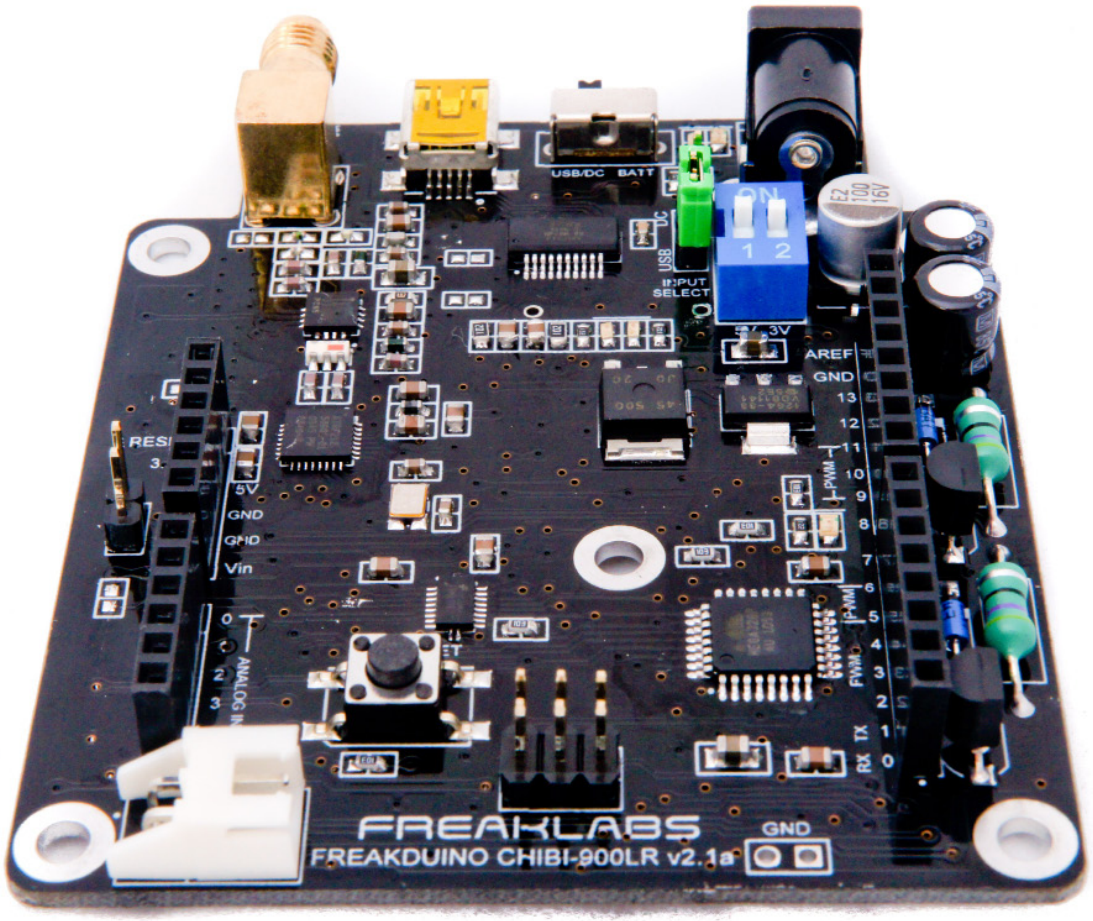
FREAKLABS

FREAKDUINO 900 MHz Long Range Wireless Arduino-Compatible Prototyping Platform v2.1a Datasheet



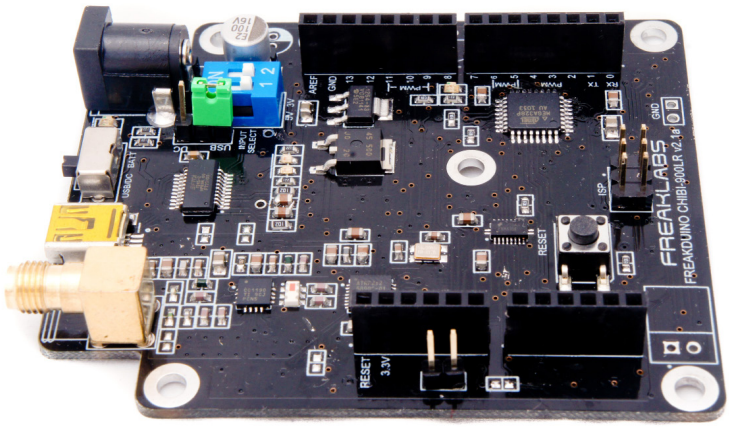
Document Revision History

Date	*Description*
2013-09-25	v2.1A Document release

Specifications

The FreakLabs Freakduino 900 MHz long range wireless board is designed for communicating over long distances as well as rapid prototyping, evaluation, and deployment of wireless devices at low cost. It combines the ease-of-use of the Arduino IDE and toolchain, compatibility with a rich assortment of peripherals in the Arduino shield form factor, and an integrated, amplified 802.15.4 wireless radio for inexpensive prototyping and testing of a wireless device.

The base board has all the functionality of an Arduino-based system with a wireless radio and is an inexpensive way to start playing with Arduino designs and wireless communications. This board also has additional amplifiers on the radio to boost both the transmitted and received signal. The result is a total increase of over 30 dB for the total link budget which means the range is drastically increased compared to a bare radio.



It also has optional features such as battery regulation circuitry, a low-cost bottom-mounted battery case, or a ruggedized enclosure with integrated battery case.

QuicK Specs

MCU: ATMega328P

Memory: 32 kB Flash/2 kB RAM

Communications: 802.15.4 wireless, USB

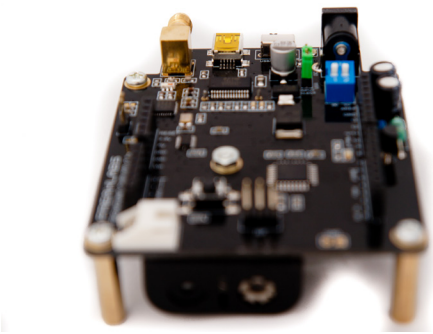
Radio Amplifiers: Tx: 500 mW (27 dBm), Rx: 11 dB

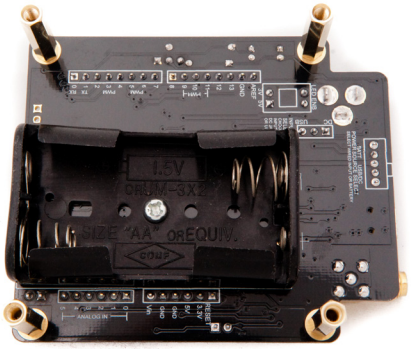
Expansion: Arduino-compatible shield connector

Power: Ext 5VDC, USB, Battery (optional)

Current Consumption: Sleep: ~350 uA, Active: 70 mA

Options: Ruggedized enclosure, battery regulation circuit, standalone battery case





This board is designed to introduce people to wireless sensor networking inexpensively and without having to deal with complex toolchains, protocol stacks, and software. It's fully compatible with the Arduino IDE and toolchain which offers a single click compile and download and a rich ecosystem of open source software and tutorials available on the internet. The electrical specifications and connector form factor are also compatible with the original Arduino hardware so that the board can interface with the large assortment of Arduino peripheral shields. The availability of third party peripheral shields and open source software allows this board to be used for many diverse applications.

Enhancements were also added to this board to increase functionality. The addition of an integrated wireless radio based on the 802.15.4 protocol (same radio protocol as the XBee) allows for wireless control of devices or wireless sensor data collection. Battery circuitry was added so that it could function as a true wireless node without any external power cables. The board is also fitted to a ruggedized enclosure so that the design can be transported safely or deployed in remote settings without worrying about damaging the circuit.

The design has also been optimized for low power. It consumes approximately 200 uA in power down/sleep mode at 3.0V which is the voltage supplied by 2 AA alkaline batteries. It consumes approximately 300 uA at 2.4V which is the voltage supplied by 2 AA NiMH rechargeable batteries. With proper power management, it's possible to have the device live for months on a single pair of batteries.

Radio

The main addition to this board is the integrated wireless radio. The radio is based on the 802.15.4 wireless protocol and is the same protocol used by XBee modules and Zigbee devices. The radio operates at 900 MHz and also contains an on-board RF front end for amplification. The transmit amplifier boosts the transmit signal from 10 mW with the bare radio to 500 mW. The receiver amplifier boosts the received signal by 11.7 dB, although there's typically a 6 dB gain when noise figure and other factors are taken into account.

It comes with an RP-SMA antenna connector which is a standard antenna connector commonly

found on Wi-Fi routers. An external antenna was chosen over other options such as a chip antenna or printed antenna because of the improved range and variety of available antennas.

The radio driver software and protocol stack are fully open source and available as an Arduino library. The protocol stack is simplified to three main library functions to make wireless communications as simple as possible. Those functions are: init, send, and receive. This makes it easy to use the radio as a simple extension of a serial port or to set up a peer-to-peer star network where each device can talk with any other device within listening range.

There are also many benefits to using 802.15.4 for communications. At 900 MHz, the signal is able to travel long distances and penetrate thick walls. There are a large variety of antennas in different sizes, shapes, and power for both frequencies. Omnidirectional antennas such as the standard whip antennas on Wi-Fi routers give moderate range and allow transmission from all directions. Directional antennas can also be used for greatly improved range if the direction of communications is fixed. The 900 MHz radio also has allows other ways to boost range such as changing the modulation from OQPSK to BPSK.

Some other benefits of using IEEE 802.15.4 is robustness such as automatic acknowledge and retries and functionality such as auto-discard of frames that don't match the particular network address, node address, or have their data corrupted. Auto retry means that if the receiver doesn't acknowledge that it received a packet, the transmitter will retry sending the packet up to some maximum number specified by the user. The auto checking of addresses and checksum means that the radio does most of the heavy lifting to ensure the packet arrives uncorrupted and the user doesn't have to do manual filtering.

The chipset used also has a hardware accelerator for AES-128 encryption. AES-128 is a strong encryption standard used by many government agencies to secure communications, especially wireless communications. The system also supports a true random number generator based on ambient RF signal levels. The chibi library includes support for AES-128 encryption using the onboard encryption engine as well as random number generation using the radio hardware.

Since 802.15.4 is also used for many communications protocols including Zigbee and 6LoWPAN, the Freakduino can be configured as a wireless protocol sniffer and used in conjunction with Wireshark to form a wireless protocol analyzer for 802.15.4 based protocols. This application is very useful to debug software for these protocols such as smart meter communications or for security testing of 802.15.4 based communications.

Since all software and hardware is open source, all registers and all features are available to the user. It allows for more flexibility for advanced users and a convenient prototyping and test platform for users that are considering an 802.15.4 wireless design of their own.

Power

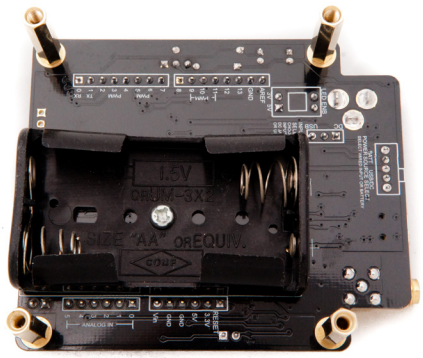
There are three options to provide power to the board. The most common is via the USB, however the board can also be powered via an external adapter connected to the DC jack. This is especially useful when more power is needed than can be provided by the USB. Finally, when no external power is available, the board can also be battery powered.

The USB connector provides up to 500 mA of current at 5V and can directly power the board for most applications. It's convenient when the device is connected to a PC since the USB to serial converter also allows for communications with the PC.

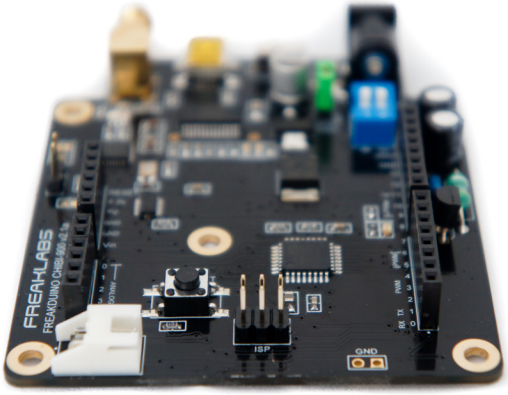
For designs with higher current requirements, such as driving motors or high power LEDs, an external DC adapter can be used. The adapter should be a minimum of 6V and a maximum of 14V.

> **Note:** The power input is determined by the position of the jumper below the DC input jack. The markings on the jumper indicate whether the USB or DC input is being used.

The board also has two separate connectors for batteries. One of the connectors allows a battery case to be mounted on the bottom side of the board. This is convenient since the battery case is directly mounted to the board and allows for battery operation in tight spaces.



The other connector is located near the bottom of the board and is a 2-wire connector that features mating and polarization. It's technically called a JST XH series connector and was chosen to allow easy attachment of external battery cables without worrying about reversing the positive and negative connections. It also makes connecting and disconnecting the battery case of the enclosure to the board much easier.



There is also a battery regulation circuit on the board. The reason this is needed is because the battery voltage varies based on the battery type and amount of charge left. The voltage regulation allows either standard alkaline or rechargeable NiMH batteries to be connected to the board and will generate a stable 5V output. The battery regulation circuit has up to two parallel stages. Each stage can provide current output of 200 mA for a maximum of 400 mA. This is fine for most sensors, however power-hungry devices like motors and power LEDs may exceed the maximum current output.

