 port for Anemometer / Wind Vane from WeatherRack
- RJ11 port for Rain Bucket from WeatherRack
- Grove Connector for connection to Mini Pro LP for Anemometer/Wind Vane
- Grove Connector for connection to Mini Pro LP for Rain Bucket



Solar Panel

Included in the Solar WXLink kit is a 330mA / 6V Solar Panel. You can combine more panels in parallel to increase the solar power gathered. SunAirPlus will use up to 1000mA ignoring any additional current.

SunAirPlus Solar Power Controller and Data Collector

Solar Charge Controller

The Solar Charge Controller on SunAirPlus is based around a CN3065 Lithium Ion Charge Controller to run the charging sequence for the batteries. The chip does an approximation of the MPPT (Maximum Power Transfer Tracking).

It is the purpose of the MPPT system to sample the output of the cells and apply the proper resistance (load) to obtain maximum power for any given battery and temperature conditions. Although the fully charged battery pack voltage may be close to the solar cell maximum power point voltage, this is unlikely to be true at sunrise when the battery has been partially discharged. Charging may begin at a voltage considerably below the solar panel maximum power point voltage.

When the batteries are fully charged and the solar power production exceeds the load of the computer, the CN3065 can no longer operate the panel at its maximum power point as the excess power has no load to absorb it. The CN3065 must then shift the solar power operating point away from the peak power point until production exactly matches demand.

The CN3065 is set in SunAirPlus to deliver a maximum of 1000ma instantaneously to the connected LiPo batteries. Now note, that doesn't mean you can't use more than 6W of solar panels. The 6W solar panels won't be delivering 1000ma most of the time.

Voltage Booster

In order to boost the nominal 3.7V - 4.2V output from the LiPo batteries up to 5V, SunAirPlus uses a voltage booster. It uses an ISL97156 PWM Step-Up Regulator to do the step up in voltage. This design uses an PWM (Pulse Width Modulated) oscillator run through an inductor with a feedback loop to increase the voltage up to a regulated 5V. Note that this voltage will start to come down below 5V when you overload the board (exceed 1000ma) or the LiPo batteries get below about 3.6V. A good system will monitor this voltage and cut the power to the computers when things get low. If you are charging your phone, it doesn't matter.

Level Converters

Outputs and inputs from the Raspberry Pi are 3.3V. You can't drive them with 5V lines without potentially destroying the Pi. Why is this? The Raspberry Pi has protection diodes between the pin and 3.3V and ground. Positive voltages greater than 3V3 + one "diode drop" (normally 0.5V) will be shorted to 5V, this means that if you put a 5V power supply on the GPIO pin you will "feed" the 3V3 supply with 4.5 Volt (5V - the diode drop of 0.5V) and that may damage 3V3 logic.

A designer should only use 3.3V outputs connected to your Raspberry Pi GPIO pins without putting a voltage level converter between the devices. Note: You can DRIVE most 5V device inputs with the Raspberry Pi 3.3V GPIO outputs. Most devices will read 3.3V inputs as a logic one. In SunAirPlus there are bidirectional level converters. Figure 4 shows a simple bi-directional logic level FET based converter. You can set VDD to either 3.3V or 5.0V and SunAirPlus will adjust the levels..

A/D Converters - ADS1015

The internal A/D converters on the Arduino are sufficient for reading the photoresistors used by SunAirPlus to track the sun, but since the Raspberry Pi has no built-in A/D converters, SunAirPlus includes a circuit to do this. SunAirPlus contains an excellent built-in 12 bit 4 channel A/D converter an I2C device known as the ADS1015. The ADS1015 is a precision analog-to-digital converter (ADC) with 12 bits of resolution. The ADS1015 features an onboard reference and oscillator. Data are transferred via an I2C-compatible serial interface; four I2C slave addresses can be selected.

The ADS1015 can perform conversions at rates up to 3300 samples per second (SPS). An onboard PGA is available on the ADS1015 that offers input ranges from the supply to as low as $\pm 256\text{mV}$, allowing both large and small signals to be measured with high resolution. The ADS1015 also features an input multiplexer (MUX) that provides two differential or four single-ended inputs.

