I. Introduction

Interrupts are one of the most fundamental and useful principles of modern embedded processors. A microcontroller normally executes instructions in an orderly *fetch then execute* sequence depending on how the user would write the code, but it should also be ready to handle any unscheduled events that might occur inside or outside the microcontroller.

The AVR can do this by performing an *Interrupt Service Routine (ISR)*. Its execution is performed by loading the beginning address of the ISR specific for the interrupt into the program counter then, the AVR processor would start running the ISR. For the ISR to function as planned, there are several conditions that it must meet. This would further be discussed along the way *(see section b.4)*. The interrupt routine is an asynchronous process as these events occur outside the regular execution thread of the main program. It does not do things simultaneously as the regular program is stopped when the ISR runs. By default, interrupts are not *interruptible*, once it runs, the AVR CPU will automatically disable the Global Interrupt Enable bit to prevent the ISR from being interrupted. This is done to prevent stack overflows from too many interrupts occurring at once and to prevent the ISR from running too long.

As microcontrollers can accept inputs from I/O ports, interrupts can also be used for accepting inputs generated by external events. The type of device associated with this is referred to as the *external interrupts*. These are called on a given status change on the *INTn* pin. On the ATMEGA328P, all pins can trigger interrupts when they change the state from *low to high* and vise versa. Only *INT0* and *INT1* can be configured to trigger interrupts in a low level and it will trigger continuously until the input is no longer low. The other non-interrupt pins serve as the *interrupt handlers*, this allows for INT0 and INT1 to handle multiple external interrupts. The pins are grouped into three pin change interrupt vectors, if any pin changes its state, an interrupt will be triggered.

Thus, interrupts are useful as it is a way for an external or internal event to pause the current processor's activity so that it can complete a brief task before resuming where it left off. Other uses of interrupts include: serial data transfers, timer interrupts, ADC conversions, etc.

II. Discussion

a. ATMEGA328P

The ATMEGA328P is a high performance, low power controller from Microchip. It is an 8-bit microcontroller based on AVR RISC architecture and it is the most popular of all the AVR controllers as it is mostly used in Arduino boards.

a.1. Features

Some notable features of the ATMEGA328P:

- High-performance and Low-power
- Advanced RISC architecture
  - 131 powerful instructions
  - 32x8 general purpose working registers
  - Fully static operation
  - Up to 16MIPS throughput at 16MHz
  - On-chip 2-cycle multiplier
- High endurance non-volatile memory segments
  - 32k-bytes of in-system self-programmable flash program memory
  - 1k-bytes EEPROM
  - 2k-bytes internal SRAM
  - Write/erase cycles: 10k flash/100k EEPROM
  - Optional boot code section (with independent lock bits)
  - Programming lock for software security
- Peripheral Features
  - Two 8-bit Timer/Counters (with separate prescaler and compare mode)
  - One 16-bit Timer/Counter (with separate prescaler, compare mode, and capture mode)
  - Real time counter (with separate oscillator)
  - Six PWM channels
  - 8-channel 10-bit ADC
  - Programmable serial USART
  - Master/slave SPI serial interface
  - Byte-oriented 2-wire serial interface
  - Programmable watchdog timer
  - On-chip analog comparator
  - Interrupt and wake-up on pin change
- Special Microcontroller Features
  - Power-on reset and programmable brown-out detection
  - Internal calibrated oscillator
  - External and internal interrupt sources
  - Six sleep modes: Idle, ADC noise reduction, power-save, power-down, standby, and extended standby
- I/O and Packages
  - 23 programmable I/O lines

- 32-lead TQFP and 32-pad QFN/MLF

- Operating Voltage
  - 2.7V to 5.5V
- Temperature Range
  - -40°C to +125°C
- Speed Grade
  - 0 to 8 MHz at 2.7 to 5.5V
  - 0 to 16MHz at 4.5 to 5.5V
- Low Power Consumption
  - Active mode: 1.5mA at 3V (4MHz)
  - Power-down mode: 1μA at 3V

