Embedded Programming for Beginners

Implementing an embedded application using Arduino

Session 4

Course Goals

- Objective
  - Introduction to external interrupts
  - Learn how to use PWM

External interrupts

- INT0-INT1
  - Rising
  - Falling
  - Change
  - LOW
- PCINT0-PCINT2
  - Not pin specific
  - Consult PIN register
  - Multiple pins hard to identify
- Do not confuse the interrupt vector with effective interruption pin
  - Ex.: PCINT2 vs PCINT20 (PD4)

Configure external interrupts

- Configure peripheric

```
// întreruperi externe
EICRA |= (1 << ISC00);    // set INT0 to trigger on ANY logic change
// întreruperi de tip pin change (activare vector de întreruperi)
PCICR |= (1 << PCIE2); // enable the pin change interrupt, set PCIE2 to enable PCMSK2 scan
// alte întreruperi
```

- INT0-INT1 – use EICRA registry
- PCINT0-PCINT2 – use PCICR registry

Configure external interrupts

- Activate interrupts

```
// întrerupere externă
EIMSK |= (1 << INT0);     // Turns on INT0
// întrerupere de tip pin change
PCMSK2 |= (1 << PCINT20); // Turns on PCINT20 (PD2)
// activare întreruperi globale
sei();
```

- Configure External Interrupt Mask Register (EIMSK)
  - Bits INT1:0
- Configure Pin Change Mask Register (EIMSK)
  - One register for each port