The ADS1015 operates either in continuous conversion mode or a single-shot mode that automatically powers down after a conversion and greatly reduces current consumption during idle periods.

Software for the ADS1015 for both the Raspberry Pi and Arduino is readily available from Adafruit and others.

The Default I2C address is 0x48. If you have more than one SunAirPlus in your system, consider using a SwitchDoc Labs I2C 4 channel Mux to avoid address conflicts.

You can use these two photoresistors A/Ds with the SunTracker tubes (shown below in the 3D print) to pinpoint where the sun is and turn our panels to face the sun using a servo motor or stepper motor.

Current/Voltage Sensors - INA3221

This is one of the most interesting parts of the SunAirPlus board. It allows you to get dynamic and accurate information on how your Solar Power system is running.

The INA3221 is a three-channel, high-side current and bus voltage monitor with an I2C interface. The INA3221 monitors both shunt voltage drops and bus supply voltages in addition to having programmable conversion times and averaging modes for these signals. There are three 0.1 Ohm shunts on the board

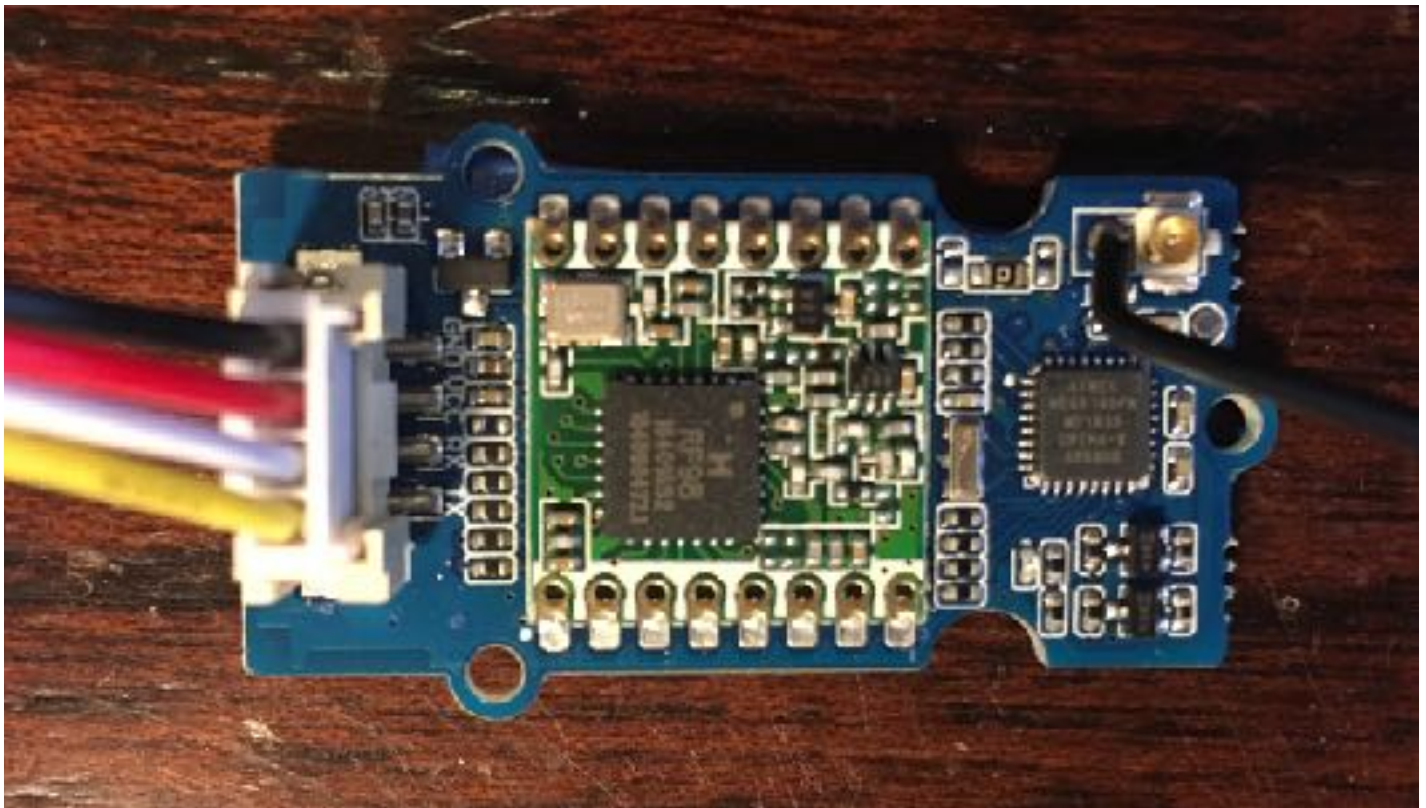
Kit Assembly

All of the parts necessary to build a functional Solar WXLink are included, with the exception of the 3.7V LiPo battery. Because this system is so low power, any size of LiPo battery greater than 1000mAh should work fine. The higher the mAh rating of the battery, the longer the streak of cloudy days your unit can run without solar power.

