Zigduino

Shield compatible Arduino variant with onboard 802.15.4 radio

1. Introduction

The Zigduino is a shield-compatible Arduino variant with an onboard 802.15.4 radio that allows it to communicate wirelessly with a large range of different devices, including not only other Zigduinos and other Atmel-based devices[1] but also a wide range of popular devices using the Zigbee protocol. A number of other protocols can also be used.[2] It uses a reverse polarity SMA connector (RP-SMA) for an external antenna. This allows the user to use nearly any existing 2.4 GHz antenna with it. The Zigduino runs on 3.3V, but all I/O pins are 5V compatible. In order to withstand the electrical environments that may often be encountered in remote sensing applications, it is designed to be more electrically robust than a stock Arduino.

The Zigduino is sold as a kit for experimenters, not a complete product. As a result, it is not currently FCC or CE certified. Compliance with local radio regulations is the responsibility of the user.

Microcontroller	ATmega128RFA1
Operating Voltage	3.3V
Input Voltage	7 – 20V (recommended) 6 – 30V (maximum, can tolerate -20V/+60V transients)
Digital I/O Pins	14 + 3 auxiliary
PWM Output Pins	6
Analog Input Pins	6 (0-1.8V range)
I/O Protection	±30V transient
	-2.5V to +5.8V continuous
DC Current per I/O pin	20 mA
DC Power Supply Current	250 mA (5V), 200 mA[3] (3.3V)
Flash Memory	128 kB (2 kB are used by the bootloader)

[1] There are a number of devices on the market now that use either the ATmega128RFA1 that the Zigduino uses or one of Atmel's 802.15.4 transceiver chips.

[2] RF4CE is supported by Atmel. Route Under MAC & 6LoWPAN are supported by third-party libraries.

[3] The 3.3V regulator draws its current from the 5V regulator. This must be kept in mind if both are being used.

SRAM	16 kB
EEPROM	4 kB
Clock Speed	16 MHz
Current Draw	30 mA (transmitting, USB active, no I/O)
	15 mA (transmitting, no USB, no I/O)
	6 mA (radio off, no USB, no I/O)
	250 μA[4] (sleep)

2. Quick Start

2.1. Kit Contents

Your kit should contain the following items:

Quantity	Item
1	Zigduino PCB
2	1x8 female pin headers (digital pins)
2	1x6 female pin headers (power & analog pins)
1	1x3 female pin header (SPI connector)
1	1x2 female pin header (I^2C/Wire connector)
1	2x3 male pin header (ICSP connector)
1	2.1mm barrel jack (DC power connector)
1	RPSMA PCB edge connector

2.2. Assembling the Board

The Zigduino is shipped without its through-hole connectors or the RP-SMA connector soldered on. None of the three solder jumpers used to configure the board are completed either. In order to use all of the functions of the board, you need to solder on the connectors and select the functions of the pins controlled by the solder jumpers. You will need a soldering iron, a supply of flux, and a supply of flux-core solder intended for electronics use.

[4] This is almost entirely due to the quiescent draw of the regulators.

It is easiest to start by soldering on the RF connector. This connector needs to be soldered on to the edge of the board. If you hold the board such that the USB connector is on the left, the RF connector pads are on the right hand side of the board, aligned with the lower edge of the main processor. There are three pads on the front side of the board and two on the back. The connector should slip easily over the edge of the board, with three fingers on top and two on the bottom. Follow these steps to install it:

1. Apply a generous coat of flux to the pads on both sides of the PCB and the connector fingers.
2. Slide the connector into place and align it with its pads.
3. Heat one of the connector fingers and its adjoining pad until the solder flows over both. Add enough solder to make a complete joint.
4. Repeat step 3 for the rest of the connector fingers, making sure to get all five fingers.
5. Inspect the connector to make sure there are no shorts between the center finger on the top side and the ground pads or other fingers. Use a continuity meter to confirm that there are no shorts.

