Mechatronics Engineering

Tutorial #1

Introduction to AVR Microcontrollers

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Tutorials Teaching Assistants:

- MSc. Eng. Menna Magdy
  Office: C3.124, Email: mennatullah.ibrahim@guc.edu.eg

- MSc. Eng. Bishoy Emil
  Office: C7.120C, Email: bishoy.emil@guc.edu.eg

- MSc. Eng. Abdallah Hossam
  Office: C3.124, Email: abdallah.mohammed@guc.edu.eg

Tutorial Contents

- Introduction to AVR Microcontrollers
- Commonly Used Circuit Components
  - LED
  - Switches
- Arduino Uno Board
- Arduino Code Example

Tutorial Contents

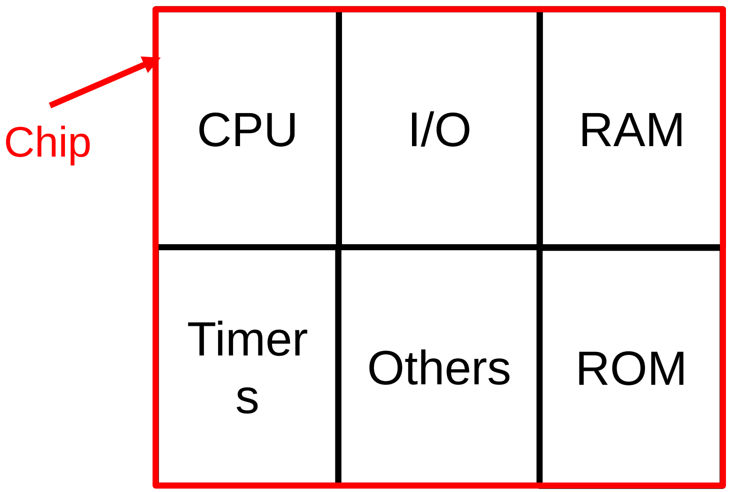
- **Introduction to AVR Microcontrollers**
- Commonly Used Circuit Components
  - LED
  - Switches
- Arduino Uno Board
- Arduino Code Example

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Introduction to AVR Microcontrollers

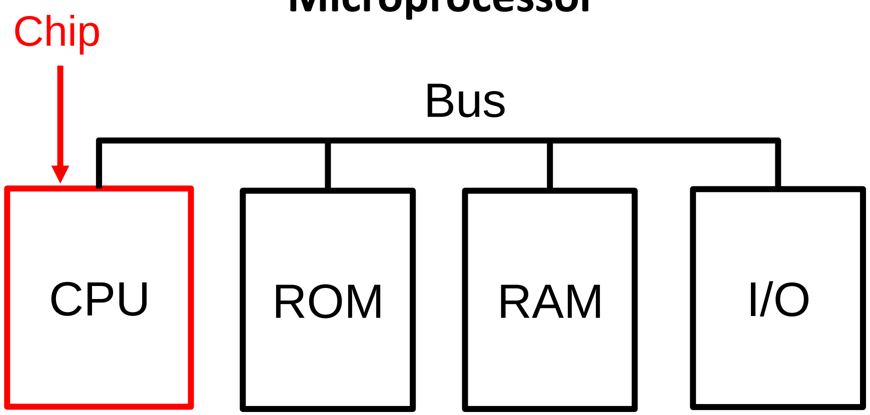
- **Microcontroller vs Microprocessor:**

Microcontroller



- CPU + Peripherals in one package (Compact).
- Usually, it is the preferred choice for many embedded systems.

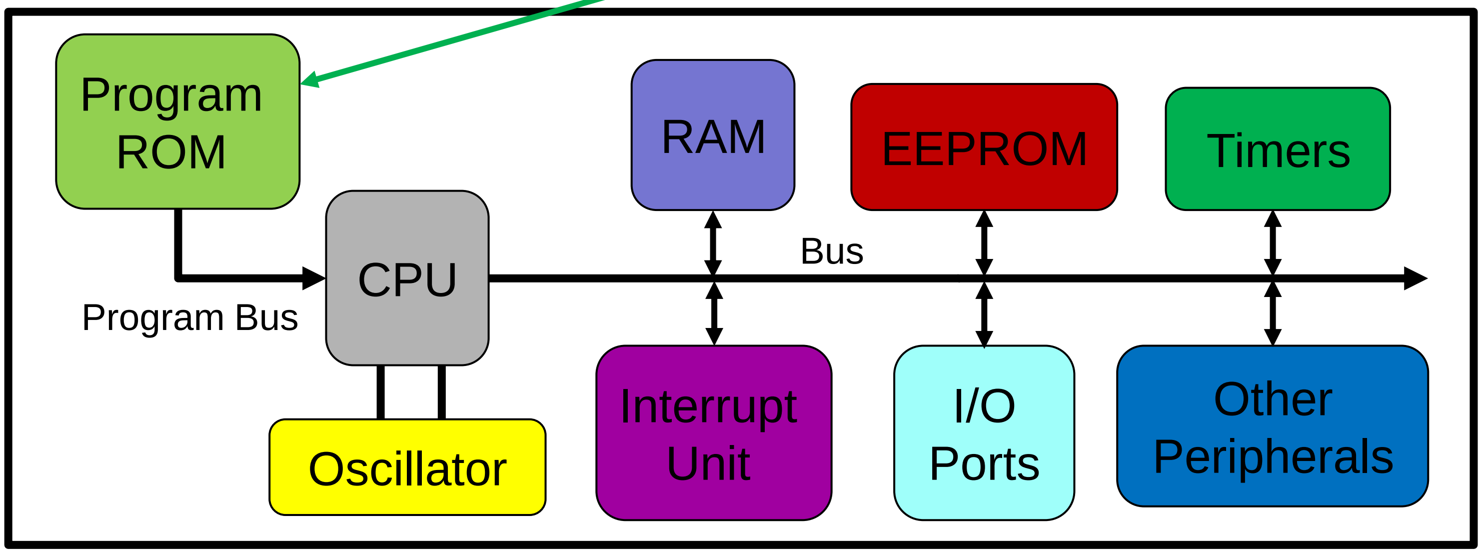
Microprocessor



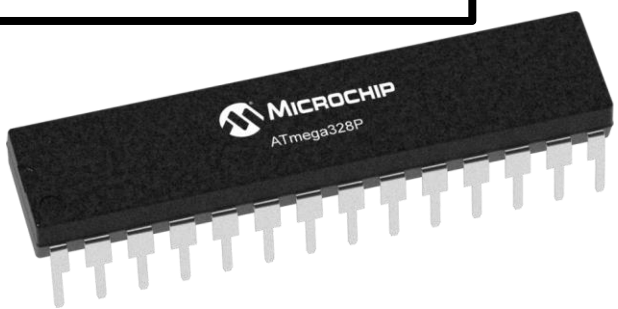
- Requires the addition of peripherals externally (Can be bulkier).
- More versatile than a microcontroller.

