F28HS Hardware-Software Interface: Systems Programming

Hans-Wolfgang Loidl

School of Mathematical and Computer Sciences,
Heriot-Watt University, Edinburgh

Semester 2 2016/17

[0]No proprietary software has been used in producing these slides

Outline

HERIOT WATT UNIVERSITY

Tutorial 3: Programming a Button input device

- In this tutorial we want to use a button, connected through a breadboard as an input device.
- This is the simplest input device that we will cover.
- The code needed to control is typical for such devices.
- This tutorial deals with **programming a button as input device.**

Core Techniques

- In the LED tutorial, we have seen that we first need to **identify the registers** that give control to the device.
- For that we will again look into the BCM Peripherals documentation.
- We will then go through a simple example of
  - reading button input data,
  - blinking an LED on button press.
- We want to connect the button with **pin 24**, using a breadboard.
- These simple activities, will also be at the core of CW2.

HERIOT WATT UNIVERSITY

GPIO Register Assignment

Registers

13
14

	GPLEV0	GPIO Pin Level 0	32	R
	GPLEV1	GPIO Pin Level 1	32	R
0x 7E20 003C	-	Reserved	-	-
0x 7E20 0040	GPEDS0	GPIO Pin Event Detect Status 0	32	R/W
0x 7E20 0044	GPEDS1	GPIO Pin Event Detect Status 1	32	R/W
0x 7E20 0048	-	Reserved	-	-
0x 7E20 004C	GPREN0	GPIO Pin Rising Edge Detect Enable 0	32	R/W
0x 7E20 0050	GPREN1	GPIO Pin Rising Edge Detect Enable 1	32	R/W
0x 7E20 0054	-	Reserved	-	-
0x 7E20 0058	GPFEN0	GPIO Pin Falling Edge Detect Enable 0	32	R/W
0x 7E20 005C	GPFEN1	GPIO Pin Falling Edge Detect Enable 1	32	R/W

[0] See BCM Peripherals Manual, Chapter 6, Table 6.1

HERIOT WATT UNIVERSITY

GPIO Register Assignment

Registers

13
14

	GPLEV0	GPIO Pin Level 0	32	R
	GPLEV1	GPIO Pin Level 1	32	R
0x 7E20 003C	-	Reserved	-	-
0x 7E20 0040	GPEDS0	GPIO Pin Event Detect Status 0	32	R/W
0x 7E20 0044	GPEDS1	GPIO Pin Event Detect Status 1	32	R/W
0x 7E20 0048	-	Reserved	-	-
0x 7E20 004C	GPREN0	GPIO Pin Rising Edge Detect Enable 0	32	R/W
0x 7E20 0050	GPREN1	GPIO Pin Rising Edge Detect Enable 1	32	R/W
0x 7E20 0054	-	Reserved	-	-
0x 7E20 0058	GPFEN0	GPIO Pin Falling Edge Detect Enable 0	32	R/W
0x 7E20 005C	GPFEN1	GPIO Pin Falling Edge Detect Enable 1	32	R/W

[0] See BCM Peripherals Manual, Chapter 6, Table 6.1

BCM2835 GPIO Peripherals

Base adress: 0x3F000000

0	GPFSEL	Pins 0-9	(3-bits per pin)
5		Pins 50-53	
7	GPSET	Pins 0-31	(1-bit per pin)
8		Pins 32-53	
10	GPCLR	Pins 0-31	(1-bit per pin)
11		Pins 32-53	
13	GPLEV	Pins 0-31	(1-bit per pin)
14		Pins 32-53	
...			