It's also possible to monitor the battery voltage through one of the analog input pins on the MCU. When combined with wireless communications, its possible to get early warning when the battery starts to get low.

There is a power switch on the board that selects between external line power or battery power. The switch can also be used as an OFF switch if either line power or battery power is not present.

There is also a 2-pin DIP switch on the board that enables or disables the LED indicators. If the board is externally powered, then it's fine to enable both LEDs, but when the board is battery powered, the LEDs will cause the battery to drain faster and should be disabled. The DIP switch allows the user to easily disable the LEDs during battery operation.

Connectors, Jumpers and Switches

The board contains a number of connectors, jumpers, and switches:

1. **Power Input Jumper.** The power input jumper is used to select between using the external DC jack or the USB to power the board.

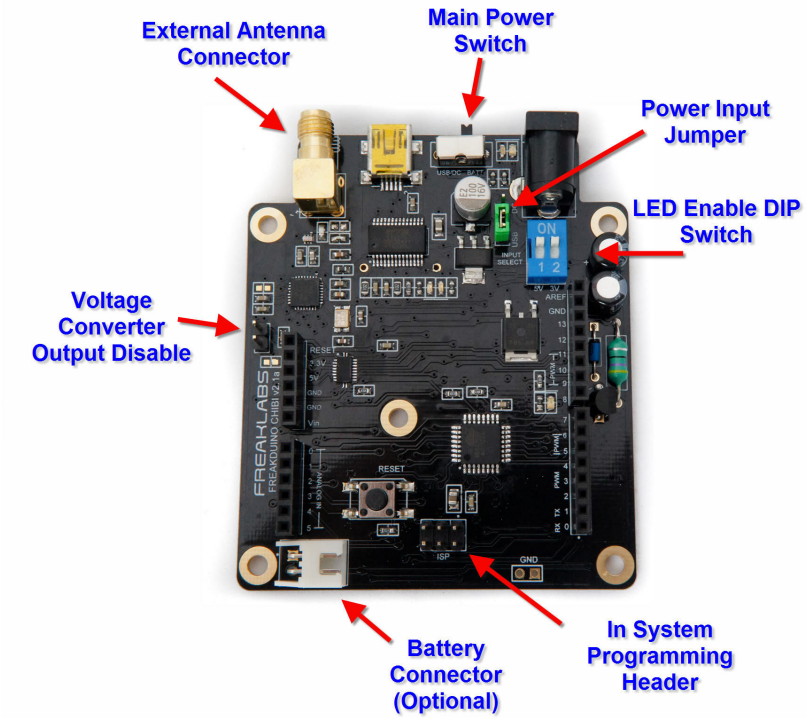
2. **Main Power Switch.** The main power switch toggles between using line power or battery power. When no battery is connected, this would also serve as the OFF position. The opposite is true when no line power is attached.

3. **LED Enable Switch.** The LED enable DIP switch is used to enable or disable the main power LEDs for 5V and 3.3V supplies. LEDs may consume unecessary power during battery operation so the DIP switch allows the user to enable/disable them when needed.

4. **External Antenna Connector.** The external antenna connector is an RP-SMA connector that attaches to a compatible antenna.

5. **Battery Connector.** The battery connector is optionally installed and comes with the battery boost kit.

6. **Voltage Converter Output Disable.** This is only used when programming the bootloader on the board to prevent collisions on the SPI bus. This jumper should rarely be used and only when flashing the bootloader or using an in-system programmer (ISP).



Due to having three ways to power the board, the power switch and power jumper settings need to be configured for your desired input. The switch and jumper is labeled to make things easy, but here's a quick reference chart on how to configure the power input. The positions in the table correspond to the labels on the board.

	Main Power Switch	Power Input Jumper
USB Power	USB/DC	USB PWR
External DC Power	USB/DC	DC PWR
Battery Power	BATT	Ignored

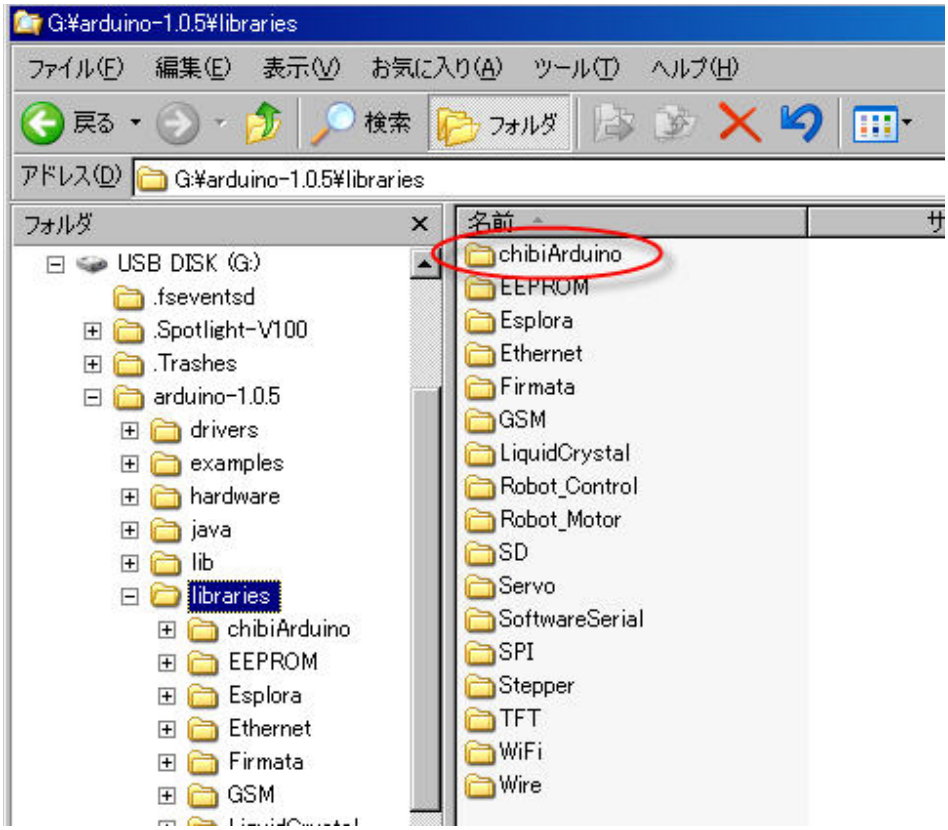
Installing the chibiArduino Library

Once you have the board configured properly, it's time to install the library. The first thing you need to do is get the Arduino IDE and install it. It can be found at the Arduino site. From there, it becomes OS specific:Download the chibiArduino library. You can get it from the FreakLabs website: http://www.freaklabs.org/index.php/chibiArduino.html

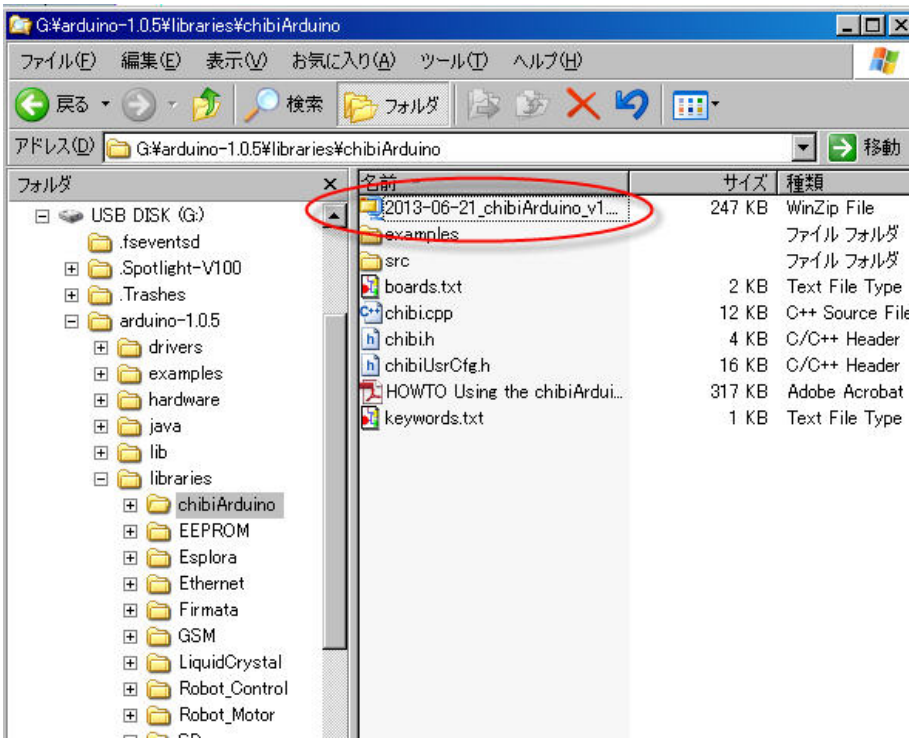
From there, it's OS-specific:

Windows

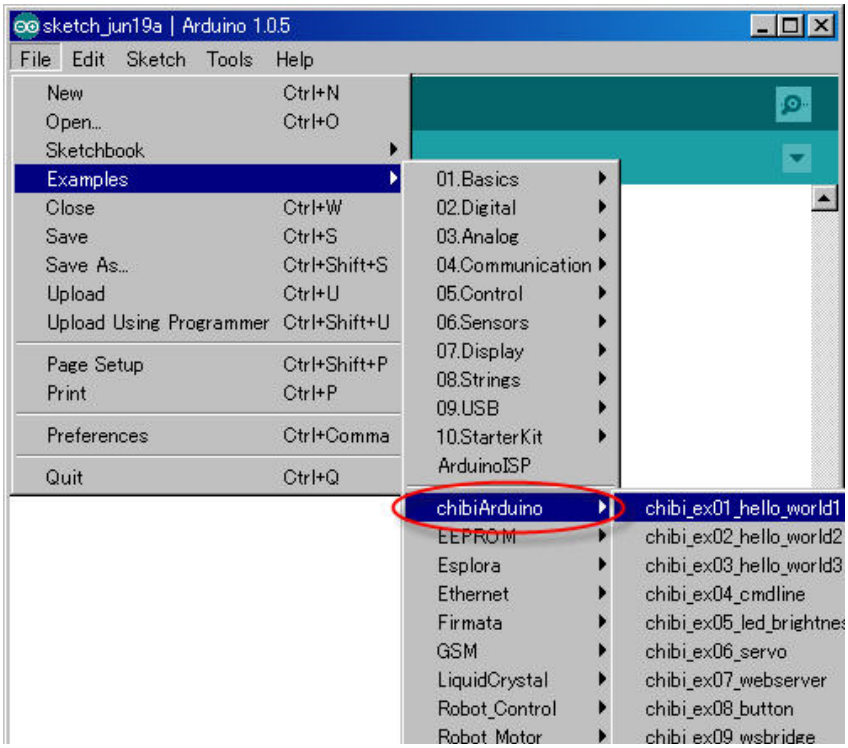
1. Copy the chibiArduino library zip file. Go to the Arduino root directory and then go into the libraries directory:



2. Create a folder called chibiArduino. Paste the chibiArduino library zip file in the folder you just created. Then unzip the chibiArduino library.

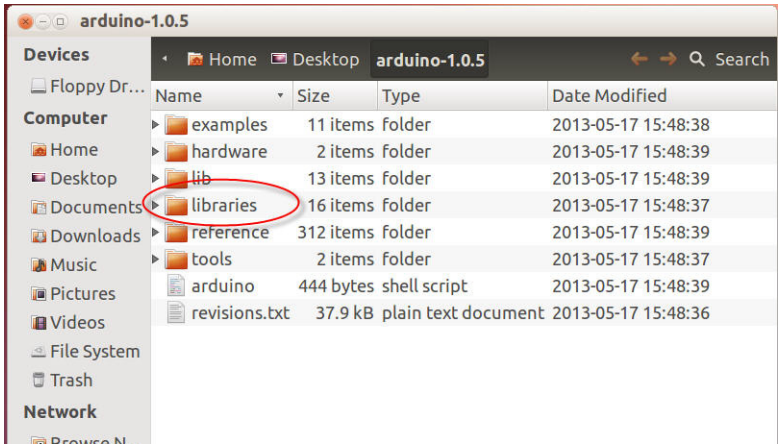


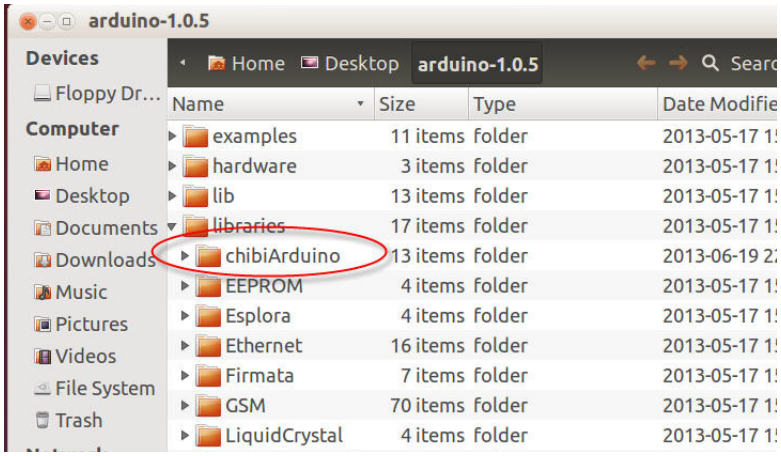
3. The library should now be installed. We're going to verify that it's in there properly. Start up the Arduino IDE. If it is already open, close it and re-start it. It only checks for new folders and libraries on startup so this is required. Once it starts up, go to the File/Examples menu item and look at the examples listing. You should see the chibiArduino examples:



