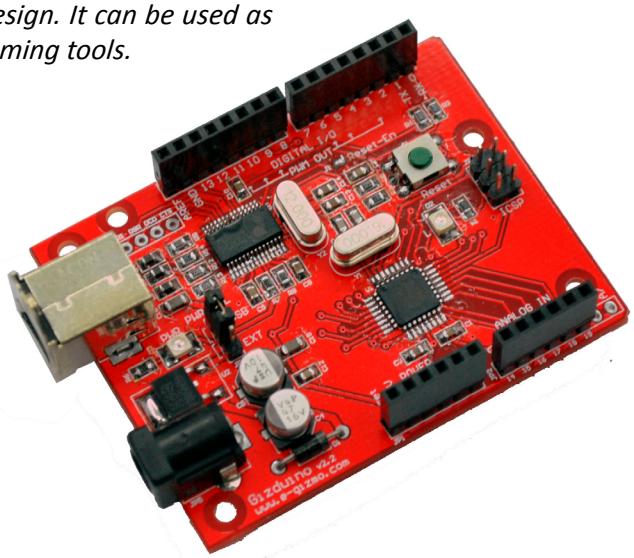
Gizduino: Arduino Compatible Kit

(Atmega168 and Atmega328)

Hardware Manual Rev 1r0

e-Gizmo Learn to use and program microcontroller the fast and easy way. e-Gizmo's Gizduino platform kit is a single board AVR microcontroller platform based on highly popular open source Arduino design. It can be used as well with AVR's tradional programming tools.



The Gizduino is a microcontroller board based on the ATmega328 and ATmega168. It has 14 digital input/output pins, 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

it is an open source computing platform based on a simple input/output (I/O) board and the use of standard programming language; in otherwords, it is a tool for implementing a program you have designed. Gizduino is programmed using the IDE (Integrated Development Environment).

Gizduino is ideal for beginner programmers and hobbyists because of its simplicity compared to other platforms. It is a multiplatform environment; it can run on Windows, Macintosh, and Linux. It is programmable via USB cable, which makes it more accessible and allows communication with the computer.

Features & Specifications

- Microcontroller: ATmega168 or ATmega328
- User Interface: USB Port, DC Jack, Reset Button, ICSP Port, Shield Connection Port
- Debbuger Port: ICSP
- Power Input: External:8V-12V USB:5V
- DC Power Output: 3.3V
- PCB Size: 2.7 x 2.1 inch
- On-board Crystal: 16MHz and 12MHz

MAJOR COMPONENTS

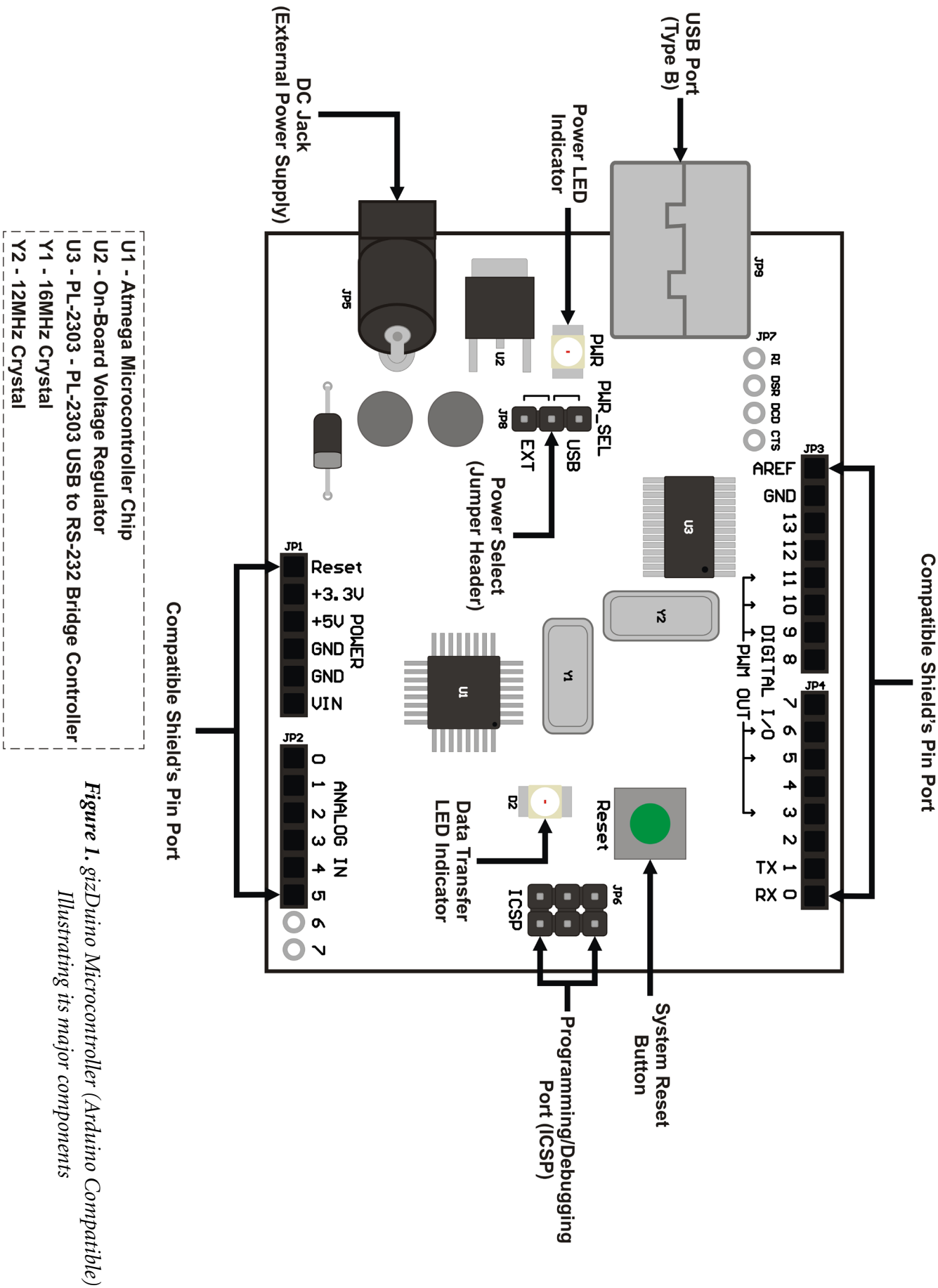


Figure 1. gizDuino Microcontroller (Arduino Compatible) Illustrating its major components