a.2. Pin Configurations and Description

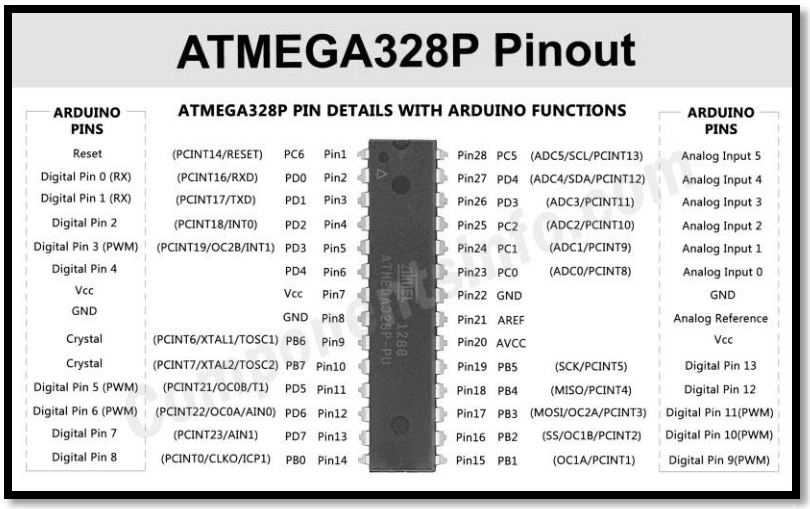

ATMEGA328P Pinout
ATMEGA328P PIN DETAILS WITH ARDUINO FUNCTIONS
ARDUINO PINS
Reset (PCINT14/RESET) PC6 Pin1
Digital Pin 0 (RX) (PCINT16/RXD) PD0 Pin2
Digital Pin 1 (RX) (PCINT17/TXD) PD1 Pin3
Digital Pin 2 (PCINT18/INT0) PD2 Pin4
Digital Pin 3 (PWM) (PCINT19/OC2B/INT1) PD3 Pin5
Digital Pin 4 PD4 Pin6
Vcc Vcc Pin7
GND GND Pin8
Crystal (PCINT6/XTAL1/TOSC1) PB6 Pin9
Crystal (PCINT7/XTAL2/TOSC2) PB7 Pin10
Digital Pin 5 (PWM) (PCINT21/OC0B/T1) PD5 Pin11
Digital Pin 6 (PWM) (PCINT22/OC0A/AIN0) PD6 Pin12
Digital Pin 7 (PCINT23/AIN1) PD7 Pin13
Digital Pin 8 (PCINT0/CLKO/ICP1) PB0 Pin14
Pin28 PC5 (ADC5/SCL/PCINT13) Analog Input 5
Pin27 PD4 (ADC4/SDA/PCINT12) Analog Input 4
Pin26 PD3 (ADC3/PCINT11) Analog Input 3
Pin25 PC2 (ADC2/PCINT10) Analog Input 2
Pin24 PC1 (ADC1/PCINT9) Analog Input 1
Pin23 PC0 (ADC0/PCINT8) Analog Input 0
Pin22 GND GND
Pin21 AREF Analog Reference
Pin20 AVCC Vcc
Pin19 PB5 (SCK/PCINT5) Digital Pin 13
Pin18 PB4 (MISO/PCINT4) Digital Pin 12
Pin17 PB3 (MOSI/OC2A/PCINT3) Digital Pin 11(PWM)
Pin16 PB2 (SS/OC1B/PCINT2) Digital Pin 10(PWM)
Pin15 PB1 (OC1A/PCINT1) Digital Pin 9(PWM)
ATMEGA328P-PU

Figure 1: ATMEGA328P Pinout Diagram

PIN NUMBER	PIN NAME	FUNCTION
1	PC6 (Reset)	Used as the reset pin. It can only be used as an I/O pin when the RSTDISBL fuse is programmed.
2	PD0 (RXD)	○ RXD (Data input for USART). ○ USART serial communication interface (can be used for programming)
3	PD1 (TXD)	○ TXD (Data output for USART)

