MCU Lab 1 - Week 7
MCU: Digital IOs - LEDs and a Pushbutton
[2.5 Marks Total]

This computer lab will introduce a microprocessor, Atmega328P, and its perpherial Digital Input/Outputs (DIOs, also called General Purpose IOs) built-in Atmega328P board. Understanding their key properties and ability to program the devices confidently are the foundation of embedded programming.

1. Objectives

- To install and set up Visual Studio Code (or Vscode, an IDE editor) and PlatformIO software (a cross-compiler for Atmega series) on your laptop, which is essential to complete the lab tasks and the MCU Project.
- To setup the DIO pins in output and input modes, and to use the internal pull-up resistors in an input mode.
- To control two LEDs using a pushbutton input. Bitwise operations (set, clear and check operations) will be used to manipulate specific bits without affecting other bits in a register.

2. Marking and Due Date

- Students can form a group (maximum of **two** students) to complete the lab tasks but the demonstration will be assessed **individually**. There is a total of **2.5 marks** allocated to this lab. You need to complete the tasks during the lab session or at latest before the following week lab (in which case, students need to upload a short video to Canvas capturing the operational tasks).

3. Activitiy #1: Preparation

- Read Chapter 14 of the datasheet (page 84) – IO Ports (a brief register summary is attached at the end of this sheet.)
- Please refer to “How to setup PlatformIO and a project.pdf” to set up the PlatformIO extension (IDE for AVR programming) and create an AVR project.
- Download “***mcu1.cpp***” from the Canvas and copy and paste the code into your main.cpp in your project (or you can add the file to your project). Connect your UNO board to your computer using a USB connector. Build the project and upload the executable to the UNO board.
- If the on-board LED (labelled as ‘L’ in UNO) is blinking at 1Hz, your system is ready for the lab. Otherwise, check your USB connection (you can manually set up the port number, e.g., “upload_port = com4” in the platformio.ini file under your VScode project.)

4. Activity #2: Control an LED using a pushbutton

Connect an LED to Atmega328P PD3 pin (Atmega328P PORT D3, or UNO pin-3), and a pushbutton to PD2 pin (PORT D2, or UNO pin-2). It will look similar to Fig.1.

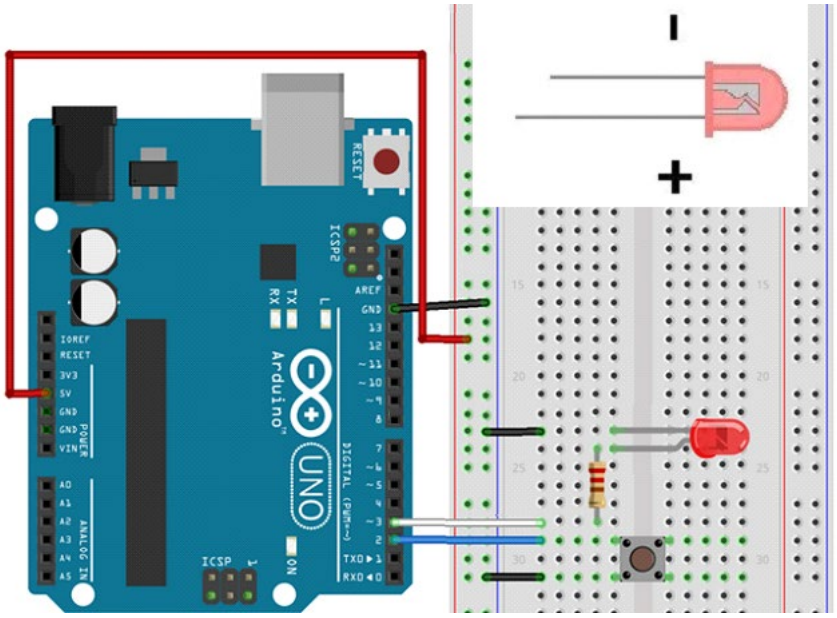

Figure 1. An LED and pushbutton connections. Note the positive terminal (a longer lead) of the LED, and the LED is connected to PORT-D together with the pushbutton.

When connecting the LED, a current-limiting resistor is necessary to protect the DIO output pins. A few notes:

- You need to check the LED datasheet to make sure. A very basic 5mm LED with a red lens has a typical forward voltage of 2.0V and a rated forward current of 20mA. This suggests when the LED is turned on, the voltage drop across the resistor should be 5V-2V = 3V, with 20mA. Thus the required resistance R = V/I = 3V/20mA = 150Ω. The MCU kit includes (220Ω, 1KΩ and 10KΩ) and 220Ω will do the job (higher resistance is okay, but it will allow less current, thus yielding dimmer than nominal).
- The longer-side lead of the LED is the anode (+) and be connected to a higher voltage.
- When the pushbutton is pressed, the PD2 pin is connected to the GND. If the PD2 is configured as an input mode, it will read 0V. If the pushbutton is released, PD2 pin is disconnected from GND, but not connected to 5V, making it a “floating” pin. How can we provide 5V input then?
- We can attach a pull-up resistor (connected to 5V) to the pushbutton. Alternatively, there is a more straightforward solution: Atmegar328P DIO supports an internal pull-up resistor which can be used to provide 5V when the pushbutton is floating. To activate the pull-up, the PORT-bit should be written to 1 in an input mode

(DDR-bit and PUD-bit are all 0 in default). Then reading input is through PIND register (refer to Appendix for details).