Table 1. *JP3,JP4 (Digital I/O) pin assignments:*

No.	I.D.	Description
1	AREF	analog reference pin for the A/D Converter.
2	GND	ground.
3	13	Digital I/O
4	12	Digital I/O
5	11	PWM OUT
6	10	PWM OUT / Digital I/O
7	9	PWM OUT / Digital I/O
8	8	Digital I/O
9	7	Digital I/O
10	6	PWM OUT / Digital I/O
11	5	PWM OUT / Digital I/O
12	4	Digital I/O
13	3	PWM OUT / Digital I/O
14	2	Digital I/O
15	1	TX / Digital I/O
16	0	RX / Digital I/O

Table 2. *JP2 (Analog I/O) pin assignments:*

No.	I.D.	Description
1	A0	Analog I/O
2	A1	Analog I/O
3	A2	Analog I/O
4	A3	Analog I/O
5	A4	Analog I/O
6	A5	Analog I/O
7	A6	Analog I/O
8	A7	Analog I/O

Table 3. *JP1 (Power) pin assignments:*

No.	I.D.	Description
1	Reset	reset.
2	+3.3V	3.3V Device Power Supply
3	+5V	5V Device Power Supply
4	GND	ground.
5	GND	ground.
6	VIN	8-12V Device Power Supply

Table 4. *Available Gizduino Microcontrollers Details:*

Device	Flash Mem.	EEPROM	RAM	Interrupt Vector Size
ATmega168PA	16K Bytes	512 Bytes	1K Bytes	2 instruction words/vector
ATmega328P	32K Bytes	1K Bytes	2K Bytes	2 instruction words/vector

its a choice of ATmega168 or ATmega328 Microcontroller

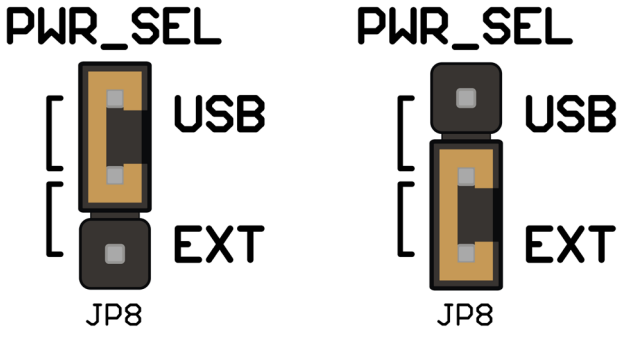


Figure 2. *illustration of PWR_SEL Jumper Header, 2 types of Power Selection*

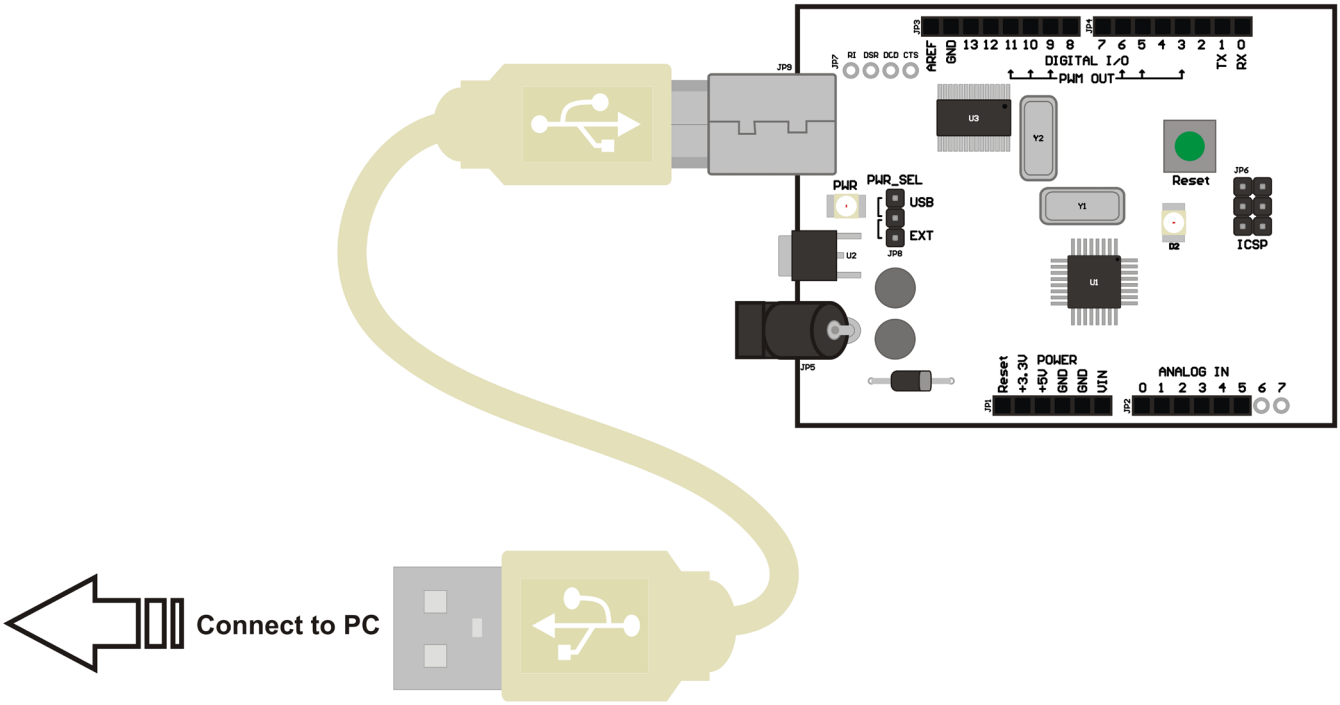
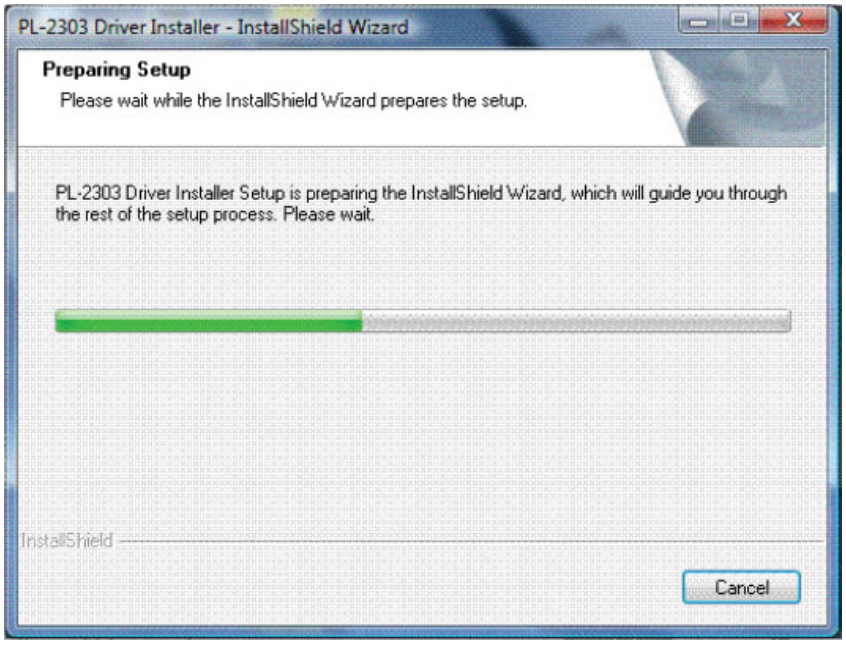


