- CO3009 -

- Switches and Buttons -

Introduction

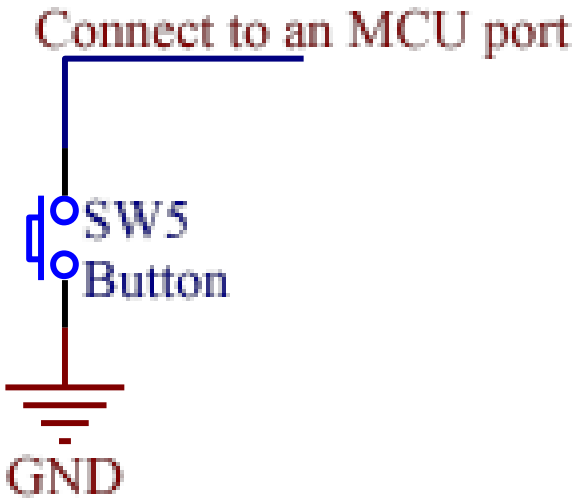
- Embedded systems usually use switches/buttons as part of their user interface.
- This general rule applies from the most basic remote-control system for opening a garage door, right up to the most sophisticated aircraft autopilot system.
- Whatever the system you create, you need to be to create a reliable switch interface.

What we will do in this lecture

- How we can read inputs from mechanical switches in an embedded application

anhpham@hcmut.edu.vn

The need for pull-up resistors



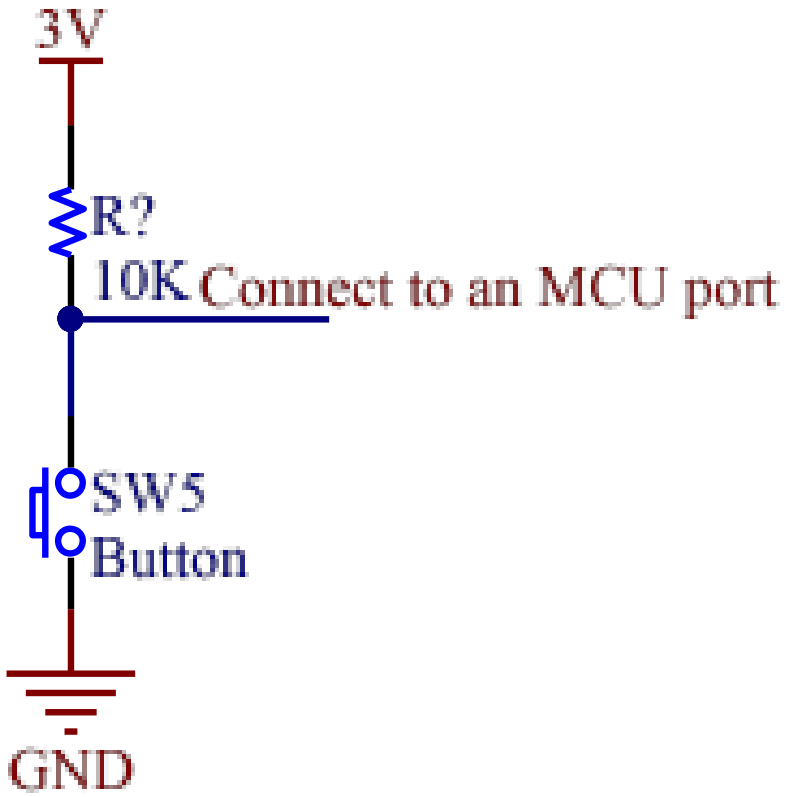
- The hardware operates as follows:
  - When the switch is open, it has no impact on the port pin. An internal resistor on the port 'pull up' the pin to the supply voltage of the MCU (typically 5V or 3V3). If we read the pin, we will see the value '1'
  - When switch is closed (pressed), the pin voltage will be 0V. If we read the pin, we will see the value '0'

The need for pull-up resistors

With pull-ups:

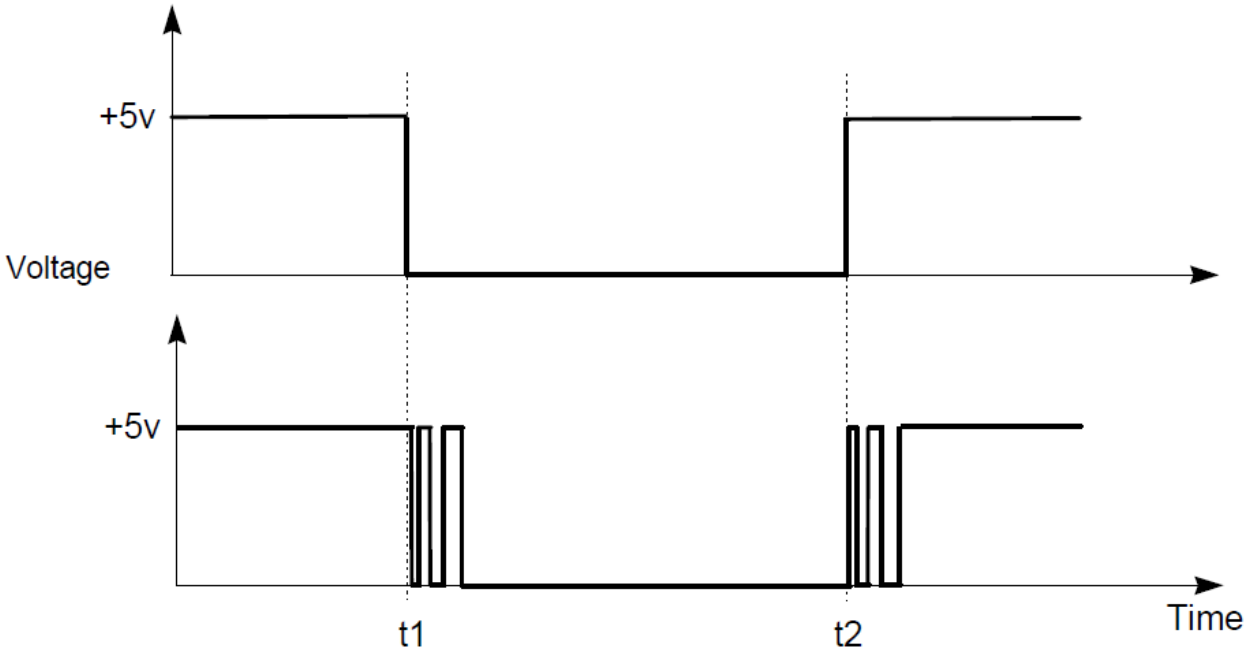
Without pull-ups:

The need for pull-up resistors



Dealing with switch bounce

- In practice, all mechanical switch contacts bounce (that is, turn on and off, repeatedly, for short period of time) after the switch is closed or opened.



Dealing with switch bounce

- As far as the MCU concerns, each “bounce” is equivalent to one press and release of an “ideal” switch.

- Without appropriate software design, this can give several problems
  - Rather than reading ‘A’ from a keypad, we may read ‘AAAAA’
  - Counting the number of times that a switch is pressed becomes extremely difficult.

