



GoDroid

Application Notes

Arduino Support and Configuration

v1.0



Copyright Statement

Copyright in this document is owned by GoWarrior Community. Any person is hereby authorised to view, copy, print and distribute this document subject to the following conditions:

- The document may be used for informational purposes only
- The document may be used for non-commercial purposes only
- Any copy of this document or portion thereof must include this copyright notice

This document is provided "as is" without any warranty of any kind, either express or implied, statutory or otherwise; without limiting the foregoing, the warranties of satisfactory quality, fitness for a particular purpose or non-infringement are expressly excluded and under no circumstances will GoWarrior Community be liable for direct or indirect loss or damage of any kind, including loss of profit, revenue, goodwill or anticipated savings.

This document is intended only to assist the reader in the use of the product. GoWarrior Community makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, which is used at your own risk and should not be relied upon. The information could include technical inaccuracies or typographical errors. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property right is granted by this document.

The product described in this document is subject to continuous development and improvements. GoWarrior Community also reserves the right to make changes to the specifications and product description at any time without notice.

Third-party brands and names mentioned in this publication are for identification purpose only and may be the property of their respective owners.

Android™ is a registered trademark of Google Inc. Linux® is a registered trademark of Linus Torvalds. Microsoft® and Windows® are registered trademarks of Microsoft Corporation. Supply of this product does not convey a license nor imply a right under any patent, or any other industrial or intellectual property right of Google Inc., Linus Torvalds, and Microsoft Corporation to use this implementation in any finished end-user or ready-to-use final product. It is hereby notified that a license for such use is required from Google Inc., Linus Torvalds and Microsoft Corporation.

For the latest version of this document refer to:

www.gowarriorosh.com

Copyright © 2015 GoWarrior Community All Rights Reserved.

Table of Contents

Preface	1
Overview	1
Audience	1
Applicable Products.....	1
Reference Documents.....	1
Conventions.....	2
How to Contact Us.....	3
 1 Introduction	 4
 2 Setting Up PyFirmata	 6
2.1 Solution.....	6
2.2 Discussion	6
 3 Writing Digital Outputs	 9
3.1 Solution.....	9
3.2 Conclusion	10
 4 Using PyFirmata with TTL Serial.....	 11
4.1 Solution.....	11
4.2 Conclusion	12
 5 Reading Arduino Digital Inputs.....	 13
5.1 Solution.....	13
5.2 Discussion	14
 Revision History	 15
Document Change History.....	15

Software Change History.....	15
------------------------------	----

List of Tables

Table 1. Typographical Conventions.....	2
Table 2. Symbol Conventions	3
Table 3. Document Change History	15
Table 4. Software Change History.....	15

List of Figures

Figure 1. An Arduino Uno Board	4
Figure 2. I/O Pins on an Arduino Uno	7
Figure 3. Wiring Diagram for Arduino and LED.....	9
Figure 4. Wiring Diagram of Using a Pair of Resistors.....	11

Preface

Overview

This manual mainly describes about Arduino support and configuration. This manual is organized into the following chapters:

- **Chapter 1: Introduction**

This chapter introduces how to connect a TIGER Board to an Arduino board

- **Chapter 2: Setting Up PyFirmata**

This chapter describes how to set up pyFirmata to control an Arduino from a TIGER Board.

- **Chapter 3: Writing Digital Outputs**

This chapter gives compact description how to write digital outputs on an Arduino from a TIGER Board.

- **Chapter 4: Using PyFirmata with TTL Serial**

This chapter provides details about how to use pyFirmata with TTL serial.

- **Chapter 5: Reading Arduino Digital Inputs**

This chapter gives insight on how to read Arduino digital input using pyFirmata.

Audience

This manual is primarily written to provide complete guidance for those who wants to exploit the GoWarrior platform, such as makers, tinkers, innovators, students, etc.

Applicable Products

This manual is applicable for the GoWarrior TIGER Board.