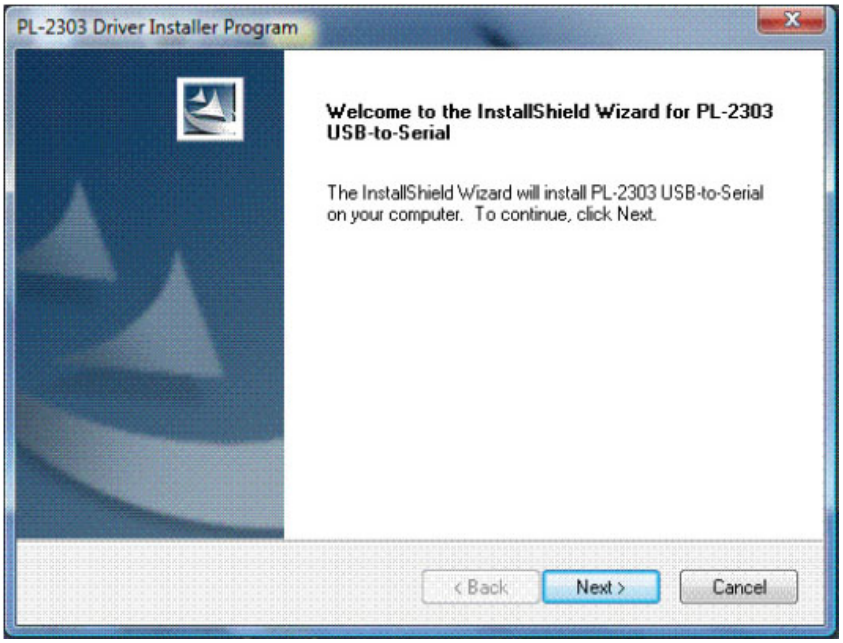
Figure 3. *illustration on how to connect the gizDuino to PC.*

To begin, attach first the USB cable to the JP9 of the Gizduino. At this point, it would not be advisable to connect the USB to the computer immediately. Before doing so, the user must acquire the software that handles the Gizduino. Because it is a clone of the Arduino, the required software is obtainable from the Arduino website (http://www.arduino.cc/en/Main/Software) and is called the Arduino IDE. At the time of the writing of this manual the latest version of the Arduino IDE is 0021 (whose file size is around 85.1 MBs). When the fileis successfully downloaded the user will notice that it takes the form of a ZIP file. As such, one needs to extract or uncompress it first. There are many ZIP file programs found over the internet if the computer is not able to recognize the ZIP file, as a suggestion to this is one of the good ZIP file programs WinRAR. Once extracted, we strictly recommend that the user does not make any changes to the internal structures and directories of the files within the extracted Arduino IDE folder (such as file names). This is because the program is written to find whatever data it requires according to exactly how the files are initially named and arranged. Changes made to these files and directories would cause confusion with the running program and will lead to errors.

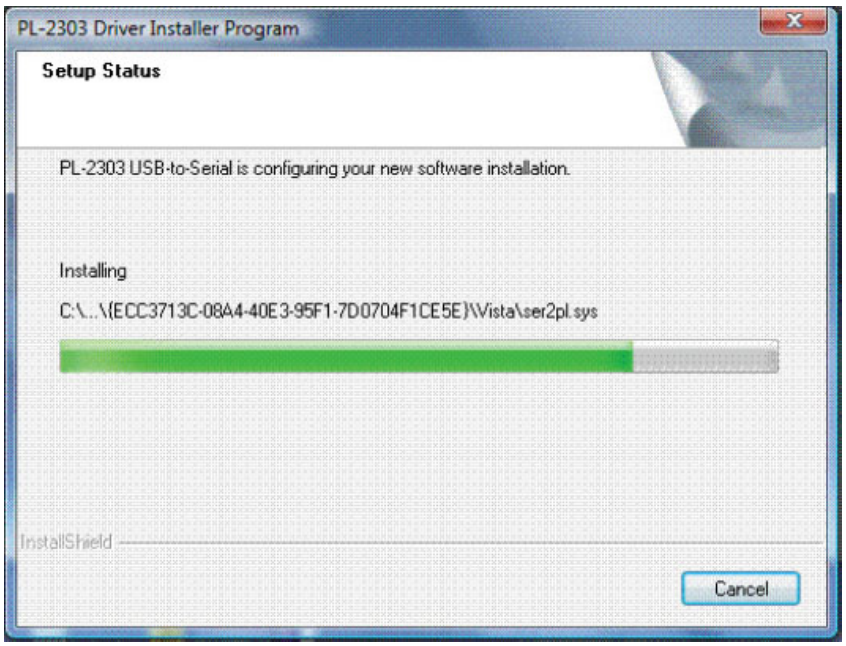
At this point, the user may now connect the free-end of the USB cable to one of the USB ports of the computer. Most computers will normally not be able to recognize the newly created Gizduino-to-PC USB connection initially. This is caused simply by the lack of the proper driver that handles the recognition of the hardware by the computer. The driver required by the Gizduino-to-PC connection is the PL-2303 USB-to-Serial driver (downloadable at http://www.prolific.com.tw/eng/downloads.asp?ID=31). When the driver is first obtained it comes in the form of a ZIP file, as such the user must extract it first. After extraction is done, run the 'PL2303_Prolific_DriverInstaller' executable file to initiate the driver installation process, there will be a brief appearance of a loading window (Install Shield Wizard) before the introductory window prompt appears.



