Embedded Systems
General-purpose input/output
(introduction to laboratory)

Krzysztof Czarnecki
Department of Marine Electronic Systems
Faculty of Electronics, Telecommunications and Informatics
Gdansk University of Technology
2015
General-purpose input/output = GPIO

WARNINGS !!!

Do not connect any wire to pins labeled by
DNC (= Do Not Connect).
Pins labeled in this manner are usually reserved for internal functions of a micro-controller. The connection of any wire may cause a device failure.

The direct connection (short) of two random pins on the supplied board can damage the entire system. The damage may be irreversible. A high degree of caution during the laboratory and classroom project is required.
Some internal components of the board can be supplied by the voltage equal to 3.3 V or 5 V depending on the platform. Intel Galileo and Raspberry PI will be used during the laboratory and classroom project. Each student should make sure each time what kind of the power supply is required. Otherwise the equipment failure is possible.
General-purpose input/output = GPIO as a peripheral system
General-purpose input/output = GPIO

as a peripheral system

GPIO ports usually consist of several pins. They play a role of an interface between the system and the outside world.

GPIO pins configurations can be different for different devices.

The pins are normally spaced about 2.54 mm (0.1 inches) - is a standard in the world of electronics.

Additionally in the universal GPIO ports can also be the power pins (standard 3.3 and 5 V), the mass and plugs for a variety of specific communication buses.

Some of the pins together are circuits and can be configured for specific communication interfaces.
General-purpose input/output = GPIO
as a serial interface

- **UART serial bus** (Universal Asynchronous Receiver and Transmitter) – in addition to the power and ground pins, two additional GPIO pins one to send and one to receive information are required. The bus speed is configurable – in systems used in the classroom is 115 200 b / s.

- **I²C serial computer bus** (Inter-Integrated Circuit) has been designed for communication between integrated circuits. Its configuration, in addition to power and ground, uses two additional GPIO pins: first to the data transmission, second to the synchronization. One bus allows you to connect multiple devices (multi-master mode).
General-purpose input/output = GPIO as a serial interface

▶ SPI bus (Serial Peripheral Interface) – is used for short distance communication, primarily in embedded systems. The SPI bus uses 4 GPIO pins / logic signals: Serial Clock (output from master), Master Output, Slave Input (output from master), Master Input, Slave Output (output from slave), and Slave Select (active low, output from master). A one physical bus enables communication between multiple devices and integrated circuits. Synchronous Serial Bus was originally designed for programming micro-controllers and other devices in the ISP mode (In-System Programming).

▶ One-wire bus Configuration of the bus is possible using only one output for data transmission and the mass as a reference. The communication is realized in the master-slave mode. It is possible to connect multiple devices (typically sensors) per a single bus.
General-purpose input/output = GPIO characteristics

GPIO Pins:

- can be configured both as inputs and outputs,
- can be pre-connected to the LEDs or buttons on the board,
- the current in the pins must not exceed a dozen mA,
- it is assumed that the current in GPIO pins can at most control one LED,
- usually they do not have additional protection against short circuits or over-voltage,
- selected pins, indicated by an PWM, have the ability to emit a signal with the pulse width modulation – they can be used to control devices by infinitely adjustable inputs.
General-purpose input/output = GPIO
pulse-width modulation (PWM)
General-purpose input/output = GPIO

connection of a LED
General-purpose input/output = GPIO

limit the current

Task: please calculate value of limit resistor.

- Let’s assume that we want connect a red LED to pin powered by voltage equal to 3.3 V.
- Conduction current is 20 mA and voltage 1.7 V.
- The LED and resistor are connected in series, and therefore the voltage across the resistor should reach $3.3 - 1.7 = 1.6$ V.
- Then from Ohm’s law, we know that resistance can be expressed by the ratio: voltage to current, hence the resistance of the resistor should be: $1.6/0.02 = 80\Omega$.
- Thus designated resistance can rarely be matched to typical resistors that available in stores. Selection of resistor with the lowest available resistance value, but at the same time greater than the specified value is recommended.
General-purpose input/output = GPIO

limit the current
General-purpose input/output = GPIO

the role of an operating system

Why involve the operating system with the Linux kernel to handle such a simple interface as the GPIO?

It is not always justified, but it has some advantages:

▶ The operating system provides hardware abstraction for application. This allows to port an application to another platforms with different micro-controller or processor without changing it.