		○ USART serial communication interface (can be used for programming) ○ INT2 (External interrupt 2 input)
4	PD2 (INT0)	External Interrupt source 0.
5	PD3 (INT1/OC2B)	○ External Interrupt source 1 ○ OC2B (PWM- Timer/Counter2 Output compare Match B Output)
6	PD4 (XCK/T0)	○ T0 (Timer0 External Counter Input) ○ XCK (USART External Clock I/O)
7	VCC	Connected to *positive* voltage.
8	GND	Connected to ground.
9	PB6 (XTAL1/TOSC1)	○ XTAL1 (Chip clock oscillator pin 1 or External clock input) ○ TOSC1 (Timer oscillator pin 1)
10	PB7 (XTAL2/TOSC2)	○ XTAL2 (Chip clock oscillator pin 2) ○ TOSC2 (Timer oscillator pin 2)
11	PD5 (T1/OC0B)	○ T1 (Timer1 external counter input) ○ OC0B (PWM- Timer/Counter0 Output compare Match B Output)
12	PD6 (AIN0/OC0A)	○ AIN0 (Analog comparator positive I/P) ○ OC0A (PWM- Timer/Counter0 Output compare Match A Output)
13	PD7 (AIN1)	AIN1 (Analog comparator negative I/P)
14	PB0 (ICP1/CLKO)	○ ICP1 (Timer/Counter 1 input capture pin) ○ CLKO (Divided system clock)
15	PB1 (OC1A)	OC1A (Timer/Counter1 Output compare Match A Output)
16	PB2 (SS/OC1B)	○ SS (SPI Slave Select Input) ○ OC1B (Timer/Counter1 Output compare Match B Output)
17	PB3 (MOSI/OC2A)	○ MOSI (Master Output Slave Input) ○ OC2A (Timer/Counter2 Output compare Match A Output)
18	PB4 (MISO)	MISO (Master Input Slave Output)
19	PB5 (SCK)	SCK (SPI Bus Serial Clock)
20	AVCC	Power for internal ADC converter.
21	AREF	Analog reference pin for ADC.
22	GND	Ground.
23	PC0 (ADC0)	ADC input channel 0.

24	PC1 (ADC1)	ADC input channel 1.
25	PC2 (ADC2)	ADC input channel 2.
26	PC3 (ADC3)	ADC input channel 3.
27	PC4 (ADC4/SDA)	○ ADC input channel 4 ○ SDA (two-wire serial bus data input/output line)
28	PC5 (ADC5/SCL)	○ ADC input channel 5 ○ SCL (two-wire serial bus clock line)

a.3. I/O Ports

The ATMEGA328P has 23 general purpose digital I/O pins assigned to 3 GPIO ports: 8-bit ports B and D, and 7-bit port C. Each I/O port pin may be configured as an *output* with symmetrical drive characteristics, each pin has 20 mA of current to drive LED displays directly. It can be configured as an *input* with or without pull-up resistors (values of 20-50k-Ω). Port B, C, and D are bi-directional I/O ports. Port B ranges from PB7 to PB0, port C ranges from PC5 to PC0, and port D ranges from PD7 to PD0.

b. Interrupts

b.1. Basics of Interrupts

Interrupt is a signal sent to the CPU by external devices, normally by I/O devices, indicating an event that needs immediate attention. It alerts the processor to a high-priority process requiring interruption of the current process. When an interrupt occurs, the controller completes the current instruction and begins an Interrupt Service Routine (ISR) or Interrupt Handler. When an interrupt happens, ISR instructs the processor or controller what to do.

b.2. Kinds of Interrupts

1. Hardware Interrupts – They are generated by hardware devices to indicate that they need attention from the OS. They may have recently obtained data (for example, keystrokes on the keyboard or data from the ethernet card), or they may have recently accomplished a task that the operating system had previously requested, such as data transfer between the hard drive and memory.

2. Software Interrupts – are developed by programs when they want the operating system to conduct a system call on their behalf. An extraordinary state or a special instruction in the instruction set that induces an interrupt when executed by the processor causes the interrupt. For instance, if the processor's arithmetic logic unit executes a command to divide a number by zero, a divide-by-zero exception is triggered. As a result, the machine will either stop calculating or show an error message.

b.3. Purpose of Interrupts

An interrupt is a device that allows modules such as I/O or memory to interrupt the CPU's normal processing. External devices are slower than the CPU in comparison. If there isn't an interrupt, the CPU can spend a lot of time waiting for external devices to catch up to the CPU's speed. The CPU's performance reflects the impact. As a result, an interrupt is needed to remove these limitations.

b.4. Interrupts Service Routine (ISR)

An ISR (also known as an interrupts handler) is a software process that is triggered by a hardware interrupt request. It takes care of the request and sends it to the CPU, interrupting the current task. The process is resumed once the ISR is completed. A basic example of an ISR is a routine that handles keyboard events, such as pressing or releasing a key. The ISR processes the input each time a key is pressed.

c. ATMEGA328p External Interrupts

c.1. Interrupt Vectors in ATmega328P

There are 25 interrupt sources supported by the ATmega328P. Each of these interrupts, as well as the separate Reset Vector, has its own program vector in the Flash program memory space, which is located at the lowest addresses.

