Galileo Board

Basic Tutorial

Sanket R. Bhimani

July 8, 2016

Contents

1 Tutorial Name:

"Basic Tutorial For Galileo Board"

This tutorial will help you to do basic interfacing with Galileo board. This board is new to market. So, you will not get enough resources to complete your tasks.

2 Prerequisites:

- Basic knowledge of Linux OS because we will run Linux on Galileo board.
- Basic Arduino programming.
- what is SSH
- read Galileo from "Sign Language Interpreter" tutorial, so you get knowledge about installing Linux on the Galileo board.

3 Requirement

- Hardware Requirement:
  - Galileo Board
  - LEDs
  - Servo Motors
  - Connecting wires
  - LAN cable or WIFI module(for connecting board with PC)
  - 5V power supply
  - SD card of minimum 8 GB

- Software Requirement:
  - Python editor
  - Leap sdk
  - SSH Client
  - Arduino IDE
  - various library
  - SSH Client
  - Wireshark(optional)

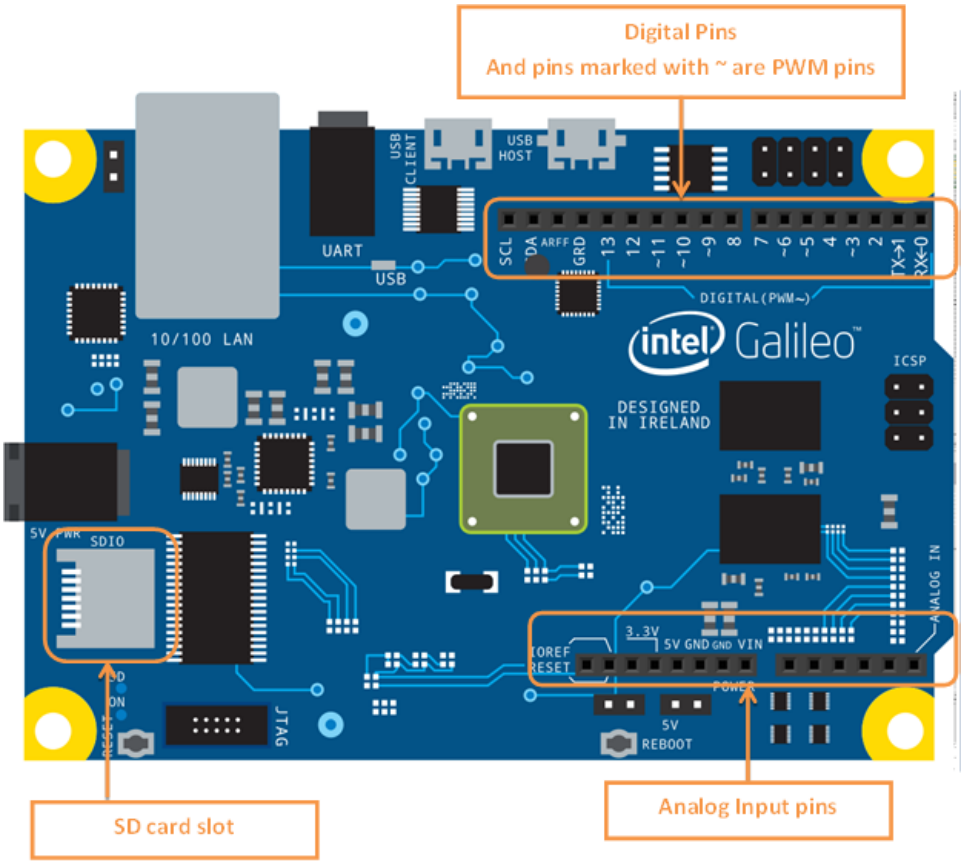

Figure 1: Pin configuration

4 Theory and Description

- Galileo board can be used in two ways :
  - As an Arduino (IDE is also same)
  - By installing a particular OS on it. Board supports three OS: Linux, Windows IOT and OSX

IOT feature is also available for both modes. This implies that the board can be connected to the Internet without installing any OS.

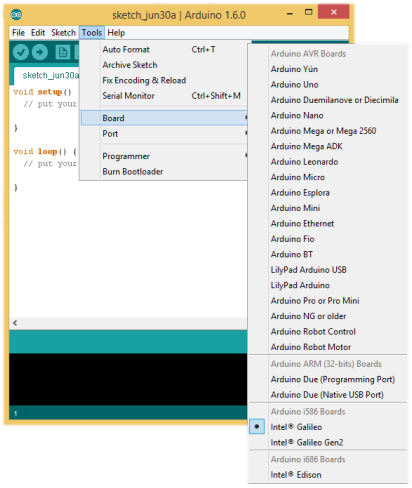

Figure 2: Arduino IDE selecting board

For using Galileo board from Arduino IDE no extra fuctions are required as all the things are same. We just need to select Galileo in board option. (Refer to Figure 1)

Tools -) Board -) Intel Galileo

Then choose the appropriate port number from:
Tools -) Port

Installation of Linux image in Galileo board:

In Galileo board, we can install Linux, windows IOT or os x. So the interface with board become easy. Here we will use Linux image. For that, we need to write Linux image on the sd card. From sd card, Galileo board will boot.

So How to prepare sd card???

Step 1:
Download Linux image from this location.
`https://downloadmirror.intel.com/26028/eng/iot-devkit-prof-dev-image-galileo-201605zip`
You will get one zip file extract it. You will find one '.direct'. You need to write this into sd card

Step 2:
Download this software from this location,
`http://download.softpedia.com/dl/d5a643aad00816ee735372c2d530a1a2/57584b56/100173006/software/cd_dvd_tools/Win32DiskImager-0.9.5-binary.zip`
this will be helpful for writing an image in sd card.