Note that you can power the Solar WXLink RX Mini Pro LP board by connecting it to a 5V I2C interface (such as on the Pi2Grover Raspberry Pi Grove Interface Board).



The two Mini Pro LP Boards are identical in hardware, but come preprogrammed with the TX (Transmitter) and RX (Receiver) software preprogrammed into the units. You can tell which is which by looking at the upper left corner of the Mini Pro LP Board. The black dot indicates that this board is an RX board.. All the software is open source and you can change the software if you wish.



Parts List

Solar WXLink Parts List	
TX Mini Pro LP Board	Marked TX
RX Mini Pro LP Board	Marked TX
DS3231 RTC Board	Plugin Board
LoRa 433MHz Board	W/Grove Cable
LoRa 433MHz Board	W/Grove Cable
WXLinkWR Board	Interface to WeatherRack
7 Grove Cables	Includes two included with LoRa 433MHz Boards

Solar WXLink Parts List

SunAirPlus	Solar Power Controller / Data Collector
4 Pin Female Header to Grove Cable	for SunAirPlus to I2C Connector on TX Mini Pro LP
Micro USB to USB A Cable	Connects from SunAirPlus to TX Mini Pro LP Board
3.7V LiPo Battery with JST-2 Plug	USER SUPPLIED (adafruit.com is a good source).

Wiring List

First STEP: Plug in the DS3231 Clock on the RX Mini Pro LP Board in the JP 10 socket with the Battery facing the top of the Mini Pro XP Board (as in the picture below). Align the GND pins carefully.



