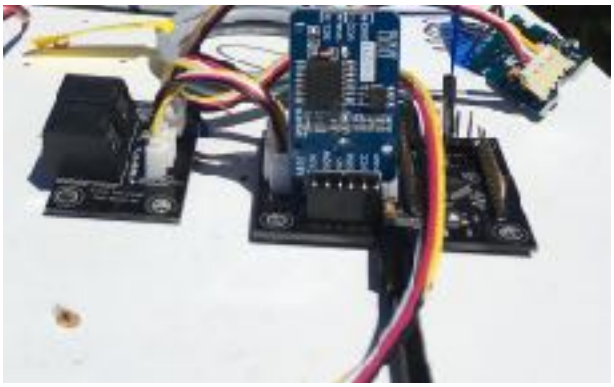
This is a Mini Pro LP Arduino based bi-directional radio link designed for hooking up the WeatherRack and AM2315 outside and then transmitting the data to a computer inside rather than using wires.

Features and Benefits:

- 600 Meter range
- Low power - Suitable for Solar
- 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

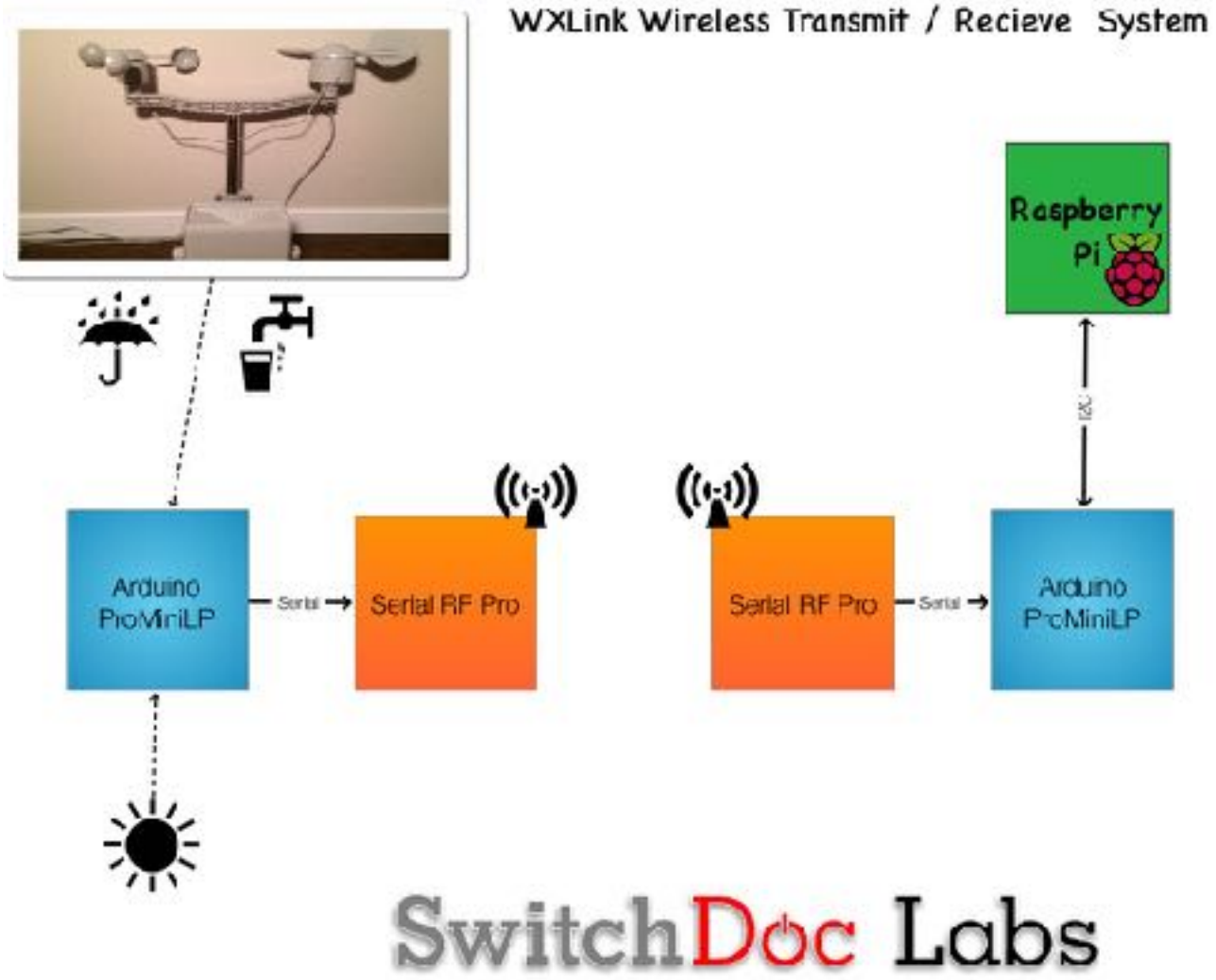
Introduction



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.

The WXLink product kit contains two Min Pro LP Arduino boards, a WXLinkWR Weather Rack Interface board, two transceivers and one DS3231 RTC Board.