Next, solder the standard Arduino headers to the board. A shield is often useful for lining up the headers and keeping them straight while you solder them in. If you wish to install the I2C, ICSP, or SPI headers, insert them into the board and solder them down. Turning the board and laying it on the bench such that the bench holds the headers in to the board makes it easier to solder these connectors on. Lastly, solder on the power connector. You'll need to hold it in with a finger or clamp while you solder on the first pin.

2.3. Configuring the Board

There are three solder jumpers on the back of the board. These jumpers are provided to maintain pin compatibility with all existing shields. They select the connections to digital pin 11 and to analog pins 4 and 5. Neither option is selected on the board as shipped. In order to make one of the two connections, follow these steps:

1. Apply a generous coating of flux over the jumper.
2. Lay your soldering iron over the two contacts you wish to join.
3. Feed solder into the joint until you have a good puddle of solder wetted to both contacts.
4. Slowly and carefully slide your iron out of the puddle sideways, leaving a solder jumper across two contacts.

Digital pin 11 may be selected to be either an analog/PWM output or the MOSI line of the SPI bus. When configured as an SPI pin, it is referenced as pin 20. MOSI is always available on the SPI connector.

Analog pin 4 may be selected to be either an analog input or the SDA line of the I^{2}C (Wire) bus. When configured as SDA, it is referenced as pin 21. The SDA line is always available in the I^{2}C connector.

Analog pin 5 may be selected to be either an analog input or the SCL line of the I^{2}C (Wire) bus. When configured as SCL, it is references as pin 22. The SCL line is always available in the I^{2}C connector.

2.4. Installing the Environment

In order to use the Zigduino with the Arduino environment, you must perform the following steps:

1. Upgrade your avr-gcc to version 4.3.3 or later and avr-libc to 1.6.7 or later. On Windows, you can do this by unzipping WinAVR-201001011 or later over the contents of your arduino-*/hardware/tools/avr directory. On OS X, upgrade to the most recent version of Crosspack. On Linux, update your installation of avr-gcc and avr-libc.
2. Copy the zigduino directory into your arduino-*/hardware/arduino/cores directory.
3. Append the contents of zigduino-board.txt to your arduino-*/hardware/arduino/boards.txt file.

After you have completed these steps, you should have a ‘Zigduino’ entry in your Tools > Boards pull down. Switch to this option to compile your sketches for the Zigduino.

2.5. Fetching & Installing RF Libraries

There are currently two 802.15.4 libraries supported by Atmel. Atmel requires registration for downloads of all of their libraries. Their EULA is not horribly onerous, but it does require that you pay them royalties if you use libraries in a commercial product.

2.5.1. IEEE 802.15.4 MAC

This is the only source-available library provided by Atmel, and therefore is the only one that can be compiled directly from the Arduino environment.[5] It is designed to abstract the hardware of the different supported Atmel platforms.

In order to install the MAC library, first download it from Atmel.[6] Unpack the zip file in to your arduino-*/libraries directory. In order to compile from the Arduino environment, you must copy all of the .c and .h files from the various subdirectories up to the main directory. Make sure you copy the specifically Zigduino versions for those header (*.h) files that have multiple versions.