The main registers that we need in this case are (see p90ff of BCM2835 ARM peripherals):

- **GPFSEL**: function select registers (3 bits per pin); set it to 0 for input, 1 for output; 6 more alternate functions available
- **GPSET**: **set** the corresponding pin
- **GPCLR**: **clear** the corresponding pin
- **GPLEV**: return the **value** of the corresponding pin

Define the button pin as an INPUT device

Write into these bits (12–14) to set the **function** for `pin 24`

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
0:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
1:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
2:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	1		2	11	10	9	8
3:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
4:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
5:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
6:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
7:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
8:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
9:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
10:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
11:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
12:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8

pin 24

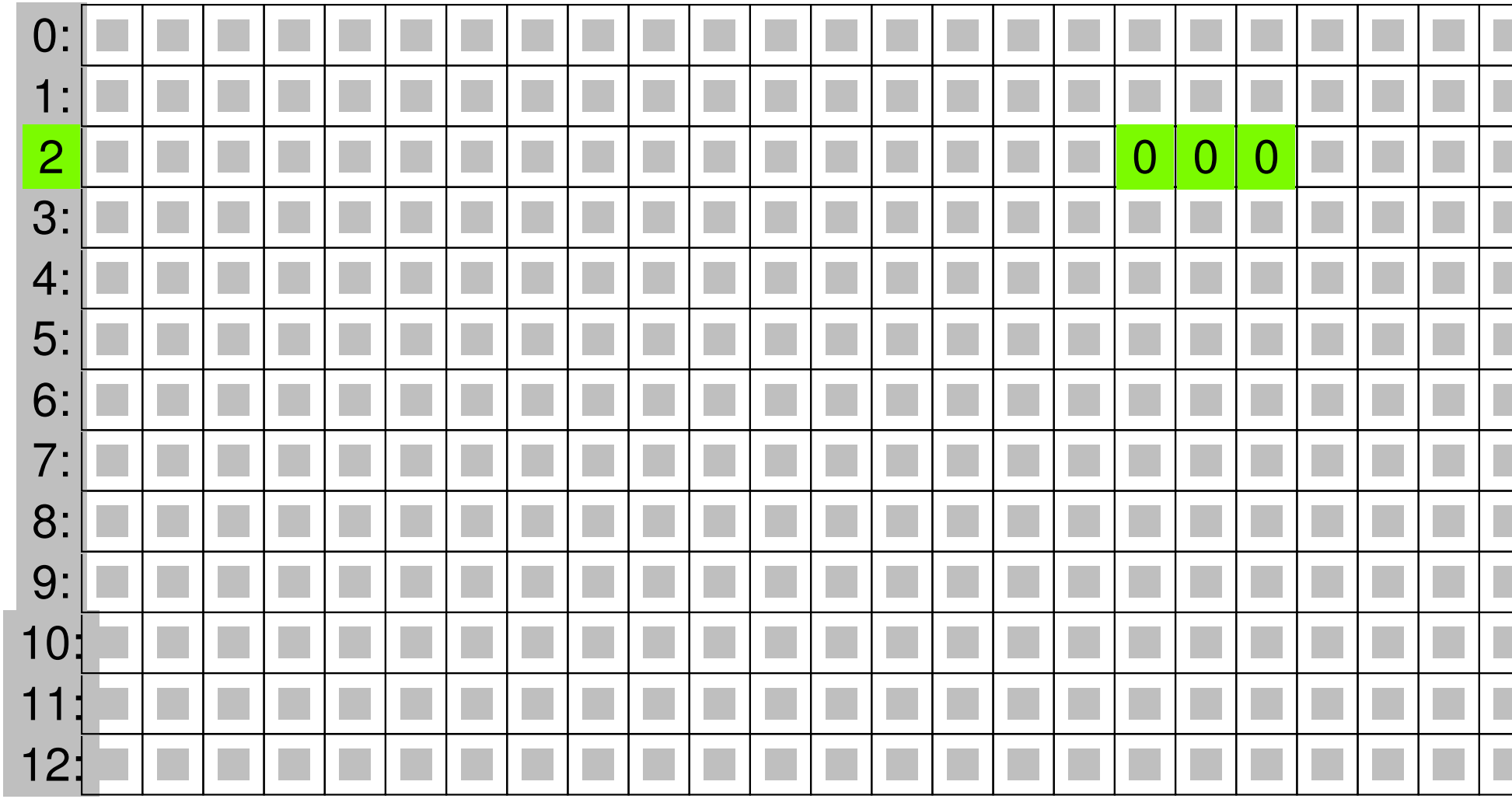
Define the button pin as an INPUT device