ATmega328P Interrupt Vector Table

VECTOR NO	**PROGRAM ADRESS**	**SOURCE**	**INTERRUPT DEFINITION**	**ARDUINO/C++ ISR() MACRO VECTOR NAME**	**ASSEMBLY NAME**
1	0x0000	RESET	Reset		
2	0x0002	INT0	External Interrupt Request 0 (pin D2)	(INT0_vect)	INT0addr
3	0x0004	INT1	External Interrupt Request 1 (pin D3)	(INT1_vect)	INT1addr
4	0x0006	PCINT0	Pin Change Interrupt Request 0 (pins D8 to D13)	(PCINT0_vect)	PCI0addr
5	0x0008	PCINT1	Pin Change Interrupt Request 1 (pins A0 to A5)	(PCINT1_vect)	PCI1addr
6	0x000A	PCINT2	Pin Change Interrupt Request 2 (pins D0 to D7)	(PCINT2_vect)	PCI2addr
7	0x000C	WDT	Watchdog Time-out Interrupt	(WDT_vect)	WDTaddr
8	0x000E	TIMER2 COMPA	Timer/Counter2 Compare Match A	(TIMER2_COMPA_vect)	OC2Aaddr
9	0x0010	TIMER2 COMPB	Timer/Counter2 Compare Match B	(TIMER2_COMPB_vect)	OC2Baddr
10	0x0012	TIMER2 OVF	Timer/Counter2 Overflow	(TIMER2_OVF_vect)	OVF2addr
11	0x0014	TIMER1 CAPT	Timer/Counter1 Capture Event	(TIMER1_CAPT_vect)	ICP1addr
12	0x0016	TIMER1 COMPA	Timer/Counter1 Compare Match A	(TIMER1_COMPA_vect)	OC1Aaddr
13	0x0018	TIMER1 COMPB	Timer/Counter1 Compare Match B	(TIMER1_COMPB_vect)	OC1Baddr
14	0x001A	TIMER1 OVF	Timer/Counter1 Overflow	(TIMER1_OVF_vect)	OVF1addr
15	0x001C	TIMER0 COMPA	Timer/Counter0 Compare Match A	(TIMER0_COMPA_vect)	OC0Aaddr
16	0x001E	TIMER0 COMPB	Timer/Counter0 Compare Match B	(TIMER0_COMPB_vect)	OC0Baddr
17	0x0020	TIMER0 OVF	Timer/Counter0 Overflow	(TIMER0_OVF_vect)	OVF0addr
18	0x0022	SPI, STC	SPI Serial Transfer Complete	(SPI_STC_vect)	SPIaddr
19	0x0024	USART, RX	USART, Rx Complete	(USART_RX_vect)	URXCaddr
20	0x0026	USART, UDRE	USART, Data Register Empty	(USART_UDRE_vect)	UDREaddr
21	0x0028	USART, TX	USART, Tx Complete	(USART_TX_vect)	UTXCaddr
22	0x002A	ADC	ADC Conversion Complete	(ADC_vect)	ADCCaddr

23	0x002C	EE READY	EEPROM Ready	(EE_READY_vect)	ERDYaddr
24	0x002E	ANALOG COMP	Analog Comparator	(ANALOG_COMP_vect)	ACIaddr
25	0x0030	TWI	2-wire Serial Interface	(I2C) (TWI_vect)	TWIaddr
26	0x0032	SPM READY	Store Program Memory Ready	(SPM_READY_vect)	SPMRaddr

c.2. Register Description

Data and addresses are stored in **registers,** which are temporary storage locations within the CPU. The register file is a component that contains all of the microprocessor's general-purpose registers. A few CPUs even have special registers in the registry file, such as the PC and the status register.