Introduction to AVR Microcontrollers

- AVR microcontrollers are used in this course.
- A simplified view of an AVR MCU:

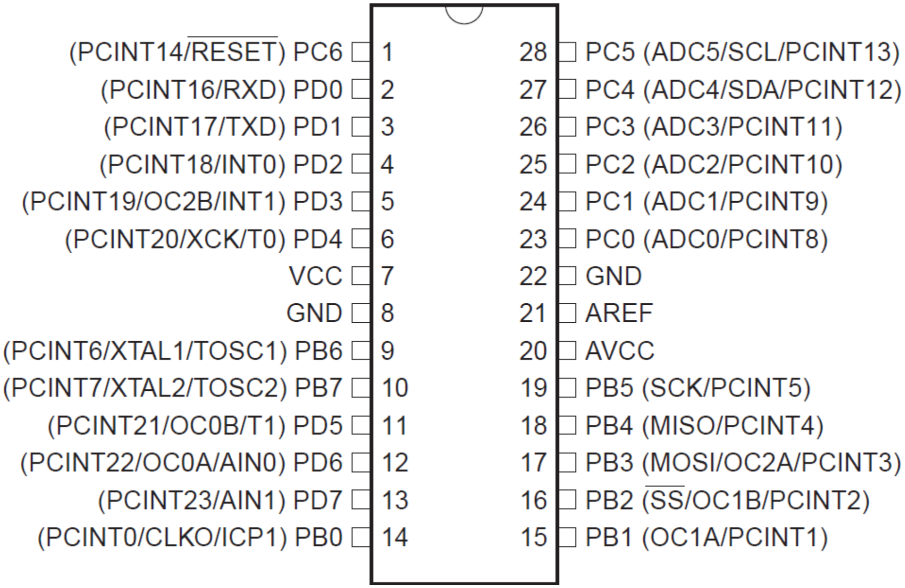


- The device used in the examples is **ATmega328P**.



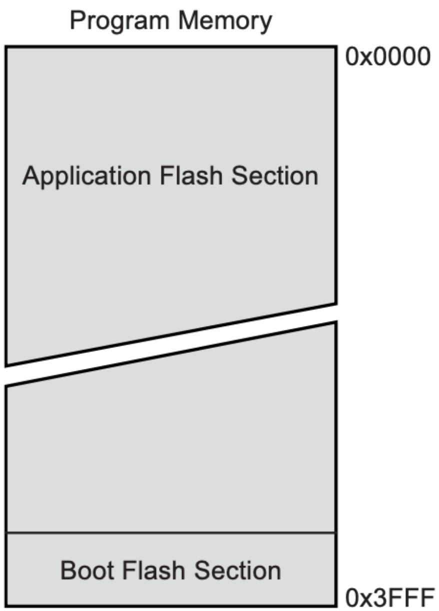
ATmega328P

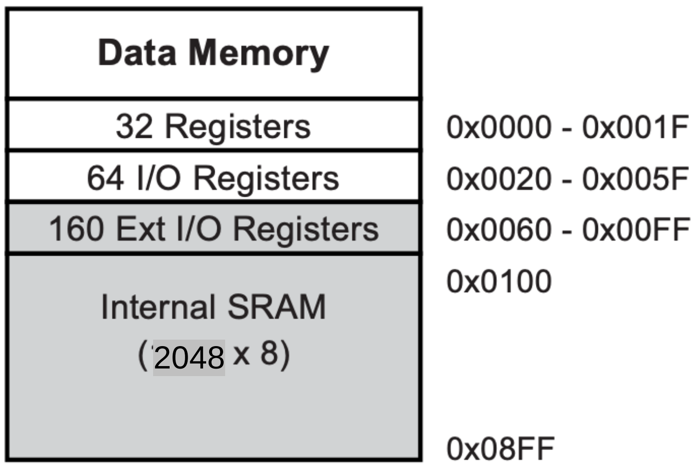
- Some features of the ATmega328P Microcontroller Unit (MCU):
  - ❖ Programmable I/O lines.
  - ❖ Non-volatile memory segments (Flash program memory, EEPROM)
  - ❖ Static RAM (Part of Data memory)
  - ❖ Three timers
  - ❖ Analog-to-Digital Converted (ADC)
  - ❖ PWM Channels
  - ❖ Communication ports (SPI, I2C, USART)
  - ❖ External/Internal Interrupts



Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

ATmega328P Memory





- Program Memory: 32K Bytes of programmable flash memory.

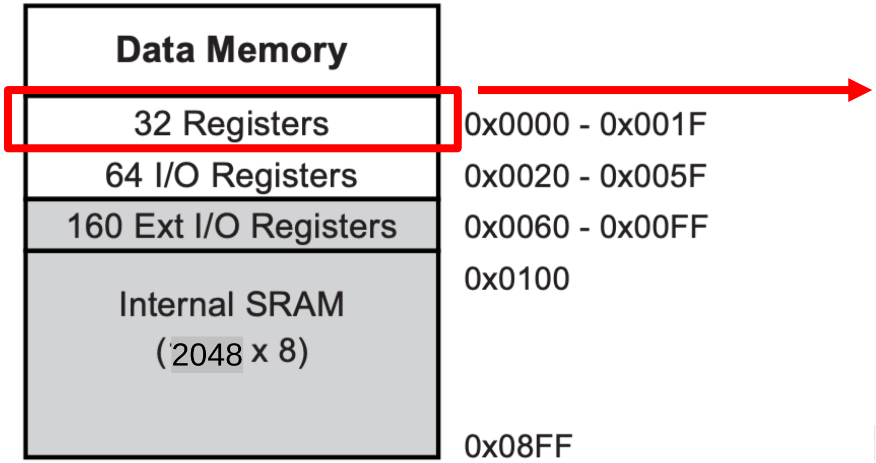
- Other non-volatile memory include: 1K Bytes of EEPROM

