**Intel Galileo**

**About:**

Intel Galileo is the first [Arduino](https://en.wikipedia.org/wiki/Arduino)-certified development board based on [Intel](https://en.wikipedia.org/wiki/Intel) x86 architecture and is designed for educational communities. Intel released two versions of Galileo, Gen 1 and Gen 2. These are sometimes called "Breakout boards". The Intel Galileo is a board based on the IntelQuark SoC X1000, a 32­bit Intel Pentium brand system on a chip. The Intel Galileo has built-in networking and more memory capability than other boards, making it suitable for libraries and incorporate large display functions. It supports Microsoft Windows, Mac OS and Linux host operating systems, and the Arduino IDE.

**Features:**

Galileo is a microcontroller board based on the Intel® Quark SoC X1000 Application Processor, a 32-bit Intel Pentium-class system on a chip. Galileo is designed to support shields that operate at either 3.3V or 5V. The core operating voltage of Galileo is 3.3V. Processor Included is [Intel® Quark™ SoC X1000 (16K Cache, 400 MHz)](https://ark.intel.com/products/79084/Intel-Quark-SoC-X1000-16K-Cache-400-MHz) with Max Memory Size of 256 MB and type DDR3 800.It has 1 memory channel with memory bandwidth of 2.5 GB/s and 32 bit extensions. It has PCI express PCI support with 1 mini card slot. It also has 3 USB 2.0 ports and integrated LAN port. Its 15mm\*15mm with only 1 CPU configuration.

**Booting up:**

To get started, simply connect the board to power with the 5V AC to-DC adapter and then connect to the computer with the micro-USB cable. By default they measure from ground to 5 volts.

1. Locate the 32GB SanDisk bootable pen drive inside the accessories box. This pen drive is

loaded with a custom operating system with all the necessary software tools pre-installed

to kick start the application development with eIoT3500.

2. You need a host machine a.k.a computer to boot the pre-installed pen drive.

Note: Please ensure that your host machinea.k.a computer has a 64 bit processor

architecture before you attempt to boot the pre-installed pen drive provided.

Check the PROCESSOR\_ARCHITECTURE environment variable on your host a.ka.

Computer to know whether the processor architecture is 64 bit or 32 bit.

Open command line in your windows PC and type

echo PROCESSOR\_ARCHITECTURE.

If the output gives AMD64 then your processor is 64 bit else the output will be x86

which tells us that the processor architecture is 32-bit.

If your host machine a.k.a computer is already running any flavor of Linux, then please

follow the instructions provided in the link below.

http://www.cyberciti.biz/faq/linux-how-to-find-if-processor-is-64-bit-or-not/

3. Shutdown your host machine a.k.a computer

4. Connect the USB bootable pen drive to your host machine a.k.a computer USB port.

5. Power up the system and enter the BIOS set up menu.

Note: Please check how to enter BIOS set up menu on your host machine. Usually

pressing the keys F1/F2 immediately after pressing power button will allow you to enter

the BIOS set up menu option.

6. Navigate through the BIOS set up menu and disable the secure boot option if present.

7. Navigate to the boot order option in the BIOS set up menu and move the USB pen drive

to the top of the list.

8. After all the above changes are done, save the BIOS settings and boot the setting. Usually

pressing F10 key will allow you to save and exit from the BIOS set up menu

environment.

9. After all the above steps are followed the system should automatically boot into the system.

## Interfacing Piezo Buzzer

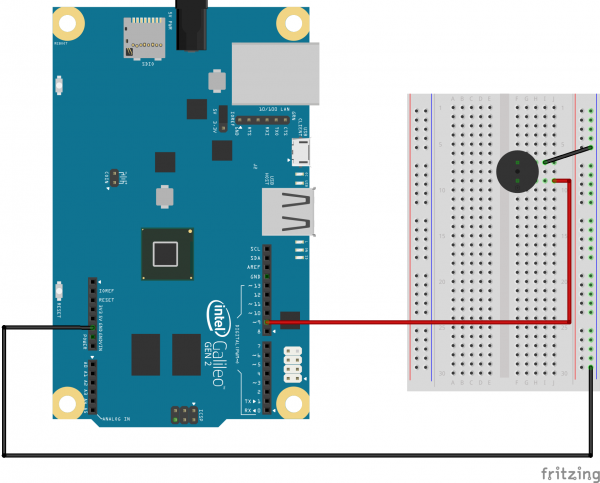
### **Parts Needed**

* **1** Breadboard
* **1** Galileo
* **1** Piezo Buzzer
* **3** Jumper Wires

### **Hardware Setup**

See the diagram below, to get an idea of connection. Look at components properly to see how to place them on the breadboard . Polarized components can only be connected to a circuit in one direction.