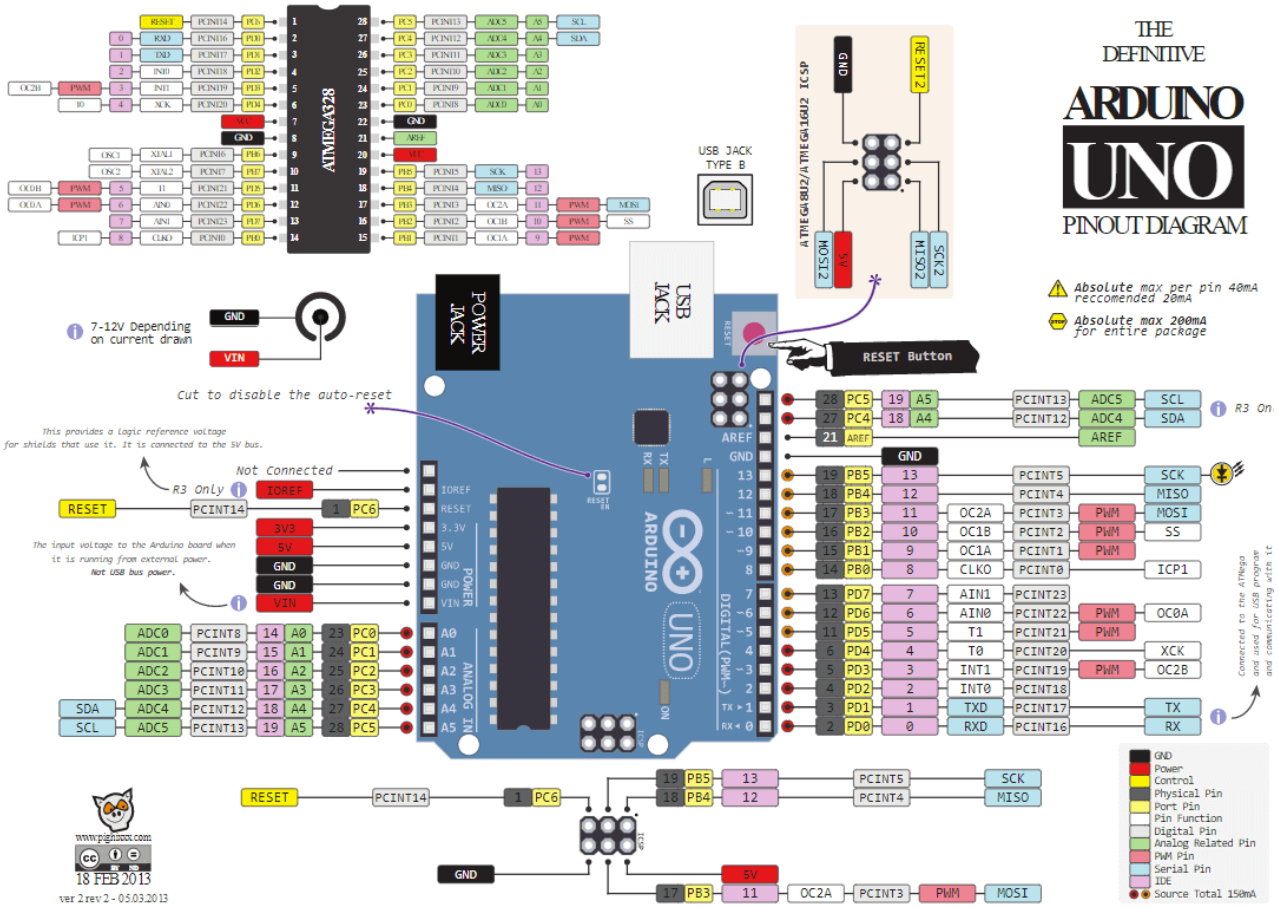
Configure external interrupts

- Write interrupt routine

```
ISR(INT0_vect)
{
  // cod întrerupere externă PD2 /INT0
  // verificare tranziție pe front crescător, descrescător sau oricare
  // (după cum este configurat INT0)
}

ISR(PCINT2_vect){
  // cod întrerupere de tip pin change
  if ((PIND & (1 << PD4)) == 0){
     // întrerupere generată de pinul PD4 / PCINT20
     // verificare nivel logic
  }
}
```