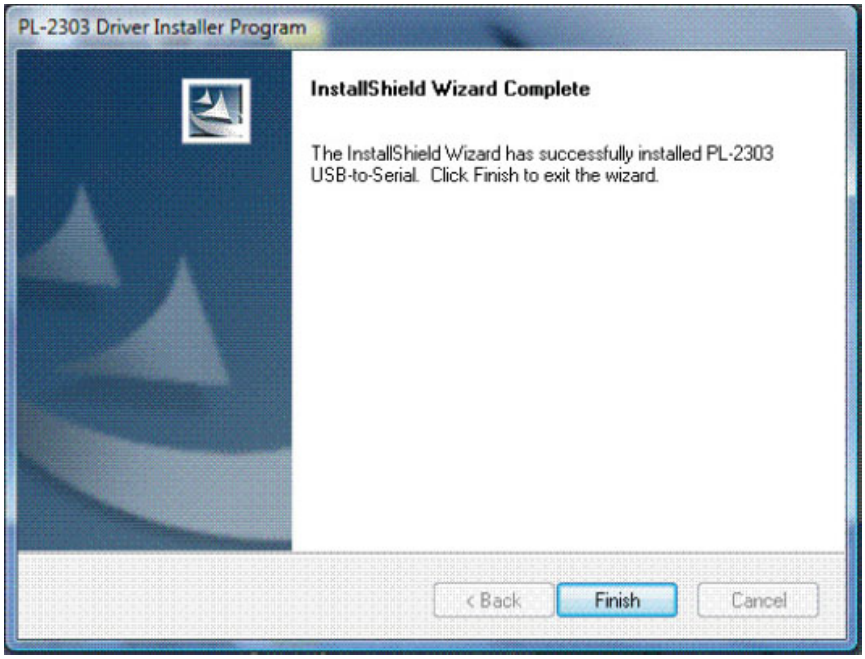
The succeeding page will just inform the user that the installer will install the PL-2303 driver, simply continue by clicking on 'Next >'.



The next page would appear like this.

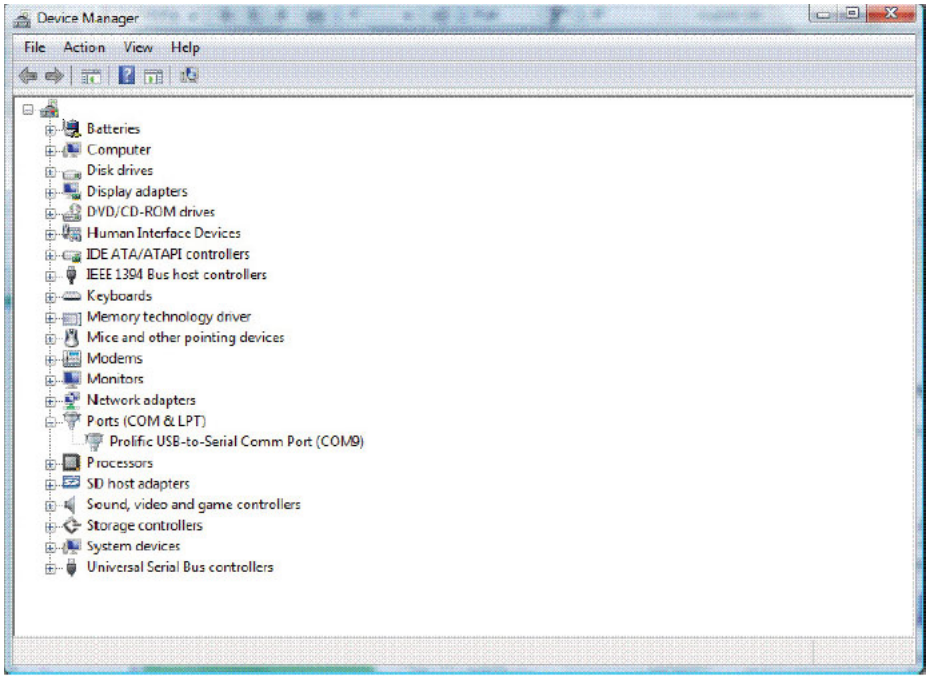


The installer will immediatly install the driver into the computer system. The installation progress will be displayed as a loading bar which will span a duration of about 20 to 30 seconds to complete. There is the chance occurence that even after the loading bar has finished and the window has minimized, the program itself might seem not to respond. This peculiar effect is normal, and usually resolves itself within around 30 seconds. If it takes more than around 5 minutes, end the program using task manager and repeat the process again, or repeat it after restarftng the computer. If all goes well, the complefton page will appear.



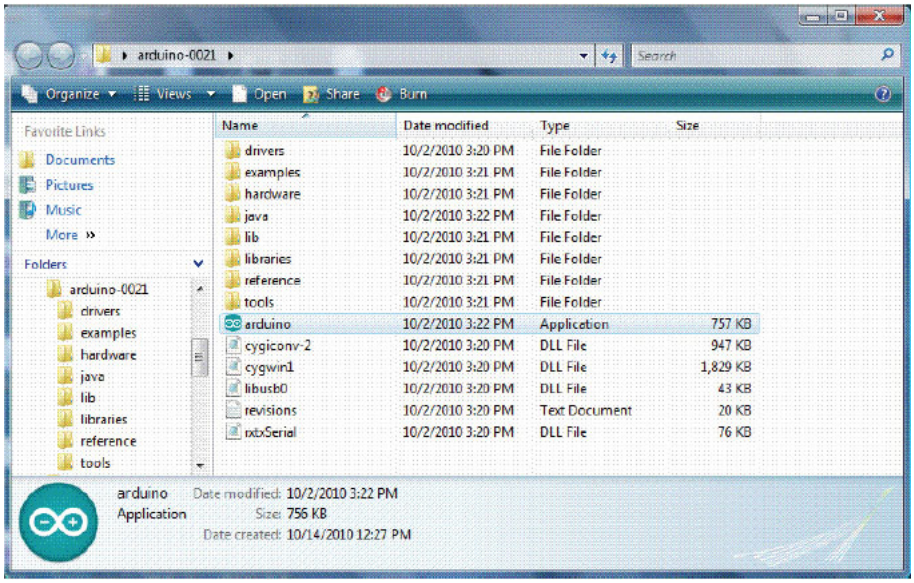
Now that we have accomplished the preliminary driver setup, we are left with the setup of the programming application Arduino IDE.