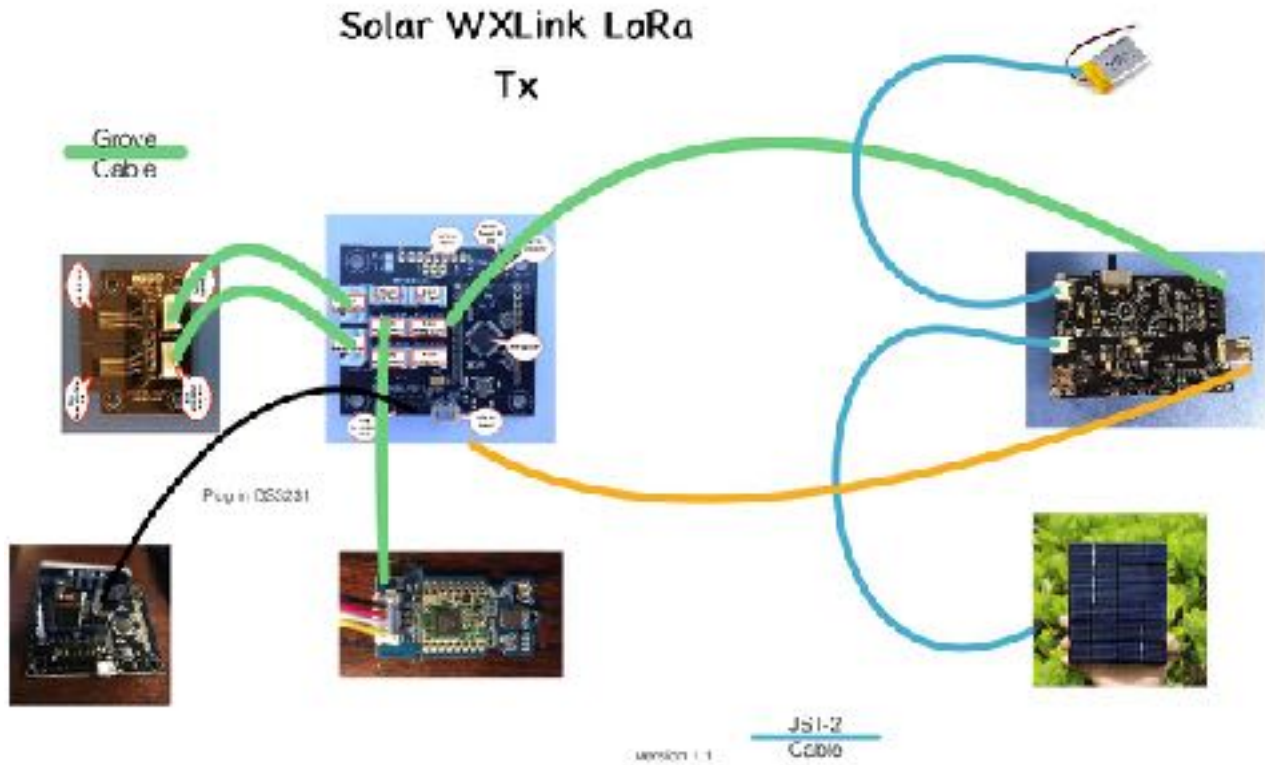
WXLink Wiring List TX Portion

From	To	Notes
TX Mini Pro LP Board / Grove Digital D2/A1	WXLinkWR / Grove J3 Anemometer/ Wind Vane	Grove Cable
TX Mini Pro LP Board / Grove Digital D3/D4	WXLinkWR / Grove J4 Rain Bucket	Grove Cable
TX Mini Pro LP Board / Grove D6/ D7	LoRa 433MHz Board for TX	Grove Cable - note: Both LoRa 433MHz boards are identical - pick one for TX and one for RX
TX Mini Pro LP Board / Micro USB Connector	SunAirPlus USB Connector	5V Power from SunAirPlus
TX Mini Pro LP Board / Grove I2C Plug (either plug)	SunAirPlus / "From Computer Header"	Wires on Grove Cable: Black - GND, Red - VDD, White - SDA, Yellow - SCL
LiPo Battery JST-2 Cable	SunAirPlus / Male JST-2 Labeled Battery	Battery to SunAirPlus
330mA/6V Solar Cell JST-2 Cable	SunAirPlus / Male JST-2 Labeled Solar	Solar Cell to SunAirPlus

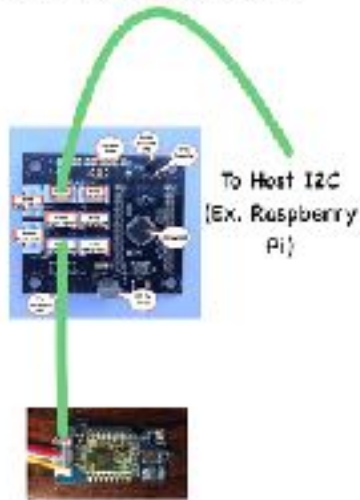
WXLink Wiring List RX Portion

From	To	Notes
RX Mini Pro LP Board / Grove D8/ D9	LoRa 433MHz Board for TX	Grove Cable - note: Both LoRa 433MHz boards are identical - pick one for TX and one for RX
RX Mini Pro LP Board /Grove I2C	Pi2Grover I2C or a 5V Arduino Grove I2C connector	

Wiring Diagram



Solar WXLink LoRa Rx



Testing

The Solar WXLink system comes programmed to send a sample every 30 seconds. After you have assembled and checked the wiring above, plug power into both the TX and RX Mini Pro LP Board (you may use any Micro USB Power supply for the TX board and either a power supply or connect one of the Grove I2C Ports to a Pi2Grover on the Raspberry Pi or an 5V Arduino Grove connector). Note that for the TX WXLink (once wired to the battery and SunAirPlus) you can supply a power supply to the Mini USB connector on SunAirPlus (thus charging the battery) or to the Micro USB connector on the TX Mini Pro LP Board.