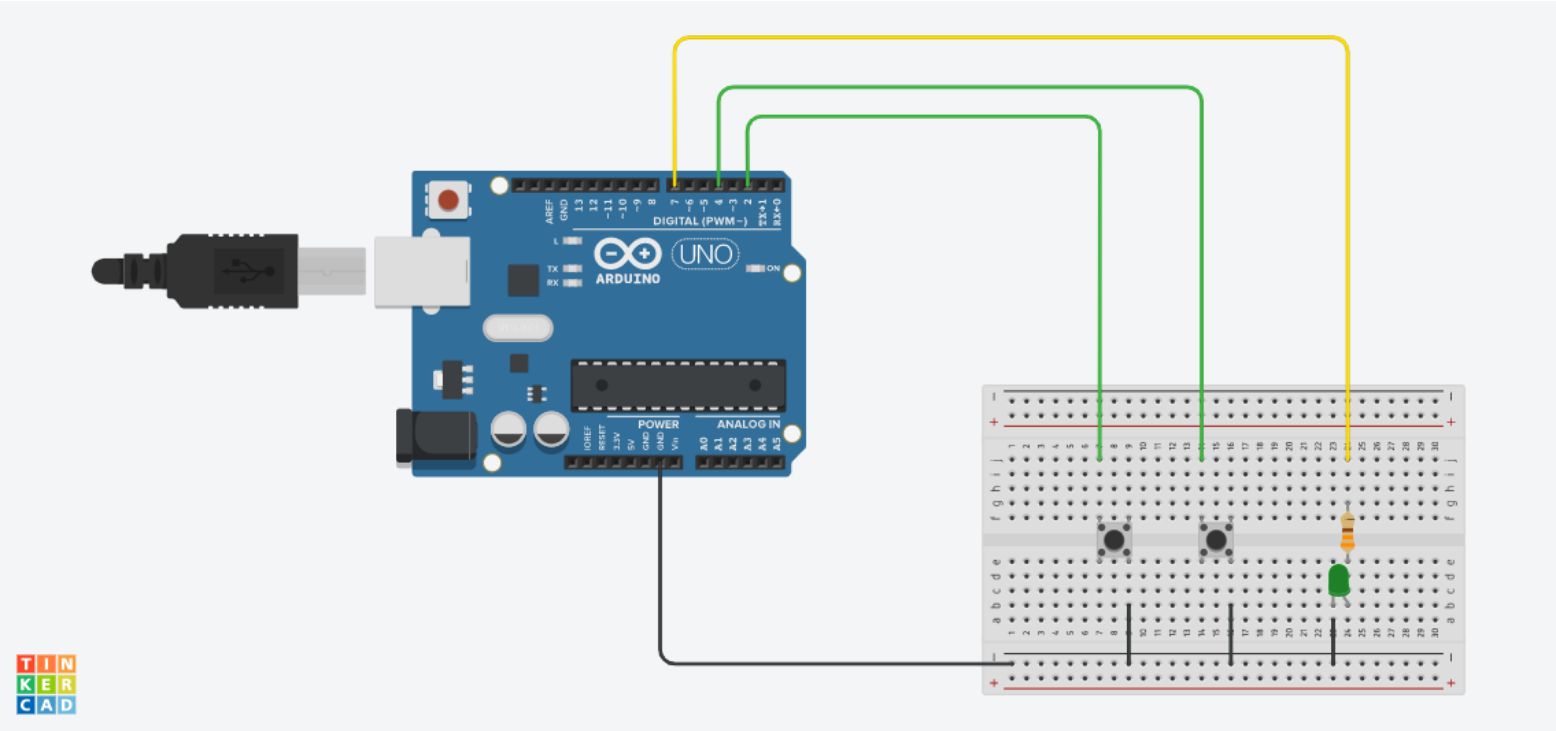
How to avoid delay in ISR

- Use millis() to check time interval
- Use timer instead of millis() function



Hands-on exercise: Button usage … again

- Use external interrupts (INT and/or PCINT) to detect the pressing of a button connected to PD2 (pin 2) and one connected to PD4 (pin 4). Change the status of an LED connected to PD7 (pin 7) in the ISR.



Pulse-width Modulation (PWM)

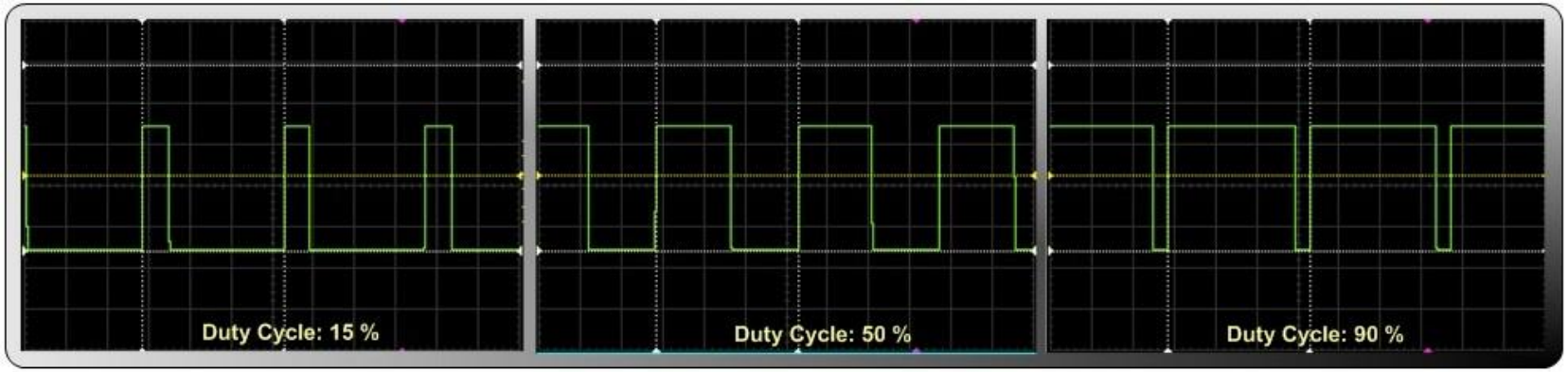
- Control voltage output to a given electronic device
- Switches fast between ON and OFF and vice-versa
- Duty-cycle = The ratio between total period of ON-OFF, represents the voltage received by electronic device
- This means that an LED can be turned on gradually and an engine can be rotated faster or slower

PWM – Principle of operation

- D * Vcc

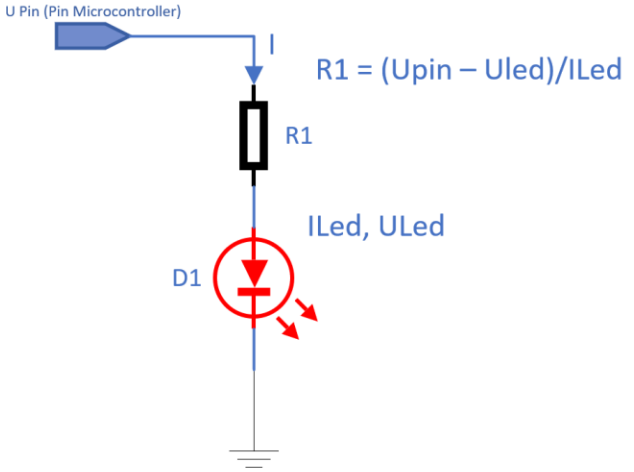
$$D[\%] = \frac{t\_on}{t\_on + t\_off} \cdot 100 = \frac{pulse\_width}{period} \cdot 100$$