Before the Arduino IDE program can communicate with the Gizduino module, the user must first set the number of the COM port assigned to the USB cable. To identify as to which COM port the USB cable is currently attached to, open Start, then the Device Manager. Amongst the great list of devices present in the computer, one should find the 'Ports' category. Expanding on this category scrolls down the list of COM ports under use. The COM port the user should take note of is that which is labeled 'Prolific USB-to-Serial Comm Port (COM9)'. This is the COM port where the Gizduino module is attached to.



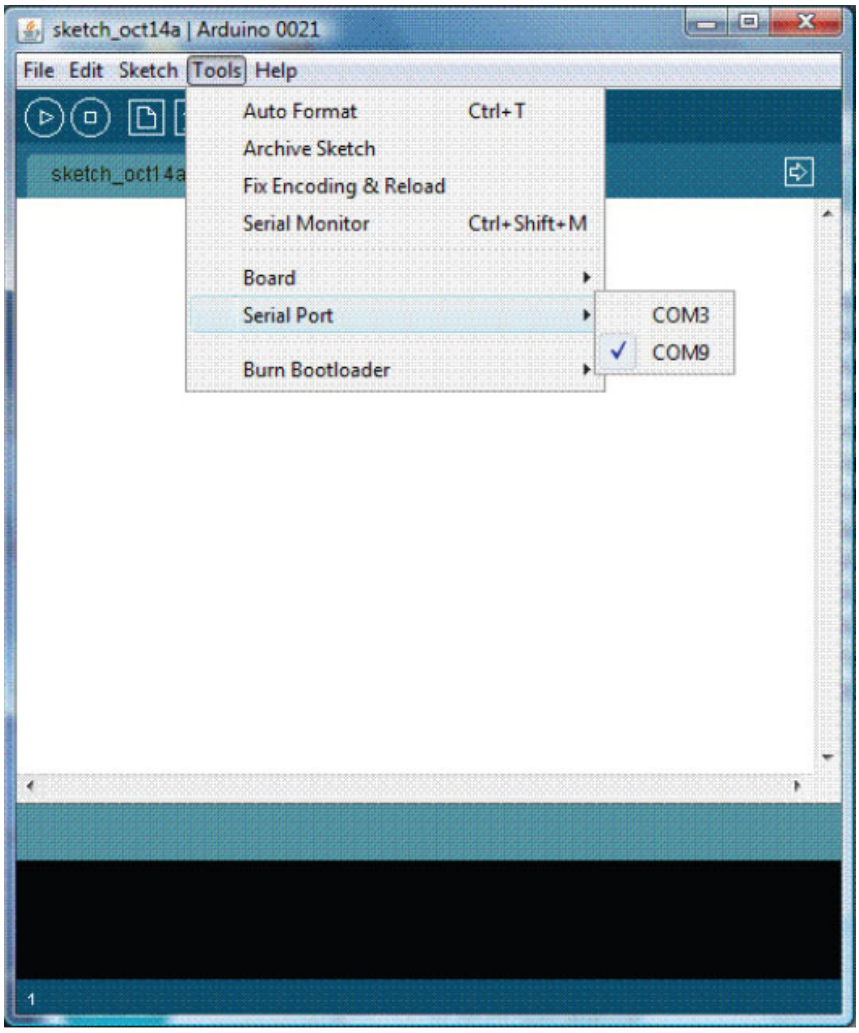
As an example, we will use COM9 as shown in the figure above. The user at this point is urged to finally open the Arduino IDE program now that we have the COM port number assigned to the Gizduino module!

To open the Arduino IDE, navigate to the folder extracted from the ZIP file previously mention. Upon opening the folder named 'arduino-0021' by default, search for the file that is highlighted in figure 3 below.