2.5.2. BitCloud – ZigBee Pro

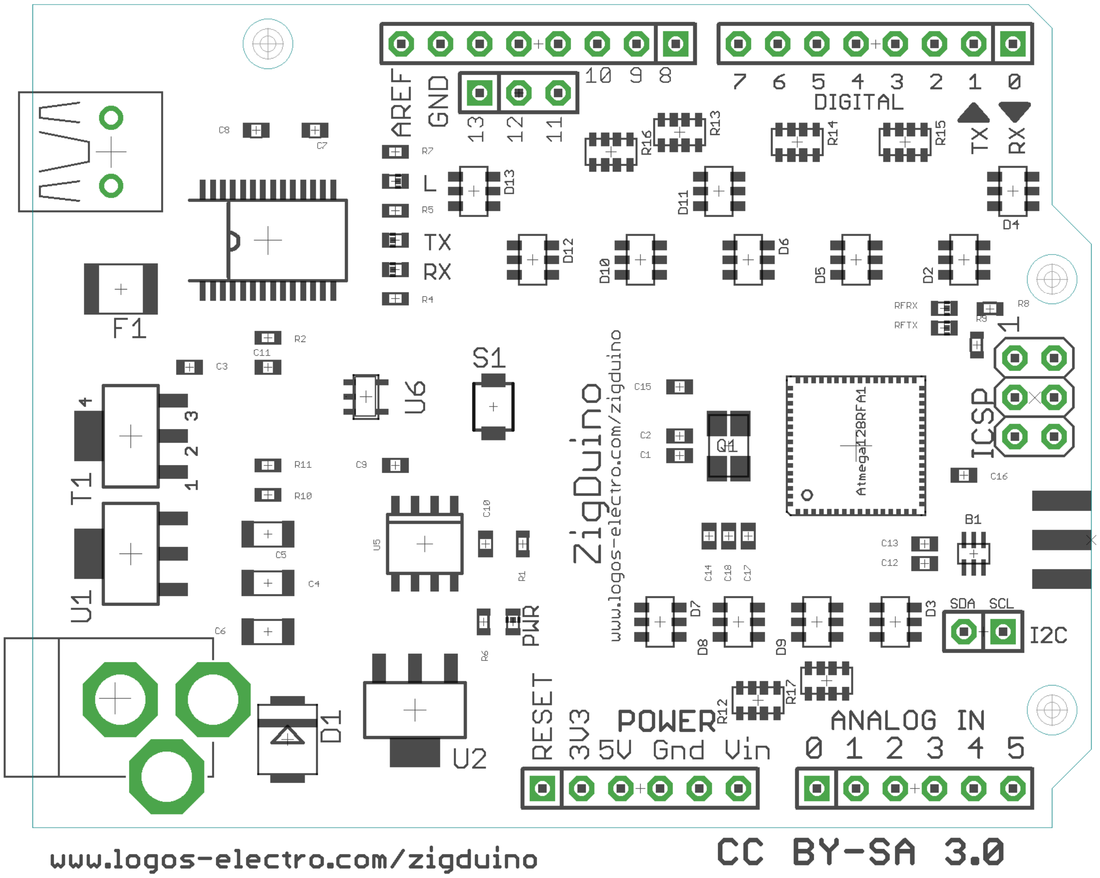
This library allows the Zigduino to interoperate with other ZigBee devices and networks. It is available only in a pre-compiled form. Therefore, you will need to use WinAVR, AVR Studio, or some related compiler in order to use it. Download the version for the megaRF zip file from Atmel,[7] unpack it, and follow the instructions in the quick start guide.

[5] The Arduino environment, as of arduino-0022, does not support pre-compiled libraries. It also does not correctly support libraries arranged in multiple subdirectories.

[6] http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4675&category_id=163&family_id=676&subfamily_id=2124

[7] http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4495&category_id=163&family_id=676&subfamily_id=2124

3. Hardware Overview



3.1. Board Layout

The layout of the Zigduino follows that of the stock Arduino, particularly with respect to the location of the standard headers. This is to provide maximum compatibility with the existing array of shields on the market. The only significant difference among these connectors is that the Zigduino uses a

There are three additional connectors as well:

- **SPI** – This connector carries the three pins of the SPI bus, **MOSI**, **MISO**, and **SCK**. While the **MOSI** pin on the ordinary digital connector (pin 11) may be configured to be a **PWM** output instead, the one on the SPI connector is always **MOSI**. See the Configuration section for more information.
- **RF** – This is the antenna connector for the integrated 802.15.4 radio transceiver.

- **I^{2}C** – This connector carries the two pins of the I^{2}C bus (accessed through the Wire library), **SDA** and **SCL**. Analog pins 4 and 5 may also be configured to carry these pins as well, but this connector always has them. See the Configuration section for more information.