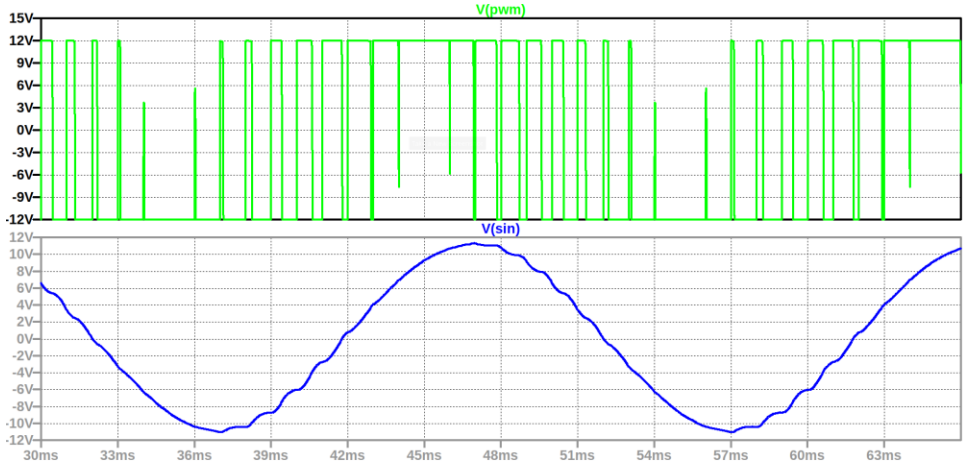
- PWM
  - Hardware: counter, connected to clock unit, reset after each cycle
  - Software: bit-banging, timer with ISR
    - Set Pin High
    - Wait for T_on
    - Set Pin Low
    - Wait For T_off

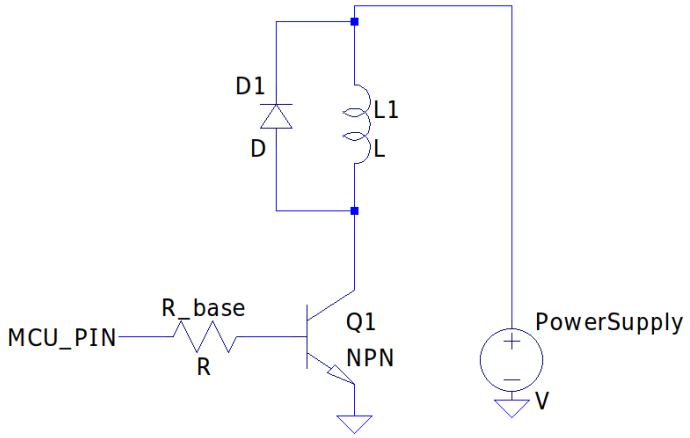


PWM applications

- LED control
- Audio amplification
- Battery charging
- Motor and servo control







Atmega328p PWM

- 6 PWM channels
  - Timer0 - OCR0A, OCR0B (8 bits)
  - Timer2 - OCR2A, OCR2B (8 bits)
  - Timer1 - OCR1A, OCR1B (16 bits)
- **Fast PWM**
- Phase Correct PWM
- Phase and Frequency Correct PWM