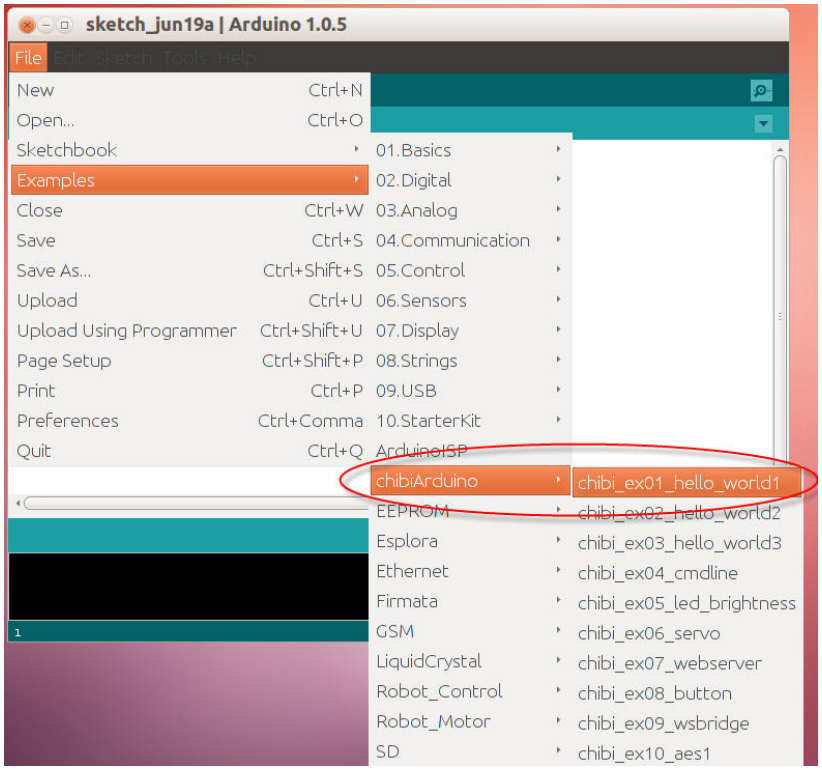
Linux

1. Download the chibiArduino library zip file. Go to your <ARDUINO_ROOT>/libraries folder and create a folder called chibiArduino inside. Move the chibiArduino zip file into the chibiArduino folder and unzip.



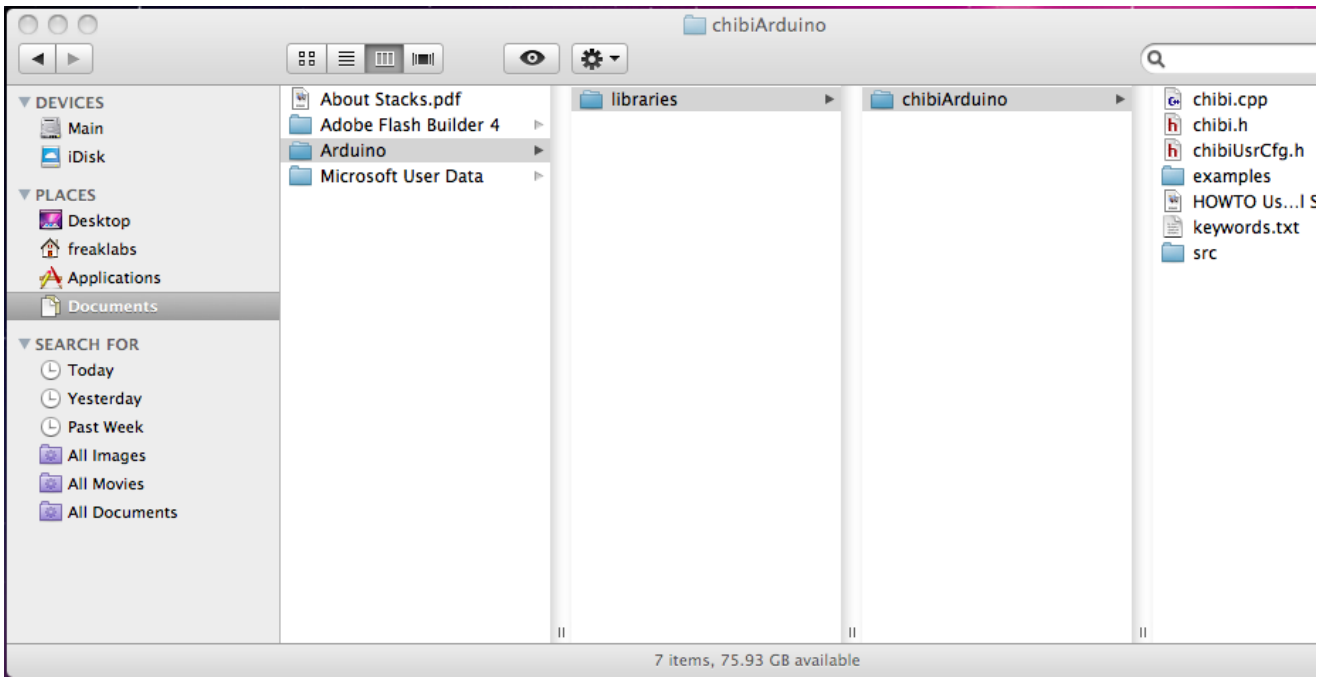


2. Once the file is unzipped inside the chibiArduino folder, the library should be installed. Now we're going to check to make sure it's seen properly. Start up the Arduino IDE. If it's already open, restart the IDE since it only scans for new folders and libraries on startup. Go to the Files/Examples menu item and look at the examples listing. You should see the chibiArduino examples. Select any of the examples to open.

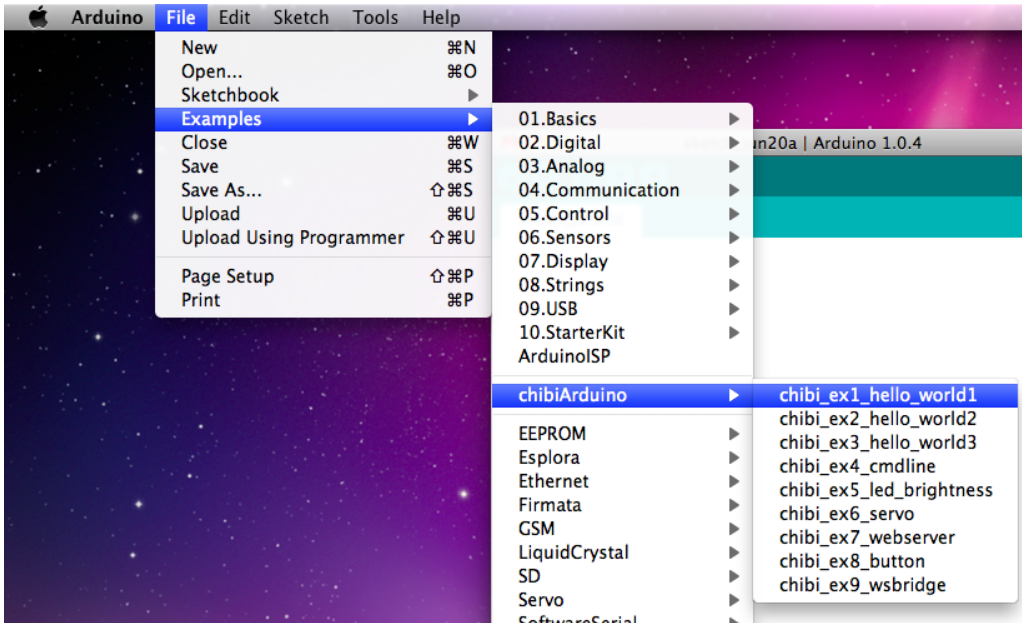


Mac OSX

1. Create a folder called "libraries" in the /Documents/Arduino folder on your system. Make sure the Arduino IDE is installed first. Create a folder called "chibiArduino" in the Documents/Arduino/libraries directory. Unzip the chibiArduino library in this folder.



2. Once the chibiArduino library is installed, then we can check to make sure the Arduino IDE sees it. Start the Arduino IDE. If it's already open, restart the Arduino IDE. It only scans for new folders and libraries on startup so this step is important. Then go to the Files/Examples menu item and check the Examples listings. You should see the chibiArduino examples.



Installing the Board Support Package

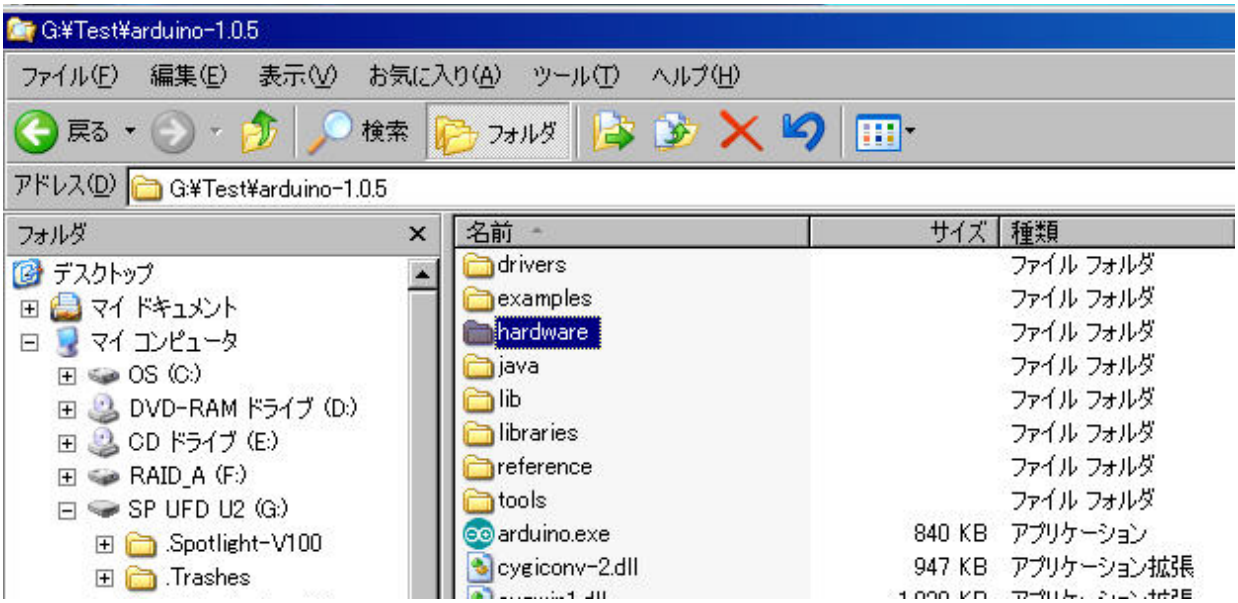
Along with installing the library, a board support package is also required to be installed. This extra step makes it possible to identify which board is being used, which is especially useful since there are multiple variations on the Freakduino design depending on features.

Windows

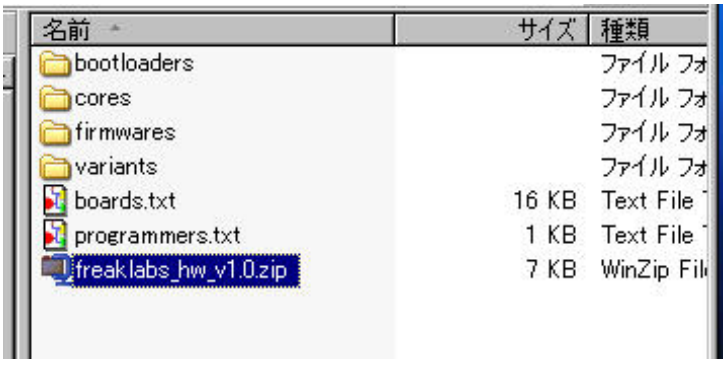
1. *First download the FreakLabs board support package located here:*

http://www.freaklabsstore.com/pub/freaklabs_hw.zip

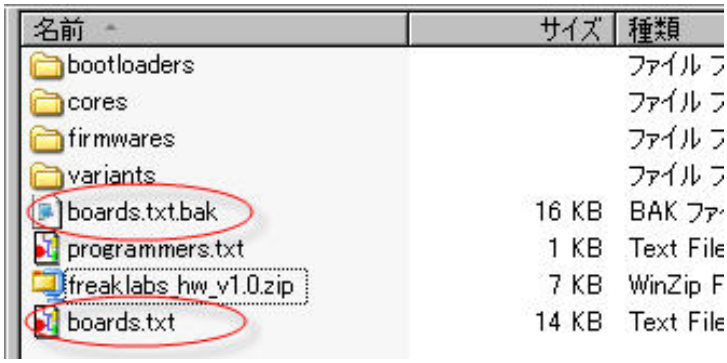
2. *In Windows, go to the Arduino root directory. You should see the "hardware" directory in the root.*