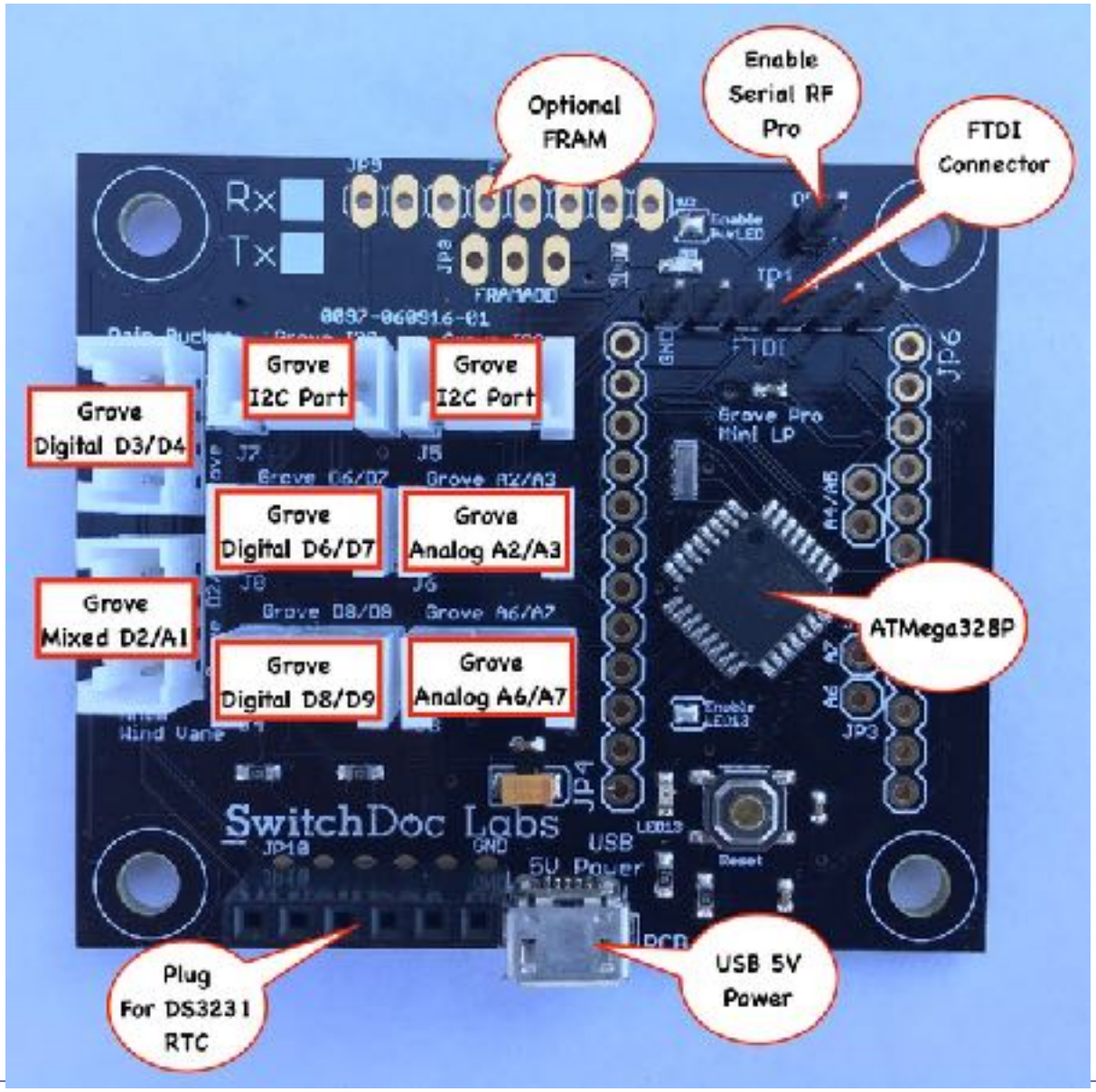
Theory of Operation



There are six main parts in this kit:

- Two Grove Pro Mini LP Arduino Boards
- Two Grove Serial RF Pro
- Grove WXLinkWR
- DS3231 RTC

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 an 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 pf 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.

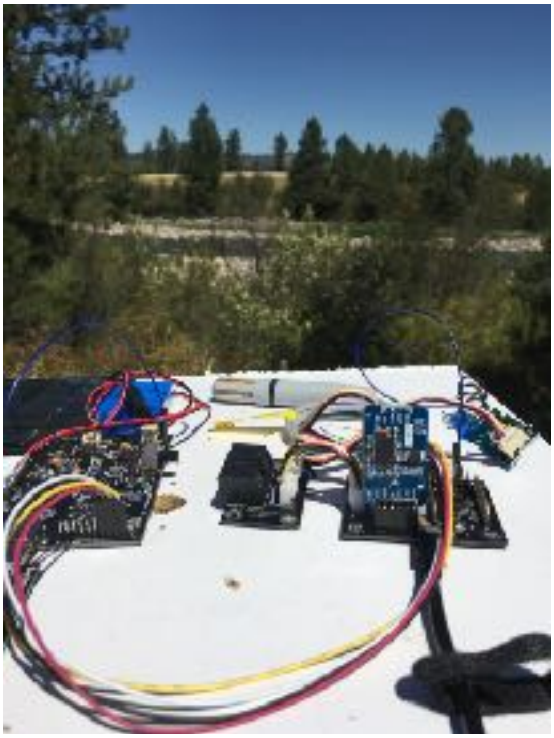
The RTC maintains seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. Two 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 µP 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 Serial RF Pro

The Grove Serial RF Pro is a low cost, high performance transparent FSK transceiver with operating at 433/470/868/915 MHz. It features small size, high output power, high sensitivity, long transmission distance and high communication data rate with auto set up for communication change and data receiving and transmission control.
Typically, you can get 600m in line of sight transmission, less if you have walls or intervening landscape features.



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 Serial RF Pro is rated for 600 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 - Maximum Transmission Distance