If you have a charged battery, you will see the power LED on SunAirPlus turn on (providing you have turned the On / Off switch on the SunAirPlus board to ON!). You will see an additional Red LED light up when your battery is being charged by the Mini USB Cable or the Solar Panel.

You will see a blue power LED on both Mini Pro LP Boards if they are powered.

Testing the Unit on the Raspberry Pi

If you are running the GroveWeatherPi software, then support for the WXLink is built in. Otherwise consider downloading and configuring the Raspberry Pi DataLogger to receive the WXLink (https://github.com/switchdoclabs/SDL_Pi_DataLogger) packets. The Raspberry Pi DataLogger has support for the WXLink built in to the software.

Both of these pieces of software read the data from SunAirPlus (Battery Voltage, Battery Current, Solar Panel Voltage, Solar Panel Current, Load Current and Load Voltage) and will store the information in the database and produce graphs of the power system data.

On the receiver end, we read the data from the ProMiniLP based receiver by using an I2C interface at 0x08. Here is an example packet of data coming from the transmitter:

In the receiver, the data is buffered and then supplied to the host computer when requested by the host computer on the I2C bus. The receiver emulates an I2C device. This makes the receiver compatible with many different types of development computers.

A key thing to know about the WXLink RX board is that it acts like an I2C device at address 0x08. That is how the hosting computer can read the information from the packets coming in from the TX board. The packets consist of 63 bytes and are formatted as follows:

Put the following python code into a file named readI2C.py.

```
# reads from WXLink RX
#

import smbus
import time
import sys
import subprocess

bus = smbus.SMBus(1)

address = 0x08
twoblockMessages = 0
oneblockMessages = 0
errorMessages = 0

#sudo modprobe -r i2c_bcm2708
#sudo modprobe i2c_bcm2708 baudrate=32000
```


while True:

```

flag = 0
data = ""
print "-----"
try:
    print "block 1"
    data = bus.read_i2c_block_data(address, 0);
    print ''.join(hex(x) for x in data)
except IOError:
    #subprocess.call(['i2cdetect', '-y', '1'])
    flag = 1 #optional flag to signal your code to resend or something
#time.sleep(3)
try:
    print "block 2"
    data = bus.read_i2c_block_data(address, 1);
    print ''.join(hex(x) for x in data)
    print "-----"
except IOError:
    #subprocess.call(['i2cdetect', '-y', '1'])
    flag = 1 #optional flag to signal your code to resend or something

print "flag=", flag
if (flag == 0):
    twoblockMessages = twoblockMessages + 1
else:
    errorMessages = errorMessages + 1

print "twoblock = %i error = %i percent error=%6.2f%%" % (twoblockMessages, errorMessages,
100*(float(errorMessages)/float(errorMessages+twoblockMessages)))
time.sleep(2);

```

Run this code as:

```
sudo python readI2C.py
```

and you should see results like:

```
pi@RPi3-61:~/I2CRead $ sudo python readI2C.py
```

```

-----
block 1
0xab 0x66 0x1 0x4d 0xa0 0xe 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
0x0 0x0 0x0 0x0 0x0 0x0
block 2
0x0 0x80 0x6a 0x7c 0x40 0xcc 0xcc 0xbc 0x41 0x32 0x33 0x53 0x41 0x29 0x5c 0x8f 0x3f 0x0 0x0 0x0 0x80 0x0 0x0
0x0 0x0 0x3e 0x7 0x0 0x0 0xdb 0x7f 0x0
-----
flag= 0

```

twoblock = 1 error = 0 percent error= 0.00%

If you see all 0x00 values, the WXLink RX has not yet received data from the WXLink TX. You should see the values change every 30 seconds by default.