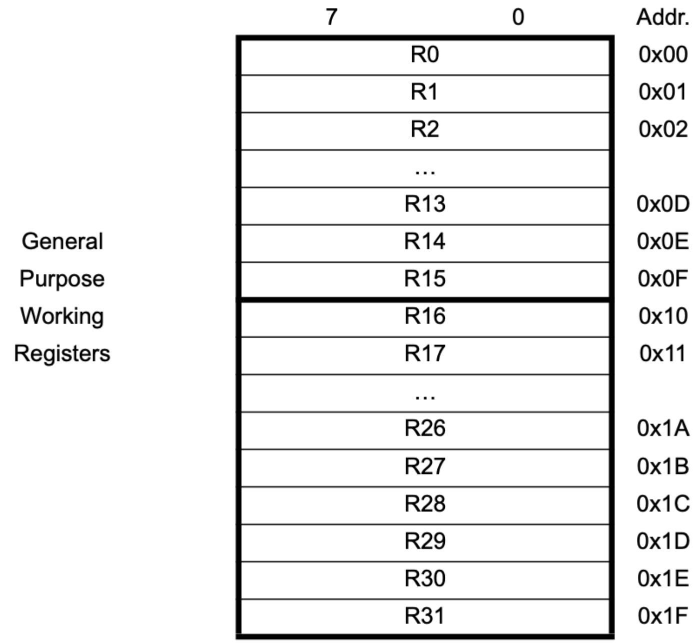
Data Memory

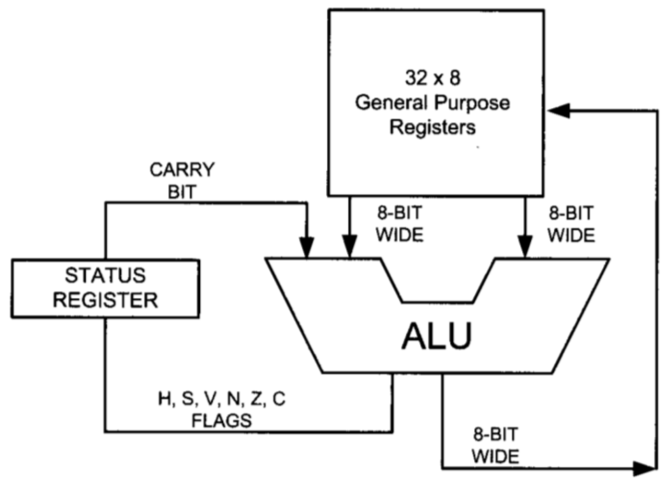
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

ATmega328P Memory

Data Memory:







- GPWR: Registers used by the CPU to store data temporarily

ATmega328P Memory

Data Memory:

Data Memory	
32 Registers	0x0000 - 0x001F
64 I/O Registers	0x0020 - 0x005F
160 Ext I/O Registers	0x0060 - 0x00FF
Internal SRAM (2048 x 8)	0x0100
	0x08FF

Address Mem.	Address I/O	Name
$20	$00	TWBR
$21	$01	TWSR
$22	$02	TWAR
$23	$03	TWDR
$24	$04	ADCL
$25	$05	ADCH
$26	$06	ADCSRA
$27	$07	ADMUX
$28	$08	ACSR
$29	$09	UBRRL
$2A	$0A	UCSRB
$2B	$0B	UCSRA
$2C	$0C	UDR
$2D	$0D	SPCR
$2E	$0E	SPSR
$2F	$0F	SPDR
$30	$10	PIND
$31	$11	DDRD
$32	$12	PORTD
$33	$13	PINC
$34	$14	DDRC
$35	$15	PORTC

Address Mem.	Address I/O	Name
$36	$16	PINB
$37	$17	DDRB
$38	$18	PORTB
$39	$19	PINA
$3A	$1A	DDRA
$3B	$1B	PORTA
$3C	$1C	EECR
$3D	$1D	EEDR
$3E	$1E	EEARL
$3F	$1F	EEARH
$40	$20	UBRRC UBRRH
$41	$21	WDTCR
$42	$22	ASSR
$43	$23	OCR2
$44	$24	TCNT2
$45	$25	TCCR2
$46	$26	ICR1L
$47	$27	ICR1H
$48	$28	OCR1BL
$49	$29	OCR1BH
$4A	$2A	OCR1AL

Address Mem.	Address I/O	Name
$4B	$2B	OCR1AH
$4C	$2C	TCNT1L
$4D	$2D	TCNT1H
$4E	$2E	TCCR1B
$4F	$2F	TCCR1A
$50	$30	SFIOR
$51	$31	OCDR OSCCAL
$52	$32	TCNT0
$53	$33	TCCR0
$54	$34	MCUCSR
$55	$35	MCUCR
$56	$36	TWCR
$57	$37	SPMCR
$58	$38	TIFR
$59	$39	TIMSK
$5A	$3A	GIFR
$5B	$3B	GICR
$5C	$3C	OCR0
$5D	$3D	SPL
$5E	$3E	SPH
$5F	$3F	SREG

AVR Programming

- How to program an AVR microcontroller?

- First Option (**Assembly language**):

  - ❖ Write instructions that can be transformed into machine language by assembler.

  - ❖ Example to add two hexadecimal values (0x25 and 0x34):

```
LDI R16,0x25 ;load 0x25 into R16(working register 16)
LDI R17,0x34 ;load 0x34 into R17 (working register 17)
ADD R16,R17 ;add value R17 to R16 (R16 = R16 + R17)
```

AVR Programming

- The Second Option (**The C Language**):

  - Example to add two hexadecimal values (0x25 and 0x34):

```c
char x = 0x25 + 0x34 ; // Add 0x25 and 0x34 literals
```

  - Compiler transforms high-level C language code into machine language.

- Assembly language programs may produce smaller sized machine language code compared to C (more efficient in terms of program memory space usage).

- Reasons to use C language instead of Assembly language:

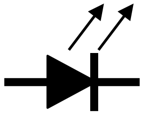
  - It is easier and less time consuming to write in C than in Assembly.
  - C is easier to modify and update.
  - C code is portable to other microcontrollers with little modifications.

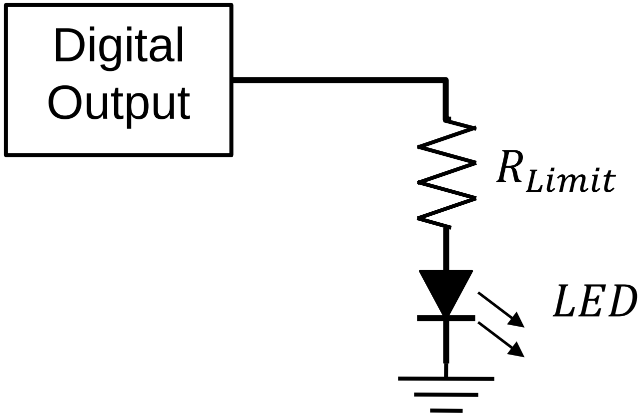
- **C language** is used in this course.

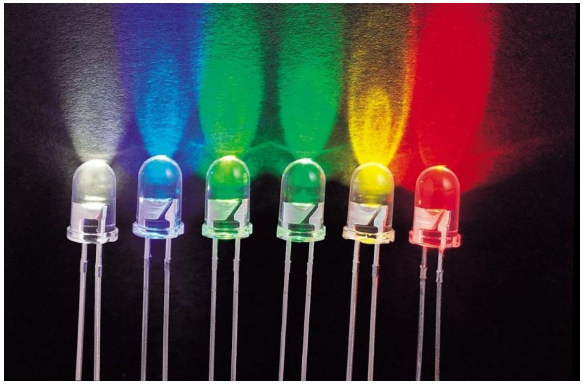
Tutorial Contents

- Introduction to AVR Microcontrollers
- **Commonly Used Circuit Components**
  - **LED**
  - Switches
- Arduino Uno Board
- Arduino Code Example

Light-Emitting Diode (LED)

- One of the few actuators that can be driven directly by a microcontroller.
- A convenient way for an embedded systems to provide a visual indication of some activity.
- LED Circuit Symbol: 
- LEDs Can be controlled directly by one of the MCU's digital I/O pins, when connected to a current limiting resistor in series.
- Connection Example:





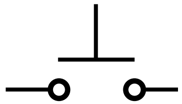
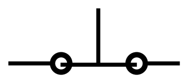
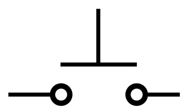
GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Tutorial Contents

- Introduction to AVR Microcontrollers
- **Commonly Used Circuit Components**
  - LED
  - **Switches**
- Arduino Uno Board
- Arduino Code Example

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Switches

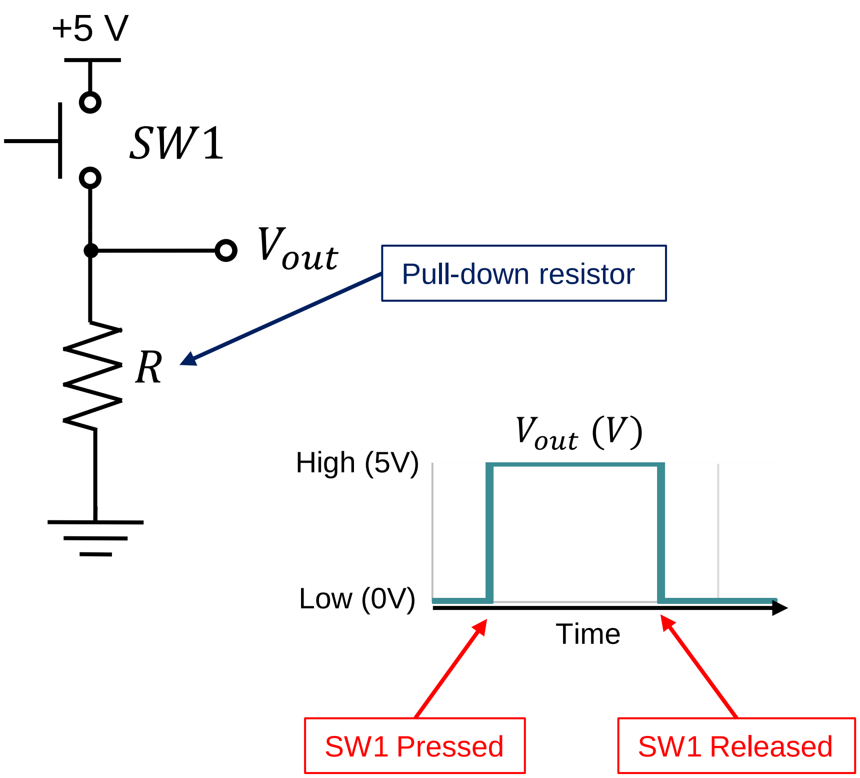
- The function of a switch is simple. When we press a switch, two contacts are joined together, and connection is made.
- Switch Symbol: 
- The button/switch is considered as a short-circuit if pressed, and an open-circuit if released:
- Pressed (short-circuit): 
- Released (open-circuit): 

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

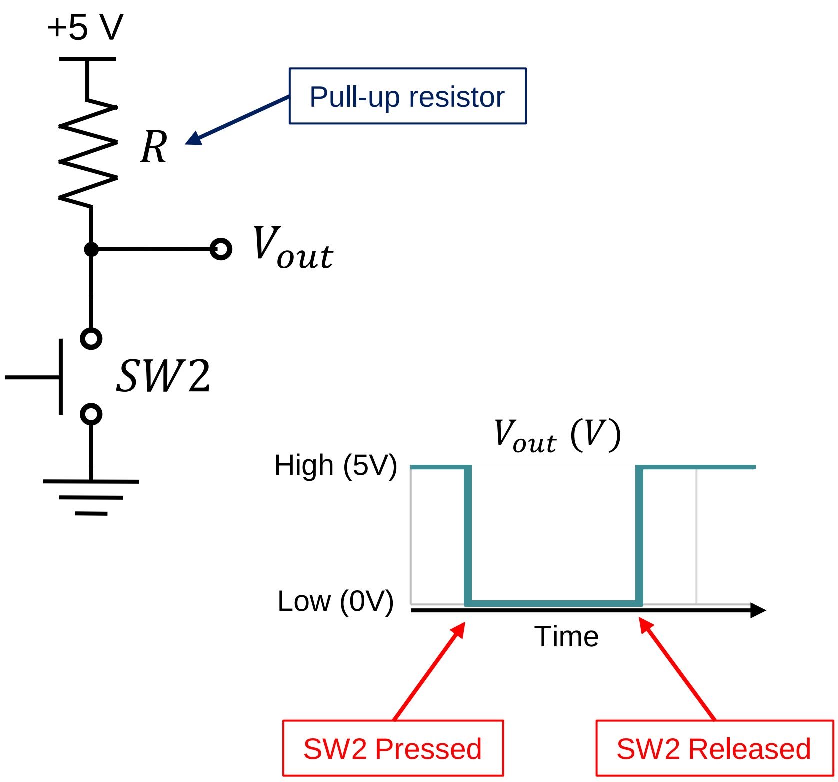
Switches

- Possible connections of a push button to send a digital signal V_{out}:

Active High



Active Low



Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Switch Bouncing

- When a switch is toggled or a push button is pressed, the contacts will move from one position to the other. As the contacts of the switch settle into their new position, they mechanically vibrate (bounce).

- This causes the connection to be opened and closed several times, which introduces a series of spikes between the high and low states.