External Interrupt Control Register

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
EICRA	-	-	-	-	ISC11	ISC10	ISC01	ISC00

External Interrupt Mask Register

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
EIMSK	-	-	-	-	-	-	INT1	INT0

External Interrupt Flag Register

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
EIFR	-	-	-	-	-	-	INTF1	INTF0

Pin Change Interrupt Control Register

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
PCICR	-	-	-	-	-	PCIE2	PCIE1	PCIE0

Pin Change Interrupt Mask Register

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
PCMSK 0	PCINT 7	PCINT 6	PCINT 5	PCINT 4	PCINT 3	PCINT 2	PCINT 1	PCINT 0

Pin Change Interrupt Flag Register

	7 bit	6 bit	5 bit	4 bit	3 bit	2 bit	1 bit	0 bit
PCIFR	-	-	-	-	-	PCIF2	PCIF1	PCIF0

c.3. External Interrupt Processing

An interrupt could be scheduled and requested by the currently running program, or it could be unplanned and triggered by an incident unrelated to the currently running program. When a specified signal is input to the dedicated external interrupt terminal, an external interrupt occurs. These interrupts may be caused by a time interval expiring, the operator pressing the interrupt key on the console, or the processor receiving a signal from another processor. These interrupts are triggered by a change in the INTn pin's status. This is basically an input interrupt, and it's ideal for situations where you need to respond quickly to something.

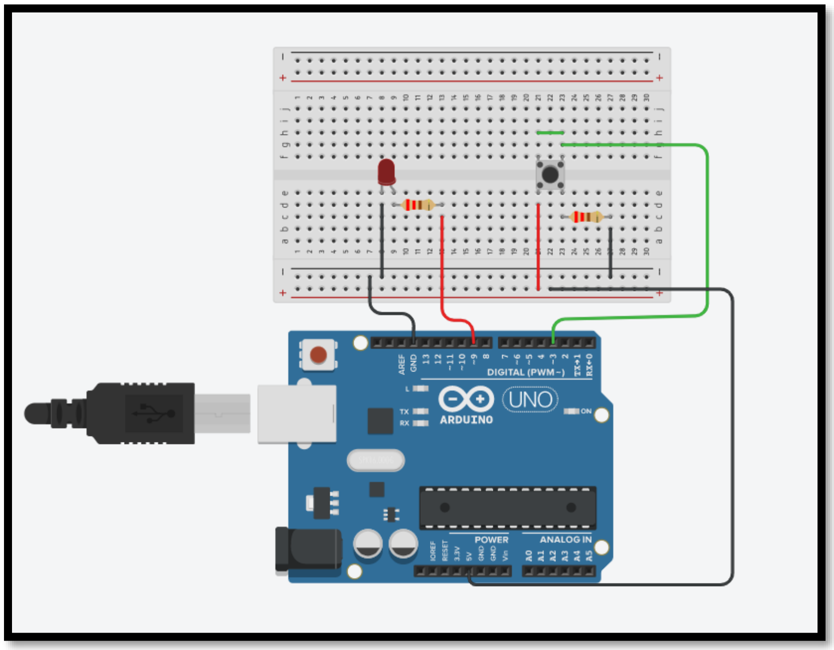
c.4. Global Manipulation of the Interrupt Flag

The Interrupt flag (IF) is a system flag bit in the FLAGS register of the x86 architecture that controls whether the central processing unit (CPU) responds to maskable hardware interrupts. The global interrupt flag is stored in the status register's I bit (SREG). When dealing with interrupts, it's important to pay attention to atomic access to objects that could be changed by code running in the interrupt context. Interrupts are sometimes disabled for a period of time in order to execute such tasks without being interrupted so that items can be taken into consideration with respect to compiler optimizations. Compilers that agree on how to handle interrupt code are almost impossible to come by. Since the C language wants to avoid machine-dependent info, each compiler author is forced to create their own support system. The vector table in the AVR-GCC setting is predefined to point to interrupt routines with predetermined names. When the subsequent interrupt happens, the routine will be called if you use the right name. If you don't specify your own interrupt routines, the system library will use the default interrupt routines.