Write into these bits (12–14) to set the **function** for `pin 24`

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
0:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
1:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
2	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
3:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
4:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
5:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
6:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
7:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
8:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
9:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
10:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
11:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
12:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8

NB: Recall, each GPFSEL register controls 10 pins (1–10, 11–20, etc), for each pin, 3-bits control the behaviour

Define the button pin as an INPUT device



NB: To set a pin to **input** we need to write the value **0** into it.

Reading from the button input

Contents:

Bit positions

GPIO registers (Base address: 0x3F200000)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
4:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
5:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
6:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
7:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
8:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
9:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
10:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
11:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
12:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
GPLEV0	30	29	28	27	26	2	pin 24		3	22	21	20	19	18	17	16	15	14	13	12	11	10	9
14:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
15:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9

Read this bit (24)

HERIOT WATT UNIVERSITY

Reading from the button input

Contents:

Bit positions

GPIO registers (Base address: 0x3F200000)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
4:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
5:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
6:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
7:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
8:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
9:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
10:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
11:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
12:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
13	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
14:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
15:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10

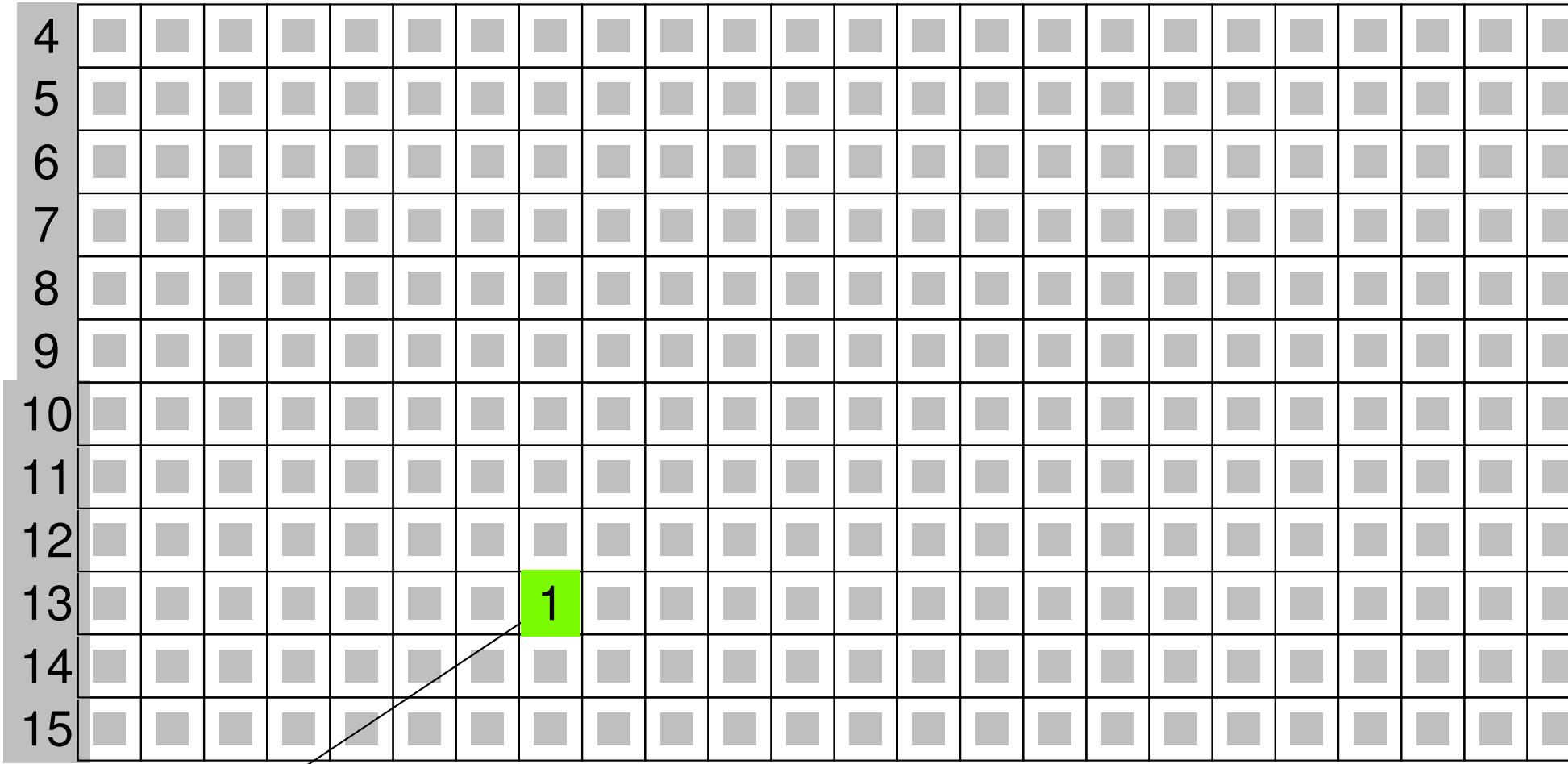
Read from this bit (24)