Select Image File which you have downloaded earlier in 'Image File' tab('.direct' file)
Select sd card in 'Device tab'
Now click on 'Write' button, it will take some time. And wait until write process completes

Step 3:

Now insert sd card in galileo board and connect LAN cable with your PC and galileo board. Now connect the power supply with galileo board.
now the board will automatically boot. There will be blinking LED near sd card port. Wait until it stops lighting. Because it shows

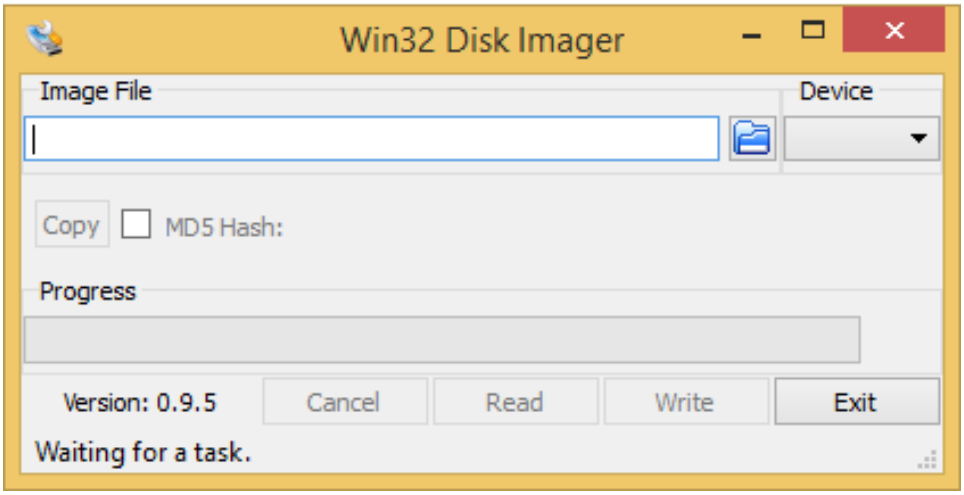

Figure 3: Win32 Disk Image windows

IO related sd card. So, when it become stop, IO related sd card is also stopped so the booting process is completed. Now you need to find the IP address of your board. For that, you can use any network packet tracer software like Wireshark, ettercap or any other.

I am using Wireshark.

Now open Wireshark select ethernet as a network adapter. It will start tracing all packet data on ethernet. But here on an ethernet network, there is only two devices are there your PC and board. So there is only two IP address data. (ignore IPs with 255, 251, 224 and more than 200 value in last two part like 192.168.0.255 or 255.255.255.0) so find your PC IP address using this command

```
ipconfig (for windows)}
ifconfig (for Linux)}
```

Now another IP showing on Wireshark is your board IP.

Your Wireshark windows:

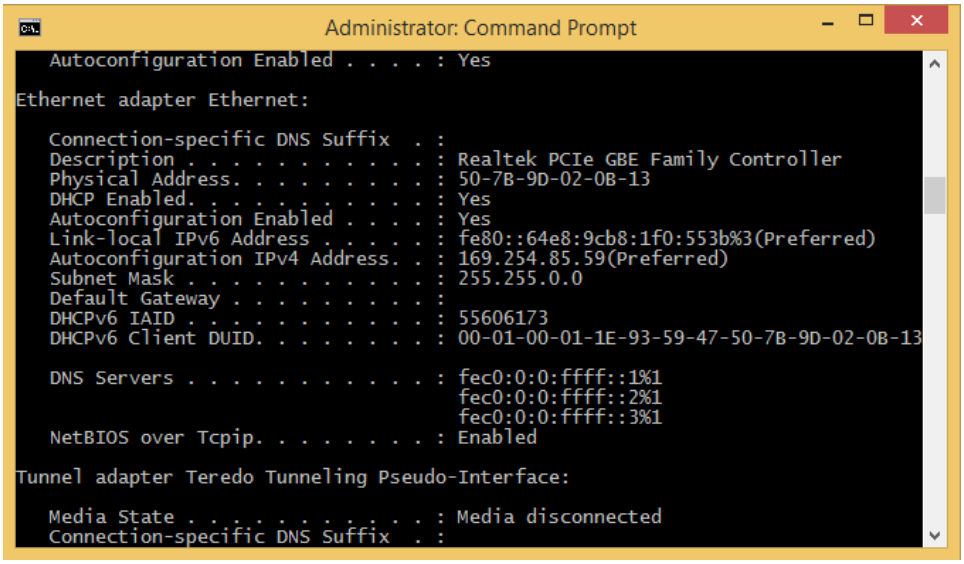

Figure 4: CMD for your IP address

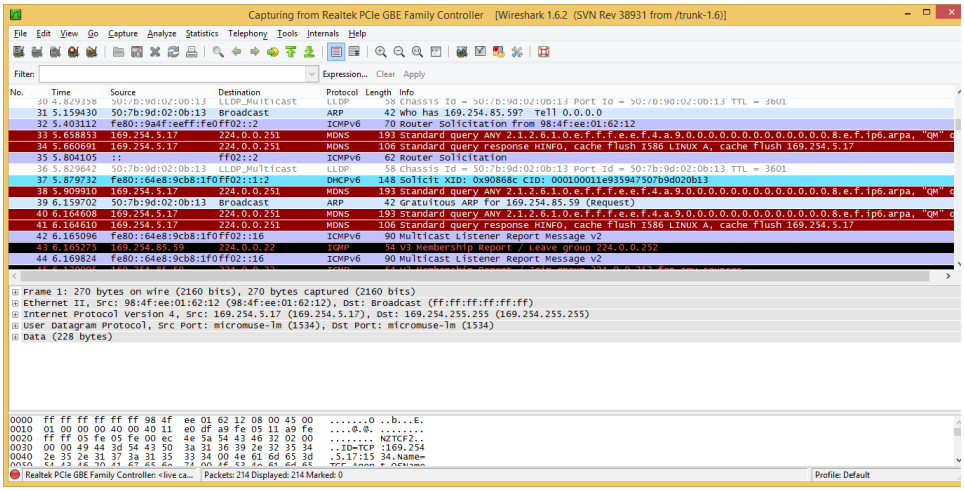

Figure 5: Wireshark windows for board IP

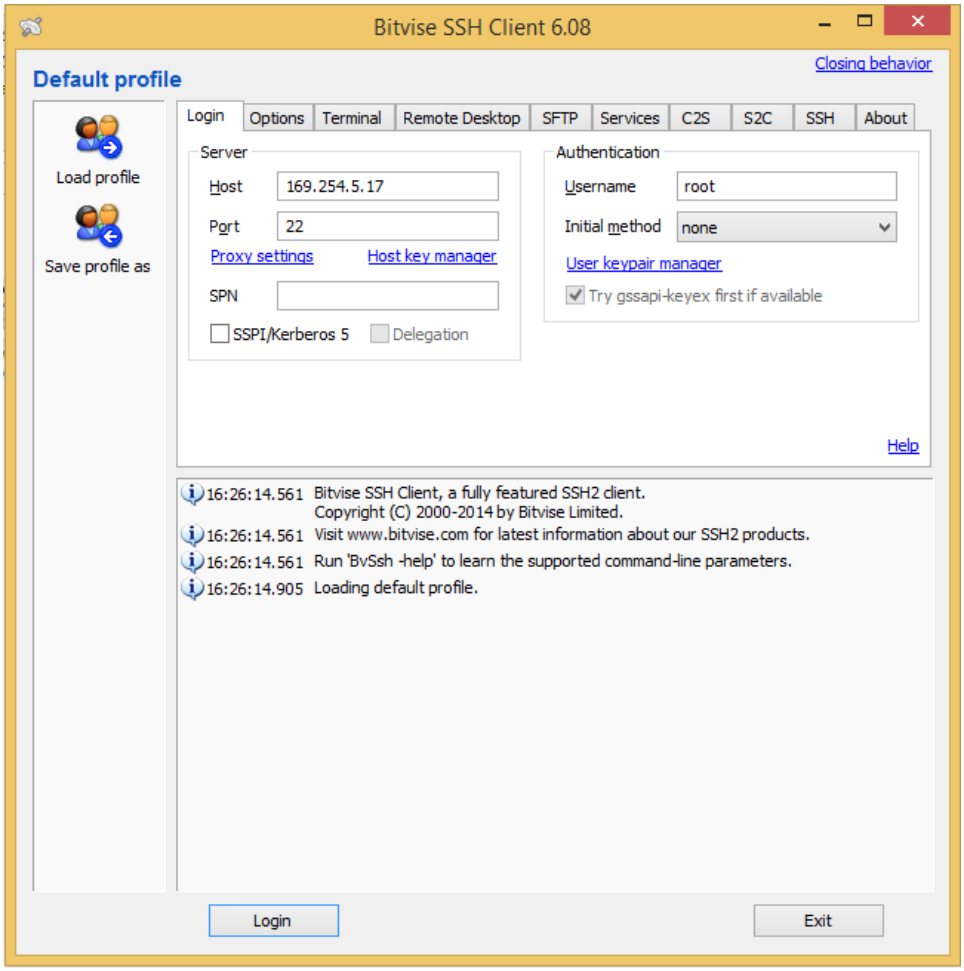

Figure 6: SSH client window

Step 3:

Now, from any SSH client make connection with board and use. :?)
(I'm using Bitvise SSH client)
Enter IP address and username: root there is no password.

Click on login, and wait. Now two windows will open one for terminal and another for file transfer. Your board is ready! :?)