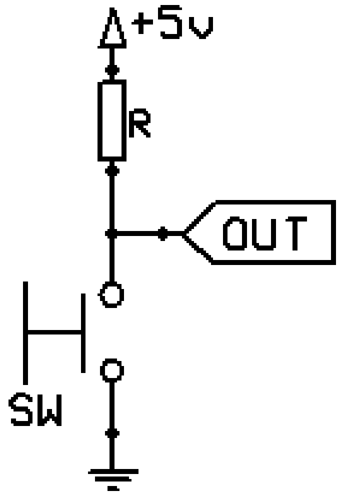


Fig. 1a: A simple switch configuration without a de-bouncing circuit

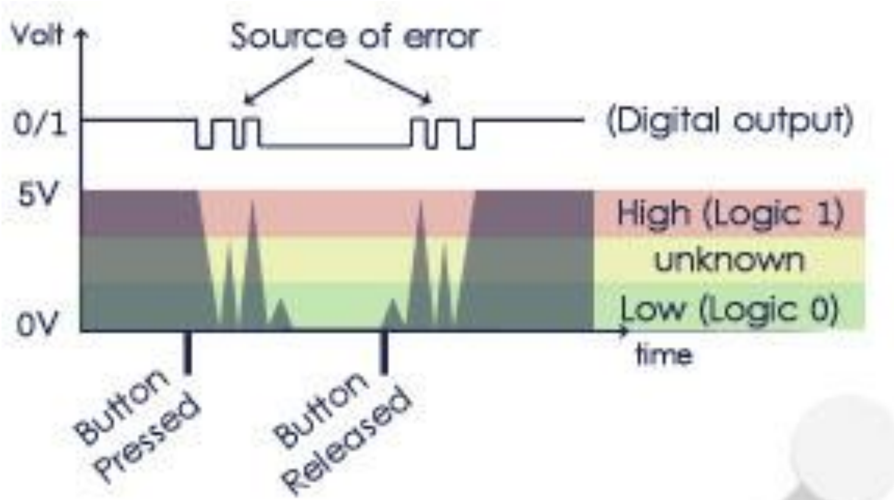


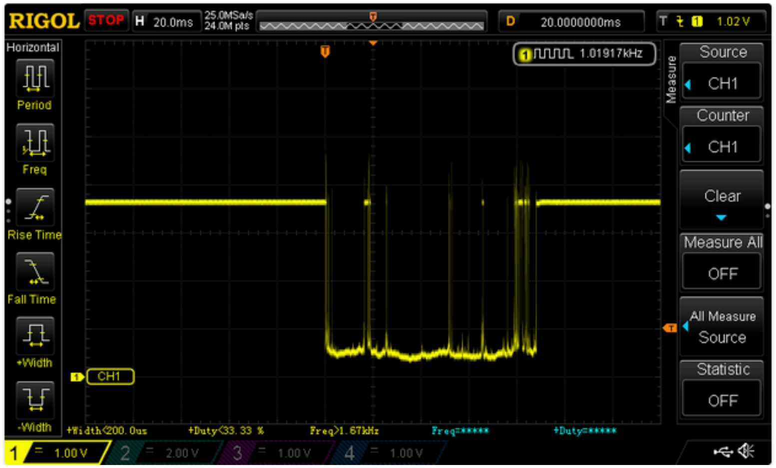
Fig. 1b: Behavior of a switch without a de-bouncing circuit

Switch Bouncing

- This series of spikes can be interpreted by a microcontroller (or any digital circuit) as if the button was pressed many times.

Bouncing while transition from high to low

Bouncing while transition from low to high



Switch bouncing for an active low configuration demonstrated by an oscilloscope capture

Switch De-bouncing

There are 2 common solutions to the switch bouncing problem. **Analog solution** and **digital micro-controller based solution**. Both are commonly used, and sometimes, both are used at the same time to provide a very stable design.

1. The analog de-bouncing circuit

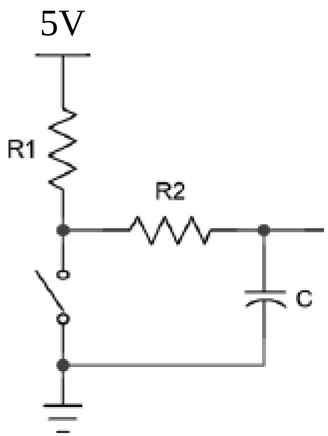


Fig. 3a: A switch with a de-bouncing circuit

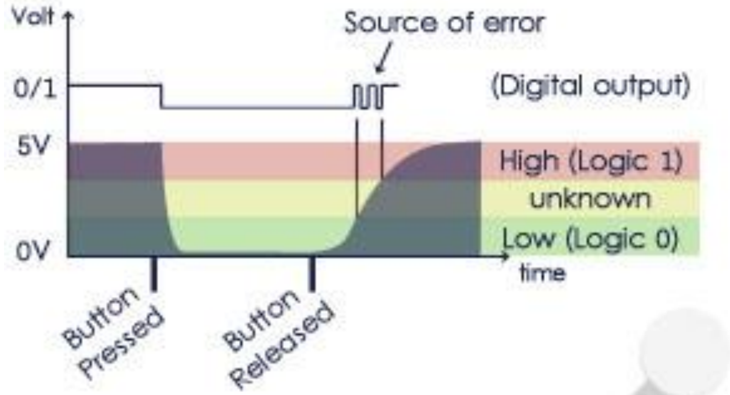


Fig. 3b: A switch with a de-bouncing circuit

The analog solution relies principally on a capacitor, which plays the role of resisting the voltage changes on the output.