Reference Documents

N/A



Conventions

Typographical Conventions

Item	Format
codes, keyboard input commands, file names, equations, and math	<code>Courier New, Size 10.5</code>
Variables, code variables, and code comments	<i>Courier New, Size, Italic</i>
Menu item, buttons, tool names	Ebrima, Size 10.5, Bold e.g. Select USB Debugging
Screens, windows, dialog boxes, and tabs	Ebrima, Size 10.5, Bold Enclosed in double quotation marks e.g. Open the “ Debug Configuration ” dialog box

Table 1. Typographical Conventions

Symbol Conventions

Item	Description
 Caution	Indicates a potential hazard or unsafe practice that, if not avoided, could result in data loss, device performance degradation, or other unpredictable results.
 Note	Indicates additional and supplemental information for the main contents.


Item	Description
 <i>Tip</i>	Indicates a suggestion that may help you solve a problem or save your time.

Table 2. Symbol Conventions

How to Contact Us

Submit all your comments and error reports to the author at:

info@gowarriorosh.com

Tel: +886-2-8752-2000

Fax: +886-2-8751-1001

For questions regarding GoWarrior, contact our support team at the email listed below:

support@gowarriorosh.com

1 Introduction

Connecting the TIGER Board to an Arduino Board is a common alternative to using an interface board, as shown in Figure 1.

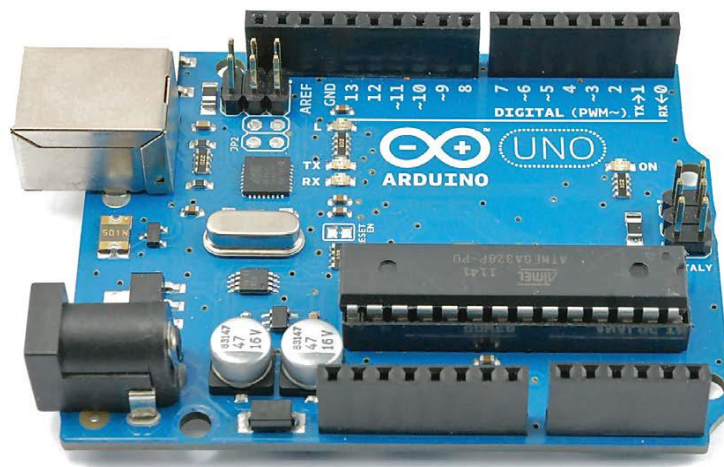


Figure 1. An Arduino Uno Board

Arduino boards are superficially similar to an external appearance of TIGER Board. However, The Arduino board is very different from the TIGER Board in a number of aspects:

- The Arduino board does not have any interface to keyboard, mouse, or screen.
- The Arduino board has just 2KB of RAM and 32KB of flash for storing programs.
- The Arduino processor runs at just 16 MHz, whereas the TIGER Board's processor comparatively runs at 1GHz.

This might leads you to wonder why you would use such an apparently feeble board rather than the TIGER Board directly.

The answer is, the Arduino Uno is the most common being of Arduino board

have slightly upper hand at interfacing to external electronics in several ways. For example, Arduino boards have:

- 14 digital inputs/outputs, like the TIGER Board's GPIO pins, but each pin can provide up to 40 mA. This enables them to power more devices without the need for extra electronics.
- 6 analog inputs. This makes connecting analog sensors much easier.
- 6 PWM outputs. These outputs are hardware-timed and produce more accurate PWM signal for controlling servo motors.
- A vast range of plug-in shields for everything from motor control to LCD displays of various kinds.

In many ways, using a TIGER Board with an Arduino to handle all the low-level stuff is a good combination, playing to the strengths of both boards. This is taken to its extreme with interface boards that have a GPIO connector and also include Arduino-compatible hardware such as the aLaMode.

2 Setting Up PyFirmata

This chapter describes how to set up pyFirmata to control an Arduino from a TIGER Board.