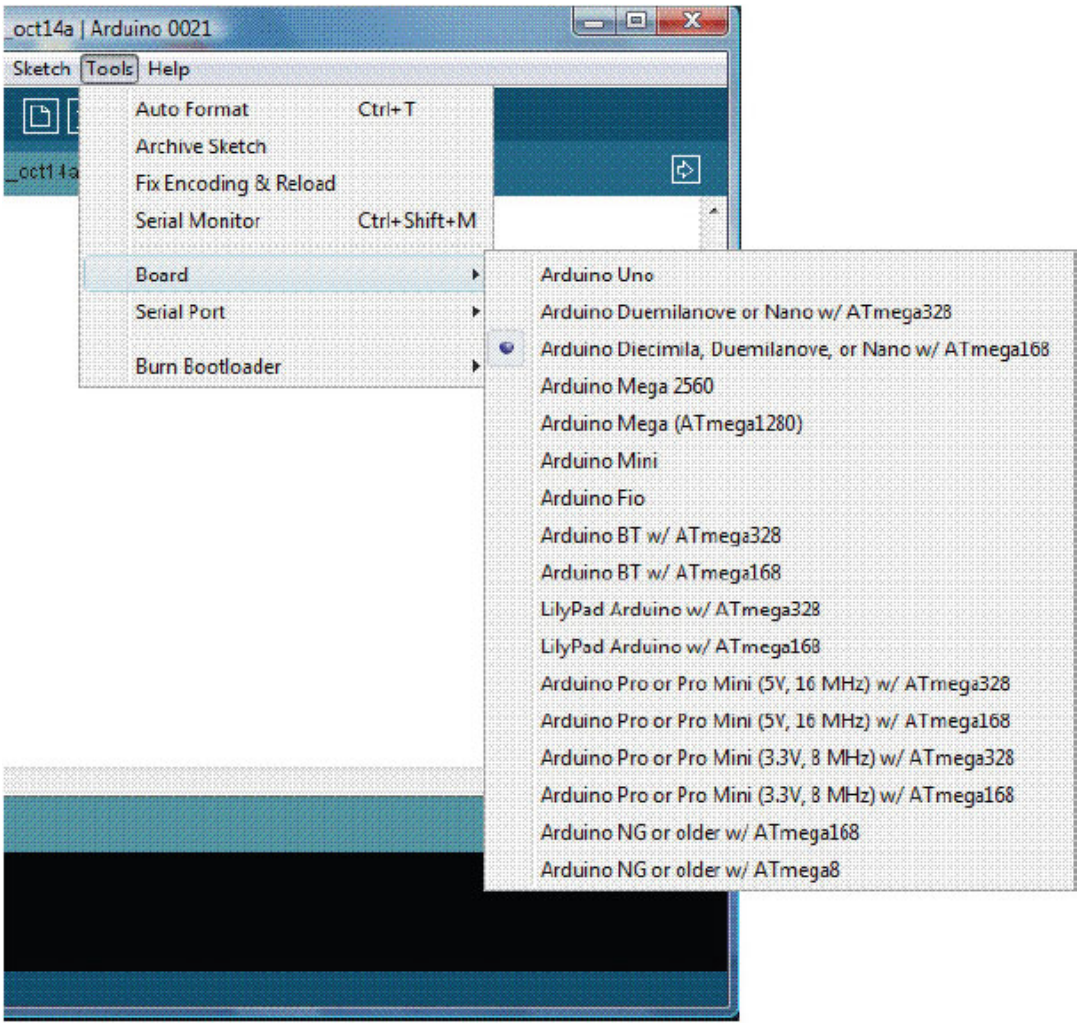


Double-click on the file to execute the Arduino IDE (Integrated Development Environment) program proper. This is where all the programming codes and instructions are made and uploaded unto the Gizduino module. These written instructions or program codes are referred to by the "-duino" community as 'Sketches' and shall here be also called as such. The Arduino upon execution will at first present a small loading screen and then proceed to the program itself. The user is here advised to hold all urge to hastily play around with the program as there are still two necessary prerequisites to complete in order to ensure that the user's creations here are communicated to the Gizduino. To do this, go to and click the option 'Tools' found at the menu bar just below the top bar of the program window. Once there, a small list will drop down. Notice that there will result a small amount of lag before the list is scrolled down, and that during this lag the D2 (orange LED) blinks briefly. This is a good sign that the program bears some communication to the COM ports.

Of primary concern are the two options found with the list are 'Board' and 'Serial Port'. First, place the mouse cursor over the 'Serial Port' option to open another short list to its right. The list will display various COM ports under use by the computer. We now assign the COM port that we earlier took note of at the Device Manager Window, which is COM9. Once the COM9 choice is selected, the 'Tools' list window will close. If the user will go to the 'Serial Ports' option again, the COM9 choice should have a check mark beside it.

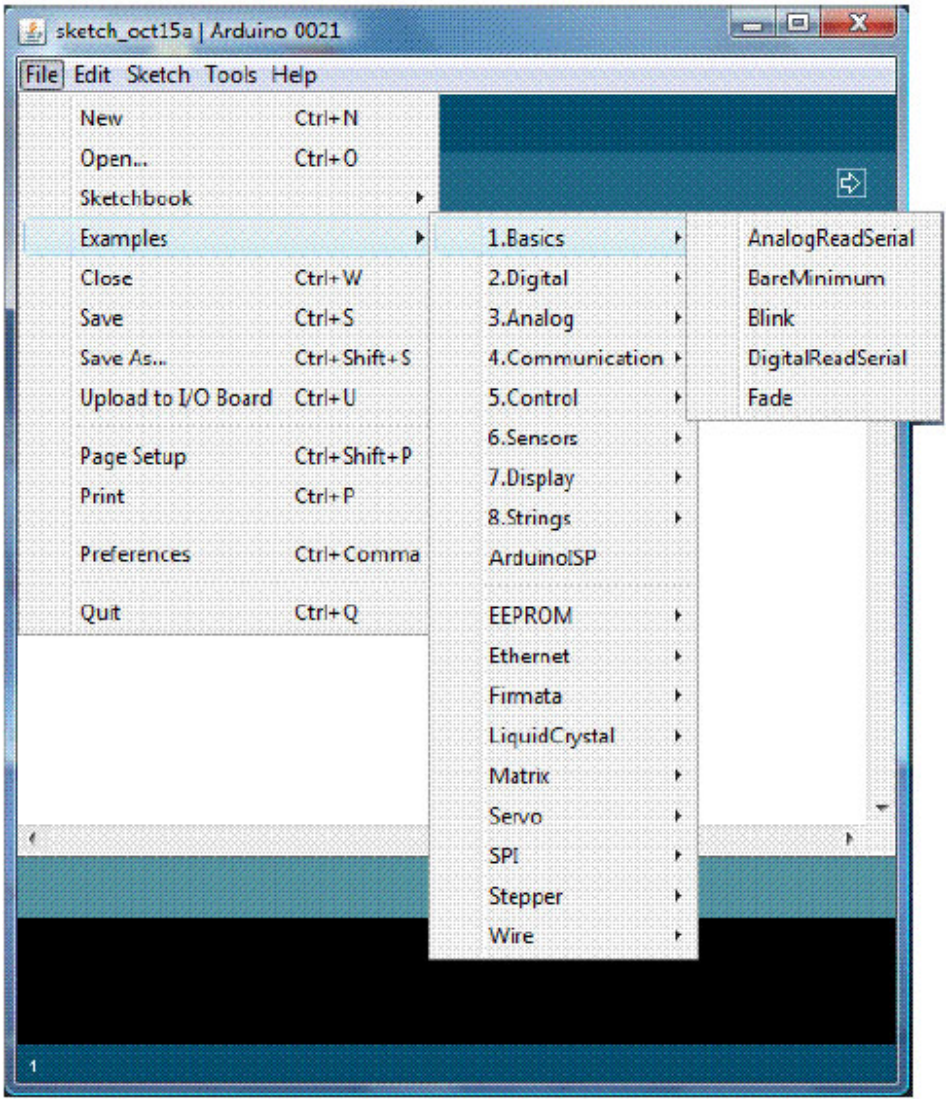


We now turn to the 'Board' option which is again found under the 'Tools' menu. Opening the 'Board' sub-list, we will encounter a window with a list of different models and versions of the Arduino modules such as the Arduino Uno or the Arduino Mega. The Gizduino module is modeled after the Arduino Diecimila with the ATmega168. As such, select the 'Arduino Diecimila, Duemilanove, Nano w/ ATmega168, Nano w/ ATmega328'.