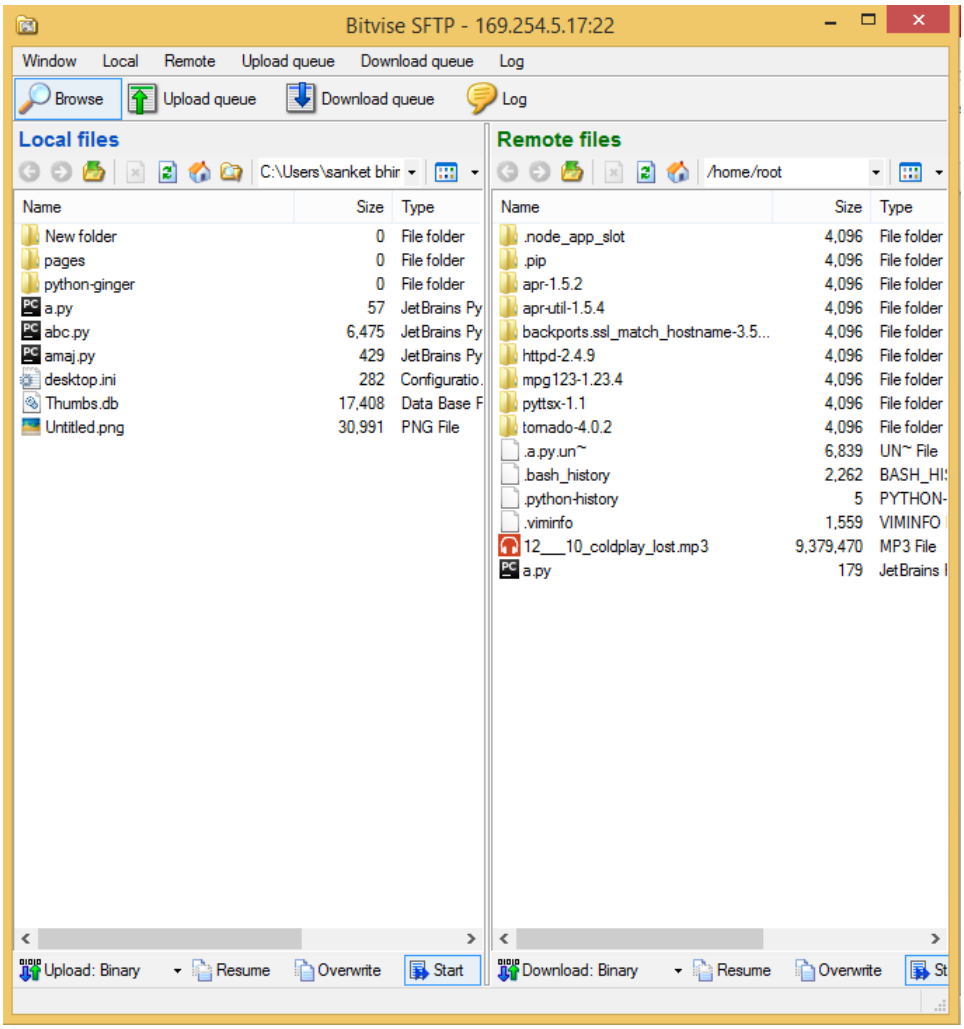

Figure 7: SSH client file transfer window

Figure 8: SSH client terminal window

Basic interface with GPIO pins:

Most of GPIO capabilities are done through Linux Sysfs. Means board can be controlled by file based I/O operation. Means Galileo linux os gives an interface to controle GPIO pins through something like file system. So using that we can get inputs and also gives outputs or generate PWM signals.

For that there is some steps tobe followed,

- First you need to export that pin
- – Defining the pin number which you are using
- Then set direction 'in' or 'out'
- – Define that you are using it as input or output
- GPIO Port Drive Configuration: 'pullup', 'pulldown', 'strong' or 'hiz' (Optional)
- – Define the intensity.
- Read or write value

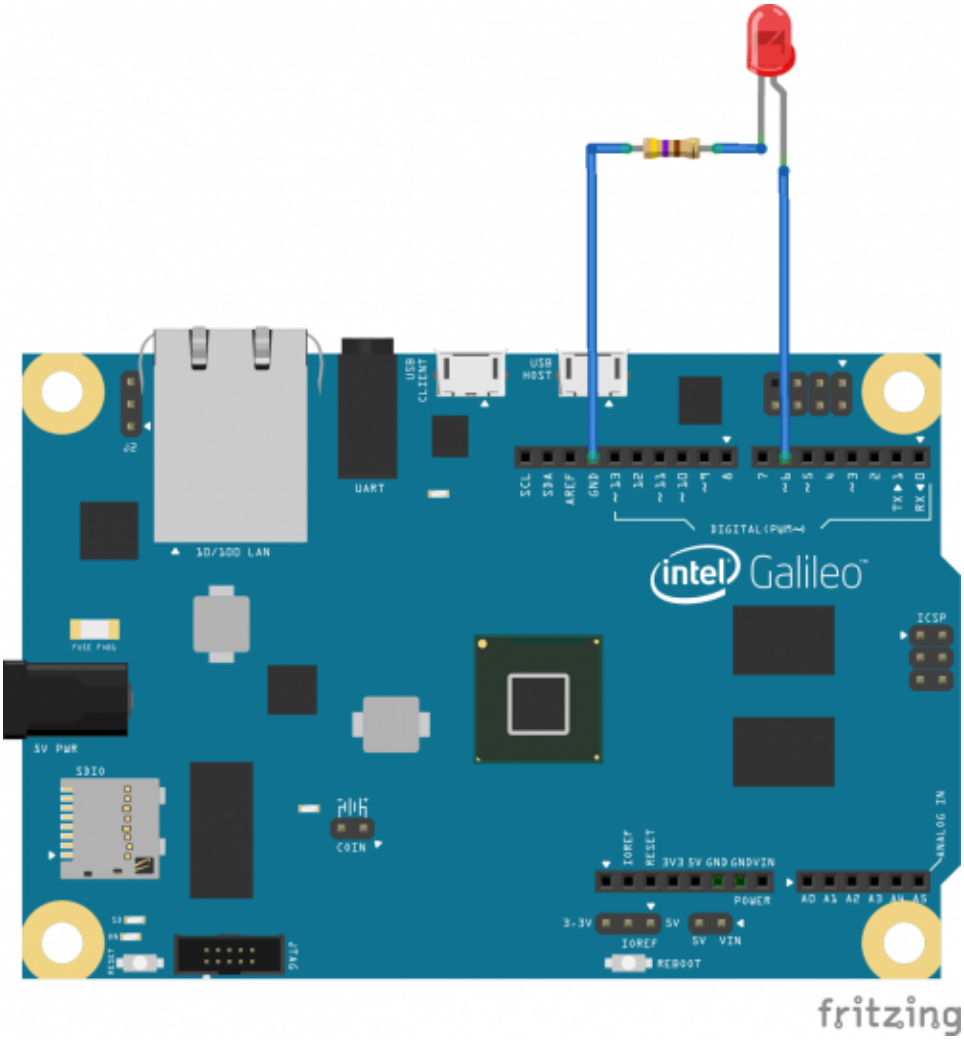

Figure 9: LED connection

Steps to be followed to blink/glow LED:

```
root@Galileo:~#echo -n "24" > /sys/class/gpio/export
root@Galileo:~#echo -n "out"> /sys/class/gpio/gpio24/direction
root@Galileo:~#echo -n "strong" > /sys/class/gpio/gpio24/drive
root@Galileo:~#echo -n "1" > /sys/class/gpio/gpio24/value
```

Here 27 is allocated for digital pin 6 so connect your LED at digital pin number 70

On-board number	Linux fs number	Settings
Digital 0	gpio50	Gpio40 = 1
Digital 1	gpio51	Gpio41 = 1
Digital 2	gpio32	Gpio31 = 1
Digital 3	gpio18	Gpio30 = 1
Digital 4	gpio28	
Digital 5	gpio17	
Digital 6	gpio24	
Digital 7	gpio27	
Digital 8	gpio28	
Digital 9	gpio19	
Digital 10	gpio16	Gpio42 = 1
Digital 11	gpio25	Gpio43 = 1
Digital 12	gpio38	Gpio54 = 1
Digital 13	gpio39	Gpio55 = 1
Analog 0	in_voltage0_raw	Gpio37 = 0
Analog 1	in_voltage1_raw	Gpio36 = 0
Analog 2	in_voltage2_raw	Gpio23 = 0
Analog 3	in_voltage3_raw	Gpio22 = 0
Analog 4	in_voltage4_raw	Gpio21 = 0 & Gpio29 = 1
Analog 5	in_voltage5_raw	Gpio20 = 0 & Gpio29 = 1