2.1 Solution

First, connect the Arduino to a USB socket of the TIGER Board so that the computer can communicate and send power to the Arduino.

For next step, install the Firmata sketch onto the Arduino. Once Firmata is installed, the Arduino waits for communication from the TIGER Board. And the PyFirmata corresponds with TIGER Board, which is pre-installed.

You can try out the PyFirmata library from the Python console. Enter the following commands to turn on the built-in LED on Arduino pin 13 (marked with an L) and then turn it off again.

```
$ sudo python
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.

>>> import pyfirmata
>>> board = pyfirmata.Arduino('/dev/ttyUSB0')
>>> pin13 = board.get_pin('d:13:o')
>>> pin13.write(1)
>>> pin13.write(0)
>>> board.exit()
```

2.2 Discussion

The preceding code first imports the PyFirmata library and then makes an instance of Arduino called board, using the USB interface (/dev/ttyUSB0) as

its parameter. We can then gain a reference to one of the Arduino pins (in this case, 13) and set it to be a digital output. The "d" is for digital, 13 is the pin number, and "o" is for output.

To set the output pin high, you just have to write (1) and to set it low just write (0). You can also use True and False in place of 1 and 0.

The Figure 2 shows an Arduino with the rows of connections down both sides of the board.

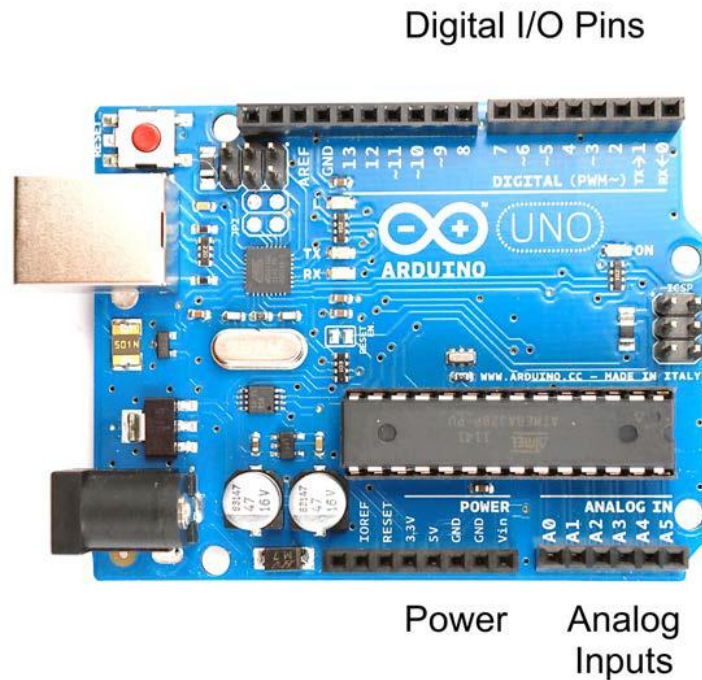


Figure 2. I/O Pins on an Arduino Uno

The pins at the top of Figure 14-4 marked 0 to 13 can be used as digital inputs or outputs. Some of these pins are also used for other things. Pins 0 and 1 are used as a serial interface, and are in use when the USB port is being used, and pin 13 is attached to the on-board LED marked with an L. The digital I/O pins 3, 5, 6, 9, 10, and 11 have a ~ symbol next to them that indicates that they can be used for PWM output (see the Figure 2). On the other side of the board, there is one set of connectors that supply power at 5V and 3.3V and six analog inputs marked A0 to A5.

An Arduino Uno on its own uses about 50 mA. Whereas the TIGER Board is probably using about 10 times more than that, which makes it perfectly possible to power the Arduino from the USB connection of the TIGER Board. However, if you start attaching a lot of external electronics to the Arduino, and the current consumption increases, then you may want to power the

Arduino from its own power adapter by using the DC barrel socket. This will accept 7V to 12V DC.