3.2. Digital Pins

Each of the 14 digital pins of the Zigduino can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead(). Each pin operates at 3.3V and can source or sink 10 mA. Each also has an internal pullup, which is disabled by default. Each pin is protected against ±30V transients and can tolerate continuous 5V input.

Care must be taken not to exceed the power ratings of the 100 ohm protection resistors, which is 62.5 mW per element. This limits the continuous voltage that the pins can tolerate to between -2.5V and +5.8V.

The digital pins include the following additional functions:

- **Serial: 0 (RX) and 1 (TX)** -- Used to transmit and receive TTL serial data. These pins are connected to the corresponding pins on the FTDI USB interface chip.
- **PWM: 3, 5, 6, 9, 10, and 11** -- Provides 8-bit PWM output with the analogWrite() function. Pin 11 must be selected for PWM operation with the solder jumper on the back of the board.
- **SPI: 11 (MOSI), 12 (MISO), 13 (SCK)** -- These pins support SPI communications using the SPI library. Pin 11 must be selected for SPI operation with the solder jumper on the back, or SPI must be accessed with the SPI connector.
- **LED: 13** -- This is the built-in LED on digital pin 13. When the pin is high, the LED is on.
- **External Interrupts: 2, 3, 6, and 7** -- These pins can be configured to trigger and interrupt on a low value, high value, or an edge. See the attachInterrupt() function for details. The two I^{2}C pins can also be used as interrupts.

3.3. Analog Pins

The six analog input pins, labeled A0 - A5, are likewise protected against ±30V spikes and can tolerate continuous 5V input. Each provides 10 bits of resolution and measures 0 - 1.8V. It is possible to change to a lower top voltage through use of the AREF pin and the analogReference() function.

There are two jumper-selectable special function pins in the analog section:

- **I^{2}C: A4 (SDA) and A5 (SCL)** -- These pins support I^{2}C communications using the Wire library. They must be selected for I^{2}C operation with the jumpers on the back or I^{2}C must be accessed through the I^{2}C connector. They can also be configured as interrupts.

3.4. Power & Ground

The Zigduino can be powered through the USB connection or with an external power supply. The power source with the highest voltage is selected automatically.

External power can be supplied via a wall wart or a battery. It can be connected with a 2.1mm center-positive plug inserted into the power jack. Alternately, external power can be connected through the GND and VIN pins of the POWER header.

The board will operate correctly on an input voltage between 6V and 30V. It will survive transients as large as -20V or +60V. However, higher supply voltages may cause excessive heat dissipation at higher current draws. The input voltage regulator has integral overtemperature protection, so you can't permanently damage the board this way. However, the board may not work correctly under these circumstances.

The power pins are as follows:

- **VIN** -- The input voltage to the Arduino board when it is running from external power, i.e. not USB bus power.
- **5V** -- The regulated 5V used to power 5V components on the board and external 5V shields. It comes either from the USB or from the VIN via the 5V regulator. Maximum current draw is 250 mA.
- **3V3** -- The regulated 3.3V supply that powers the microcontroller. It is derived from the 5V bus via a second regulator. Maximum current draw is 200 mA.
- **GND** -- Ground pins.

3.5. Wireless

There are two distinct groups of wireless-related components on the board. The first group, consisting of the blocking capacitors, the balun, and the antenna connector, is located just below the ICSP connector.

The second group is the wireless link lights and their resistors, located above and to the left of the ICSP connector. There are two of them, **RFRX** and **RFTX**. These two LEDs are controlled by digital pins 23 and 24, respectively. The Atmel MAC configuration files we provide map these LEDs appropriately.

3.6. ICSP

The ICSP programming connector is located in the standard place for an Arduino board. This provides compatibility with certain shields that use it to access the SPI bus.

3.7. Configuration

There are three solder jumpers on the back of the board. These jumpers are provided to maintain pin compatibility with all existing shields. They select the connections to digital pin 11 and to analog pins 4 and 5. All jumpers are open as shipped. Each is located as close as practicable to the pin they control.