Reading from the button input

Contents:

Bit values

GPIO registers (Base address: 0x3F200000)



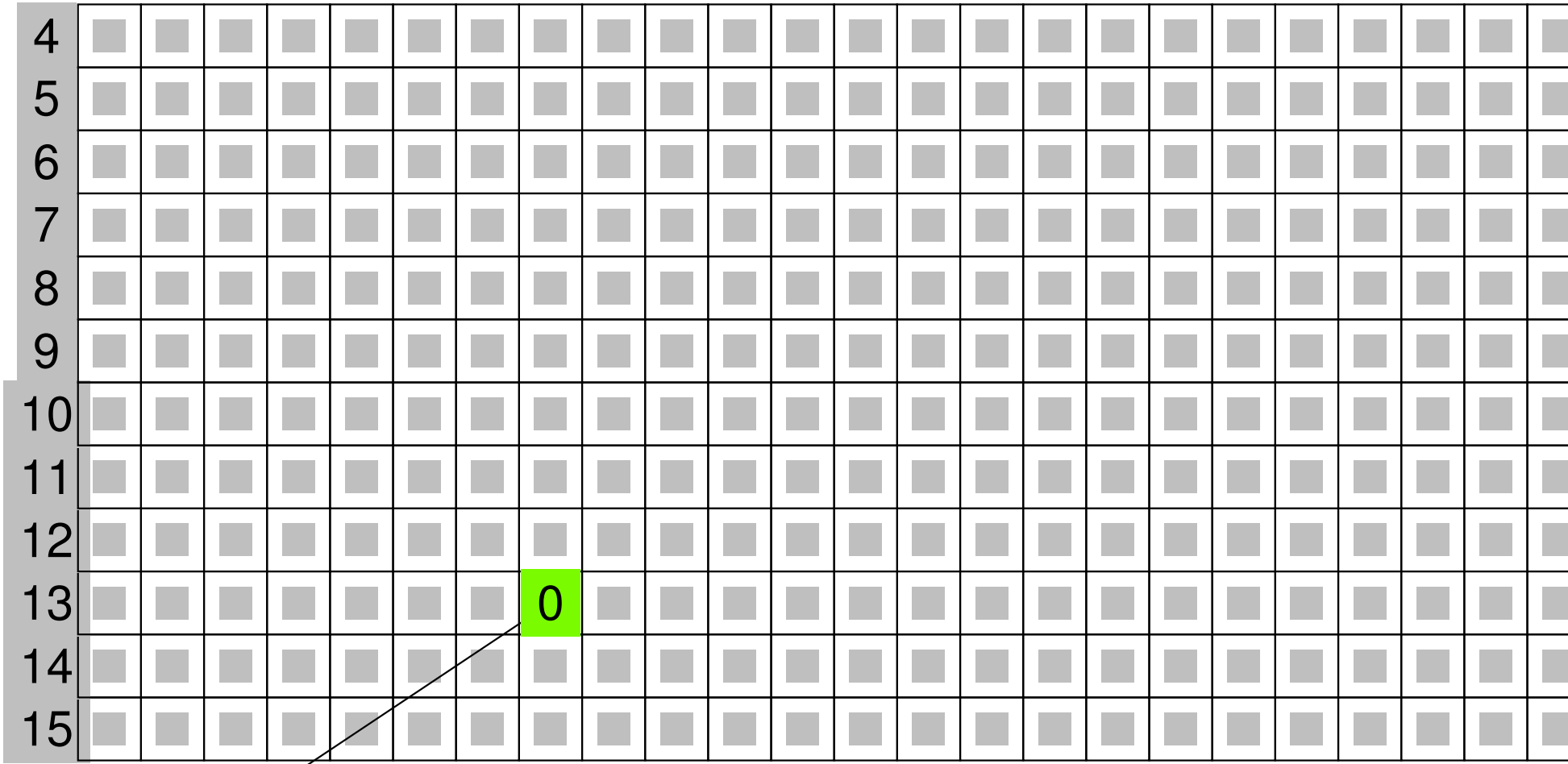
HERIOT WATT UNIVERSITY

Reading from the button input

Contents:

Bit values

GPIO registers (Base address: 0x3F200000)



Input: `LOW`

HERIOT WATT UNIVERSITY

Sample C code: Button input

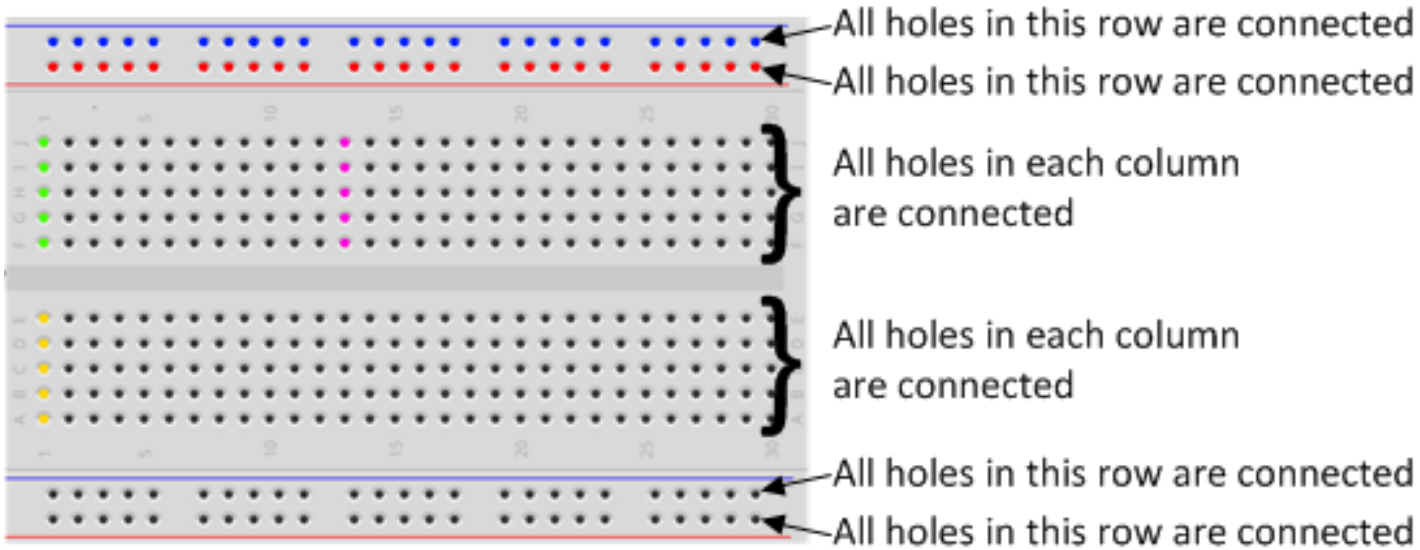
First we define some constants that we will need.

```
// Tunables:
// PINs (based on BCM numbering)
#define LED 23
#define BUTTON 24
// delay for loop iterations (mainly), in ms
#define DELAY 200

#define INPUT                   0
#define OUTPUT                  1

#define LOW                     0
#define HIGH                    1
```