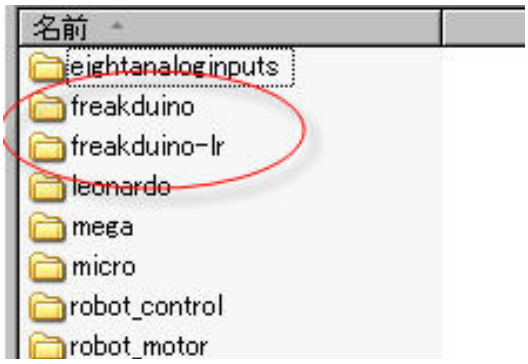
3. *Put the board support package in the "<ARDUINO_ROOT>/hardware/arduino" directory. It's best to also make a backup of your "boards.txt" file.*



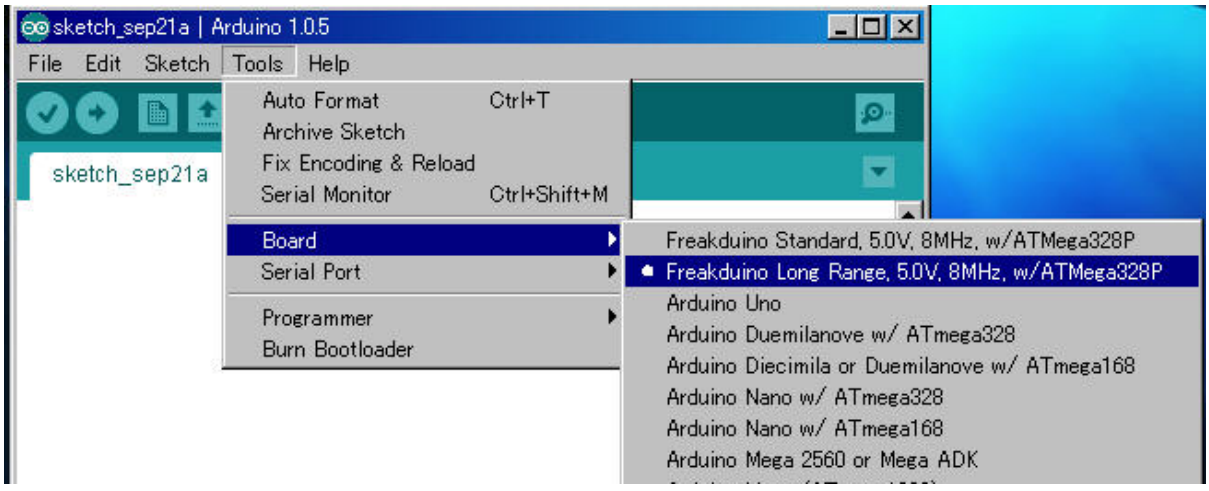
4. *Unzip the file from the board support zip file into its the directory. It should overwrite the current "boards.txt" file which is why its best to backup the current boards.txt file before performing this operation.*



5. It will also add the following two directories to the "variants" directory. These board variants are for the "freakduino" and "freakduino-lr" and contain information that will allow the Arduino IDE to add the particular boards to the Tools/Board menu. This is the view inside the "<ARDUINO_ROOT>/hardware/arduino/variants" directory.



6. Verify that the boards are installed properly. Open the Arduino IDE, go to the Tools/Board menu option and check to see that the boards are installed properly. You can also remove any unwanted boards from the "boards.txt" file to avoid cluttering the Tools/Board menu.

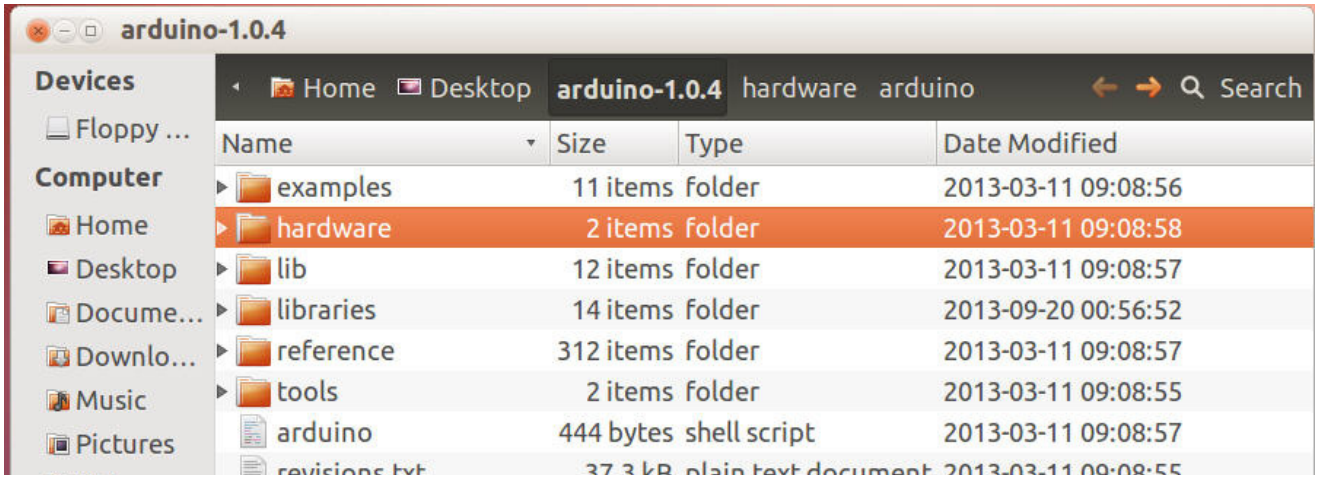


Linux

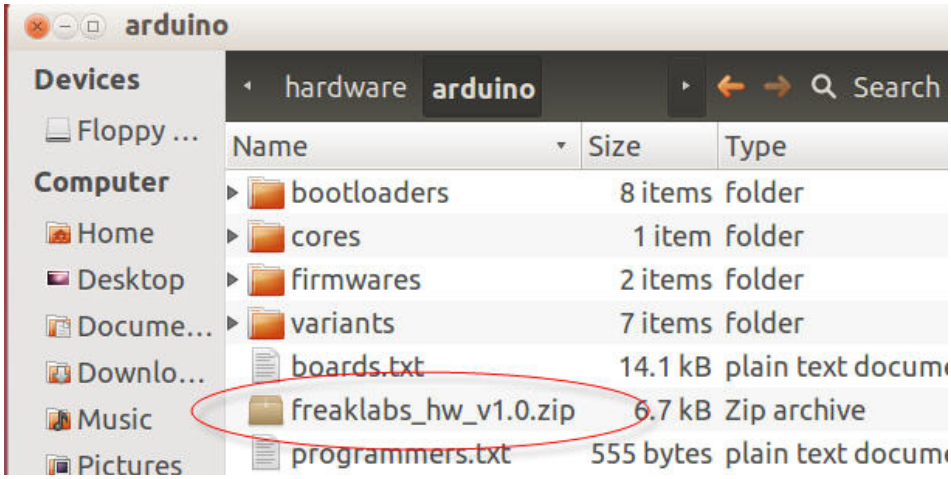
1. Download the FreakLabs board support package located here:

http://www.freaklabsstore.com/pub/freaklabs_hw.zip

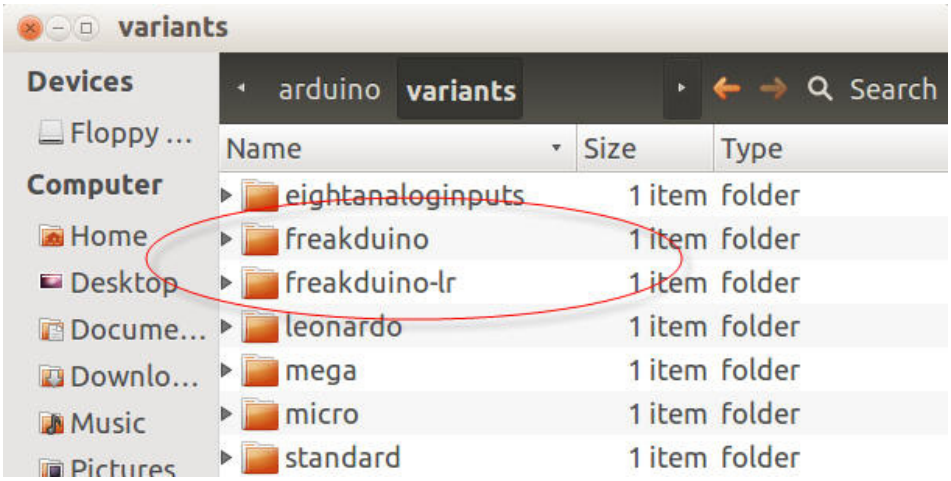
2. Go to the Arduino IDE root directory. You should see the "hardware" folder.



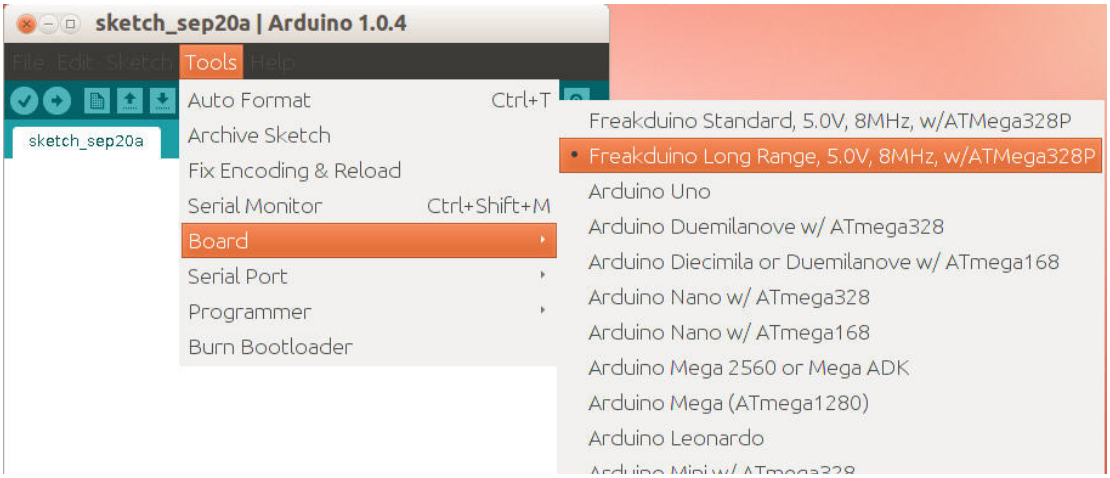
3. Put the board support package in the "<ARDUINO_ROOT>/hardware/arduino" directory. It's best to also make a backup of your "boards.txt" file. Unzip the board support zip file in this directory. It will overwrite the the current "boards.txt" file so its best to have this backed up.



4. It will also add the following two directories to the "variants" directory. These board variants are for the "freakduino" and "freakduino-lr" and contain information that will allow the Arduino IDE to add the particular boards to the Tools/Board menu. This is the view inside the "<ARDUINO_ROOT>/hardware/arduino/variants" directory.



5. Verify that the boards are installed properly. Open the Arduino IDE, go to the Tools/Board menu option and check to see that the boards are installed properly. You can also remove any unwanted boards from the "boards.txt" file to avoid cluttering the Tools/Board menu.

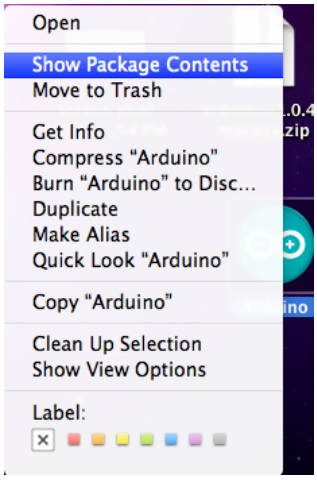


Mac OSX

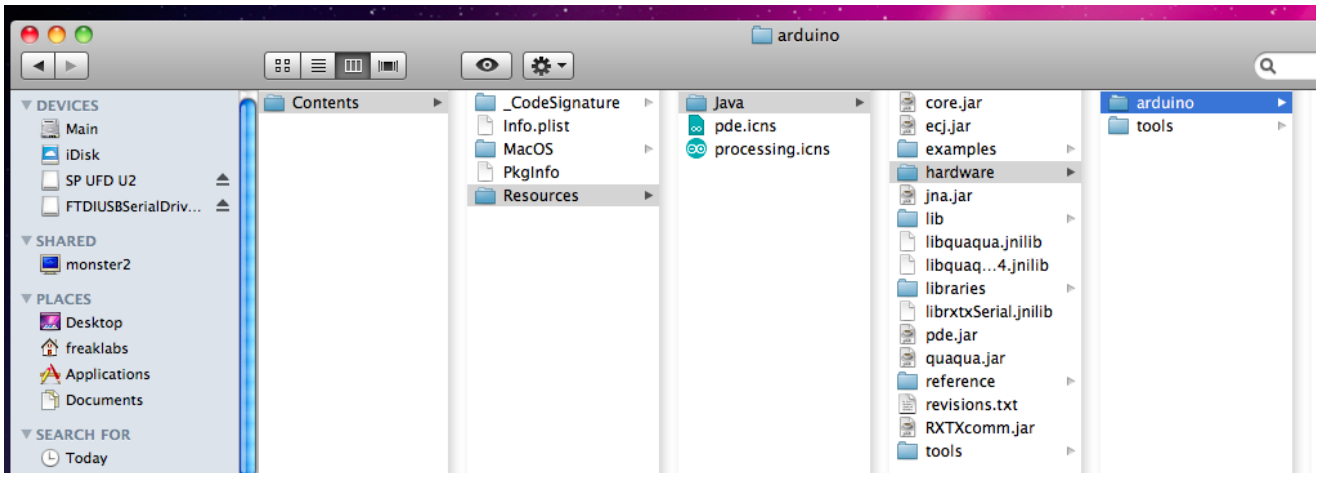
1. First download the FreakLabs board support package located here:

http://www.freaklabsstore.com/pub/freaklabs_hw.zip

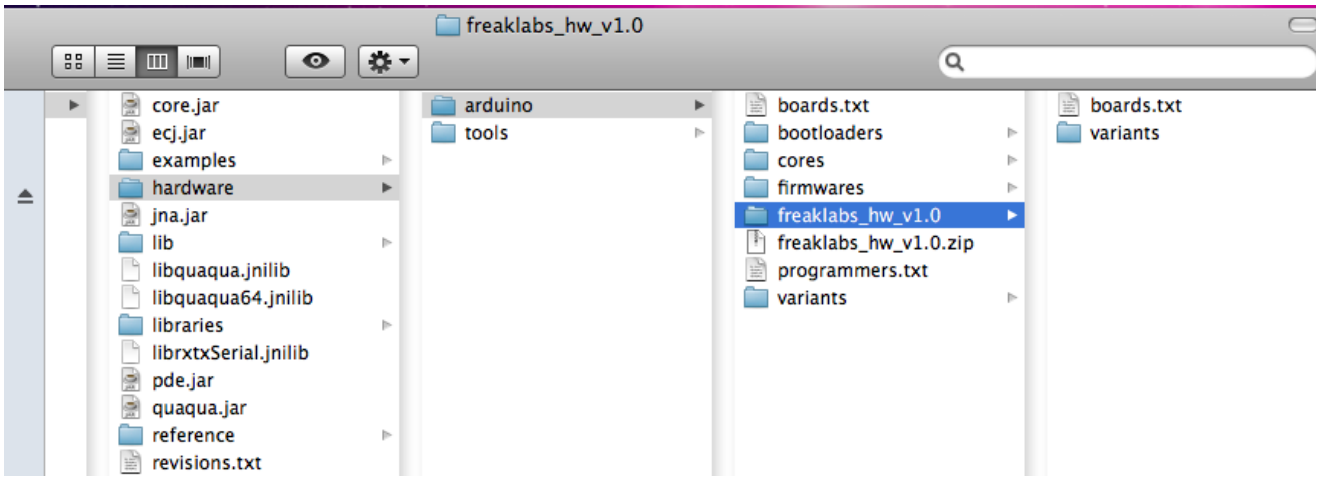
2. In Mac OSX, right-click or ctrl-click on the Arduino icon to get the context menu. Choose "Show Package Contents" and navigate to the "Contents/Resources/Java" menu. This will take you to the Arduino IDE root directory.



3. In the Arduino IDE root directory, navigate to the "<ARDUINO_ROOT>/hardware/arduino" directory.

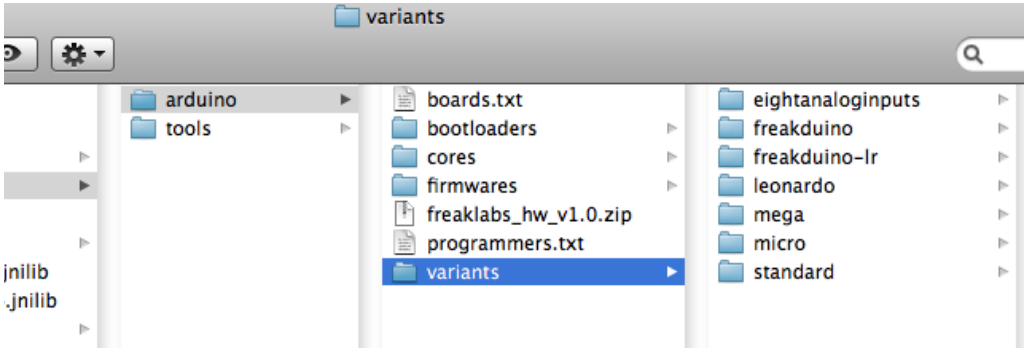


4. Put the board support package in the "<ARDUINO_ROOT>/hardware/arduino" directory. It's best to also make a backup of your "boards.txt" file. Unzip the board support zip file in this directory. It will create a new directory with the name of the zip file. Inside that directory, you should find the "boards.txt" file and a folder called "variants".

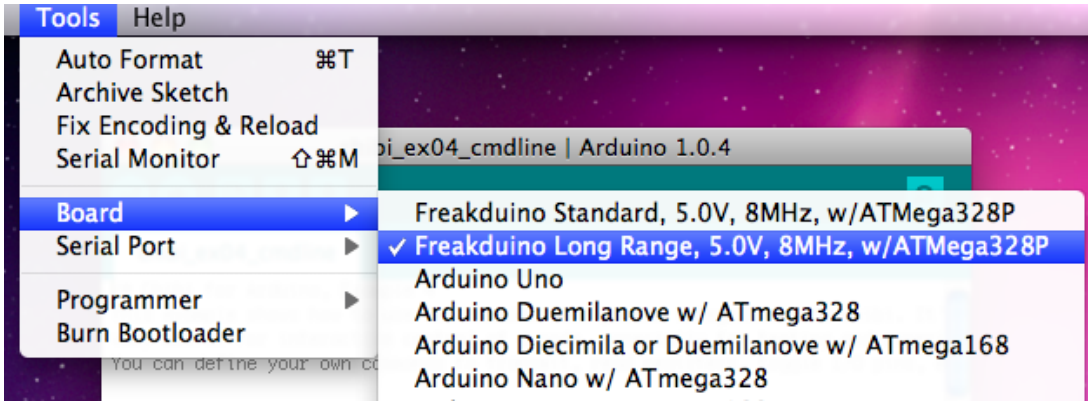


*5. While inside the newly created directory, copy the new "boards.txt" file into the "<ARDUINO_*

ROOT>/hardware/arduino" directory. Go into the variants directory and copy the folders "freakduino" and "freakduino-lr". Paste those folders into the "<ARDUINO_ROOT>/hardware/arduino/variants" directory. It should look like the following.



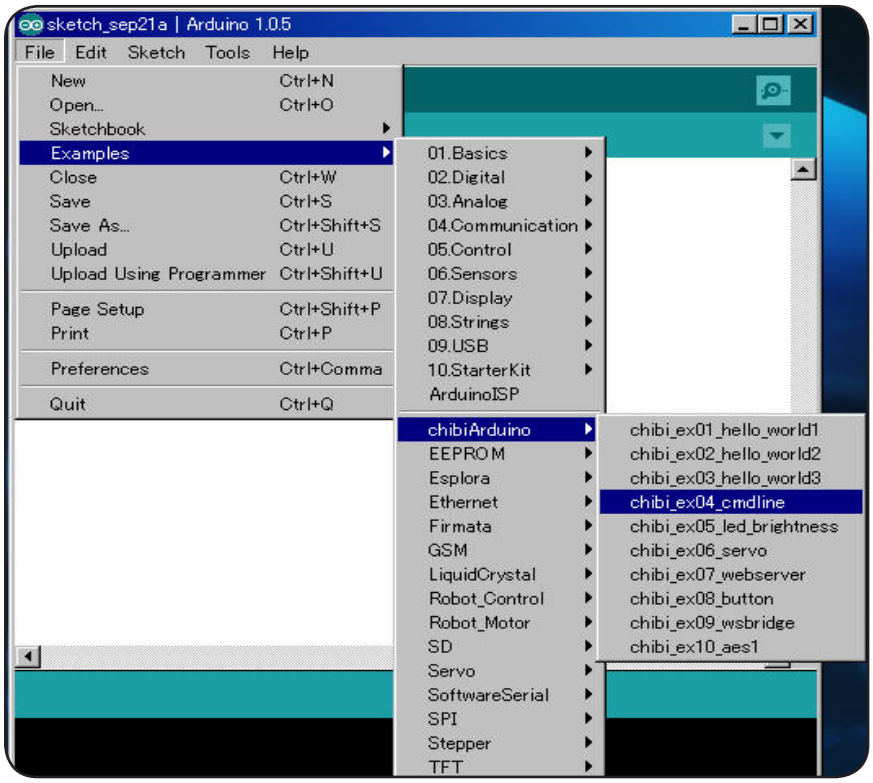
6. Verify that the boards are installed properly. Open the Arduino IDE, go to the Tools/Board menu option and check to see that the boards are installed properly. You can also remove any unwanted boards from the "boards.txt" file to avoid cluttering the Tools/Board menu.



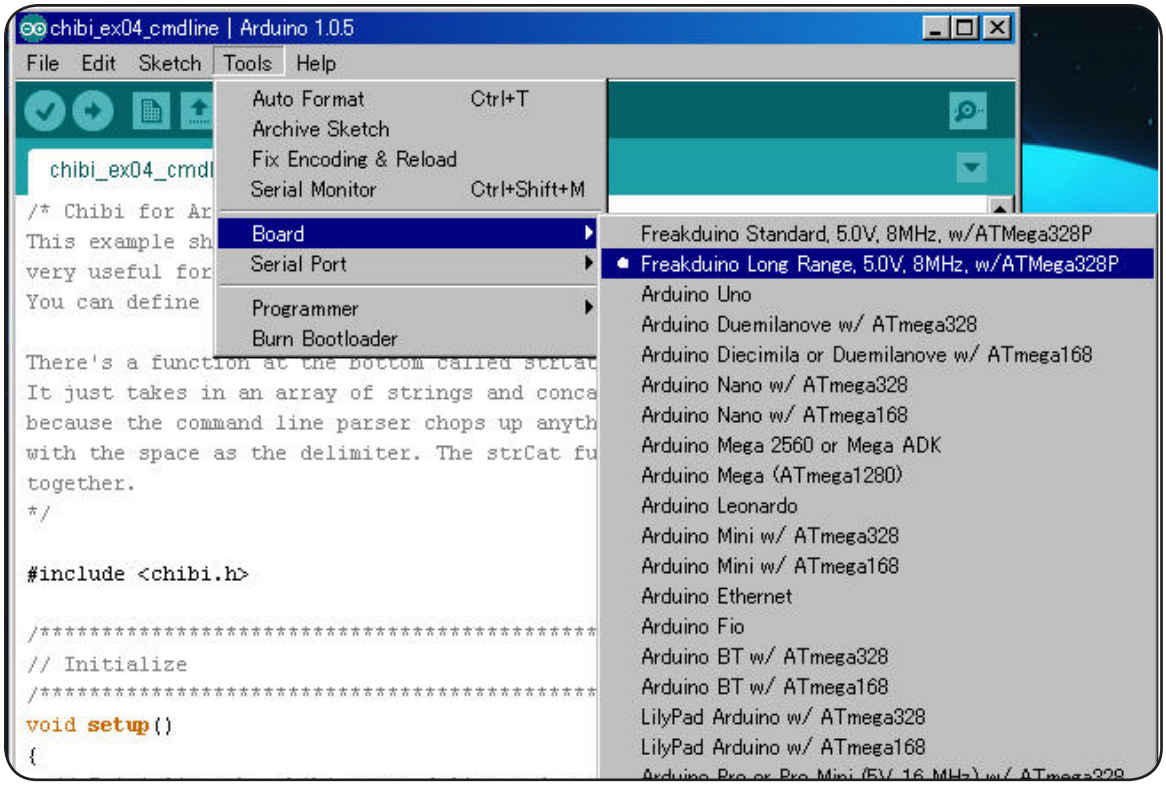
Uploading Code

At this point, you should be able to upload code to the board. This is the procedure.

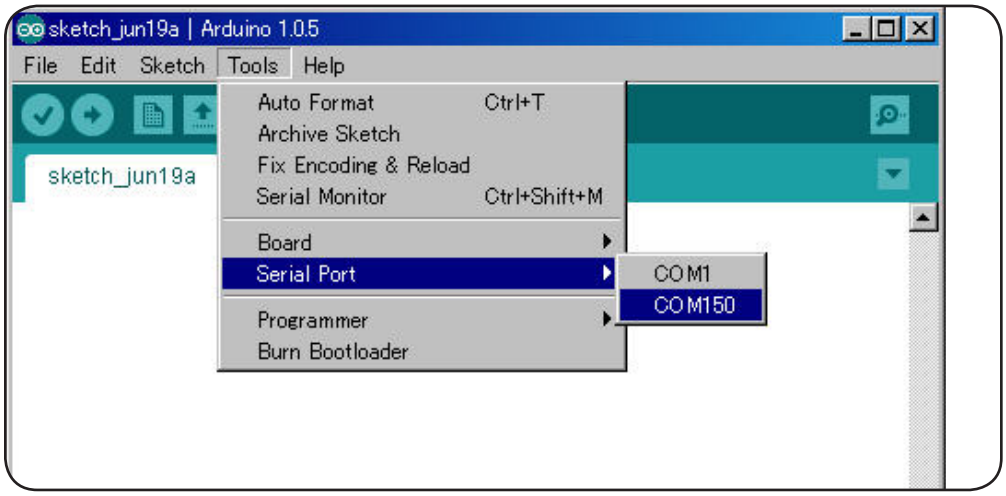
1. **Open the IDE and add code.** We're going to take a shortcut and open up one of the default examples that comes with the Arduino software. Go to the "File/Examples/ chibiArduino" menu and select "chibi_ex04_cmdline".