3.8. Pin Arrangement

All of the pins, and the two status LEDs, are all accessible as digital pins. Here is how they are arranged:

Pin Name	Digital Equivalent	Alternate	Function
0	0		Serial receive
1	1		Serial transmit
2	2		Interrupt 0
3	3		PWM, Interrupt 1
4	4		PWM
5	5		
6	6		PWM, Interrupt 2
7	7		Interrupt 3
8	8		PWM
9	9		PWM
10	10		PWM
11	11	SPI MOSI, defaults to not connected	PWM, SPI MOSI, or not connected, depending on jumper configuration.
12	12		SPI MISO

Pin Name	Digital Equivalent	Alternate	Function
13	13		SPI SCK
A0	14		Analog Input
A1	15		Analog Input
A2	16		Analog Input
A3	17		Analog Input
A4	18	I^2C SDA, defaults to not connected.	Analog input, I^2C SDA/Interrupt 5, or not connected, depending on jumper configuration
A5	19	I^2C SCL, defaults to not connected.	Analog input, I^2C SCL/Interrupt 6, or not connected, depending on jumper configuration
SCK	12		SPI MISO
MISO	13		SPI SCK
MOSI	20		SPI MOSI
SDA	21		I^2C SDA, Interrupt 5
SCL	22		I^2C SCL, Interrupt 6
RFTX	23		Transmit status light for the radio
RFRX	24		Receive status light for the radio

4. Software Overview

4.1. The Arduino Environment

The current Zigduino environment is based on the arduino-0018. It will be updated to Arduino 1.0 shortly after that baseline is released. In order to use the Zigduino with the standard Arduino environment, you must upgrade to a version of avr-gcc and avr-libc that support the Atmega128RFA1 chip. You must also add the Zigduino to your boards.txt file and the zigduino/ core files directory to your hardware/cores/ directory. Here are the instructions for Windows:

- Back up your arduino installation
- Download the zipball from http://www.github.com/logos-electromechanical/Zigduino/master/zipball.
- Unzip the archive in your arduino-00**/hardware directory.
- Restart the Arduino IDE. There will be a new entry for the Zigduino in the Tools>Boards drop-down.

For non-Windows platforms, don't unpack the tools/avr directory of the archive.[8] You will need to install avr-gcc-4.3.2 or later and avr-libc-1.6.7 or later in order to compile for the Atmega128RFA1. This should not negatively impact compilation for other chips.

4.2. Arduino Libraries

The Arduino libraries are based on the core libraries, and therefore most of them should work correctly. However, they have not been completely tested with the Zigduino and therefore are not guaranteed to work at this time.

4.3. Atmel IEEE 802.15.4 MAC

The Atmel IEEE 802.15.4 MAC provides a hardware abstraction layer and the link layer[9] for connections for user applications and the more sophisticated libraries offered by Atmel. It can also be used for direct point to point networking with little overhead. Follow these steps to install it:

- Download the IEEE 802.15.4 MAC library from Atmel (http://atmel.com/dyn/products/tools_card.asp?tool_id=4675) and unpack it to a convenient location.
- Copy the contents of the MAC_v_2_4_2 directory over the MAC installation directory. This puts the board definition files in the correct place in the tree.
- Copy all header and source files (*.h & *.c) files into the top level directory. In the cases of architecture headers, make sure you copy the Zigduino versions.
- Read the user guide in MAC_v_2_4_2/Doc/User_Guide/AVR2025_User_Guide.pdf

4.4. BitCloud

Atmel BitCloud is a full-featured ZigBee PRO stack. This software is completely compatible with the ZigBee PRO platform. It has both an easy to use C API and accepts serial AT commands. It is capable of supporting large mesh networks. You must register with Atmel in order to download it, and it is not open source, but it will give you full compatibility with other ZigBee devices, including XBee modules. In order to install the software, follow these steps:

- Register and Download the software from Atmel at http://www.atmel.com/dyn/resources/prod_documents/BitCloud_MEGARF_1_11_0.zip

[8] This contains the latest version of WinAVR, which installs avr-gcc and avr-libc under Windows.
[9] OSI Layer 2

- Unpack the contents in a convenient location
- Read the user guide in

5. Zigduino Usage

5.1. Powering the Zigduino

There are three ways to power the Zigduino – through the DC power jack, through the USB port, or through the VIN/GND pins. The DC power jack accepts a 2.1mm barrel connector with the outer conductor grounded and the center pin attached to power. The board automatically draws its power from the highest voltage available source.