$$f_{OCnX} = \frac{f_{clk}}{N \cdot (TOP + 1)} = \frac{f_{clk}}{N \cdot 256}$$

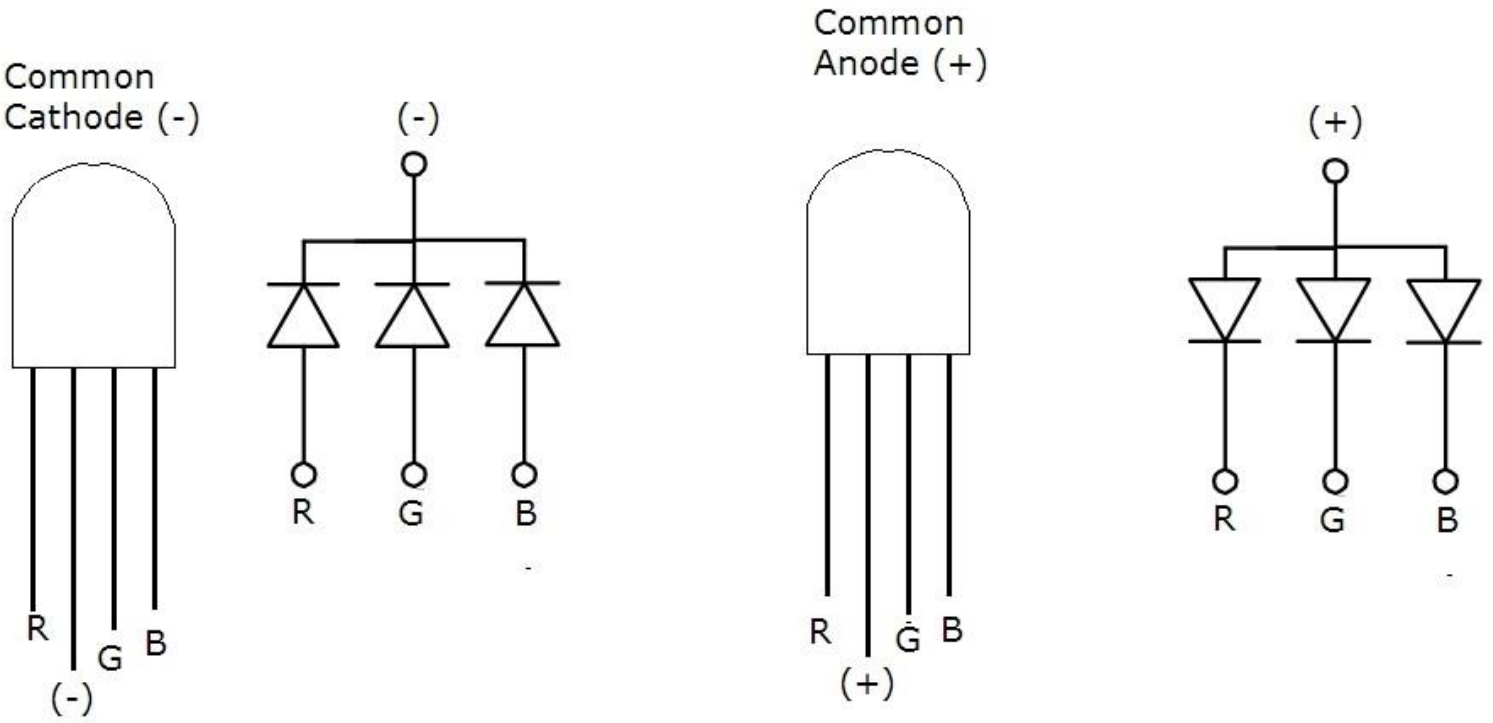
Configure PWM

```
TCCR1A = 0;
TCCR1B = 0;
TCNT1  = 0;
//PB1 output - OC1A este PB1
DDRB |= (1 << PB1);
//pentru modul Fast PWM 8-bit, biții WGMn0 si WGMn2 au valoarea 1
TCCR1A |= (1 << WGM10);
//TCCR1A conține doar biții WGM10 si WGM11, WGM12 și WGM13 se găsesc in TCCR1B
TCCR1B |= (1 << WGM12);
//pentru modul non-inverting, COM1A1 = 1 și COM1A0 = 0
TCCR1A |= (1 << COM1A1);
//pentru Prescaler de 1024 scriem 1 pe CS12 si CS10
TCCR1B |= (1 << CS12) | (1 << CS10);
//Pragul la care se schimbă semnalul pentru a obține un factor de umplere de 0.5
//Deoarece în acest mod TOP este 0xFF, OCR1A va fi 50 * 255 / 100 = 127
OCR1A = 127;
sei();
```

