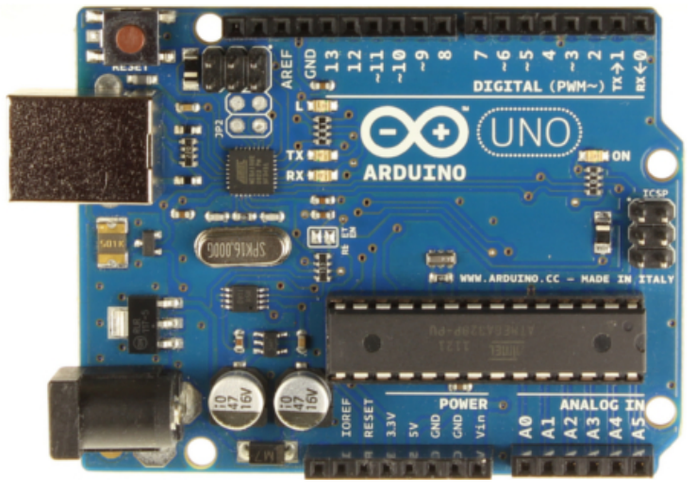
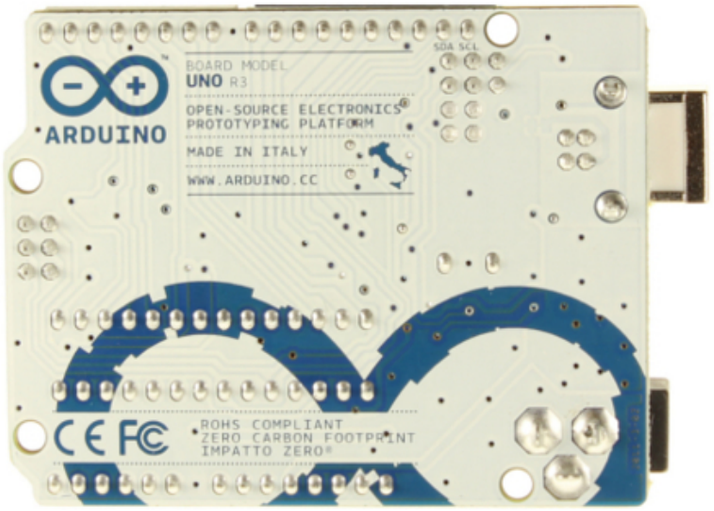
Arduino Timers

Reference: Russell Chapter 7

Timers in Arduino





- Arduino's **wiring library** has many useful time related built in functions: delay(), millis() and micros() and delayMicroseconds().
- The PWM analogWrite(), the tone() and the noTone() function also uses time related function.
- Even the Servo library uses timers.

Wiring delay() function

```
int ledPin = 13;                  // LED connected to digital pin 13

void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop()
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(1000);                  // waits for a second
}
```

- Pauses the program for the amount of time (in miliseconds) specified as parameter.
- There are 1000 milliseconds in a second.
- Syntax: delay(**ms**)
- Where ms is the number of milliseconds to pause (unsigned long).

Wiring analogWrite() function

```
int ledPin = 9;      // LED connected to digital pin 9
int analogPin = 3;   // potentiometer connected to analog pin 3
int val = 0;         // variable to store the read value

void setup()
{
  pinMode(ledPin, OUTPUT);   // sets the pin as output
}

void loop()
{
  val = analogRead(analogPin);   // read the input pin
  analogWrite(ledPin, val / 4);  // analogRead values go from 0 to 1023, analogWrite values from 0 to 255
}
```

- Function analogWrite() writes an analog value (PWM wave) to a pin.

- Can be used to light a LED at varying brightnesses or drive a motor at various speeds.

Wiring analogWrite() function

- After a call to analogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite().