De-bouncing

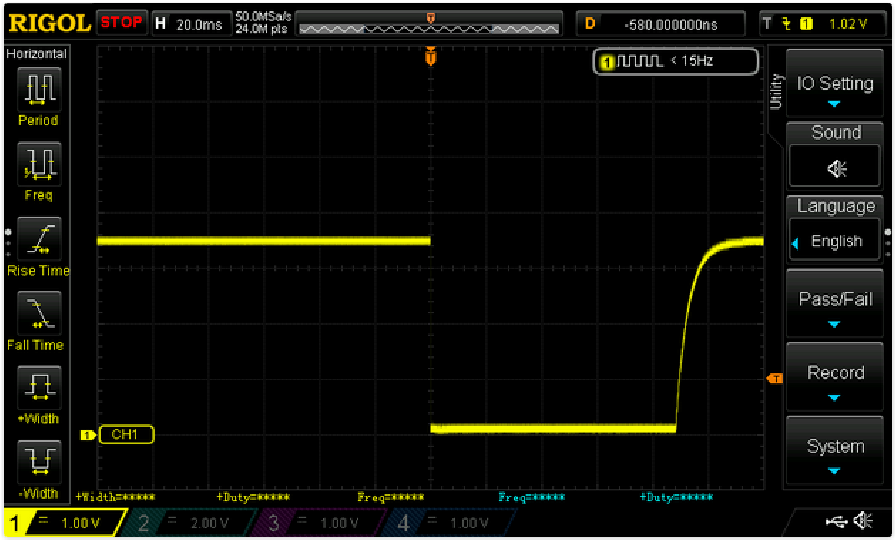


Fig.3a: switch response after de-bouncing with an RC filter

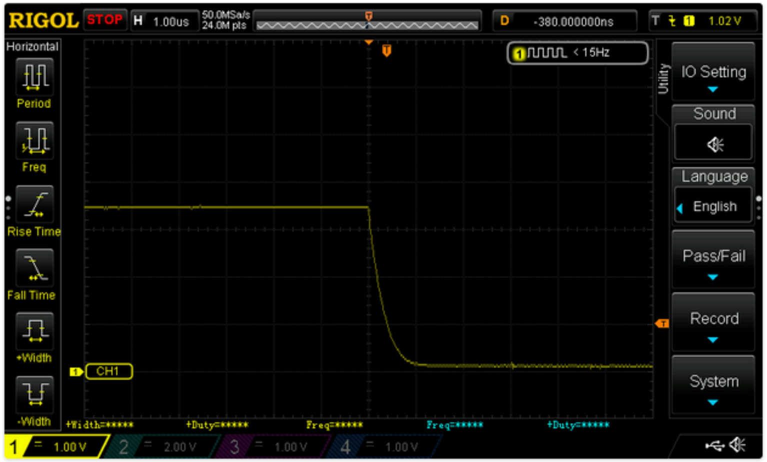


Fig.3b: falling edge on a smaller time base

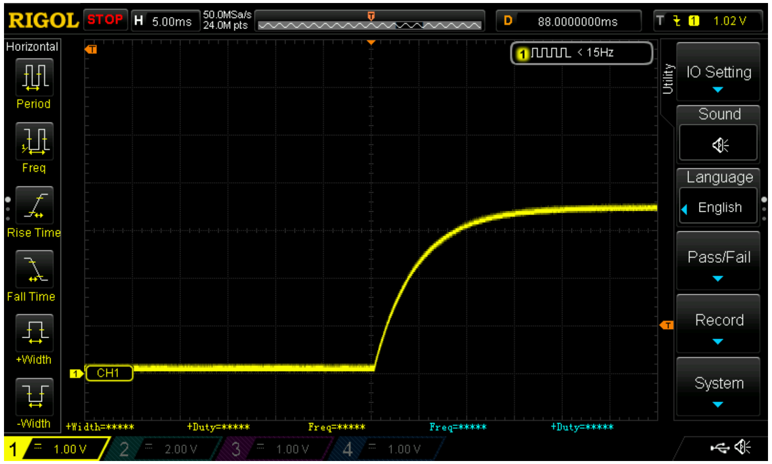
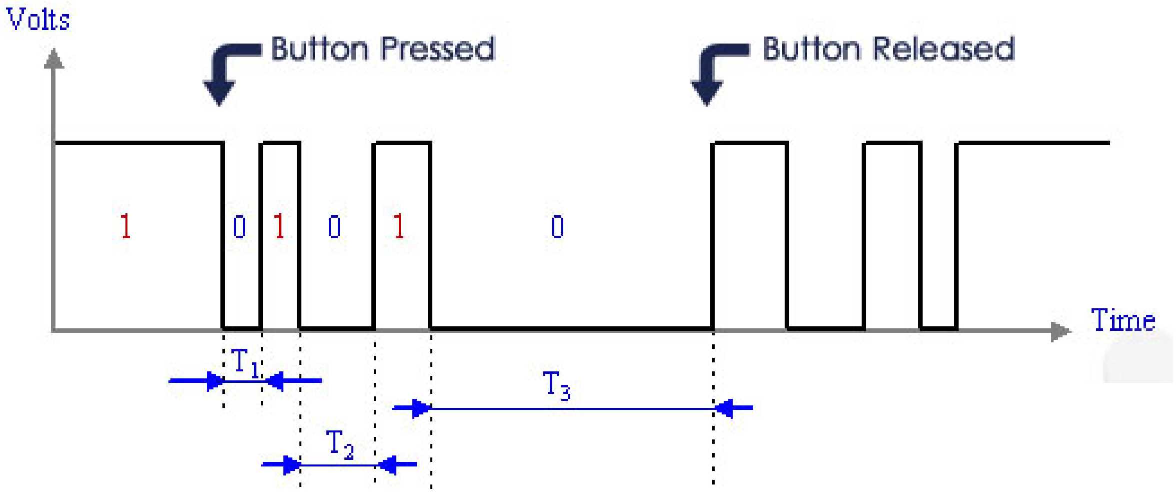


Fig.3c: rising edge on a smaller time base

De-bouncing

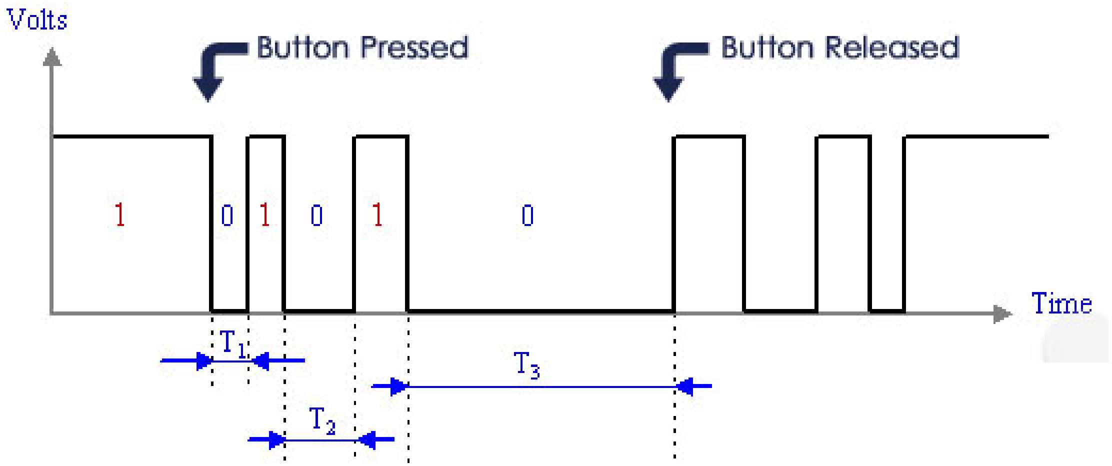
2. The software based de-bouncing

The idea here is not to prevent voltage spikes to occur, but rather to record them and analyze them in real-time using simple software routines.



- The timing T1, T2 and T3 corresponds to the low (logic =0) pulses being sent to the microcontroller from the switch.
- T3 is a **valid** reading that we want to take in account.
- T1 and T2 are the length of the reading we want to get rid off, or more scientifically, that we want to **filter**.
- It is clear that the difference between those three periods is their length, and that is how the micro-controller will be able to differentiate between valid and un-valid pulses.

De-bouncing



There are many ways to implement digital de-bouncing or filtering using a microcontroller;

- one simple method is to make a **counter** count up as long as the signal is Low, and reset this counter when the signal is high. If the counter reach a certain fixed value, which should be 1 or 2 times bigger than T1 or T2 (noise pulses), this means that the current pulse is a valid pulse (corresponds to T3 in our example).

- Another way to de-bounce using software is to **wait** for a certain time interval until the switch oscillations disappear. i.e. after the MICROCONTROLLER realizes the first state transition it waits for a certain period say (100 ms) then it takes the action that corresponds to this transition.

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Tutorial Contents

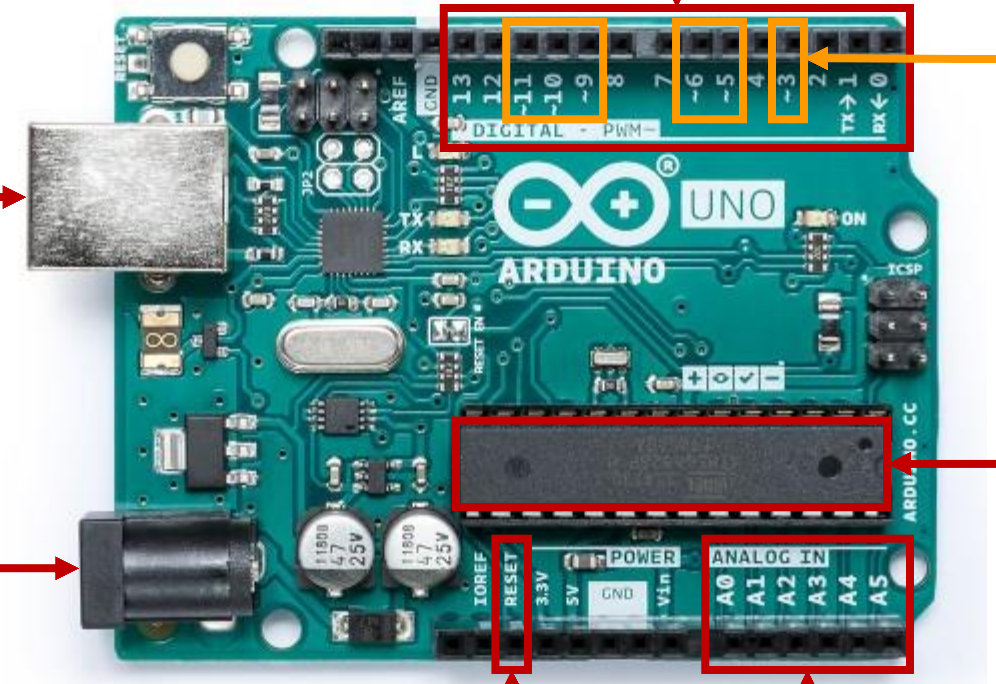
- Introduction to AVR Microcontrollers
- Commonly Used Circuit Components
  - LED
  - Switches
- **Arduino Uno Board**
- Arduino Code Example