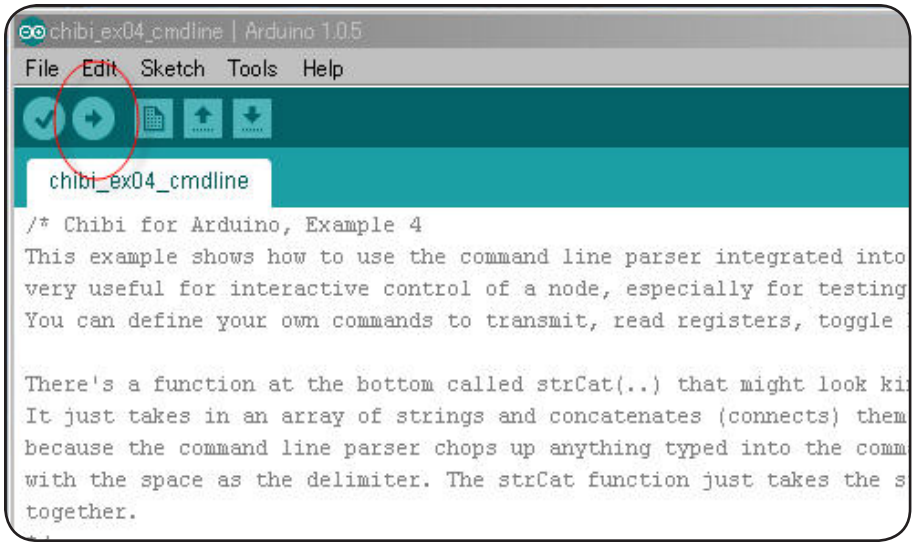
2. **Select the board.** Go to the "Tools/Board" menu and select "Freakduino Long Range, 5.0V, 8MHz, w/ATMega328P".



3. **Select the serial port.** Plug the USB connector into the Freakduino board and select the serial port that it's connected to.



4. **Click on the "Upload" icon.** The code should compile and start uploading. You'll also see the serial port's TX and RX LEDs blinking on the Freakduino Long Range board.



Congratulations! You just uploaded your code to the Freakduino Long Range board :)

Compatibility

The FREAKDUINO 900 MHz Long Range board is essentially an Arduino-based system and a shield integrated together. This means that the peripheral will require using some dedicated pins for the wireless functionality.

Two pins on the analog input connector are not available for use except for standard digital I/O. These are pins Analog 2 and Analog 3. Analog 2 controls the sleep mode and Analog 3 controls the chip select for communications between the microcontroller and radio IC. If the wireless functionality is not used, these pins are available as standard digital I/O but not as analog inputs. If possible, its best to avoid using these pins.

If the wireless functionality is being used, the SPI bus is also required to communicate with the radio. That means that digital pins 10-12 (PB3 to PB5 or MOSI, MISO, and SCLK) will be dedicated SPI pins. Please be especially careful when using the AUX LED on the board. It is located on the SCLK pin required by the SPI. If the radio will be used, then the AUX LED should not be used in the sketch.

If there is any question about compatibility with a particular shield, please post to the FreakLabs forums or email *support@freaklabsstore.com*.

License

Disclaimer

The FREAKDUINO 900 MHz Long Range board is NOT FCC approved. It is designed to comply with FCC Part 15 rules. However this board is not in a finished product form and is only intended for experimental and research/development purposes. If you wish to use this board in an actual product, you will need to attain certification with the appropriate local regulatory body for the complete system. Additionally, please use the wireless equipment in a responsible manner with regard for others and your surroundings.

Troubleshooting

There are some common problems that people run into when using the Freakduino board. The following is a list of issues people have had and the fixes for them:

1. **Serial port does not work at speed specified in Arduino sketch.** This is most likely

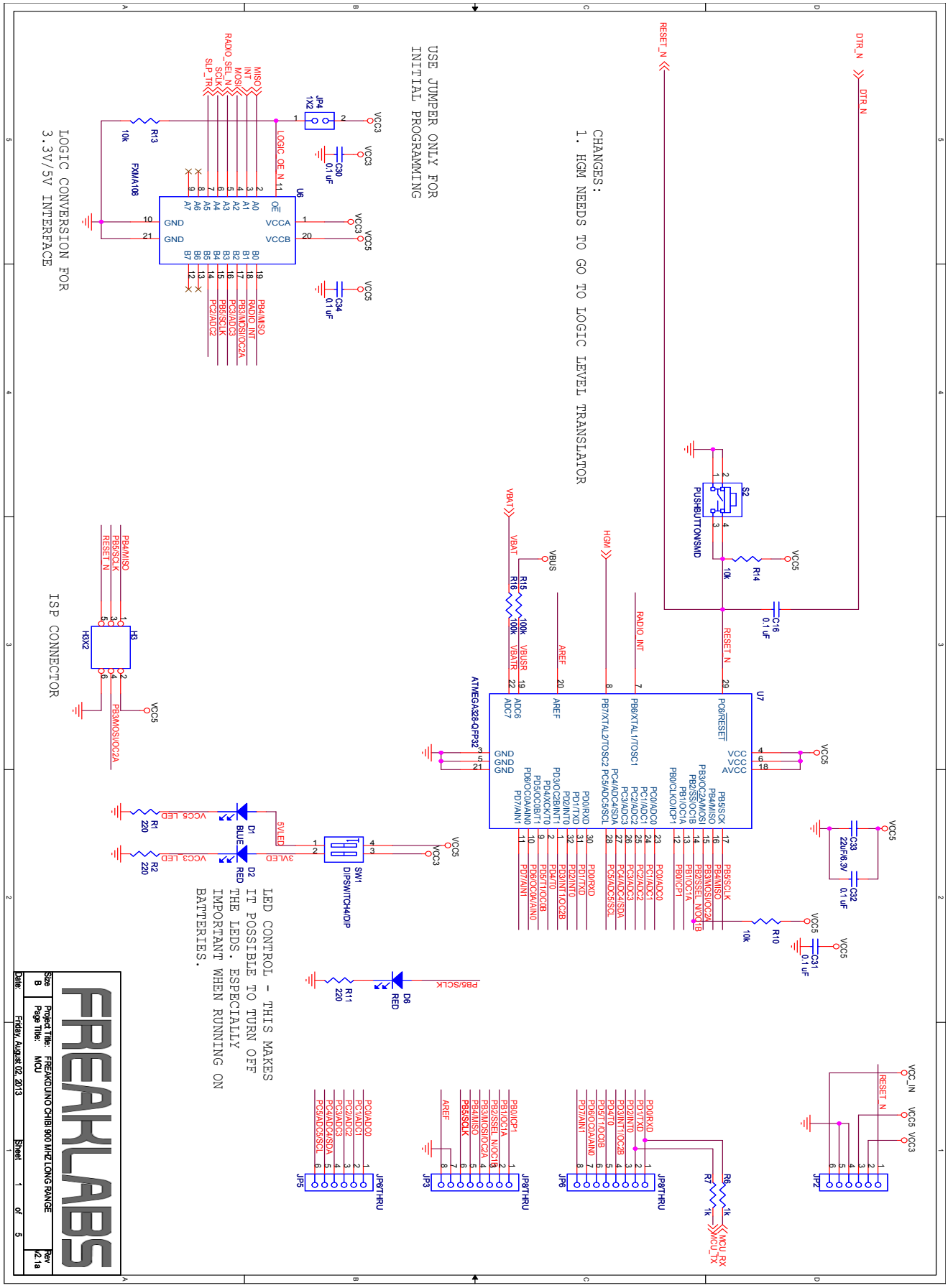
due to having the wrong "Board" setting in the Arduino IDE. Make sure the board "Freakduino Long Range, 5.0V, 8 MHz, w/ATMega328P" is selected in the Tools/Board menu.

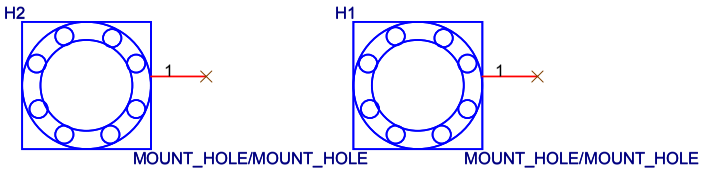
2. **Radio is not working or displays "RADIO NOT INITIALIZED PROPERLY" on serial terminal.** First, make sure that the voltage converter disable jumper is not mounted. It should not be on in normal operation. The next thing to try when you see that message is to load the example sketch "chibi_ex4_cmdline" with nothing connected to the Freakduino. If the error message disappears, then its likely that some pins needed for radio communications (like the SPI pins) are also being used in the sketch. One of the main culprits of this issue is when a sketch uses the AUX LED on the board. The LED is standard on Arduinos but is unfortunately connected to the SCLK pin required by SPI. If the radio is going to be used, the AUX LED should not be used. Please see the "Compatibility" section of this document.

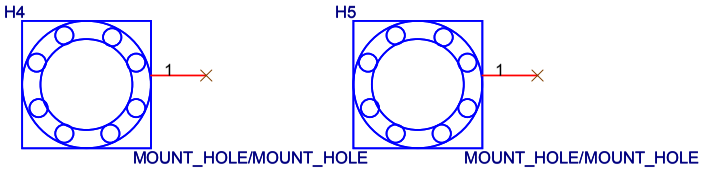
If the error message still shows with the example sketch, then there is likely a hardware problem. Please contact me through the FreakLabs website, email me at chris(at)freaklabs(dot)org, or post on the forums.

Schematics

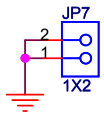
Schematics can be found on the following page:


CHANGES:
1. HGM NEEDS TO GO TO LOGIC LEVEL TRANSLATOR
USE JUMPER ONLY FOR INITIAL PROGRAMMING
LOGIC CONVERSION FOR 3.3V/5V INTERFACE
ISP CONNECTOR
LED CONTROL - THIS MAKES IT POSSIBLE TO TURN OFF THE LEDS. ESPECIALLY IMPORTANT WHEN RUNNING ON BATTERIES.
FREAKLABS
Project Title: FREAKDUINO CHIBI 900 MHZ LONG RANGE
Page Title: MCU
Rev: 2.1a
Size: B
Date: Friday, August 02, 2013
Sheet 1 of 5