The only real downside of using Firmata is that because all instructions have to come from the TIGER Board, it doesn't make much use of Arduino's ability to run independently. For advanced projects, you will probably end up writing your own Arduino code that receives instructions from the TIGER Board and/or sends messages to the TIGER Board, while it gets on with other tasks.

3 Writing Digital Outputs

This chapter describes how to write digital outputs on an Arduino from the TIGER Board.

3.1 Solution

In chapter 2, the built-in LED (labeled with an L) has enabled to flash on the Arduino board. Here, you will attach an external LED and write a short Python program to make the LED blink.

For this process, the following components are required:

- Arduino Uno
- Breadboard and jumper wires
- 270 Ω resistor
- LED

Connect the breadboard, holding the components to the Arduino as shown in Figure 3.

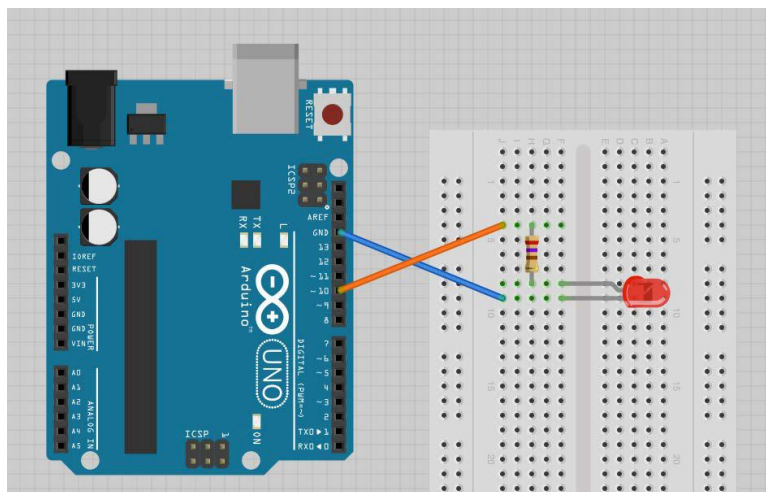


Figure 3. Wiring Diagram for Arduino and LED

The following Python script makes the LED blink at the rate of about 1Hz. Open an editor (Nano or IDLE) and paste in the following code.

```
import pyfirmata
import time
board = pyfirmata.Arduino('/dev/ttyUSB0')
led_pin = board.get_pin('d:10:o')
while True:
    led_pin.write(1)
    time.sleep(0.5)
    led_pin.write(0)
    time.sleep(0.5)
```

3.2 Conclusion

This is very similar to connecting the LED to the TIGER Board. However, note that since Arduino outputs can supply a lot more current than TIGER Board outputs, you can use a low value resistor and make the LED a bit brighter. Arduino outputs are also 5V rather than 3.3V.

4 Using PyFirmata with TTL Serial

This chapter describes how to use pyFirmata with TTL serial.

4.1 Solution

You want to use PyFirmata, but over the serial connection (RXD and TXD on the GPIO connector) rather than by USB. Using the pair of resistors, then connect the breadboard as shown in Figure 4.