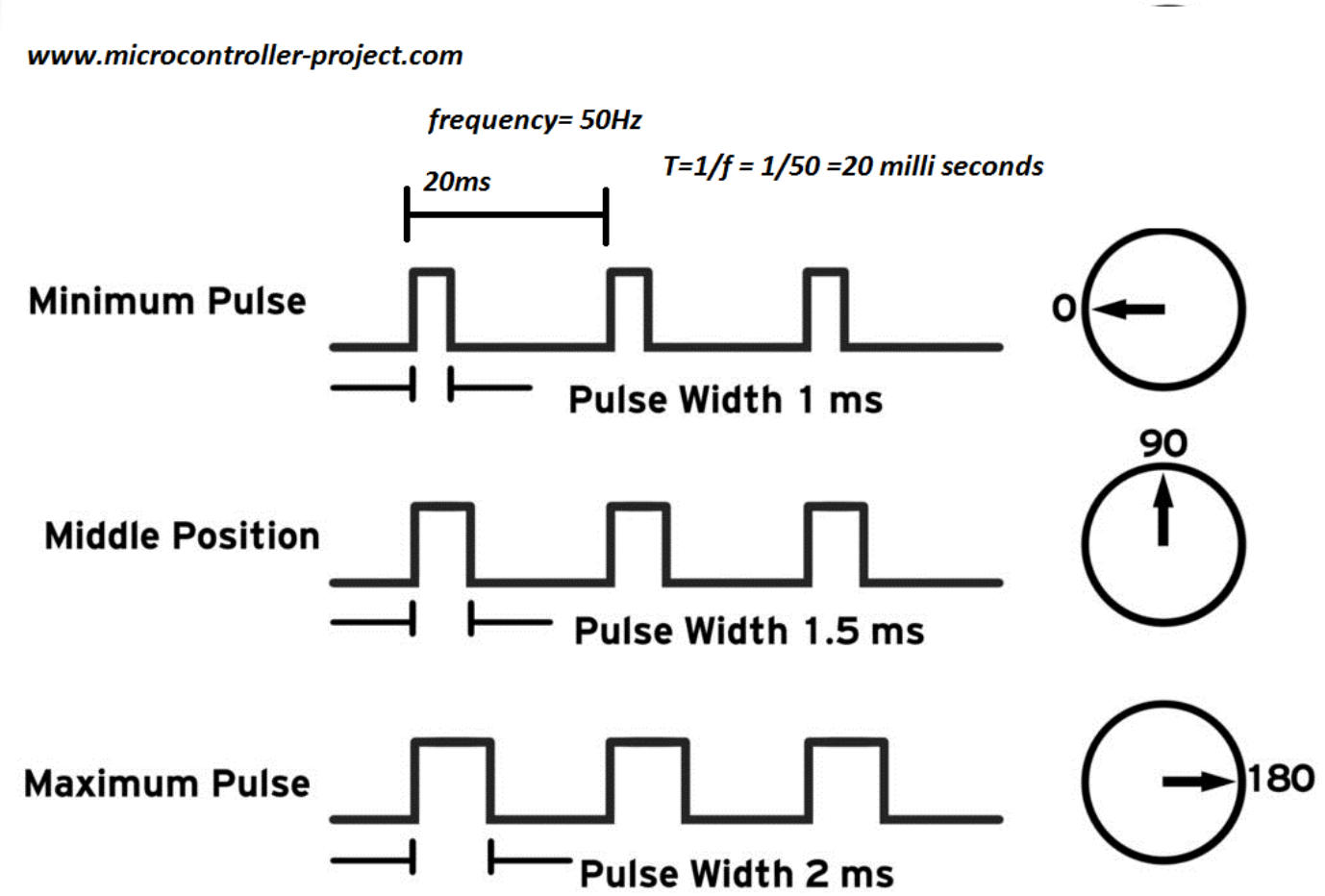
AnalogWrite

Pin Arduino	Pin Atmega328p	Timer output	Frecvența PWM (default)
5	PD5	OC0B (Timer0)	980 Hz
6	PD6	OC0A (Timer0)	980 Hz
9	PB1	OC1A (Timer1)	490 Hz
10	PB2	OC1B (Timer1)	490 Hz
11	PB3	OC2A (Timer2)	490 Hz
3	PD3	OC2B (Timer2)	490 Hz

RGB LED



Servo motors



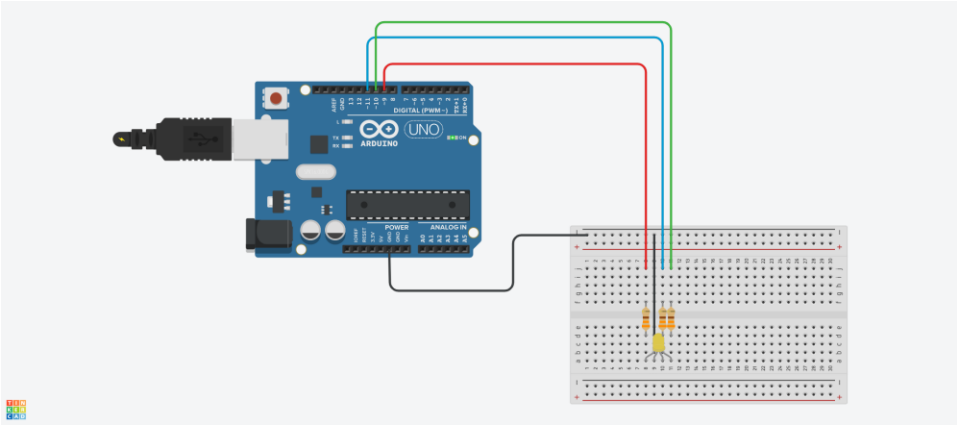
Arduino servo example

```
#include<Servo.h>

Servo myservo; // creates the servo object

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  for (int pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (int pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

Hands-on exercise: PWM RGB LED