#### **Fritzing Diagram**

[](https://cdn.sparkfun.com/assets/learn_tutorials/2/7/5/circuit_11_bb.png)

Having a hard time seeing the circuit? Click on the Fritzing diagram to see a bigger image.

### **Code To Note**

Characters (single, printable, letters, numbers and other symbols) have their own type, called “char”. Array of characters can be defined as a string.

In Arduino there is a tone() function which set the frequencies to play the perfect tone for buzzer. If the duration is set then buzzer runs for that time and then stops if not mentioned then it runs continuously.

**BUZZER**

Buzzer have a coil inside it. When a current is passed through it, it gets attached to the valve which create a sound. As this takes place many times in a second it form a tone.

Hardware connections:

The buzzer has two pins. One is positive and one is negative.

The postitive pin is marked by a "+" symbol on both the top

and bottom of the buzzer.

Connect the positive pin to Arduino digital pin 9.

Connect the negative pin to GND.

Tip: if the buzzer doesn't fit into the breadboard easily,

try rotating it slightly to fit into diagonal holes.

This sketch was written by SparkFun Electronics,

with lots of help from the Arduino community.

(This sketch was originally developed by D. Cuartielles for K3)

This code is completely free for any use.

Visit http://learn.sparkfun.com/products/2 for SIK information.

Visit http://www.arduino.cc to learn about the Arduino.

note frequency

c 262 Hz

d 294 Hz

e 330 Hz

f 349 Hz

g 392 Hz

a 440 Hz

b 494 Hz

C 523 Hz

For more information, see <http://arduino.cc/en/Tutorial/Tone>

const int buzzerPin = 9;

const int songLength = 18;

// Notes is an array of text characters corresponding to the notes

// in your song. A space represents a rest (no tone)

char notes[] = "cdfda ag cdfdg gf "; // a space represents a rest

// Beats is an array of values for each note and rest.

// A "1" represents a quarter-note, 2 a half-note, etc.

// Don't forget that the rests (spaces) need a length as well.

int beats[] = {1,1,1,1,1,1,4,4,2,1,1,1,1,1,1,4,4,2};

// The tempo is how fast to play the song.

// To make the song play faster, decrease this value.

int tempo = 150;

void setup()

{

pinMode(buzzerPin, OUTPUT);

}

void loop()

{

int i, duration;

for (i = 0; i < songLength; i++) // step through the song arrays

{

duration = beats[i] \* tempo; // length of note/rest in ms

if (notes[i] == ' ') // is this a rest?

{

delay(duration); // then pause for a moment

}

else // otherwise, play the note

{

tone(buzzerPin, frequency(notes[i]), duration);

delay(duration); // wait for tone to finish

}

delay(tempo/10); // brief pause between notes

}

}

int frequency(char note)

{

// This function takes a note character (a-g), and returns the

// corresponding frequency in Hz for the tone() function.

int i;

const int numNotes = 8; // number of notes we're storing

char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };

int frequencies[] = {262, 294, 330, 349, 392, 440, 494, 523};

// Now we'll search through the letters in the array, and if

// we find it, we'll return the frequency for that note.

for (i = 0; i < numNotes; i++) // Step through the notes

{

if (names[i] == note) // Is this the one?

{

return(frequencies[i]); // Yes! Return the frequency

}

}

return(0);

}

**IDE:**

Initially, when you write code for your Galileo, you will use a customized version of the Arduino IDE from intel. This is because the Galileo is an "[Arduino Certified](http://arduino.cc/en/ArduinoCertified/Products" \o "Arduino Certified Product Page)" board.

1. Intel has a customized version of the Arduino IDE as part of their "[Software package release version 1.0.3](https://communities.intel.com/docs/DOC-22226)". There are versions for multiple operating systems.
2. Once you have download the .zip file, make sure to "Unblock" it before you extract it. Right click on the file, then select **Properties** from the pop-up menu. Then in the properties window, click the **Unblock** button, then "OK".
3. Next, extract the downloaded .zip file to a folder off the root of your C:\ (or the drive of your choice). Make sure to extract to a folder as close to the root as possible.
4. Rename the extracted folder **"Arduino 1.5.3".**
5. Great, we are getting closer, but first, we need to install the USB driver for the Galileo on our machine and make sure that our Galileo's firmware is up to date.

**References:**

* <https://www.packtpub.com/mapt/book/hardware_and_creative/9781782174585/2/ch02lvl1sec14/downloading-and-installing-the-ide>
* <https://www.intel.in/content/www/in/en/support/articles/000006387/boards-and-kits/intel-galileo-boards.html>
* <http://www.libelium.com/libelium-connects-intel-galileo-to-sensors-for-the-internet-of-things/>
* <http://www.nskelectronics.com/intel_galileo.html>