Figure 10: Table for GPIO pin and SYSFS pin

Here is the chart for mapping with on board pin to linux SYSFS system for GPIO

Here in setting column you need to set given value to use that pin. This way you can perform any tasks like,

- Digital input
- Digital output
- Analog input
- PWM singal
- SPI I/O
- I2C I/O
- Fast I/O

Digital Input:

It will help you to take input from digital pins

```
root@Galileo:~#echo -n "27" > /sys/class/gpio/export
```

here 27 is gpio pin number

and /sys/class/gpio is a path of something like file from whch you can access gpio pins

```
root@Galileo:~#echo -n "in"> /sys/class/gpio/gpio27/direction
```

'in' means using as input

```
root@Galileo:~#cat /sys/class/gpio/gpio27/value
1
```

reading the value 1

This will give you the value input on digital pin 7 in form of 0 or 1

Digital Output:

It will help you to give output to digital pins

```
root@Galileo:~#echo -n "27" > /sys/class/gpio/export
```

here 27 is gpio pin number

and /sys/class/gpio is a path of something like file from whch you can access gpio pins

```
root@Galileo:~#echo -n "out"> /sys/class/gpio/gpio27/direction
```

'out' means use as output

```
root@Galileo:~#echo -n "strong" > /sys/class/gpio/gpio27/drive
```

. set intensity of pin 7

```
root@Galileo:~#echo -n "1"> /sys/class/gpio/gpio27/value
```

set value

Analog Input:

This will helps you to take input from analog pin.
If we want to take input from analog pin 0 we need to set GPIO37 = 0 (as shown in figure)

```
root@Galileo:~#echo -n "37" > /sys/class/gpio/export
root@Galileo:~#echo -n "out"> /sys/class/gpio/gpio37/direction
root@Galileo:~#echo -n "0"> /sys/class/gpio/gpio37/value
```

. we need to do some setting for take input from A0 as per figure: 10
And now read the value from A0 pin at this way,

```
root@Galileo:~#catsys/bus/iio/devices/iio\:device0/
    in_voltage0_raw
2593
```

Generating PWM signal:

Here is the mapping with digital pin number and PWM SYSFS gpio no.

Now, If we want to generate PWM signal of period 1 millisecond and

duty cycle 50% then follow this command,

```
root@Galileo:~#echo -n "3" > /sys/class/pwm/pwmchip0/export
```

To enable a PWM on a port write "1" to a corresponding enable file. To disable a port write "0".

```
root@Galileo:~#echo -n "1">/sys/class/pwm/pwmchip0/pwm3/enable
```

To set a PWM period write period in nanoseconds to period file. In example below it is set to 1000000 nanoseconds or 1 millisecond:

```
root@Galileo:~#echo -n "1000000">/sys/class/pwm/pwmchip0/pwm3/
    period
```

.

To set a PWM duty cycle write its length in nanoseconds to duty_cycle file. In example below we set duty cycle to 50% (500000/1000000*100%):

Digital Pin Number	PWM Channel Number
3	3
5	5
6	6
9	1
10	7
11	4

Figure 11: Digital pin number with PWM channel number

```
root@Galileo:~# echo -n"500000" > /sys/class/pwm/pwmchip0/pwm3/
    duty_cycle
```

Fast I/O:

Galileo supports fast I/O on digital pins 2 and 3 which can be connected using multiplexer directly to Intel Quark X1000

Normally, we are not developing system using this way, we are using some library like mraa, wiringx86, jonny-five or many others.

Wiring-x86:

This library is for python programming

This library do same thing like normal GPIO SYSFS. But here ready to us functions which will be helpful for changing values in file so it will be more easy to work with this library.

But still some thing we can't do with library, like running a servo. Because for running servo we need to generate PWM signal of 20 ms period and 0-2 ms duty cycle. Library accepts PWM signal values in nano seconds range.PWM signal values in range of millisecond

seconds exceeds the maximum range of input values.

So, in this case this library fails to woks. But still, for prforming small taskes like running leds or any other GPIO process we can use this library.

This library woks with,

- Intel Galileo Board
- Intel Galileo Board Gen 2
- Intel Edison Board

Here are some functions which will be use to interfaceing with GPIO:

- GPIO()
  - It's a constructer, From it we can create object of GPIO class it accepts one argument "debug=boolean" means if debug=True, it will be run in debuging mode mean you can get descrption of every task and log. And if it False it will work in normal mode
  - Example:

```
from wiringx86 import GPIOEdison as GPIO
gpio = GPIO()
  #or
gpio = GPIO(debug=False)
```

- gpio.analogRead()
  - - here gpio is object. This function is used to read data from anaog pins. It accepts one argument which is the anlog pin number on board. And it's returns digital representation with 10 bits resolution. Means 0 to 1023

- Example:

```
value = gpio.analogRead(analogpin)
```

- gpio.analogWrite()
  - This function is used to write analog data on pins. But you can't do that right!!! So, this function used for writing value in PWM pins. It accepts two perameters one is pin number and second is value. Value will be between 0 to 255. Means at 255 complete signal will be gone means always 1 and at 0 always 0.
- gpio.cleanup()
  - Do a general cleanup. Close all open handlers for reading and writing. Unexport all exported GPIO pins and unexport all exported. Calling this function is not mandatory but it's recommended once you are finished using the library and if it is beingused with a larger application that runs for a long period of time.
  - Example:

```
gpio.cleanup()
```