```
void setup() {
  // Start off with the LED off.
  setColorRGB(0,0,0);
}

void loop() {
  unsigned int rgb[3];

  // Start off with red.
  rgb[0] = 255;
  rgb[1] = 0;
  rgb[2] = 0;

  // Choose the colors to increment and decrement.
  for (int dec = 0; dec < 3; dec += 1) {
    int inc = dec == 2 ? 0 : dec + 1;

    // cross-fade the two colors.
    for (int i = 0; i < 255; i += 1) {
      rgb[dec] -= 1;
      rgb[inc] += 1;
      setColorRGB(rgb[0], rgb[1], rgb[2]);
      delay(5);
    }
  }
}

void setColorRGB(unsigned int red, unsigned int green, unsigned int blue) {
  analogWrite(9, red);
  analogWrite(10, green);
  analogWrite(11, blue);
}
```

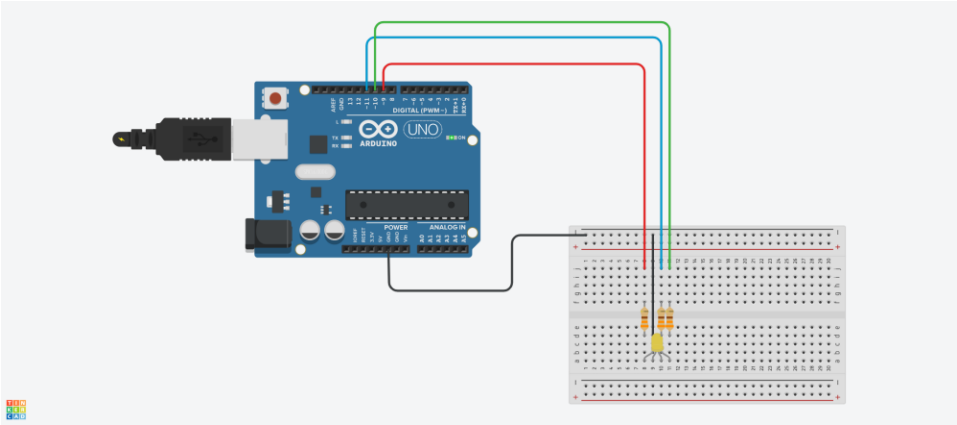
- Connect cathode to pins 9, 10 and 11 using 330ohm resistors and run the following program:



Hands-on exercise: PWM RGB LED Serial

```
int parseCSV(char* inputString, int *outputArray, int outputArraySize) {
  char *pch;
  int val = 0;
  int index_serial_data = 0;
  pch = strtok(inputString, ",");