Packet and I2C Data Format

Transmitted Data Format				
Value	Format	Notes	Byte Value Start	I2C Block Number
Start Byte #0	0xAB	Constant	0	0
Start Byte #1	0x66	Constant	1	0
Wireless ID/ Software Version	Byte	WIRELESSID is changed if you have more than one unit reporting in the same area. It is coded in protocol as WIRELESSID*10+ SOFTWAREVERSION	2	0
TimeStamp	long - 4 bytes	Approximate Seconds since reboot of TX unit. Only guaranteed to advance with time.	3	0
Wind Direction	int - 2 bytes	degrees	7	0
Average Wind Speed (KPH)	float - 4 bytes	KPH	9	0
Wind Clicks	long - 4 bytes	clicks since last packet generation	13	0
Total Rain Clicks	long - 4 bytes	Since boot up	17	0
Max Wind Gust	float - 4 bytes	KPH - Not Implmented	21	0

Outside Temperature	float - 4 bytes	Degrees C	25	0
Outside Humidity	float - 4 bytes	%	29	Split between 0/1
Battery Voltage	float - 4 bytes	Volts	33	1
Battery Current	float - 4 bytes	mA	37	1
Load Current	float - 4 bytes	mA	41	1
Solar Panel Voltage	float - 4 bytes	Volts	45	1
Solar Panel Current	float - 4 bytes	mA	49	1
Aux A	float - 4 bytes	For future use	53	1
Message ID	long - 4 bytes	Increments from 0 at boot up sequentially per message packet generated and sent	57	1
Checksum High	Byte	High byte of CRC 16XModem checksum of message - see code for generation	61	1
ChecksumLow	Byte	Low byte of CRC 16XModem checksum of message - see code for generation	62	1
Fill byte	Byte - 0x00	constant	63	1

The WXLink TX/RX Software

The complete software source for the WXLink software is located on github.com/switchdoclabs

https://github.com/switchdoclabs/SDL_Arduino_WXLink_Tx

https://github.com/switchdoclabs/SDL_Arduino_WXLink_Rx

We wrote all the software using the Arduino IDE. The software for the transmitter can be found in [SDL_Arduino_WeatherLink_Tx](https://github.com/switchdoclabs/SDL_Arduino_WeatherLink_Tx) [ref: github.com/switchdoclabs/SDL_Arduino_WeatherLink_Tx] and the software for the receiver can be found in [SDL_Arduino_WeatherLink_Rx](https://github.com/switchdoclabs/SDL_Arduino_WeatherLink_Rx) [ref: github.com/switchdoclabs/SDL_Arduino_WeatherLink_Rx]. I'm not going through most of the software as it is pretty straightforward and you can look at the source code for the details. The most complicated part of the transmitter software is the sleeping code, specifically the code designed to minimize current consumption. We will go through the two methods we use to sleep the Pro Mini.

Overall, the transmitter gathers the data from the connected weather instrument (and an optional SunAirPlus solar controller [ref: [Raspberry Pi Geek Magazine on Solar Tracking](#)] if you are building a solar powered unit), formats it in a 62 byte string, adds a check sum and then transmits it to the receiver. The serial debugging output from the WXLink TX of a transmission cycle is below:

Using DS3231 to Wake Up

ALARM_1 Found

16:01:36 4/1/2000

wakeState=5

MessageCount=27

Outside Temperature (C): 25.00

Outside Humidity (%RH): 34.40

Wind Vane Voltage =5.00

Wind Vane Degrees =0

TotalRainClicks=1

windClicks=0

Average Wind Speed=0.00 KPH

shortestWindTime (usec)10000000

LIPO_Battery Load Voltage: 3.91 V

LIPO_Battery Current: 46.40 mA

Solar Panel Voltage: 1.02 V

Solar Panel Current: 0.00 mA

Load Voltage: 4.98 V

Load Current: 27.60 mA

crc = 0xFEB0

-----Sending packet-----

Note that this packet was from a WXLink with the Solar option added. A non-solar unit will transmit all zeros for the SunAirPlus data.