BASIC PROGRAM SKETCHING

For the user with little to no knowledge at all with the C - based programming language, amongst the best ways to learn is to observe and analyze a simple existing working program code. In the Arduino IDE we will find such simple works as sample file in the program. To access these sample program codes, while on the Arduino IDE click on the 'File' menu to open a sub-list of other options. Some of these options have the basic program operations such as 'New'; which creates a new blank document, and 'Open...'; which allows the program to open an already existing document. The user will also see an 'Examples' category, and if the mouse cursor is placed on it, another list under it will appear containing the different kinds of sample codes made available to the user.It is highly recommended for those users who are new to the Arduino/Gizduino modules to first open samples under the '1.Basics' option (It is common for new timers to use the 'blinking LED' program code as a learning basis for coding).



UPLOADING CODES

While it is beyond the scope of the manual to discuss in greater detail on the subjects about the Gizduino code programming, the user may wish to instead visit the e-Gizmo official website (http://e-gizmo.com/wordpress/) and the Arduino official website 'learning' page (http://arduino.cc/en/Tutorial/HomePage). Both these websites offer tutorials regarding programming in the Arduino IDE and the programming tutorials available at the Arduino page offer codes ranging from the simple and basic to the more complex and advanced levels. However, there are some differences between the Arduino and the Gizduino board, and those who wish to learn basic code compatible with a Gizduino module should start of at the e-Gizmo Gizduino tutorial web pages (such as http://e-gizmo.com/wordpress/?p=639).

Another indispensible page found at the Arduino website is the 'References' page (http://arduino.cc/en/Reference/HomePage). It is the web page where most of the important and common language codes recognized and used by the Arduino IDE are given description. Users are encouraged to have this page open whilst they are along the analysis of a sample codes. Doing so will provide easier understanding of workings of a given code.

For the users who have inquiries about the Gizduino whose answers lie beyond that which was provided in the web pages suggested above, they are encouraged to register at the e-Gizmo technical blog/forum section (http://e-gizmo.com/wordpress/) and post their inquiries and curiosities there. The Arduino forum webpage is also good for those who have their questions or ideas about the Arduino technologies (http://www.arduino.cc/cgi-bin/ yabb2/YaBB.pl).

UPLOADING CODES TO GIZDUINO

Assuming that the user has now created a body of programming code in the Arduino IDE, it now must be uploaded to the Gizduino module. To do this, simply go to 'File' in the Arduino IDE main menu, and select 'Upload to I/O Board'. However, it is highly recommended that the user save the program code in order to ensure that whatever error might happen in the upload the program code is backed-up. In addition to this the user is also encouraged to practice the habit of first pretesting the body of programming before it is uploaded to the Gizduino module by selecting the 'Verify/Compile' command found under the 'Sketch' main menu option. The command simulates the programming code and returns any errors about the writing of the code, saving users the trouble of errors and crashes resulting for a hasty upload of an erroneous code unto the Gizduino.

For gizDuino and other Shields operations or manuals simply scroll down or click the "Manual Selection" on the Lower right of the manual's page.