This assumes that we have wired-up the button with GPIO pin 24 and the LED with GPIO pin 23.

Using a breadboard

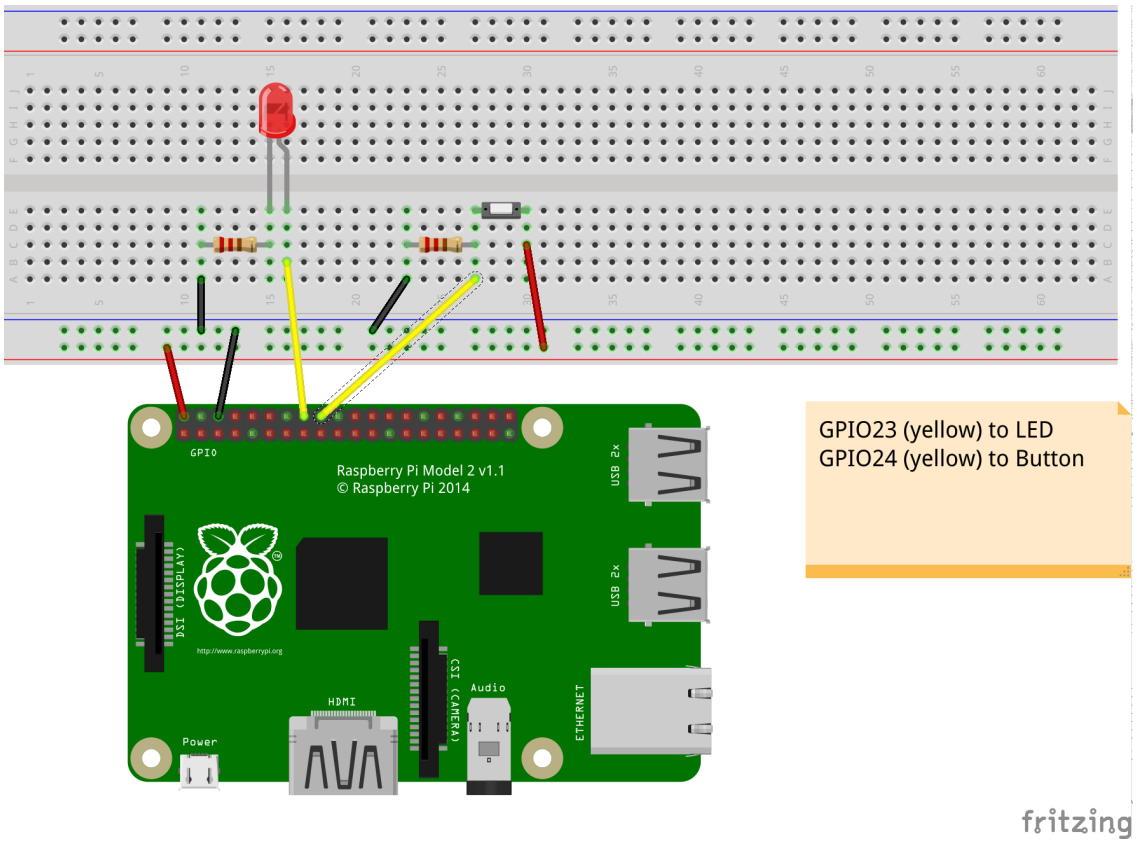
To control an external LED, you could directly connect GPIO pins with the LED and a resistor using jumper cables.
However, a breadboard is a more flexible way of wiring peripherals, such as LEDs or buttons, to the RPi.
You need to understand how the columns and the rows on a breadboard are connected, though.