Do you see the current measurement from SunAirPlus (27.60 mA)? That’s a lot higher than 5mA. Why? It is because the radio is on when we are transmitting a packet and reading the solar data. Then the software turns it off and goes back to sleep. Back to 5mA.

On the receiver end, we read the data from the ProMiniLP based receiver by using an I2C interface at 0x08. Here is an example packet of data coming from the transmitter:

```

-----
block 1
0xab 0x66 0x1 0x4d 0xa0 0xe 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
0x0 0x0 0x0 0x0 0x0 0x0
block 2
0x0 0x80 0x6a 0x7c 0x40 0xcc 0xcc 0xbc 0x41 0x32 0x33 0x53 0x41 0x29 0x5c 0x8f 0x3f 0x0 0x0 0x0 0x80 0x0 0x0
0x0 0x0 0x3e 0x7 0x0 0x0 0xdb 0x7f 0x0
-----

```

In the receiver, the data is buffered and then supplied to the host computer when requested by the host computer on the I2C bus. The receiver emulates an I2C device. This makes the receiver compatible with many different types of development computers.

The Sleep Code - Sleepy

The most complex part of the transmitter code is the sleep code. We use two different methods to sleep. The first is a software sleep mechanism based on the internal hardware watchdog timer. We are using a library called Sleepy from JeeLib [ref: jeelabs.org/pub/docs/jeelib/classSleepy.html]. It is a great wrapper for the all the myriad bits that need to be set and reset for the Arduino ATmega328P to dramatically reduce power by sleeping. When the ATmega328P is in the sleep state (low-power mode) only the watchdog, the two hardware interrupts or a pin change can wake up the processor. In this design we use the two hardware interrupts (INT0/INT1 - connected to pin D2 and D3) to count the tips of the rain bucket and the number of times the anemometer is turning to calculate wind speed.

The function “loseSomeTime” is called for 16msec at a time so as not to overflow the watchdog timer and to more importantly allow the processor to go back to sleep immediately after processing an interrupt. The

processor immediately wakes up on an anemometer interrupt, counts the interrupt and goes right back to sleep because of this loop below. After the loop has been exhausted, then we transmit a packet to our receiver. While the timing is perfect, it does work and it dramatically reduces power consumption (down to below 5mA).

```
for (long i = 0; i < nextSleepLength / 16; ++i)
  Sleepy::loseSomeTime(16);
```

Why a loop? You need to put the processor back to sleep after an interrupt, but you want to send a packet when the full sleep cycle is done, at the termination of the loop. Note that the index variable "i" is defined "long" as we want to count beyond 65,000. Using this Sleepy library and the loop scheme you can send packets from anywhere from 2 or 3 times a second to waiting for hours.

The Sleep Code - DS3231

The second way we put the processor to sleep (actually wake the processor) is by using the special timer hardware with in the optional DS3231 clock that can be plugged into the ProMiniLP. We did an analysis of the accuracy of the DS3231 in a post on SwitchDoc Labs and the DS3231 was the clear winner. The DS3231 has two sets of alarm registers that can be used to periodically interrupt the ATmega328P based on the DS3231 time / day / date. You are somewhat limited as to how fast you can send packets (1 per second is the maximum) and you can't send them say every 32 seconds, but you can set up every 30 seconds, 1 minute, 1 hour, etc. Even send yourself a message on on December 21, 2017 if you like. We still use the Sleepy library to do the power down except that here we use the powerDown function which makes the processor go away forever until it gets an interrupt. By checking to see the source of the interrupt (DS3231 or the Rain Bucket) we can decide whether to send a packet or go right back to sleep. Below is the code to wake up every 30 seconds.

```
// choose once per 30 seconds

RTC.setAlarm(ALM1_MATCH_SECONDS, 30, 0, 0, 0);
RTC.alarm(ALARM_1);
RTC.setAlarm(ALM2_EVERY_MINUTE, 0, 0, 0, 0);
RTC.alarm(ALARM_2);
RTC.alarmInterrupt(ALARM_1, true);
RTC.alarmInterrupt(ALARM_2, true);
```

Honestly, the DS3231 is a very, very nice Real Time Clock chip with lots of good features. It even has a temperature sensor! We are providing a DS3231 plug in board with each ProMiniLP. Cutting the LED on the DS3231 saves about 2mA. The DS3231 itself only uses 0.4mA.