5.2. Digital Pins

The digital pins on the Zigduino are set up to act as similarly to those on the stock Zigduino. Since the Atmega128RFA1 is a 3.3V part, these pins cannot supply the same voltage or current as the Atmega328 of the stock 5V Duemilanove or the Uno. Each digital pin is protected by a 100Ω resistor and a pair of diodes wired in series between the pin and the power rail. This limits the maximum output current to 33 mA or less. However, the pin can absorb transient spikes are large as ±30V, or between -2.5V and +5.8V continuous.

While most modern 5V logic will correctly interpret a 3.3V voltage as a logical true, this is NOT true of all logic, in particular CMOS of the C, AC, HC, and AHC series.[10] If this is a problem, it may manifest in unpredictable and strange ways, because the response to a 3.3V input may be undefined. When in doubt, check the datasheets of the parts used in the particular shield in question.

5.3. Analog Pins

The analog pins of the Zigduino may be used as digital pins, and have the same protection described above. However, in analog mode, they can only measure voltages from 0-1.8V, or a little more than a third of the range measured by more conventional 3.3V Arduino boards. This is a hardware limitation and cannot be modified by supplying a higher voltage to the AREF pin.

[10] See http://www.interfacebus.com/voltage_threshold.html for a chart of the voltage thresholds of different types of logic, or the datasheets of the parts in question.

Usage of the A4 and A5 analog inputs requires configuration of the solder jumpers on the back of the board, as discussed above.

5.4. Wired Communications

Wired communications on the Zigduino work much like they do on the standard Arduinos.

5.4.1. USB/Serial

The Zigduino has a single UART, which is connected to digital pins 0 and 1. These pins are connected to the corresponding pins on the FTDI serial interface. In order to use them, you use the Serial library, described in the Arduino reference library at http://arduino.cc/en/Reference/Serial.

5.4.2. SPI

The SPI bus is brought out three different places:

1. Through the ISP programming connector, as is typical for all Arduinos.
2. Through the dedicated SPI connector set alongside the left hand bank of digital pins.
3. Through digital pins 11, 12, and 13, if the jumpers are set correctly.

Note that the hardware CS pin is not brought out in the ZIgduino. This should only require a small amount of extra housekeeping, since a CS line must be chosen and operated from user code. This is expected to be rolled into future versions of the SPI library, and will not require further user attention. In order to use the SPI library, see the documentation for the SPI library at http://arduino.cc/en/Reference/SPI

5.4.3. I^{2}C (Wire)

The I^{2}C bus, also known as Two Wire (TWI) or simply Wire, is a popular interface for networking Arduinos and a variety of commercial sensors and other peripheral chips together. It is located on the permanent I^{2}C connector, and can be taken through the A4 and A5 pins (as it is on other non-Mega Arduinos) by making the two jumpers on the back of the board, under the I^{2}C connector. Wherever it is brought through, it can be accessed with the Arduino Wire library, http://arduino.cc/en/Reference/Wire

5.5. Wireless

5.5.1. Hardware

The wireless hardware on the Zigduino is integrated into the core microcontroller, except for two blocking capacitors, a transformer/balun, and the antenna connector. In order to use the wireless, you must solder the antenna connector to the board and attach a suitable antenna, such as the one provided with the kit. Failure to do so may result in damage to the wireless hardware.

5.5.2. Software

You will need to compile your code with one of the available stacks for the Atmega128RFA1, as described above. I recommend the excellent user guides that come with Atmel's 802.15.4 MAC and BitCloud.

6. References