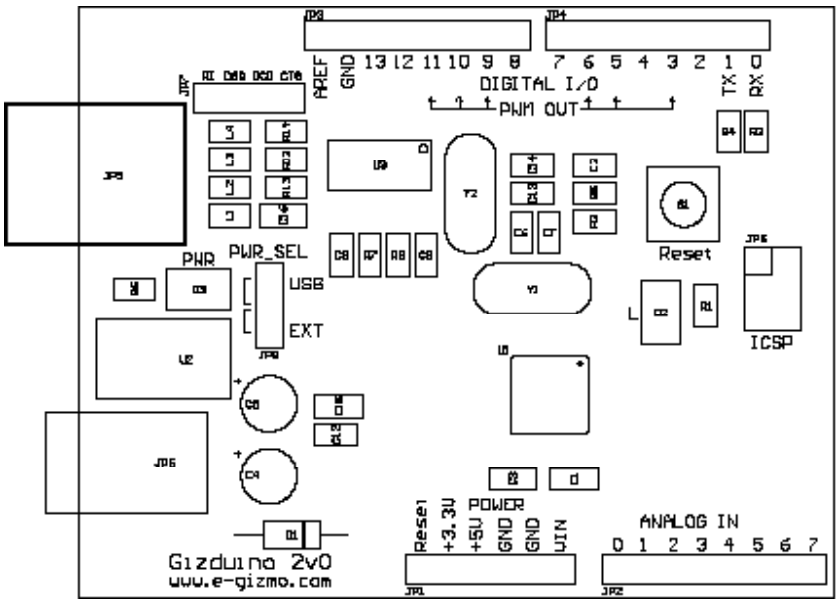


Figure 4. *Gizduino Arduino Compatible Kit (silkscreen layout)*

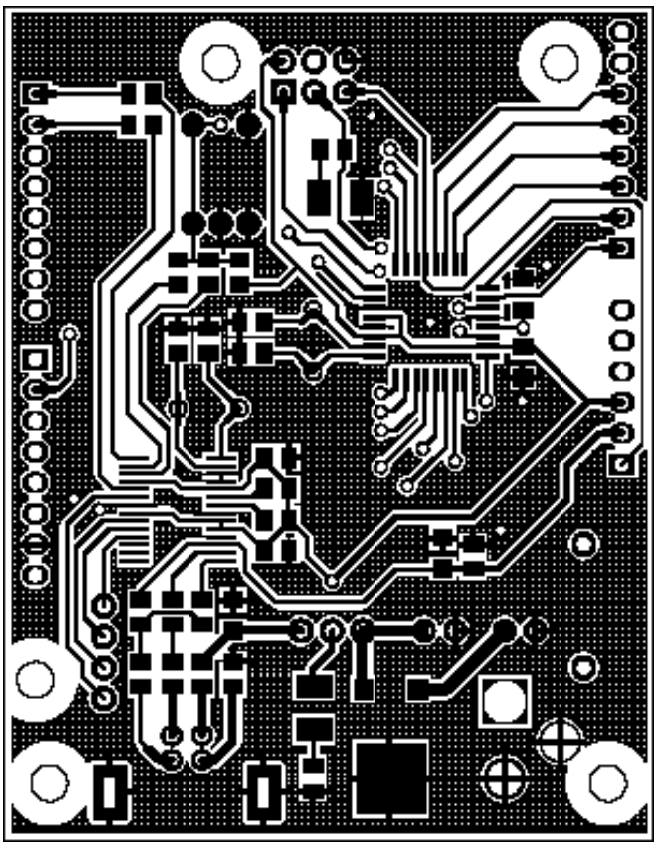


Figure 5. *Gizduino Arduino Compatible Kit Copper Pattern (Top Layer)*

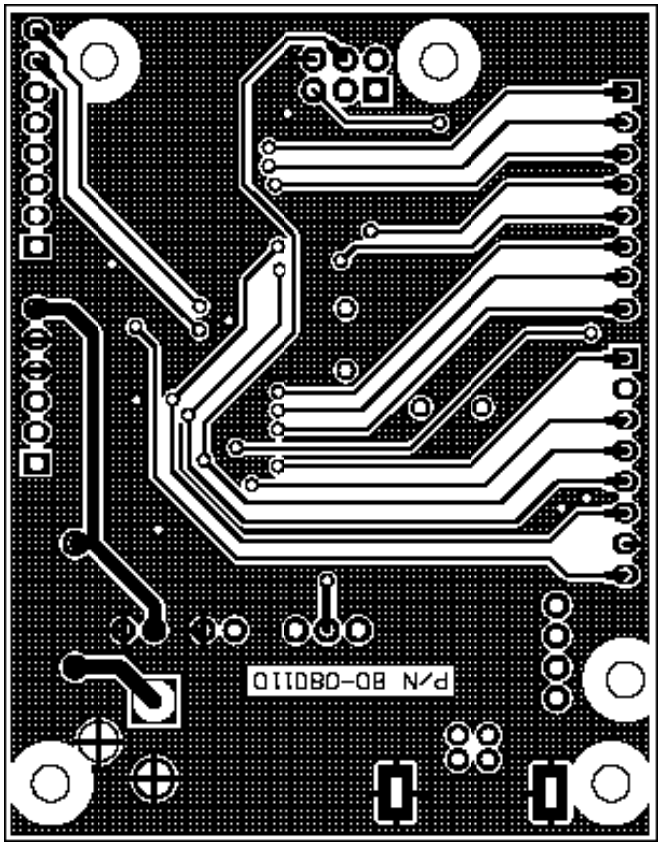


Figure 6. *Gizduino Arduino Compatible Kit Copper Pattern (Bottom Layer)*

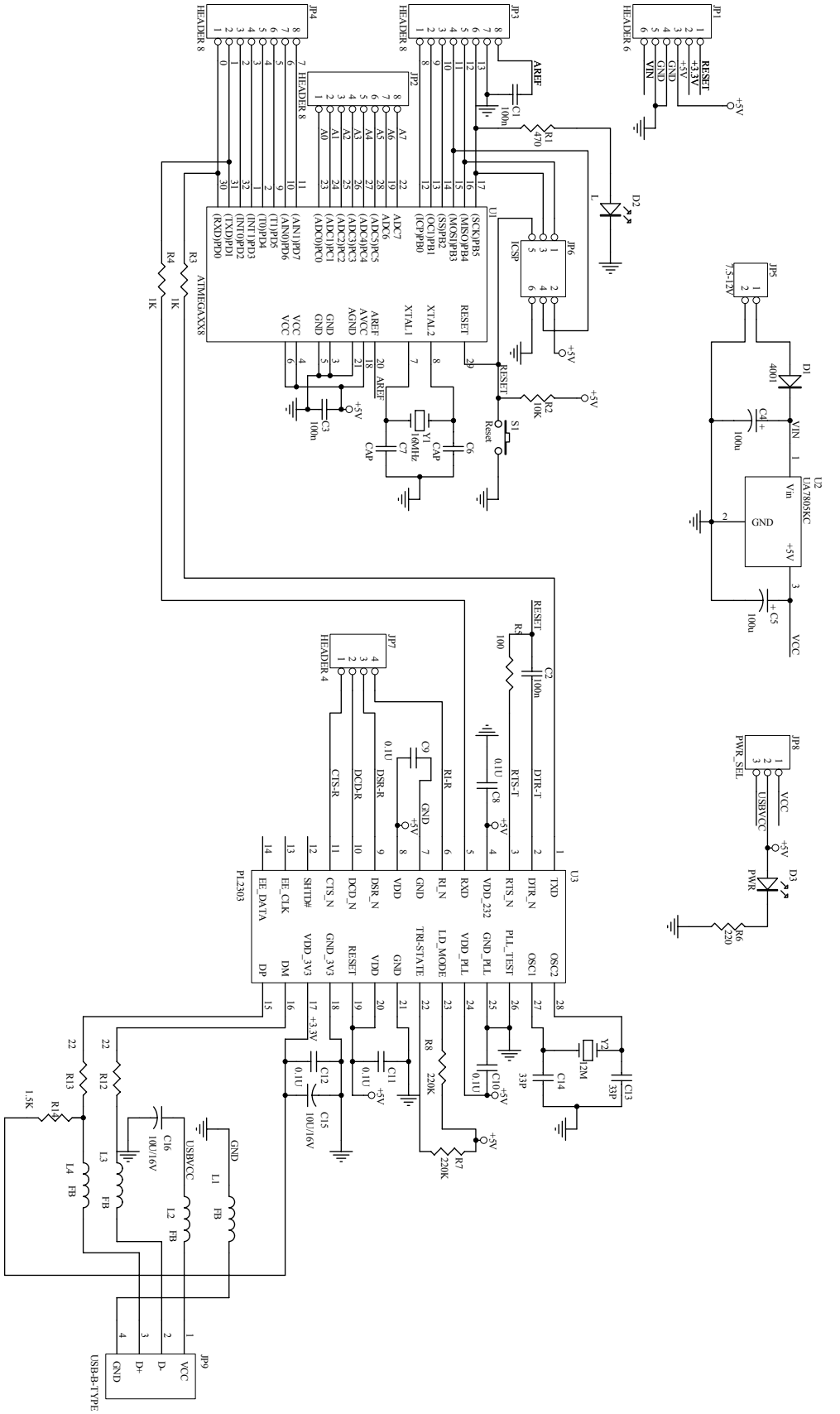


Figure 7. *Schematic Diagram of Gizduino Arduino Compatible Kit*