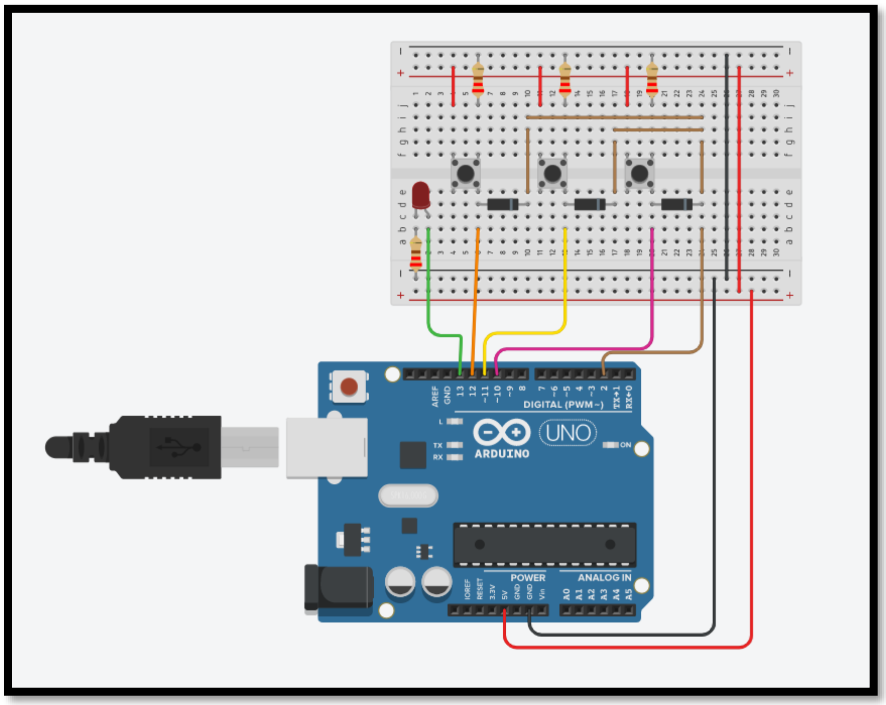
III. Circuit Configurations

a. Basic LED Toggle State

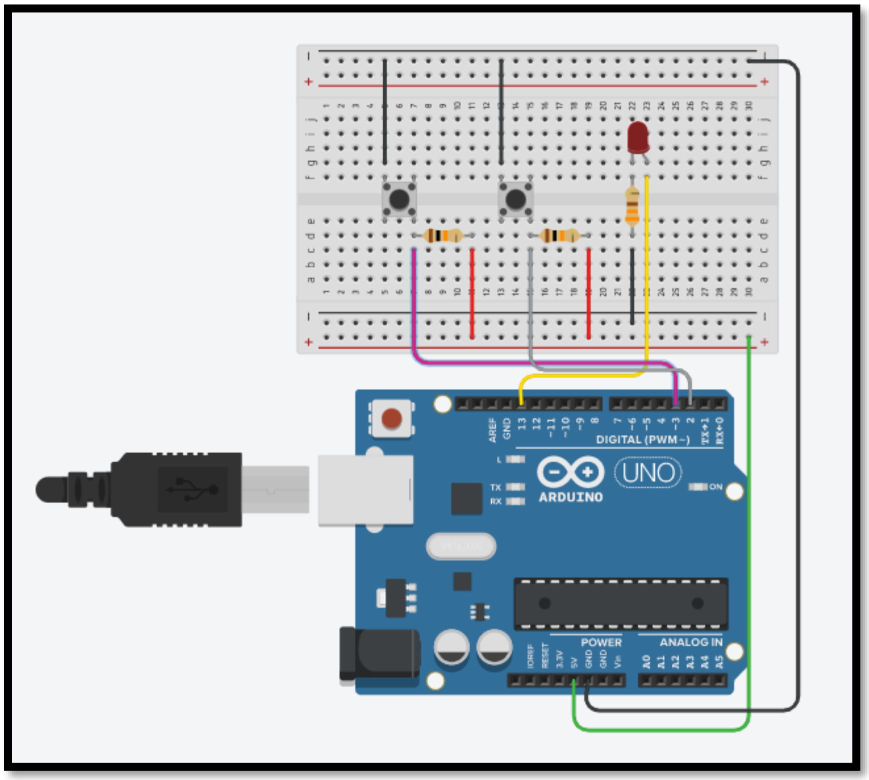
a.1. Using One Pushbutton as an External Interrupt