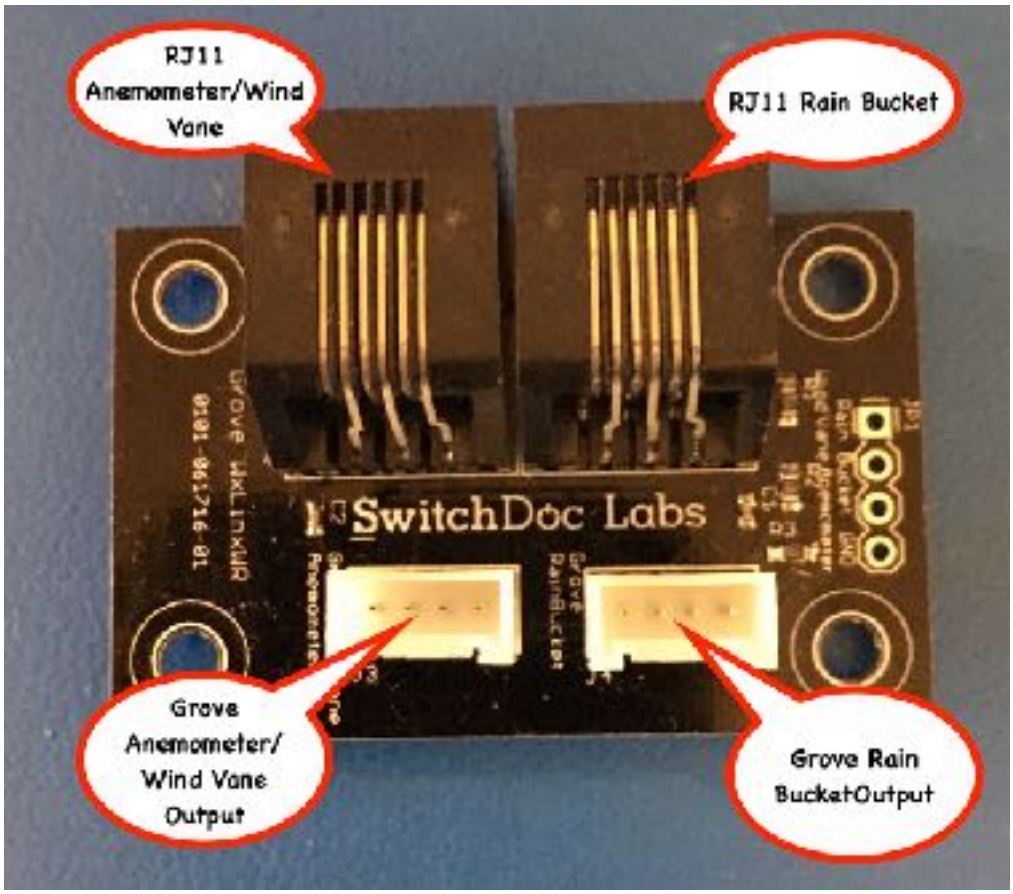
Receiver Condition	Maximum Transmitter Distance
Behind 3 Interior / 1 Exterior Wall	220 meters / 720 feet
Behind 1 Exterior Wall	438 meters / 1437 feet

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:

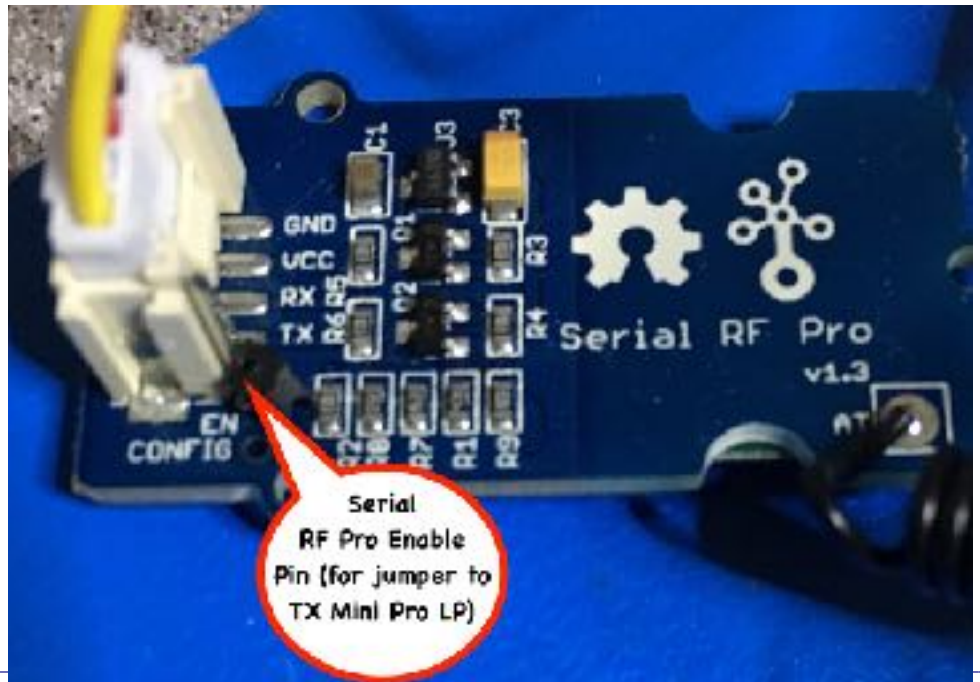
- RJ11 port for Anemomenter / Wind Vane from WeatherRack
- RJ11 port for Rain Bucket from WeatherRack
- Grove Connector for connection to Mini Pro LP for Anemeomter/Wind Vane
- Grove Connector fo connection to Mini Pro LP for Rain Bucket