  while (pch != NULL && pch != "\n") {
    sscanf (pch, "%d", &val);
    outputArray[index_serial_data] = val;
    index_serial_data++;
    if (index_serial_data == outputArraySize) {
      break;
    }
    pch = strtok(NULL, ",");
  }
  return index_serial_data;
}
```

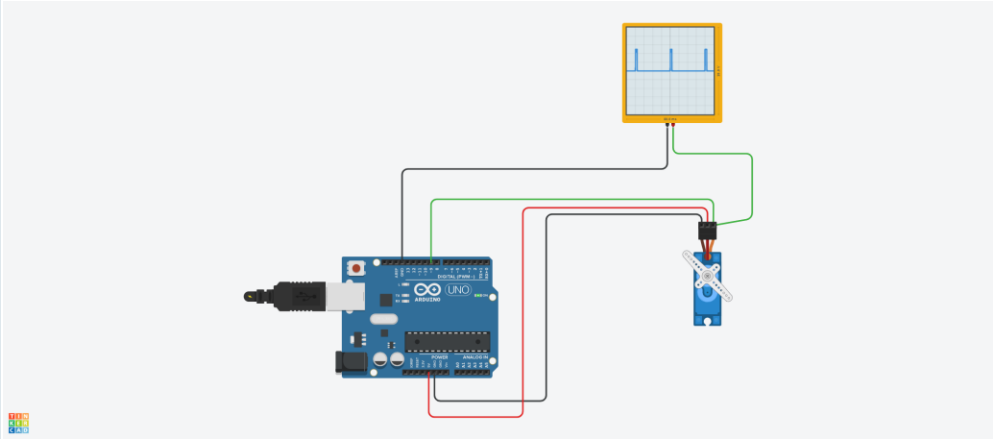
- Modify previous code to HSV color format
- Write on serial RGB values in the format (R,G,B) format 0-255



Hands-on exercise: PWM Servo Sweep

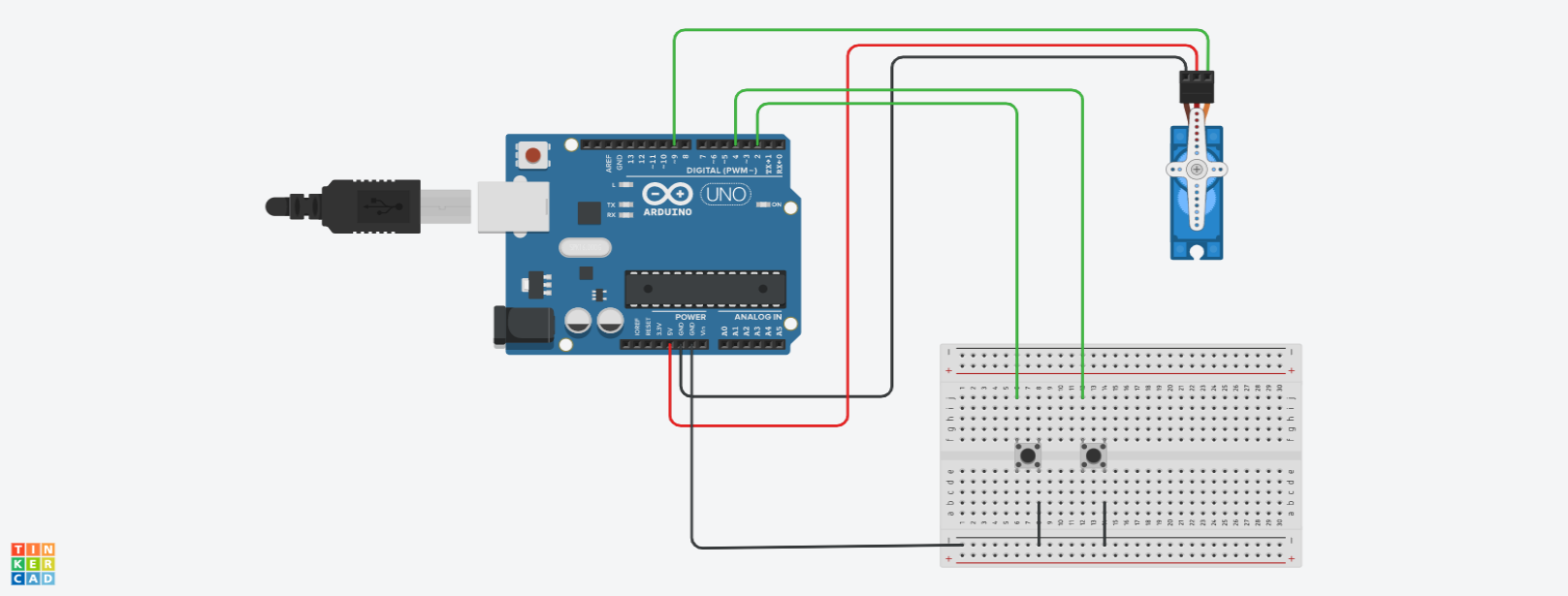
```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
  // test led
  DDRD |= (1 << PD7);
  PORTD &= ~(1 << PD7);
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

- Connect servo to pin 9 and sweep it from 0 to 180 and back.
- Use also Tinkercad and its oscilloscope tool to simulate setup



Hands-on exercise: PWM Servo-Button

- Using an interrupt connected to button pin try to modify incrementally the servomotor position
- Pay attention to (min, max) limits in order to avoid breaking the servo



Closing remarks

- Servo
- HSV
- RGB LED
- CSV
- External interrupts
- Debouncing
- AnalogWrite
- FastPWM