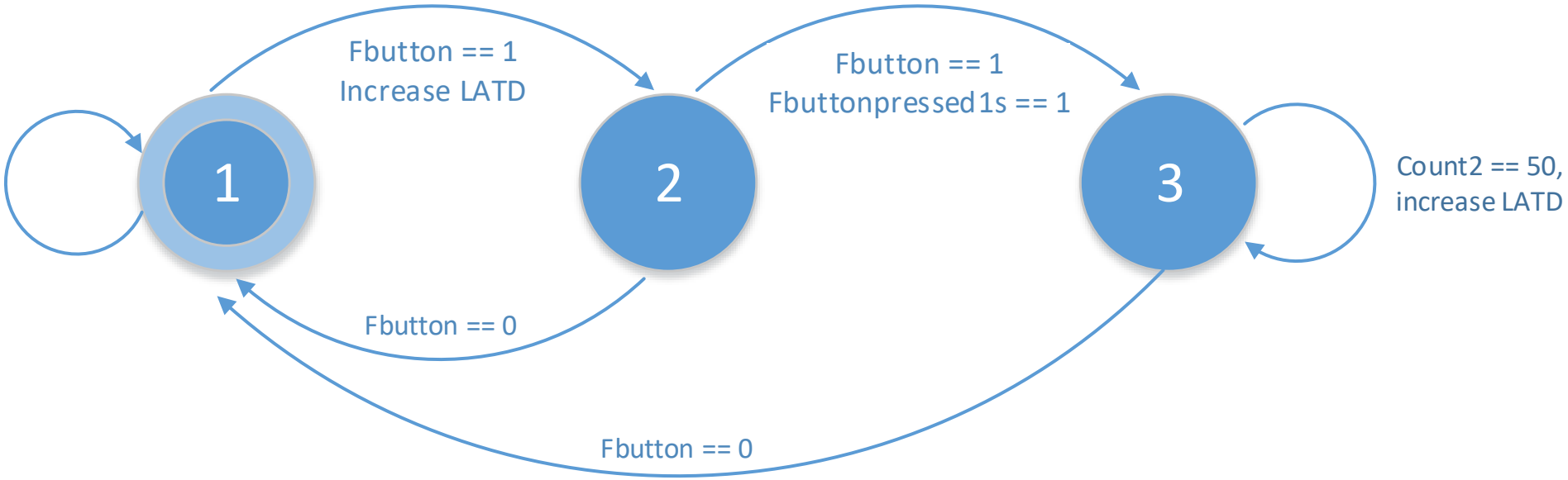
Dealing with switch bounce

- Creating a simple software to check for a valid switch input is straightforward:
  - Read the relevant port pin
  - If we think we have detected a switch depression, we wait for 20ms and then read the pin again
  - If the second reading confirms the first reading, we assume the switch really has been depressed.
- Note that the figure of '20ms' will depend on the switch used.

Round robin with timer interrupts example

- **W**rite a program that
  - Has a timer which has an interrupt in every 10 milliseconds.
  - Reads values of button **RA5** every 10 milliseconds.
  - Increases the value of LEDs connected to **PORTD** when the button **RA5** is pressed.
  - Increases the value of **PORTD** automatically in every 0.5 second, if the button RA5 is pressed in more than 1 second.

Finite State Machine



Example Code

```
void main (void) {
    enum {STATE1, STATE2, STATE3} eState;
    while (TRUE) {
        switch (eState){
          case STATE1:
            if (fbutton) {
                increase LATD;
                eState = STATE2;
            }
          break;
          case STATE2:
            if (fbutton == 0) eState = STATE1;
            else if (fbuttonpress1s) eState = STATE3;
          break;
          case STATE3:
            if (normalCountUpFlag) {
                increase LATD;
                normalCountUpFlag = 0;
             }
            if (fbutton == 0) eState = STATE1;
                break;
          }
        }
}
```

```
void Readbutton(void) {
    firstReadRA5 = secondReadRA5;
    secondReadRA5 = digitalRead(BUTTON_1);
    if (secondReadRA5 == firstReadRA5) {
        if (firstReadRA5) {
            fbutton = 1;
            count1++
            if (count1 >= 100){
                fbuttonpressed1s = 1;
                count2 ++;
                if (count2 >= 50){
                    normalCountUpFlag = 1
                    count2 = 0;
                }
            }
        } else {
            fbutton = 0;
            fbuttonpressed1s = 0;
            count1 = 0;
            count2 = 0;
        }
    }
}
```


Exercise: Counting Goats