Kit Assembly

All of the parts necessary to build a functional WXLink are included. A power supply is not included (any Micro USB 5V Power supply will work). Note that you can power the 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



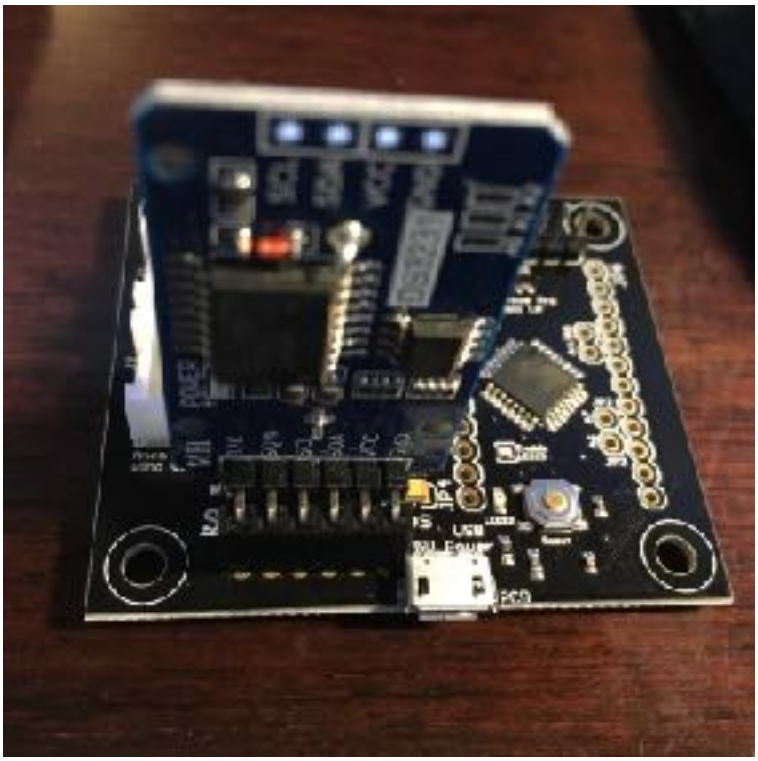
upper left corner of the Mini Pro LP Board. The black dot indicates that this board is an RX board.. All the software is open source and you can change the software if you with.

Parts List

WXLink Parts List	
TX Mini Pro LP Board	Marked TX
RX Mini Pro LP Board	Marked TX
DS3231 RTC Board	Plugin Board
Serial RF Pro	W/Grove Cable
Serial RF Pro	W/Grove Cable
WXLinkWR Board	Interface to WeatherRack
One Pin Female to Female Jumper Wire	For TX to Serial RF Pro enable/disable (for power)
7 Grove Cables	Includes two included with Serial RF Pro boards

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.