For a good basic intro on how to use a breadboard follow this link

The wiring as a Fritzing diagram

To describe a specific wiring, we use **Fritzing diagrams** like this:



An **LED**, as output device, is connected to the RPi2 using **GPIO pin 23.**

A **button**, as input device, is connected to **GPIO pin 24.**

Sample C code: Button input

We memory-map the addresses for the GPIO registers (as before).

```
gpiobase = 0x3F200000;
// memory mapping
if ((fd = open ("/dev/mem", O_RDWR | O_SYNC |
   O_CLOEXEC) ) < 0)
  return failure (FALSE, "setup: Unable to open /
     dev/mem: %s\n", strerror (errno)) ;
// GPIO:
gpio = (uint32_t *)mmap(0, BLOCK_SIZE, PROT_READ|
   PROT_WRITE, MAP_SHARED, fd, gpiobase) ;
if ((int32_t)gpio == -1)
  return failure (FALSE, "setup: mmap (GPIO)
     failed: %s\n", strerror (errno)) ;
```

Sample C code: Button input

We set the modes for the LED pin (OUTPUT) and the button pin (INPUT).

```
// setting the mode
fSel  = 2; // register 2 (GPFSEL2)
shift = 9; // slot 3 (shift 3*3)
// set the above pin to output mode
*(gpio + fSel) = (*(gpio + fSel) & ~(7 << shift))
   | (1 << shift) ;  // Sets bits to one = output

fSel  = 2; // register 2 (GPFSEL2)
shift = 12; // slot 4 (shift 4*3)
// set the above pin to input mode
*(gpio + fSel) = (*(gpio + fSel) & ~(7 << shift))
   ; // Sets bits to zero = input
```

Sample C code: Button input

Inside the main loop, we first read from the bit associated with the button input in the `GPLEV0` register.

```
for (j=0; j<1000; j++) {
  if ((*(gpio + 13 /* GPLEV0 */) & (1 << (BUTTON &
     31))) != 0)
    theValue = HIGH ;
  else
    theValue = LOW ;
```

Sample C code: Button input

Further down the loop, we write to the bit associated with the LED output in the `GPLCR0` or `GPSET0` register.

```
if (theValue == LOW) {
  clrOff = 10; // GPCLR0 for pin 23
  *(gpio + clrOff) = 1 << (LED & 31); // 23-rd bit
      in the register
} else {
  setOff = 7; // GPSET0 for pin 23
  *(gpio + setOff) = 1 << (LED & 31); // 23-rd bit
      in the register
}
// delay ...
```

Sample C code: Button input

Finally, we want to clean-up by setting the LED to LOW. Which kind of code do we need here?

```
// clean-up by setting the LED pin to LOW
```

Summary

- Reading input from a button works in the same way as writing to the LED:
  - We need to identify the relevant registers and bits for our pin
  - We declare the pin an INPUT device in the `GPFSEL` register
  - We read from the associated bit in the `GPLEV` register to get the input
- With the button you have a basic input device to communicate with the system
- In the CW we will combine a button (for input), an LED (for output) and an LCD display (for nicer output) and implement a small app for this configuration.

See sample source: `tut_button.c`

HERIOT WATT UNIVERSITY