Arduino Uno

14 Configurable Digital
Input-Output Pins

These pins can be
configured as PWM
output pins also

USB connection
for programming



ATmega328P
microcontroller

AC-to-DC adapter
goes here

Reset

6 Analog
Input Pins

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Arduino Board

- Why use Arduino ?
    - Easy to use.
    - Wide collection of libraries for different purposes.
    - Useful for rapid prototyping and initial testing of small projects.

- Caution when using Arduino:
    - Arduino code preconfigures some resources of a microcontroller, if application code uses the same resources a conflict may occur!

      For example: Arduino libraries use a hardware timer to implement the delay functionality, using the same timer in user code may result in unexpected behavior.

- It is recommended to program the MCU using the C language directly.

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Tutorial Contents

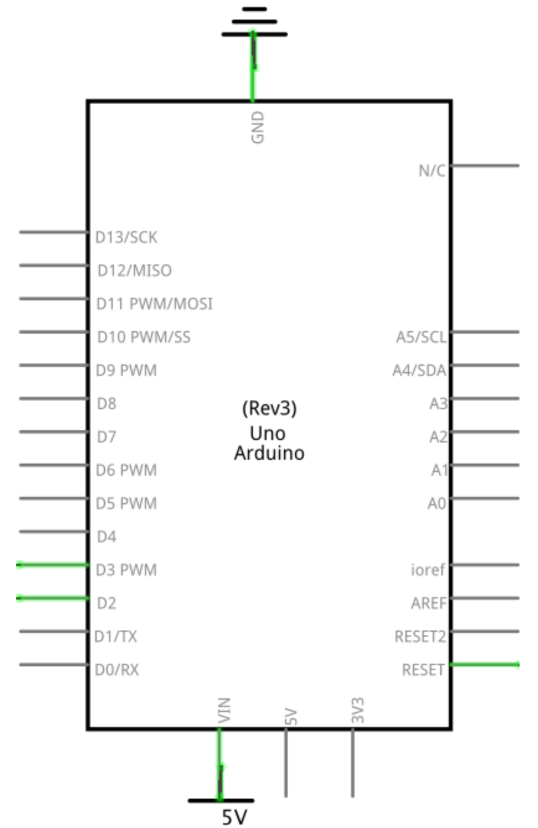
- Introduction to AVR Microcontrollers
- Commonly Used Circuit Components
  - LED
  - Switches
- Arduino Uno Board
- **Arduino Code Example**

Arduino Code Example:

Problem:

Write an Arduino code that turns an LED on when a button T2 is pressed and turns it off when a button T1 is pressed. If both buttons are pressed, the LED status should not change. You must overcome the switch bouncing problem in your software.

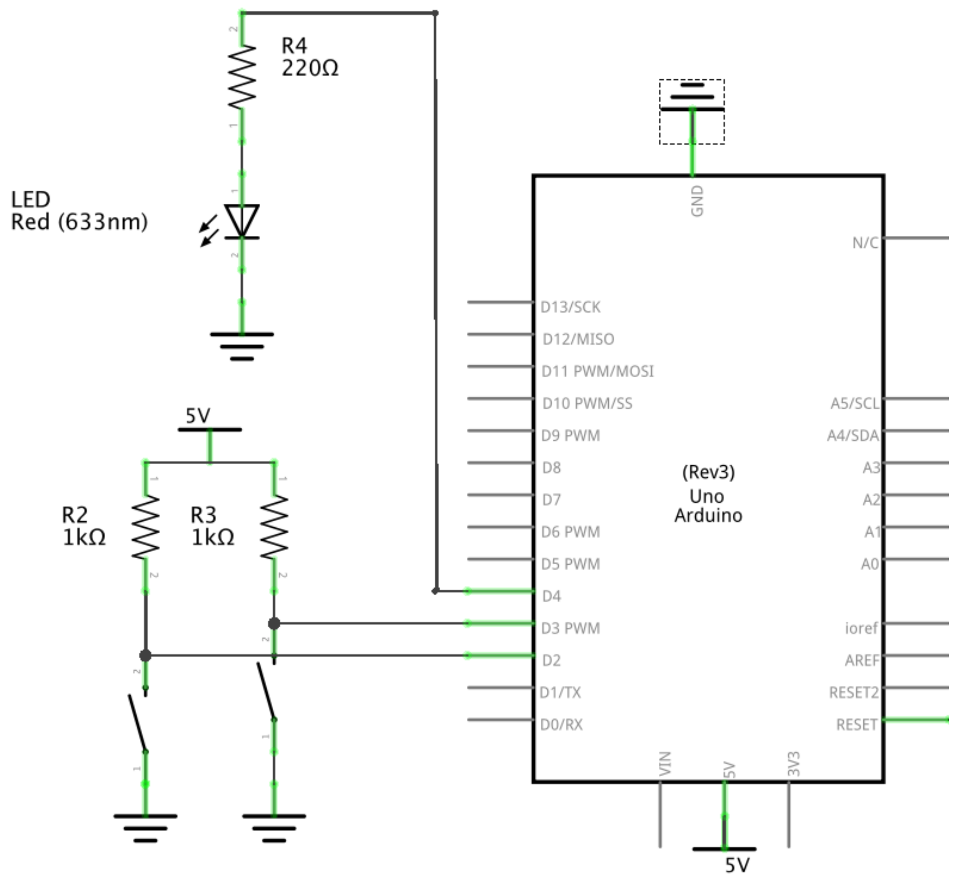
Complete the hardware connection diagram. The buttons should be connected in an **active low** configuration. **T1** should be connected to Arduino **digital pin 2** and **T2** should be connected to Arduino **digital pin 3**. The **LED** should be connected to **digital pin 4**.



Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Arduino Code Example:

Solution:



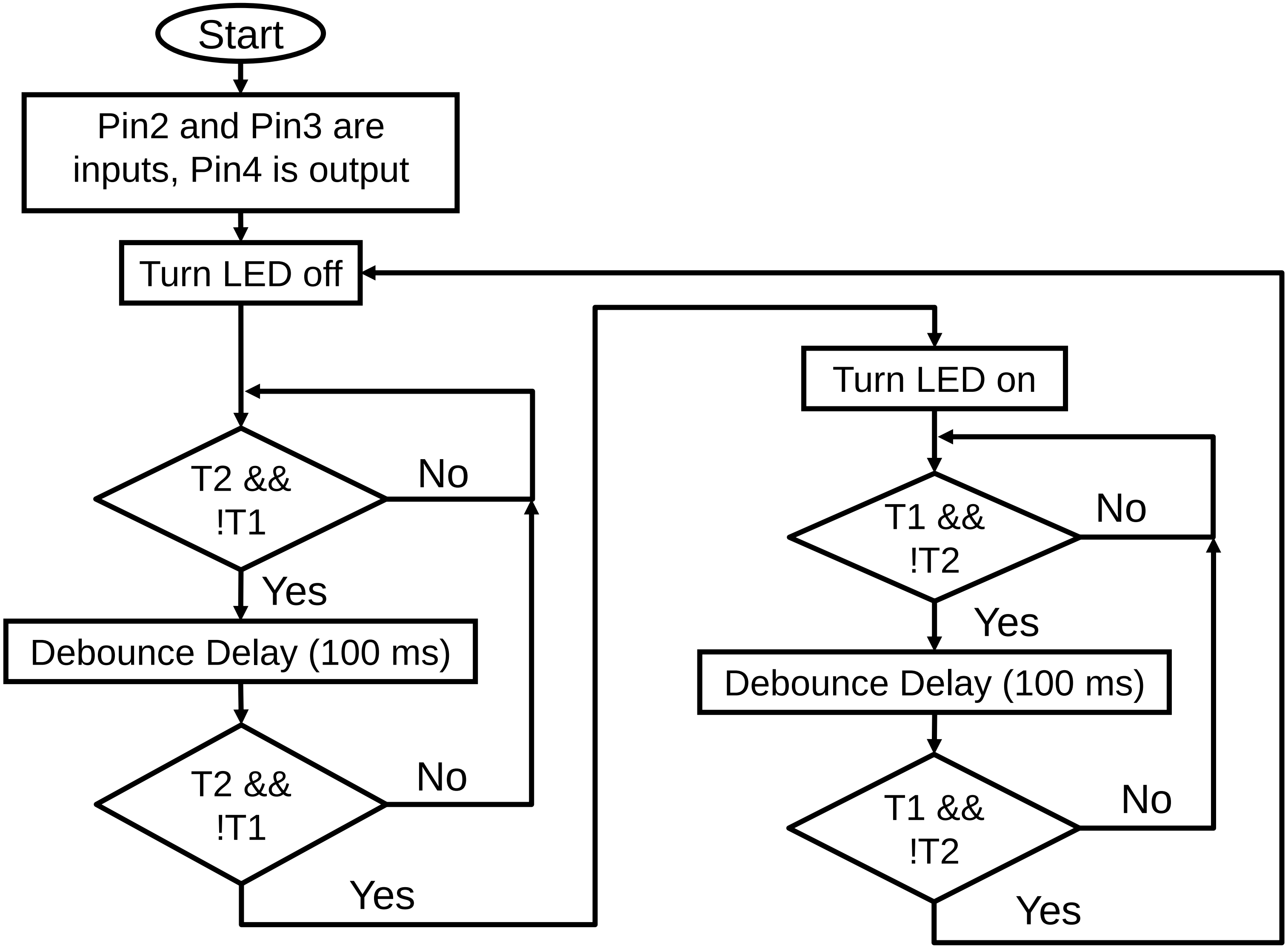
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Arduino Code Example:

Solution:

T1	T2	LED
Not Pressed	Not Pressed	No change
Pressed	Not Pressed	Turn off
Not Pressed	Pressed	Turn on
Pressed	Pressed	No change

Solution – cont'd:



Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Arduino Functions

Syntax: pinMode(pin, mode)
Parameters:
Pin = Arduino pin number to set the mode of.
Mode = INPUT, OUTPUT or INPUT_PULLUP
Function: Configures the specified pin to behave either as an input or an output. Additionally, it is possible to enable the internal pull-up resistors with the mode INPUT_PULLUP. The INPUT mode explicitly disables the internal pull-up resistors.

Syntax: digitalWrite(pin, value)
Parameters:
Pin = Arduino pin number t set the mode of.
Value = LOW or HIGH
Function: Write a HIGH or a LOW value to a digital pin.

Syntax: delay(ms)
Parameters: ms = number of milliseconds.
Function: Pauses the program for the amount of time (in milliseconds) specified as parameter.

Solution – cont'd:

Arduino Code Steps

1. Configure the inputs and outputs and initializations in setup() (Executed only once).

```
bool ledON; // LED is off --> ledON = false
            // LED is on  --> ledON = true
const int buttonOnePin = 2;
const int buttonTwoPin = 3;
const int ledPin = 4;

void setup() {
  // put your setup code here, to run once:
  pinMode(buttonOnePin,INPUT); // configure push button 1 "T1" as input
  pinMode(buttonTwoPin,INPUT); // configure push button 2 "T2"as input
  pinMode(ledPin,OUTPUT); // configure LED indicator as output
  digitalWrite(ledPin,LOW); // turn off LED indicator at start
  ledON = false; // LED is off initially
}
```

setup() method is executed only once at the beginning. Initalization and configuraiton of the Arduino should be written in this method.

Solution – cont'd:

Arduino Code Steps

2. Write your main program code in loop().

```
void loop() {
  // put your main code here, to run repeatedly:
  bool T1;
  bool T2;
  T1 = digitalRead(buttonOnePin);
  T2 = digitalRead(buttonTwoPin);
  // condition for turning the LED on: T2 pressed and T1 not pressed
  if(ledOn == false && T1 == HIGH && T2 == LOW)
  {
    delay(100); //delay for debouncing.
    T1 = digitalRead(buttonOnePin); // read the buttons again.
    T2 = digitalRead(buttonTwoPin);
    if(T1 == HIGH && T2 == LOW) // recheck the buttons
    {
      digitalWrite(ledPin,HIGH); // turn on the LED
      ledOn = true; // set flag
    }
  }
```

loop() method is executed contionously after setup() method. The main code should be written here.

Solution – cont'd:

Arduino Code Steps

Loop() continued:

```
  // condition for turning the LED off: T1 pressed and T2 not pressed
  if(ledOn == true && T1 == LOW && T2 == HIGH)
  {
    delay(100); // delay for debouncing.
    T1 = digitalRead(buttonOnePin); // read the buttons again.
    T2 = digitalRead(buttonTwoPin);
    if(T1 == LOW && T2 == HIGH) // recheck the buttons
    {
      digitalWrite(ledPin,LOW); // turn off the LED
      ledOn = false; // clear flag
    }
  }
}// end of loop()
```