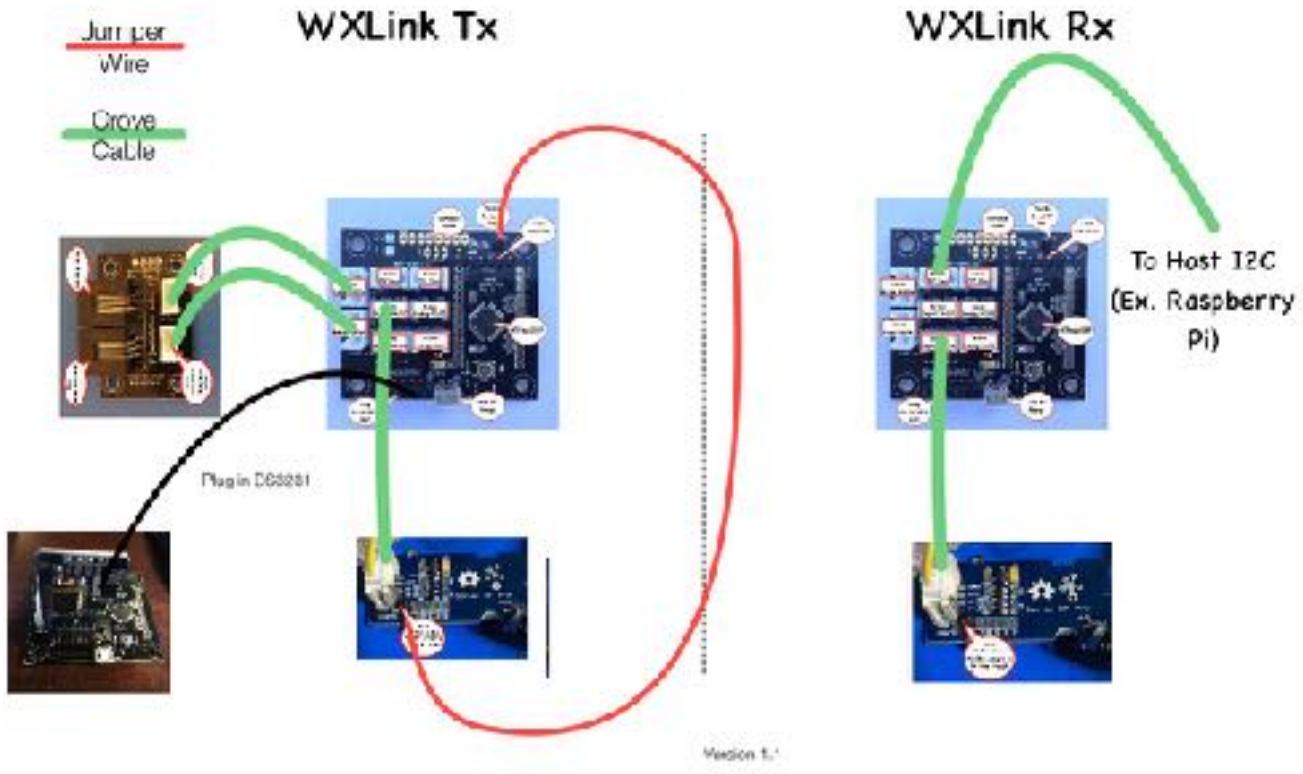


WXLink Wiring List TX Portion		
From	**To**	**Notes**
TX Mini Pro LP Board / Grove Digital D2/A1	WXLinkWR / Grove J3 Anemometer/ Wind Vane	Grove Cable
TX Mini Pro LP Board / Grove Digital D3/D4	WXLinkWR / Grove J4 Rain Bucket	Grove Cable
TX Mini Pro LP Board / Grove D6/ D7	Serial RF Pro Board for TX	Grove Cable - note: Both Serial RF Pro boards are identical - pick one for TX and one for RX

WXLink Wiring List TX Portion		
From	**To**	**Notes**
TX Mini Pro LP Board / Serial RF Pro Jumper (top right corner)	Serial RF Pro Boar TX / Serial RF Pro Enable Line	Female to Female Jumper Wire

WXLink Wiring List RX Portion		
From	**To**	**Notes**
RX Mini Pro LP Board / Grove D8/ D9	Serial RF Pro Board for TX	Grove Cable - note: Both Serial RF Pro 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	Note: The WXLink TX needs to operate on 5V for the Serial RF Pro to function correctly.

Wiring Diagram



Testing

The 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). The RX Board needs to operate at 5V for the Serial RF Pro to function correctly. You will see a blue power LED on both Mini Pro LP Boards if they are powered.

Look at the LEDs on the bottom of the Serial RF Pro boards. If you see them flicker approximately every 30 seconds your board is operating.

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.

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 he 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:

Type 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 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_Arudino_WeatherLink_Tx [ref: github.com/switchdoclabs/SDL_Arduino_WeatherLink_Tx] and the software for the receiver can be found in 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 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 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