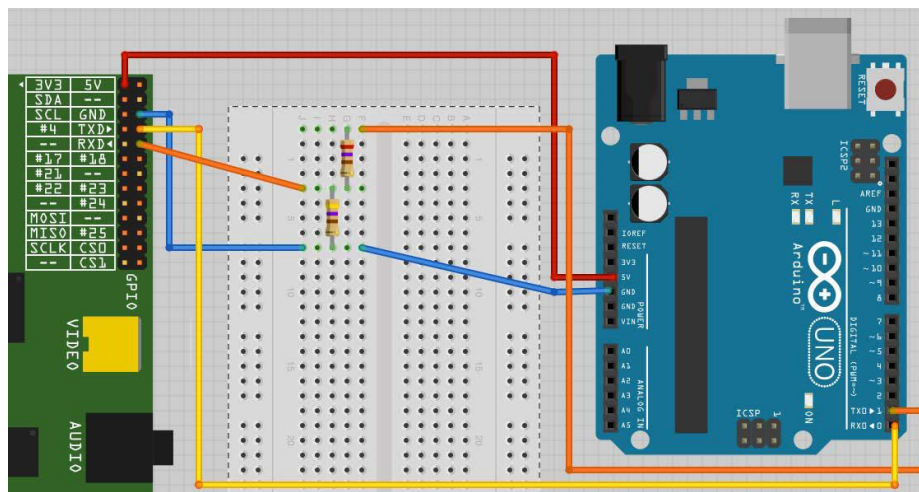


Figure 4. Wiring Diagram of Using a Pair of Resistors

The Arduino Rx input is fine with just 3.3V from the TIGER Board TXD pin; however, the 5V coming from the Arduino Tx pin must be dropped to the 3V expected by the TIGER Board.

You will need to set up PyFirmata—see Recipe 14.3. The Arduino side of the project remains exactly the same as Recipe 14.4, where USB is used instead of the serial connection. There is only one change that you need to make to the Python program running on the TIGER Board—change the device name from `/dev/ttyUSB0` to `/dev/ttyS0`, as the serial port has a different device name from the USB interface.

The following Python script makes the LED blink at a rate of about 1 Hz. Open an editor (vi) and paste in the following code.

```
import pyfirmata
import time
board = pyfirmata.Arduino('/dev/ttyS0')
led_pin = board.get_pin('d:13:o')

while True:
    led_pin.write(1)
    time.sleep(0.5)
    led_pin.write(0)
    time.sleep(0.5)
```

4.2 Conclusion

While it is fine for the 5V Arduino to use a 3V signal, where as its reverse order is not applicable. A 5V signal connected to the 3V RXD pin will likely damage the TIGER Board.

5 Reading Arduino Digital Inputs

This chapter describes how to read Arduino digital input using pyFirmata.

5.1 Solution

Use PyFirmata to read a digital input on the Arduino.

For this process, the following components are required:

- Arduino Uno
- Breadboard and jumper wires
- 1k Ω resistor
- Tactile push switch

Connect the breadboard, holding the components to the Arduino as alamode. The following Python script prints out a message every time the switch is pressed. Open an editor (vi) and paste in the following code.

```
import pyfirmata
import time
board = pyfirmata.Arduino('/dev/ttyUSB0')
switch_pin = board.get_pin('d:4:i')
it = pyfirmata.util.Iterator(board)
it.start()
switch_pin.enable_reporting()
while True:
    input_state = switch_pin.read()
    if input_state == False:
        print('Button Pressed')
    time.sleep(0.2)
```

When you run it, nothing will happen for one or two seconds, while the Firmata sketch starts and establishes communication with the TIGER Board.

But, once it starts up, each time you press the button, a message will appear.

5.2 Discussion

PyFirmata uses the concept of an Iterator to monitor the Arduino input pin. The reasons for this are bound up in the implementation of Firmata. This means that you can't simply read the value of an Arduino input pin on demand; instead, you have to create a separate Iterator thread that manages the reading of the switch using the commands:

```
it = pyfirmata.util.Iterator(board)

it.start()
```

You then also have to enable reporting for the pin you are interested in using the command:

```
switch_pin.enable_reporting()
```

A side effect of this mechanism is that when you press Ctrl-C to exit the program, it won't exit properly. There is no nice way to kill the Iterator thread other than to open another Terminal window or SSH session and kill the process. If the only Python process running is this program, you can kill it with the command:

```
sudo killall python
```

Simply disconnecting the Arduino from the TIGER Board, breaking the communication link, will also cause the Python program to exit.

Revision History

Document Change History

Revision	Changes	Date
v1.0	Initial Release	September 07, 2015

Table 3. Document Change History

Software Change History

Revision	Changes	Date
v1.0	Initial Release	September 07, 2015

Table 4. Software Change History



www.gowarriorosh.com

Headquarters

Tel: +886-2-8752-2000

Fax: +886-2-8751-1001