a.2. Connecting Multiple Pushbuttons in one External Interrupt

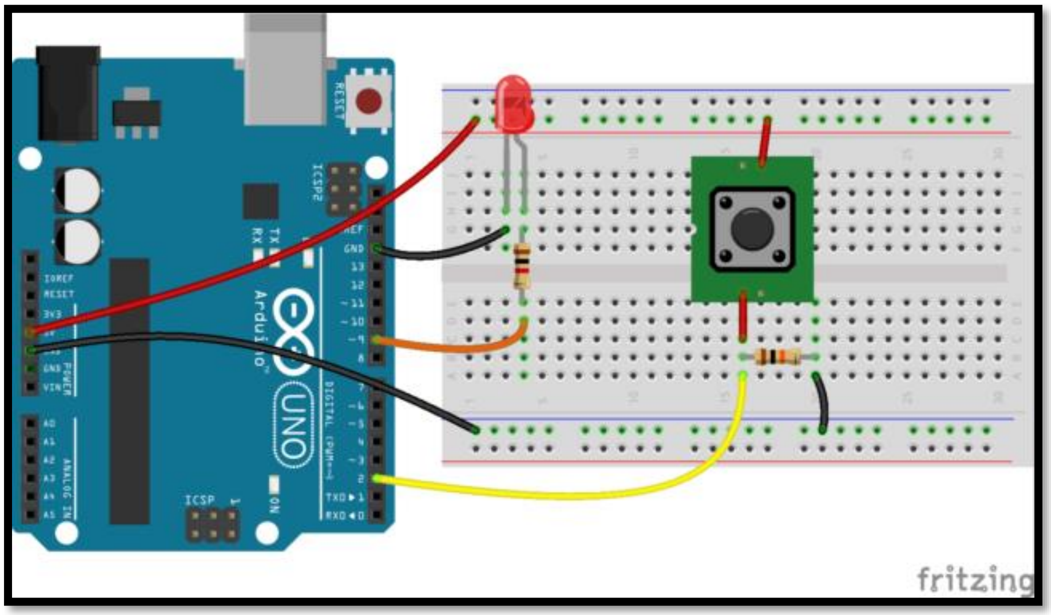


a.3. Connecting Multiple Pushbuttons in all External Interrupt

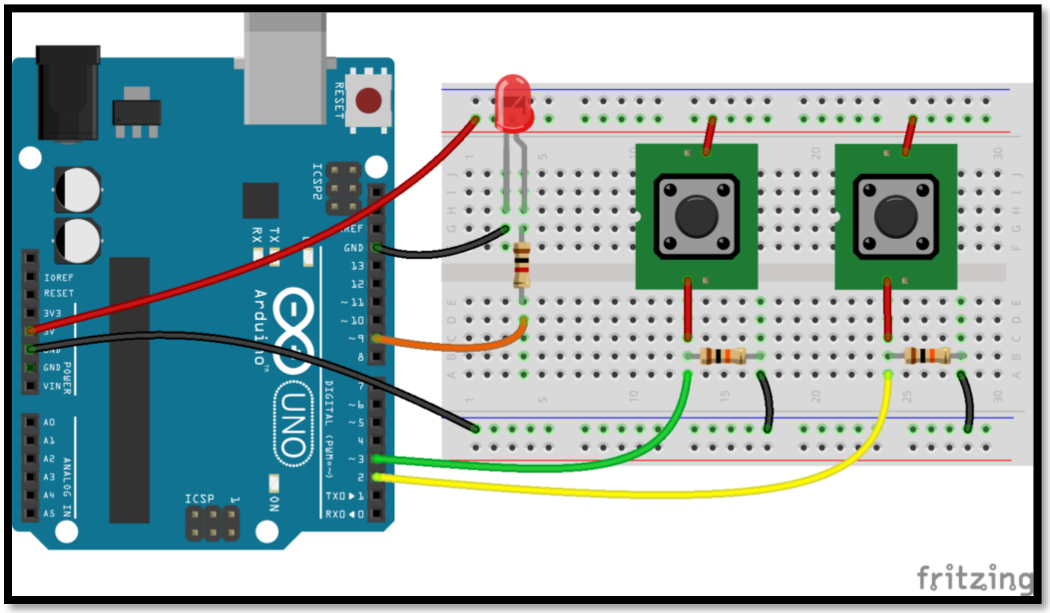


b. Basic Binary Counter

b.1 Using One Push Button to Count Up



b.2. Using One Pushbutton to Count Up and Another to Reset Counting


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX▶1
RX◀0
DIGITAL (PWM=~)
TX
RX
Arduino
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
fritzing