- The connection summary is as below,

DIO pins	Setup	Connection
PD3 (PORT D3, or UNO pin-4)	Output mode	An LED is connected
PD2 (PORT D2, or UNO pin-3)	Input mode with an internal pull-up resistor activated	A pushbutton is connected

- Modify the code to blink the LED at 1Hz (or ON for 0.5 sec and OFF for 0.5 sec) connected at PD3, while the pushbutton is not pressed. If the button is pressed (0V), turn off the LED. If the button is released, blink the LED again. **Show the results to tutors to get marked (0.5 marks).**

5. Activity #3: Blink two LEDs using a pushbutton

Modify the connections so that an additional LED is connected to PD5 (PORT D5) as shown in Fig.2. Write code so that the pushbutton controls the blinking of each LED as described in the table below. Pay attention to the *bit-wise operations* to avoid accidentally affecting other bits (Note that we can only write/read in a unit of bytes (not bits) from a register). Show the results to tutors to get marked.

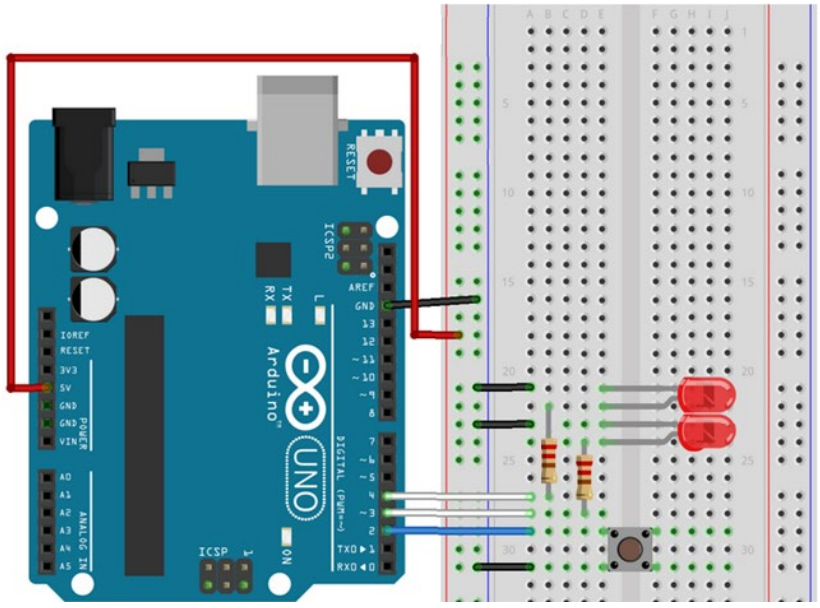

Figure 2. Two LEDs and pushbutton connections.

- Connection Summary:

DIO pins	Setup	Connection

PD3 (PORT D3, or UNO pin-4)	Output mode	An LED1 is connected
PD4 (PORT D4, or UNO pin-5)	Output mode	An LED2 is connected
PD2 (PORT D2, or UNO pin-3)	Input mode with an internal pull-up resistor activated	A pushbutton is connected

- Show the results to get marked:

Input	**LED output**	**Marking**
Pushbutton is released (HIGH)	LED1(PD3) blinks at 1Hz, and LED2 (PD4) is off	/1 mark
Pushbutton is pressed (LOW)	LED1(PD3) is off, and LED2(PD4) blinks at 1Hz	/1 mark

Appendix: Related Atmega328P Registers

There are 3 Digital input and output ports in Atmegar328P:

- PORT B[0-7] – UNO pin [8-13]
- PORT C[0-6] – UNO pin [A0-A5] (A6 pin is not mapped to UNO headers)
- PORT D[0-7] – UNO pin [0-7]

Registers used in this lab:

14.4.2 PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.4.3 DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.4.8 PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.4.9 DDRD – The Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.4.10 PIND – The Port D Input Pins Address(1)

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Each port has 3 registers (x = B, C, or D):

- PORTx – Output register in an output mode
- DDRx – Set direction (output or input). In default, it is in input mode.
- PINx – Input register to read in an input mode

To use a pin in an input mode:

- Read the PINx register to read the pin data (default mode)
- To activate the internal pull-up resistor, write ‘1’ to the PORTx-bit. Usually, we don’t write at PORTx in an input mode, and it is reserved to activate the resistor connection

Table 18-1. Port Pin Configurations

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

To use a pin in an output mode:

- Write ‘1’ at the bit position of the DDRx register
- Write data at the bit position of the PORTx register