- gpio.digitalRead()
  - Read the state of digital pin meand itwill be used totakeinput from digital pin. It accepts one aguments which is pin number.
  - Example:

```
state = gpio.digitalRead(pin)
```

- gpio.digitalWrite()
  - this is used to give output from digital pin. It accepts two arguments one is pin number and second is state (states are as shown earlier in SYSFS.)
  - Example:

```
gpio.digitalWrite(13, gpio.HIGH)
```

- gpio.pinMode()
  - this is used for setting the pin mode. Pin modes are shown below,

        * OUTPUT: pin used as output. Use to write into it.
        * INPUT: pin used as input (high impedance). Use to read from it.
        * INPUT_PULLUP: pin used as input (pull up resistor). Use to read from it.
        * INPUT_PULLDOWN: pin used as input (pull down resistor). Use to read from it.
        * ANALOG_INPUT: pin used as analog input (ADC).
        * PWM: pin used as analog output (PWM).
    - This function accepts two arguments one is pin number and another is string specified pin mode
    - Example:

```
gpio.pinMode(pin, gpio.OUTPUT)
```

- gpio.setPWMPeriod()
    - this is used for setting the PWM period. And it accepts two argument one is pin and another is PWM period value

Still this library is not suitable for servo. Because, basically it uses SYSFS GPIO system and there we can not set 20 ms as time period of PWL signal so we cannot drive servo using this library

Wiring-x86:

This library is not based on SYSFS. It works independantly so it is very good and easy to use. This library supports many boards like Galileo Gen1 and Gen2, Edison, Raspberry Pi, Banana Pi and many more.

We will keep the focus on Galileo board.

Installation:

Normally it's comes with OS. But still if you want to install,

```
1 echo"srcmraa-upm http://iotdk.intel.com/repos/3.0/intelgalactic/
    opkg/i586"> /etc/opkg/mraa-upm.conf
2 opkg update
3 opkg install mraa
```

use this command on board with internet connection on board

this library is avilable for many plateforms like python, c/c++, node.js and java.

- Analog Input:

```
1 x = mraa.Aio(0)
2 print (x.read())
3 print ("%.5f" % x.readFloat())
```

here mraa.Aio() gives the object of analog pin '0'. Using that object you can read the valus of that analog pin.

Using this x.read() you can read value in integer. But using x.readFloat() function you can read values more precisely.

- Digital Output:

```
1 x = mraa.Gpio(8)
2 x.dir(mraa.DIR_OUT)
3 x.write(1)
```

here mraa.Gpio(8) gives object of digital pin '8'. Using that object you can write high or low voltage on pin number '8'

and x.dir() specifies the direction means input or output. mraa.DIR_OUT specifies output and mraa.DIR_IN specifies input. And x.write() writes the values on that pin.

- Digital Input:

```
x=mraa.Gpio(8)
x.dir(mraa.DIR_IN)
a = x.read()
```

this way we can read from digital pin 8. And returns 1 or 0

- PWM:

```
x = mraa.Pwm(3)
x.period_us(1000)
x.enabe(True)
x.write(.5)
```

here mraa.Pwm(3) specifies pwm pin and using period_us() specifies the period of pwm signal and input is in micro second. There is period_ms() is also available for input in milliseconds. And using enable(True) starts sending pwm signal and with enable(False).

And write() method use to specify duty cycle. The input in in range of 0 to 1 1 secifies 100% duty cycle and 0 specifies 0% duty cycle.

- UART: Reciver:

```
u = mraa.Uart(0)
u.setBaudRate(9600)
u.setMode(8, mraa.UART_PARITY_NONE, 1)
while True:
  if u.dataAvailable():
    data_byte = u.readStr(1)
    print(data_byte)
```

here mraa.Uart(0) gives uart object and 0 is init perameter. And using setBaudRate() method we can set baudrate. And using setMode() method have four perameters in sequnce:

  - The UART context
  - data bits
  - Parity bit setting
  - stop bits

if you do not specifies this method it will take default value which is suitable for most of the cases.

And method dataAvailabe() we can get knowledge weather more bits available or not. And using readStr() method we can get content in string data type.

Sender:

```
u = mraa.Uart(0)
u.setBaudRate(9600)
u.setMode(8, mraa.UART_PARITY_NONE, 1)
msg_b = bytearray("Hello, mraa byte array!", "ascii")
u.write(msg_b)
u.flush()
msg = "Hello, mraa byte array!"
u.writeStr(msg)
u.flush()
```

here mraa.Uart(0) gives uart object and 0 is init perameter. And using setBaudRate() method we can set baudrate. And using setMode() method have four perameters in sequnce:

- The UART context
- data bits
- Parity bit setting
- stop bits

here u.write() send data but it accepts input only in byte array. But here writeStr() method is also avilabe to dirct input into string.

5 Prerequisites:

Experiment 1: Basic example to use Galileo board without loading any OS:

This experiments shows how to start led using arduino IDE:
First follow this steps:

- select board
- select port
- connect board with laptop

```
/*
  Blink
  Turns on an LED on for one second, then off for one   second,
    repeatedly on ppin nuumber 13
*/
void setup() {
  pinMode(13, OUTPUT);      //defining the pin as output
}
void loop() {
  digitalWrite(13, HIGH);     //writing value HIGH
  delay(1000);          //delay of one second
  digitalWrite(13, LOW);      //writeing low value
  delay(1000)         //delay of one second
}
```

Connection: See figure: 12

Experiment 2: Starting LED using SYSFS interface of Linux OS of Galileo board:

This experiments shows how to start led using SYSFS

```
root@Galileo:~#echo -n "39" > /sys/class/gpio/export
root@Galileo:~#echo -n "out"> /sys/class/gpio/gpio39/direction
```

select 13 pin as output

```
root@Galileo:~#echo -n "strong" >/sys/class/gpio/gpio39/drive
```

set intensity

```
root@Galileo:~#echo -n "1" > /sys/class/gpio/gpio39/value
```

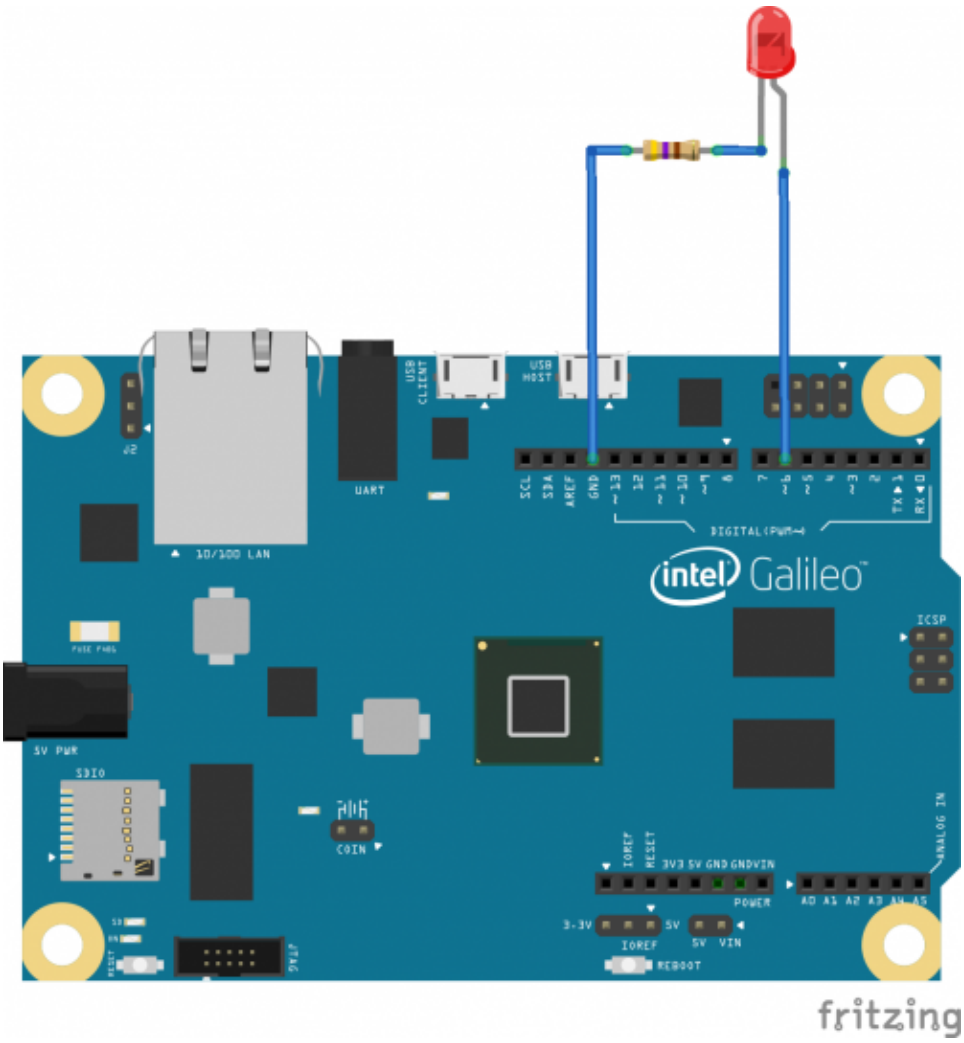

Figure 12: LED connection

write value

Connection: See figure: 12 **Experiment 3: Starting LED**

using wiring-x86 library:

This experiments shows how to start led for 3 second using wiring-x86 library

```
from wiringx86 import GPIOGalileo as GPIO
import time
gpio = GPIO(debug=False)      #create object with debug = false
    so  no log will be displayed
gpio.pinMode(13, gpio.OUTPUT)   #set pin 13 as output
gpio.digitalWrite(13, gpio.HIGH)    #write 1 in pin 13
time.sleep(1)          #delay of one second
gpio.digitalWrite(13,gpio.LOW)    #write 0 in pin 13
gpio.cleanup()           #undefine all the pins
```

Connection: See figure: 12

Experiment 4: Starting LED using MRAA library:

This experiments shows how to start led for 3 second using wiring-x86 library

```
import mraa
x = mraa.Gpio(13)     #get objetc of pin 13
x.dir(mraa.DIR_OUT)     #set as output pin
x.write(1)         #set value HIGH
time.sleep(1)       #delay of 1 second
x.write(0)         #set value low
```