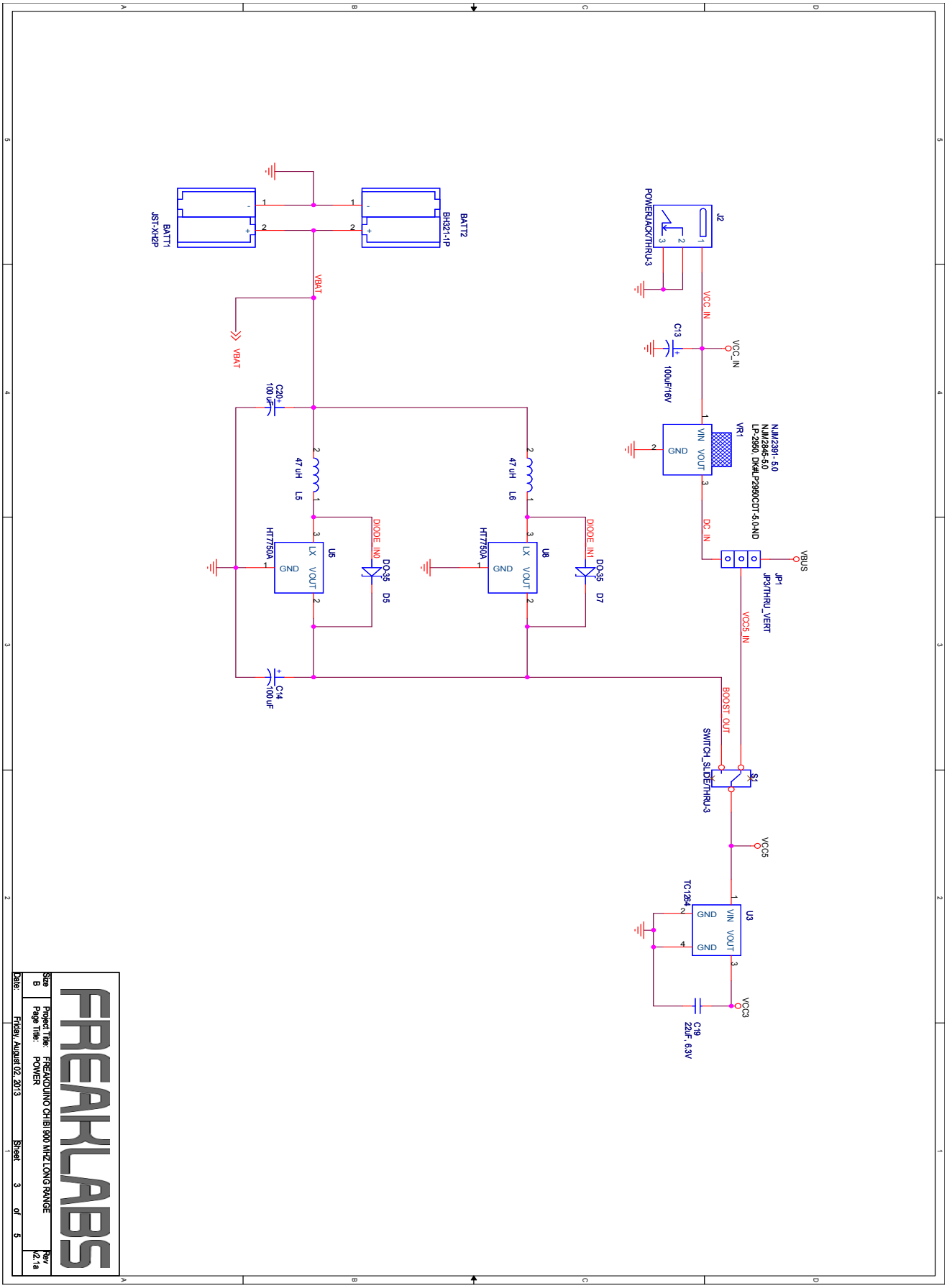


GND TESTPOINTS FOR DEBUGGING



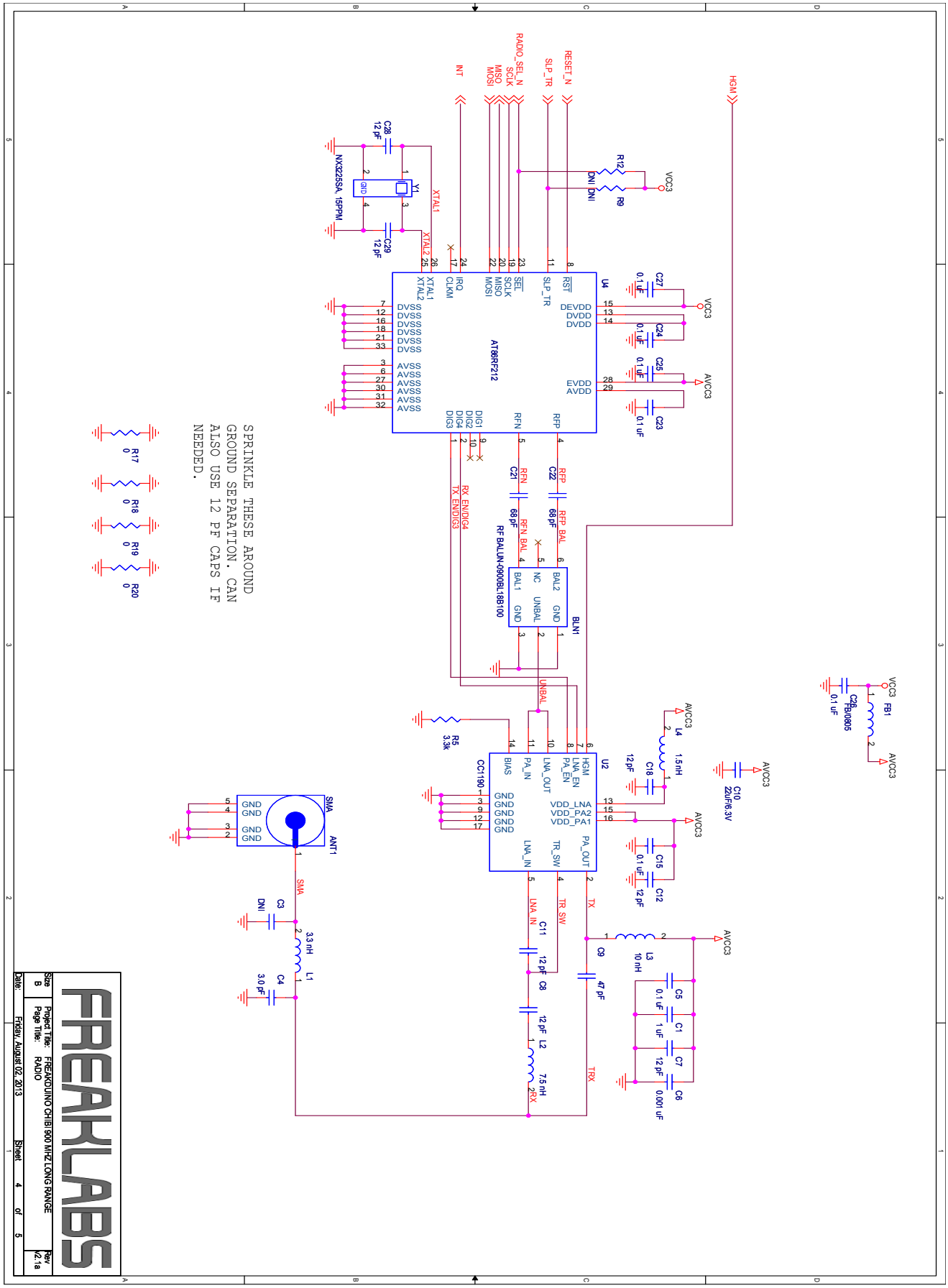
FREAKLABS

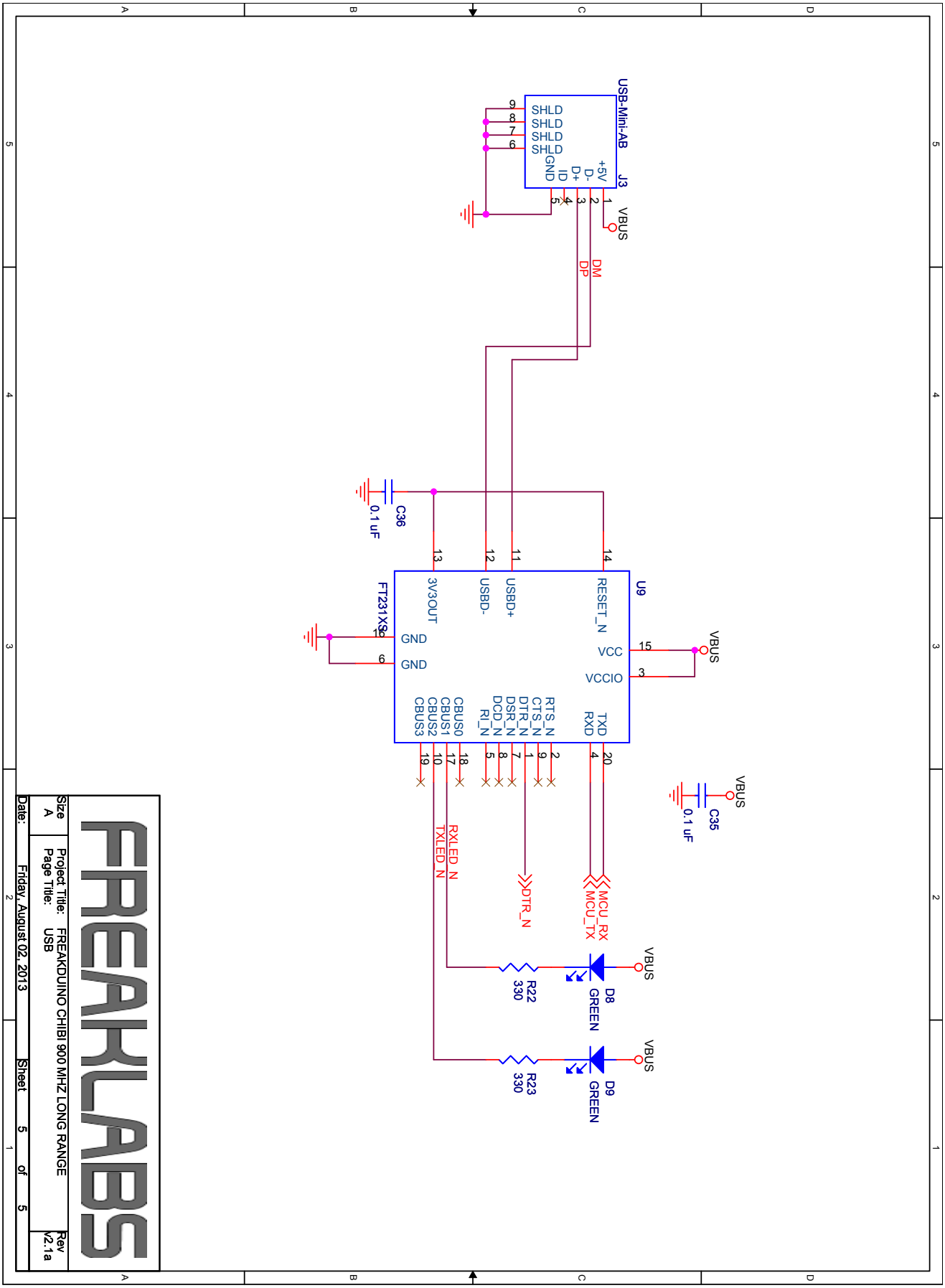
Size A	Project Title: FREAKDUINO CHIBI 900 MHZ LONG RANGE Page Title: MISC	Rev v2.1a
Date: Friday, August 02, 2013	Sheet 2 of 5	



FREAKLABS

Size: B	Project Title: FREAKDUINO CHIBI 900 MHZ LONG RANGE	Rev: V2.1a
Date: Friday, August 02, 2013	Page Title: POWER	Sheet 3 of 5

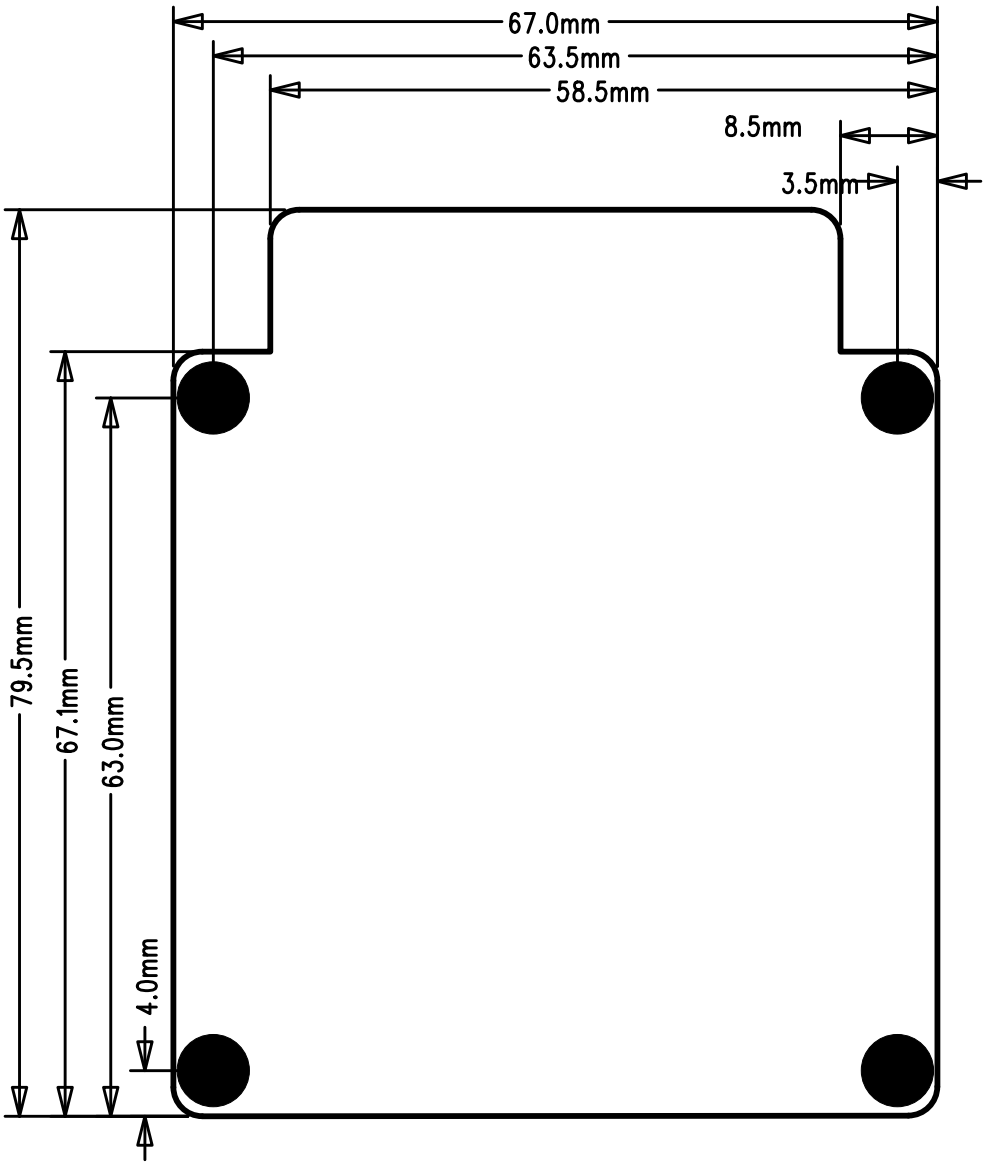

SPRINKLE THESE AROUND GROUND SEPARATION. CAN ALSO USE 12 PF CAPS IF NEEDED.
FREAKLABS
Project Title: FREAKDUINO CHIBI 900 MHZ LONG RANGE
Page Title: RADIO
Size: B
Date: Friday, August 02, 2013
Sheet 4 of 5
Rev: 2.1a