▶ Using the operating system allows you to use ready-made tools and libraries for building applications such as high-level: web servers, file servers, libraries with advanced signal processing algorithms, FFT, or graphical user interfaces for example.
General-purpose input/output = GPIO
the configuration of an operating system

Support of GPIO ports, LEDs or communication buses can be turned on during compilation of the kernel of an operating system. Please read the following configuration sections:

▶ GPIO Support
▶ LED Support
▶ I2C Support
▶ SPI Support
▶ Pin controllers
▶ itp.
General-purpose input/output = GPIO
control of the GPIO ports

kernel configuration and compilation by
make menuconfig:

Device Drivers --->
  -*- GPIO Support --->
    [*] /sys/class/gpio/... (sysfs interface)
General-purpose input/output = GPIO
everything is a file in the Linux

GPIO ports can be controlled directly from the console.

All available GPIO pins can be represented in the /sys/class/gpio/ directory by the relevant files.

Used GPIO pins should be transferred from the kernel space to the user’s space. It can be done by writing a pin number into the export file.

This will create a directory that represents this pin e.g.:
echo 12 > /sys/class/gpio/export

We should obtain gpio12 as follows:
lsof /sys/class/gpio/gpio12

In order to relieve, its number should be write into the unexport file.
echo 12 > /sys/class/gpio/unexport
General-purpose input/output = GPIO
everything is a file in the Linux

In the new directory we can find two following files: direction and value.

The /sys/class/gpio/gpio12/direction file is used to configure the pin. If the pin should play a role of output or input, respectively out or in should be write inside:

```
echo out > /sys/class/gpio/gpio12/direction
echo in > /sys/class/gpio/gpio12/direction
```

If the pin is an input, its value can be read by using the value file as follows:

```
cat /sys/class/gpio/gpio12/value
```

Analogously if the pin is an output, we can send a signal by writing a value following:

```
echo 1 > /sys/class/gpio/gpio12/value
echo 0 > /sys/class/gpio/gpio12/value
```
General-purpose input/output = GPIO
everything is a file in Linux (LEDs)

All available built-in LEDs are represented by appropriate files in `/sys/class/leds/` directory.

The following request should list all of them: `ls -l /sys/class/leds/`

The brightness of each LED can be set by writing a binary value (0 or 1) to the representing file. If a considered LED is linked with PWM output then 8-bits integer value (0-255) can be written and the brightness can be changed comparatively continuously.

```
echo 1 > /sys/class/leds/led2/brightness (switch on)
echo 0 > /sys/class/leds/led2/brightness (switch off)
cat /sys/class/leds/led2/brightness (print value)
```
General-purpose input/output = GPIO
everything is a file in Linux

#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

#define LED2 "/sys/class/leds/led2/brightness"
#define LED3 "/sys/class/leds/led2/brightness"

int main(int argc, char **argv){

    int fd_led2, fd_led3;

    if ((fd_led2 = open(LED2, O_RDWR)) < 0){
        fprintf(stderr, "Error in open %s", LED2);
        return -1;
    }

    if ((fd_led3 = open(LED3, O_RDWR)) < 0){
        fprintf(stderr, "Error in open %s", LED3);
        return -1;
    }

}
General-purpose input/output = GPIO
everything is a file in Linux

```c
char data_0 = '0';
char data_1 = '1';

for(;;){
    write(fd_led2, &data_1, sizeof(char));
    write(fd_led3, &data_0, sizeof(char));
    usleep(100000);

    write(fd_led2, &data_0, sizeof(char));
    write(fd_led3, &data_1, sizeof(char));
    usleep(100000);
}

return 0;
```
General-purpose input/output = GPIO

everything is a file in Linux

Functions in C language which allow file manipulation:

<table>
<thead>
<tr>
<th>low-level</th>
<th>high-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>fopen</td>
</tr>
<tr>
<td>close</td>
<td>fclose</td>
</tr>
<tr>
<td>read</td>
<td>fread</td>
</tr>
<tr>
<td>write</td>
<td>fwrite</td>
</tr>
</tbody>
</table>

- The low-level interface allow to call directly to the kernel. It means faster single operations but also higher load.

- The high-level functions use an additional software buffer in the user space that optimizes the number of requests to the hardware and globally reduces the load.
General-purpose input/output = GPIO

cross compilation on ARM

Makefile example

CROSS=arm-none-linux-gnueabi-
CFLAGS=-Wall

leds: leds.c
    $(CROSS)gcc -o leds leds.c $(CFLAGS)

clean:
    @rm -vf leds *.o
General-purpose input/output = GPIO
pins control by the physical memory

The `/dev/mem` file represents the whole physical memory of the board directly. It gives access to all internal registers and peripheral devices. However this solution is not recommended because:

- we lose the benefits of the use of the operating system: hardware abstraction, memory protection, debugging system, application portability, etc.

- a relatively small mistake can lead to failure of the entire system.

Please do not use this way in the laboratory!
General-purpose input/output = GPIO

Literature

- Intel Galileo Board User Guide
- Raspberry Pi User Guide
- Linux podstawy i aplikacje dla systemów embedded, (in Polish) Łukasz Skalski.