What are Grove Connectors?



A Grove connector is a four pin standardized size connector used to plug into the Pi2Grover base unit and Grove devices and modules. These standardized connectors (common to all types of Grove Connectors) are the key to making this system work. They are keyed to prevent plugging them in backwards, and the four types of connectors (see below) are all designed so that if you plug the wrong type of device into the wrong type of base unit, there is no problem. They just won't work. This is a good thing. Less smoke, more prototyping!

The Four Types of Grove Connectors

Below are some of the specifics of each of the four types of connectors. First of all, physically all of them are the same. Exactly. The differences are in the signal types that are provided. Now, note. You will never short out power and ground by mis-plugging one type of Grove connector in the other. **However, it is possible to plug a 3.3V Grove Module into a 5.0V Grove connector and damage the device.** The same could happen with an output coming back from a Grove button or switch for example into another output. While you do need to be careful and think about what you are doing, it is a lot less risky than soldering or using just jumpers to wire up devices to your Pi or Arduino.

Generically, all of the Grove connectors are wired the same: Signal 1, Signal 2, Power, Ground.

Grove Digital

A digital Grove connector consists of the standard four lines coming into the Grove plug. The two signal lines are generically called D0 and D1. Most modules only use D0, but some do (like the LED Bar Grove display) use both. Often base units will have the first connector called D0 and the second called D1 and they will be wired D0/D1 and then D1/D2, etc.

Grove Digital		
Pin 1	D0	Primary Digital Input/Output
Pin 2	D1	Secondary Digital Input/Output
Pin 3	VCC	Power for Grove Module (5V or 3.3V)
Pin 4	GND	Ground

Examples of Grove Digital modules are: Switch Modules, the Fan Module, and the LED Module.

Grove Analog

An Grove Analog connector consists of the standard four lines coming into the Grove plug. The two signal lines are generically called A0 and D0. Most modules only use A0. Often base units will have the first connector called A0 and the second called A1 and they will be wired A0/A1 and then A1/A2, etc.

Grove Analog		
Pin 1	A0	Primary Analog Input
Pin 2	A1	Secondary Analog Input
Pin 3	VCC	Power for Grove Module (5V or 3.3V)
Pin 4	GND	Ground

Examples of Grove Analog modules are: Potentiometer, Voltage Divider and the Grove Air Quality Sensor.

Grove UART

The Grove UART module is a specialized version of a Grove Digital Module. It uses both Pin 1 and Pin 2 for the serial input and transmit. The Grove UART plug is labeled from the base unit point of view. In other words, Pin 1 is the RX line (which the base unit uses to receive data, so it is an input) where Pin 2 is the TX line (which the base unit uses to transmit data to the Grove module).

Grove UART		
Pin 1	RX	Serial Receive (from base point of view)
Pin 2	TX	Serial Transmit (from base point of view)
Pin 3	VCC	Power for Grove Module (5V or 3.3V)
Pin 4	GND	Ground

Examples of Grove UART modules are: XBee Wireless Sockets, 125KHz RFID Reader

Grove I2C

The Grove I2C connector has the standard layout. Pin 1 is the SCL signal and Pin 2 is the SDA signal. Power and Ground are the same as the other connectors. This is another special version of the Grove Digital Connector. In fact, often the I2C bus on a controller (like the ESP8266, Raspberry Pi and the Arduino) just uses Digital I/O pins to implement the I2C bus. The pins on the Raspberry Pi and Arduino are special with hardware support for the I2C bus. The ESP8266 is purely software.

Grove I2C		
Pin 1	SCL	I2C Clock
Pin 2	SDA	I2C Data
Pin 3	VCC	Power for Grove Module (5V or 3.3V)
Pin 4	GND	Ground