```
// Author : Jose Goncalves
// jose.braga.pt@gmail.com

// Teacher : Nuno Peixoto
// University : IPCA
// 17.03.2014

// Classe Gpio
// BeagleBoard XM
// Port GPIO5: 32 bits : MSB 128....159 LSB


// Function Gpo(Port,status)
// Port must be between 128...159  (GPIO 5)
// status : Output value , have to be 0 : OFF or 1 : ON

// Function Gpi(Port)
// Port must be between 128...159  (GPIO 5)
// return value of the input ( 0 or 1 ), return value 2 if error


// #define GPIO_ENABLE   0x00000C00
// #define GPIO_DISABLE  0xFFFFF3FF

#include "Gpio.h"


// ---------------------------------------  Construtor  ---------------------------------------------

Gpio::Gpio() {

                // constructor, tem de ter o mesmo nome que a class

                // declarações não necessárias, definidas em Gpio.h :
                // int fd;
                // volatile ulong *gpio;

                fd = open("/dev/mem", O_RDWR | O_SYNC);
                if (fd < 0)  printf("Could not open memory\n");
                if (fd > 0)  printf("memory map open successfully, file descriptor : %d\n",fd);

                gpio = (ulong*) mmap(NULL, 0x10000, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0x49050000);
                if (gpio == MAP_FAILED) {
                                         printf("Gpio Mapping failed\n");
                                         close(fd);
                                        }
            }

// ---------------------------------------- Método Gpo ----------------------------------------------

void Gpio::Gpo(int Port,int Status)
    {

       port = Port;
       status = Status ;

       // GPIO5 : 32 bits
       // MSB 159.........128 LSB
       port = 0xFFFFFFFF & ( ~(1<< (Port-128))) ;

       // Configure Port as ouptut
       // bit=0 -> output   bit=1 -> input

       // GPIO_OE Output Enable  Address Offset : 0x034
       gpio[0x6034/4] = gpio[0x6034/4] & port ;

       if ( status == 0 )
           {
           // GPIO DataOutput   Address Offset : 0x03C
           gpio[0x603C/4] = gpio[0x603C/4] & port ;
           }

       if ( status == 1 )
           {
           gpio[0x603C/4] = gpio[0x603C/4] | ~port ;
           }
    }


// ---------------------------------------- Método Gpi ----------------------------------------------

int Gpio::Gpi(int Port)
    {
       int input = 2 ;
       port = Port;

       // GPIO5 : 32 bits
       // MSB 159.........128 LSB
       // reset bit to enable GPIO pin / GPIO5
       port = 0xFFFFFFFF & ( ~(1<< (Port-128))) ;

       // Configure Port as input
       // bit=0 -> output   bit=1 -> input

       // GPIO_OE Input Enable , Address Offset : 0x034
       gpio[0x6034/4] = gpio[0x6034/4] | ~port ;

       // GPIO DataInput   Address Offset : 0x038

       if ( (gpio[0x6038/4] = gpio[0x6038/4] & ~port) == 0 )
              input = 0;
```

```
       if ( (gpio[0x6038/4] = gpio[0x6038/4] & ~port) == (1<<(Port-128)) )
              input = 1;

       return (input);
    }


// ---------------------------------------- Destrutor ----------------------------------------------------

Gpio::~Gpio()
    {
      close(fd);
    }
```