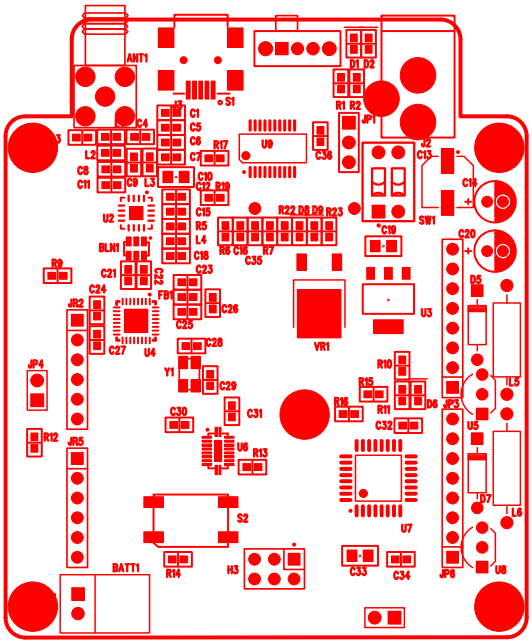

USB-Mini-AB
J3
SHLD
SHLD
SHLD
SHLD
GND
ID
D+
D-
+5V
VBUS
DM
DP
C36
0.1 uF
U9
FT231XS
3V3OUT
USBD-
USBD+
RESET_N
VCC
VCCIO
GND
GND
TXD
RXD
RTS_N
CTS_N
DTR_N
DSR_N
DCD_N
RI_N
CBUS0
CBUS1
CBUS2
CBUS3
C35
0.1 uF
RXLED_N
TXLED_N
DTR_N
MCU_RX
MCU_TX
R22
330
R23
330
D8
GREEN
D9
GREEN
FREAKLABS
Size
A
Project Title:
FREAKDUINO CHIBI 900 MHZ LONG RANGE
Page Title:
USB
Rev
v2.1a
Date:
Friday, August 02, 2013
Sheet
5
of
5

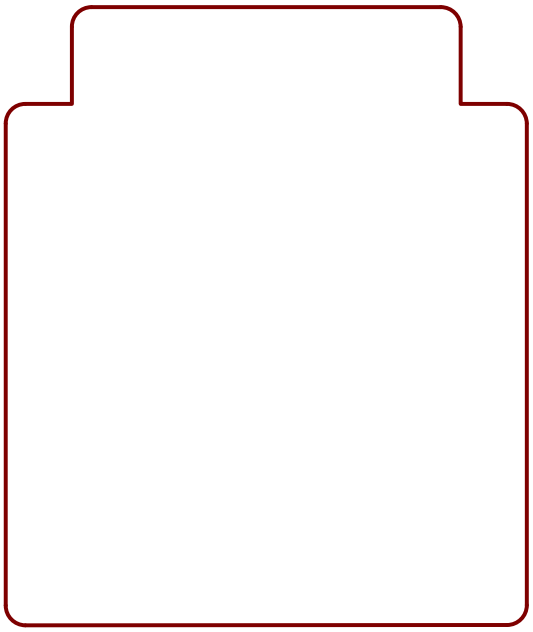
PCB Layout

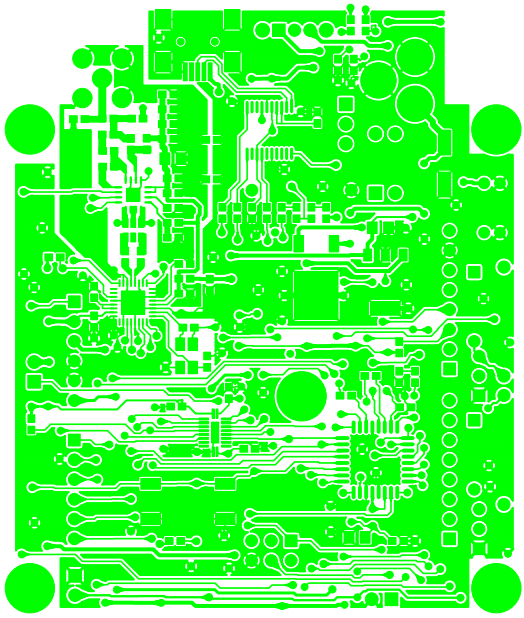
PCB layout file order:

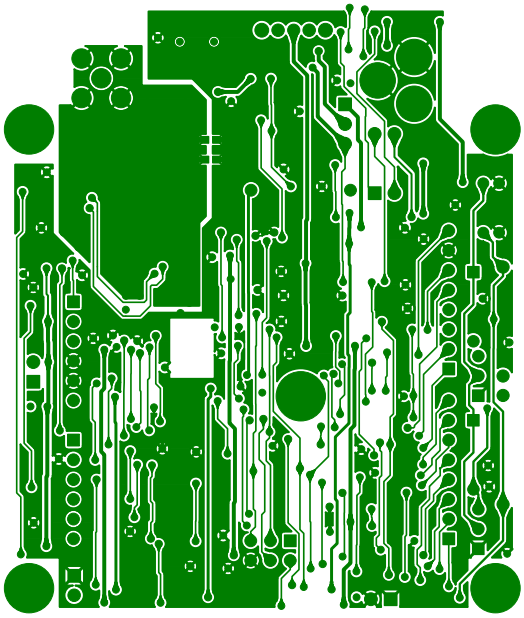
1. *Mechanical Dimensions*
2. *Top Layer*
3. *Bottom Layer*
4. *Assembly Drawing - Silkscreen View*
5. *Assembly drawing - Reference Designator View*

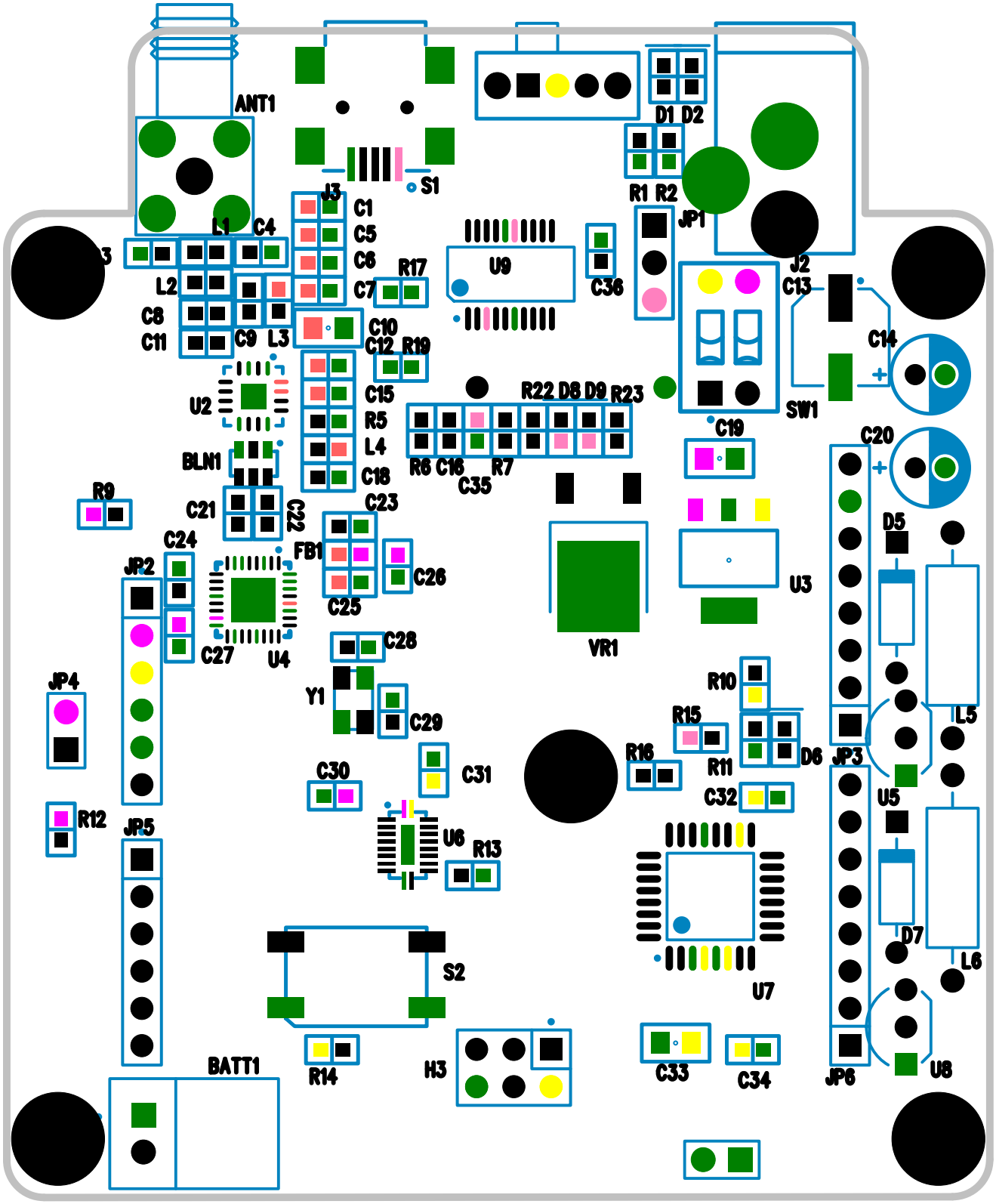











ANT1
S1
J3
C1
C5
C6
C7
L1
C4
L2
C8
C11
C9
L3
C10
C12
R17
R19
U2
C15
R5
L4
C18
BLN1
C21
C22
C23
FB1
C24
JP2
C27
U4
C26
C25
R9
JP4
R12
JP5
BATT1
C28
Y1
C29
C30
C31
U6
R13
S2
R14
H3
U9
C36
D1
D2
R1
R2
JP1
J2
C13
C14
SW1
R22
D8
D9
R23
R6
C16
R7
C35
C19
C20
D5
U3
VR1
R10
R15
R11
D6
R16
C32
JP3
U5
L5
D7
L6
U7
C33
C34
JP6
U8

Bill of Materials

Quantity	Reference	Manufacturer	Part Number	Description
1	ANT1	ChangHong	SMA-02-113-TGG	RP-SMA Connector
1	BATT1	JST	JST-XH2P	2-pin mated, polarized connector, male
1	BATT2	COMF	BH321-1P	2-AA battery case
1	BLN1	Johanson Technology	0900BL18B100E	900 MHz, 100/50 ohm balun
1	C1	Various	1 uF	1 uF/50V
3	C3,R9,R12		DNI	
1	C4	Various		3.0 pF/50V, 0603 RF
14	C5, C15, C16, C23, C24, C25, C26, C27, C30, C31, C32, C34, C35, C36	Various		0.1 uF/50V, 0603
1	C6	Various		0.001 uF/50V, 0603
7	C7, C8, C11, C12, C18, C28, C29	Various		12 pF/50V, 0603
1	C9	Various		47 pF/50V, 0603
2	C10, C19, C33	Various		22uF/6.3V, 0805
1	C13	Various		100uF/16V, electrolytic
2	C14, C20	Various		100 uF/35V, electrolytic
2	C21, C22	AVX	06035A680JAT2A	68 pF/50V, 0603, RF
1	D1	Various		Blue LED, 0603
2	D2, D6	Various		Red LED, 0603
2	D5, D7	Vishay	BAT43	Schottky Diode
2	D8, D9	Various		Green LED, 0603
1	FB1	Various		Ferrite Bead, 0603
1	H3	Various	H3X2	3x2 Straight male header, 0.100”
1	JP1	Various	JP3/THRU_VERT	1x3 Straight male header, 0.100”
2	JP3, JP6	Various	JP8/THRU	1x8 Straight female header, 0.100”
2	JP4, JP7	Various	JP2/THRU	1x2 Straight male header, 0.100”
1	JP2, JP5	Various	JP6/THRU	1x6 Straight female header, 0.100”

1	J2	4UCON	5537	DC Power Jack, 2.0 mm center conductor
1	J3	4UCON	9558	USB Mini-AB connector
1	L1	Johanson Technology	L-14C3N3SV4T	3.3 nH RF inductor
1	L2	Taiyo Yuden	HK10057N5J-T	7.5 nH
1	L3	Panasonic	ELJ-RE10NGFA	10 nH
1	L4	Panasonic	ELJ-RE1N5DFA	1.5 nH
2	L5, L6	Various		47 uH inductor, 200 mA
3	R1, R2, R11, R22, R23	Various		220 ohms, 0603
1	R5	Various		3.3 kohm, 0603
2	R6, R7	Various		1 kohm, 0603
3	R10, R13, R14	Various		10 kohms, 0603
2	R15,R16	Various		100 kohms, 0603
4	R17, R18, R19, R20	Various		0 ohms, 0603
1	SW1	Various		2-input DIP switch
1	S1	Various		Right angle slide switch
1	S2	Various		SMD SPST Tactile Switch/Pushbutton
1	U2	TI	CC1190	900 MHz RF Front End IC
1	U3	Microchip	TC1264-3.3	3.3V Regulator
1	U4	Atmel	AT86RF212	900 MHz, 802.15.4 radio
2	U5, U8	Holtek	HT7750A	Voltage level converter
1	U6	Fairchild Semi	FXMA108	5V Boost Converter
1	U7	Atmel	ATMEGA328-QFP32	AVR Microcontroller
1	U9	FTDI	FT231XS	USB to Serial Converter
1	VR1	NJRC	NJM2391- 5.0	5.0V Regulator
1	Y1	NDK	NX3225SA, 15PPM	16 MHz crystal, 15 ppm