Connection: See figure: 12

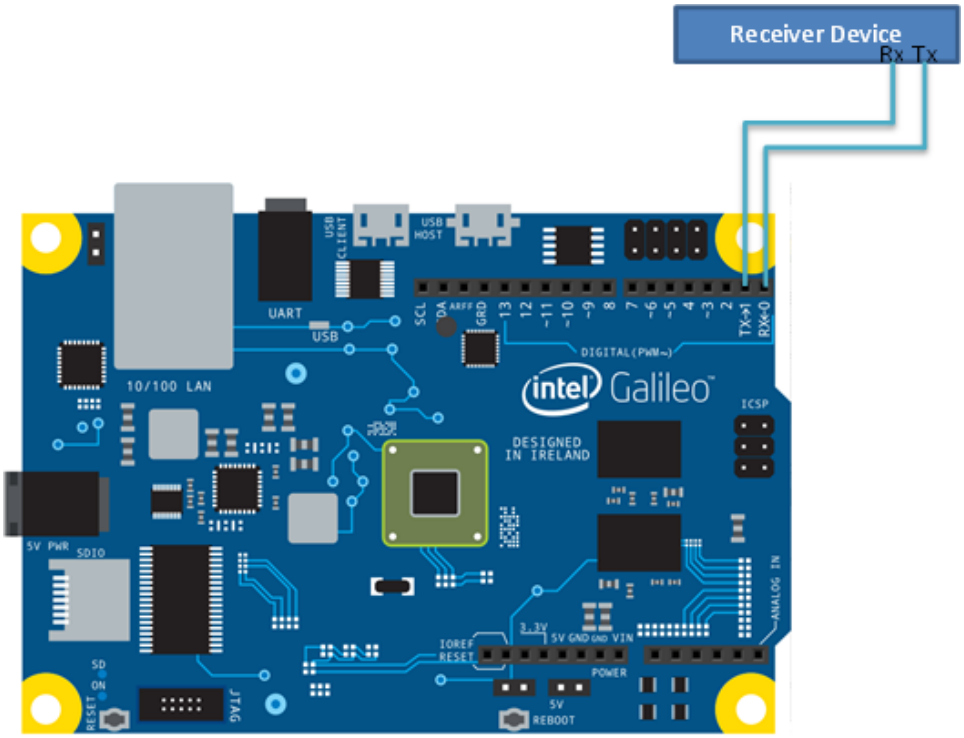

Figure 13: UART connection

Experiment 5: Send data on UART:

This experiment shows how to send data on UART port.

```
import mraa
import sys
sys.stdout.write("Initializing UART...")
u=mraa.Uart(0)              #get object of uart
u.setBaudRate(115200)         #set baudrate
msg_b = bytearray("Hello, mraa byte array!", "ascii") #convert
    string to byte array
u.write(msg_b)             #send data
u.flush()
msg_s = "Hello, mraa string!"
u.writeStr(msg_s)       #send direct to string
```

Connection: See figure: 13

Experiment 6: Receive data from UART:

This experiment shows how to send data on UART port.

```
u=mraa.Uart(0)
u.setBaudRate(115200)
while True:                                #always keep watching if
    data is avilable
if u.dataAvailable():
    data_byte = u.readStr(1)        #read data if avilable
    print(data_byte)
```

Connection: See figure: 13

Experiment 7: Generate PWM using mraa library:

This experiments shows how to generate PWM signal.

```
import mraa
import time
x = mraa.Pwm(3)     #select pwm pin
x.period_ms(20)     #set time period
x.enable(True)      #start signaling
x.write(0.05)     #write duty cycle
```

Connection: See figure: 14

Experiment 8: Run servo using mraa library:

```
import mraa
import time
pwm = mraa.Pwm(10)
pwm.period_us(4640)       #here we are using 4640us instad of 20
    ms. Just find from trial and error
pwm.enable(True)
def turn_servo(angle):
    pwm_value = float(angle)*((.26+0.070)/180)+0.070      #
    mapping angle with duty cycle.
  print pwm_value
    pwm.write(pwm_value)
def main():
  turn_servo(0)
    while (True):
        input = raw_input("enter angle between 0 to 180: ")
```

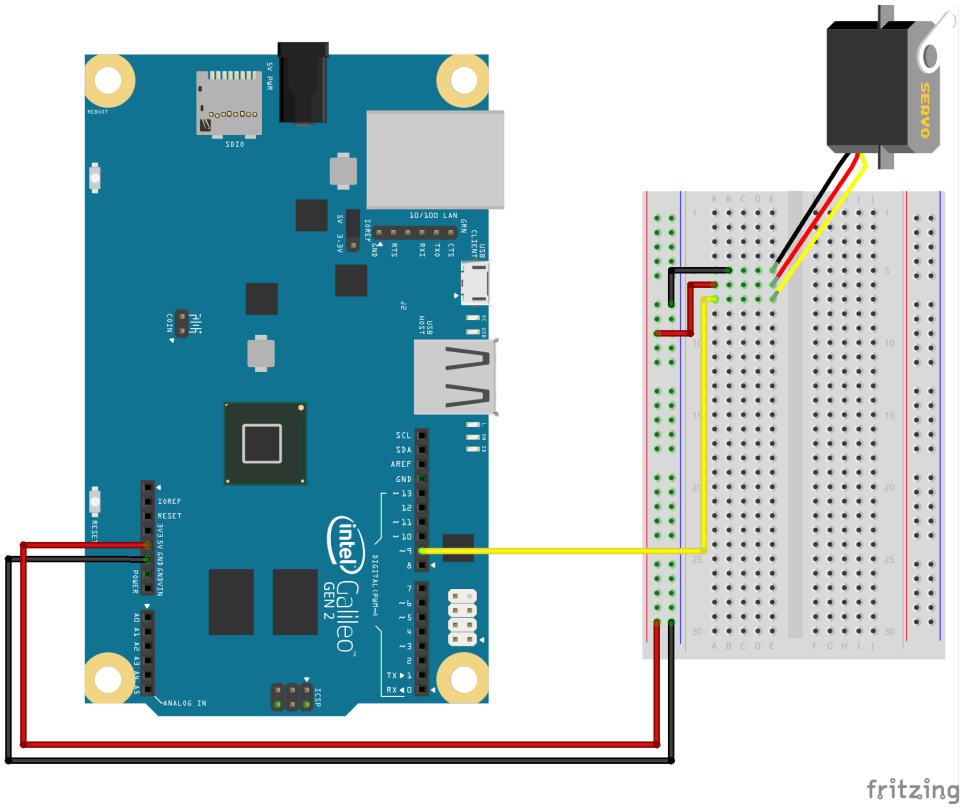

Figure 14: servo connection

```
14          turn_servo(input)
15 if __name__ == "__main__":
16     main()
```

Connection: See figure: 14

6 References

- http://www.malinov.com/Home/sergey-s-blog/intelgalileo-programminggpiofromlinux
- https://makernotes.wordpress.com/tag/galileo/
- https://github.com/intel-iot-devkit/mraa/
- http://iotdk.intel.com/docs/master/mraa/
- https://github.com/emutex/wiring-x86
- https://learn.sparkfun.com/tutorials/galileo-getting-started-guide
- http://stackoverflow.com/questions/26689434/control-servo-with-galileo-mraa-for-node