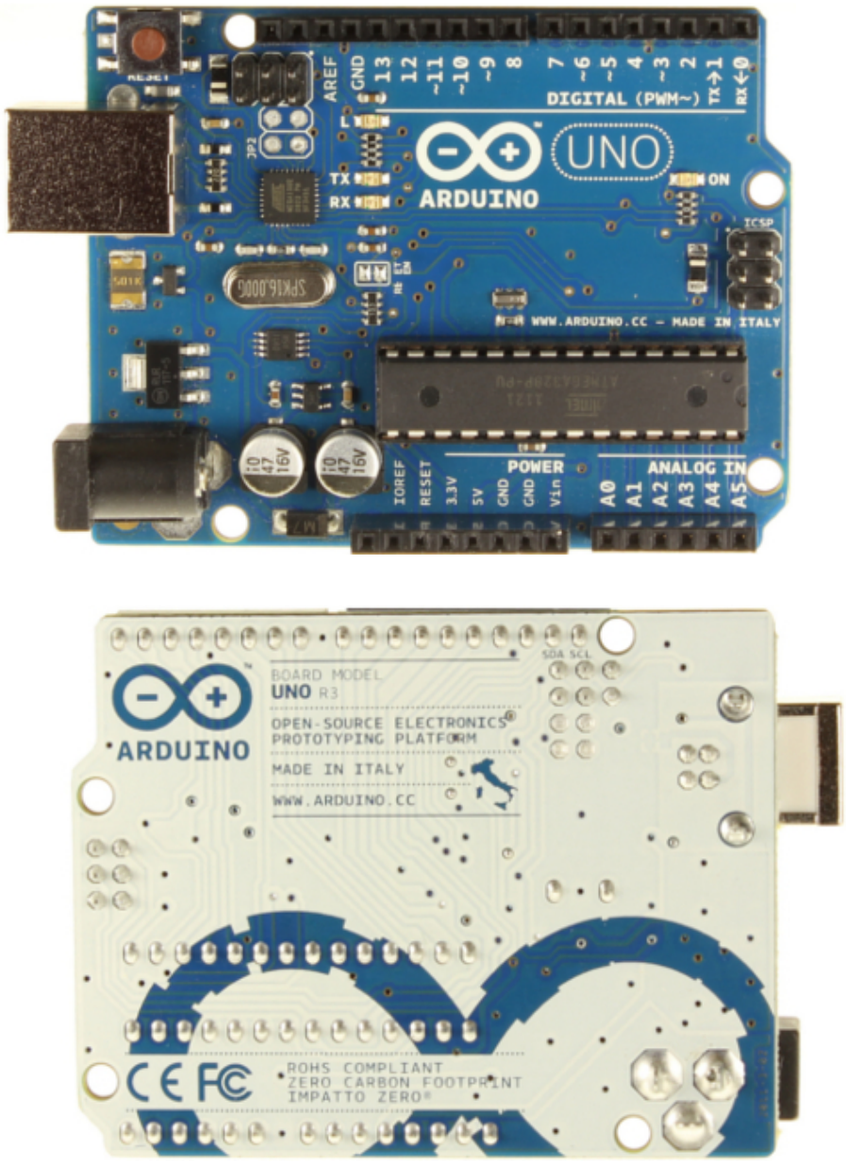
- The frequency of the PWM signal is approximately 490 Hz.

  - On the Arduino UNO, this function works on pins 3, 5, 6, 9, 10, and 11.
  - You do not need to call pinMode() to set the pin as an output before calling analogWrite().

  - The analogWrite function has nothing whatsoever to do with the analog pins or the analogRead function.

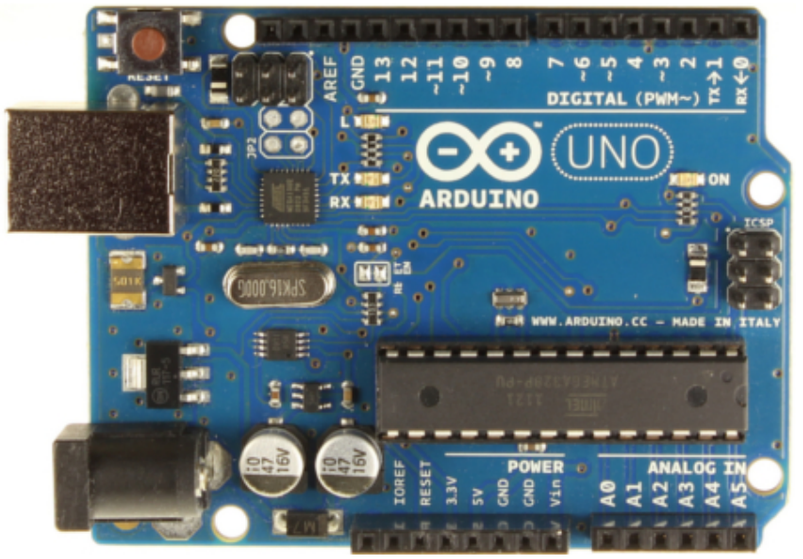
What is a timer?

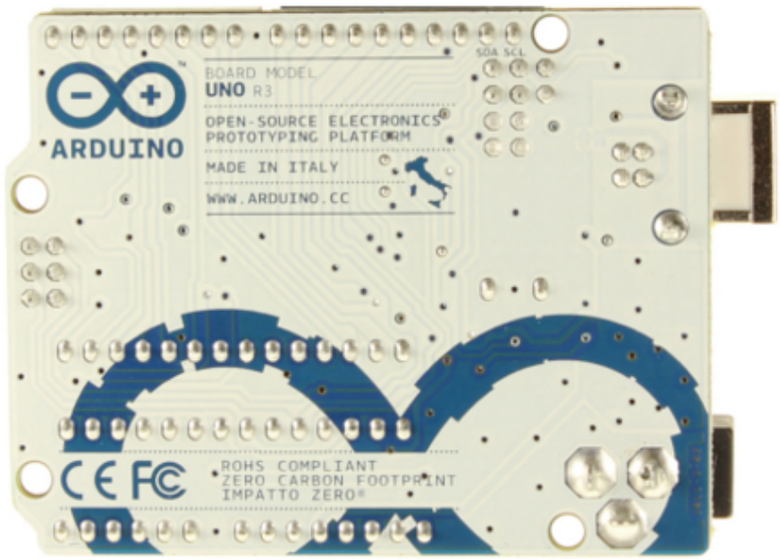
- A timer is a piece of hardware built-in the Arduino controller.
- It is like a clock, and can be used to measure time events.
- The timer can be programmed by some special **registers**.
- You can configure the pre-scalar for the timer, the mode of operation and many other things.
- The controller of the Arduino is the Atmel AVR ATmega328.



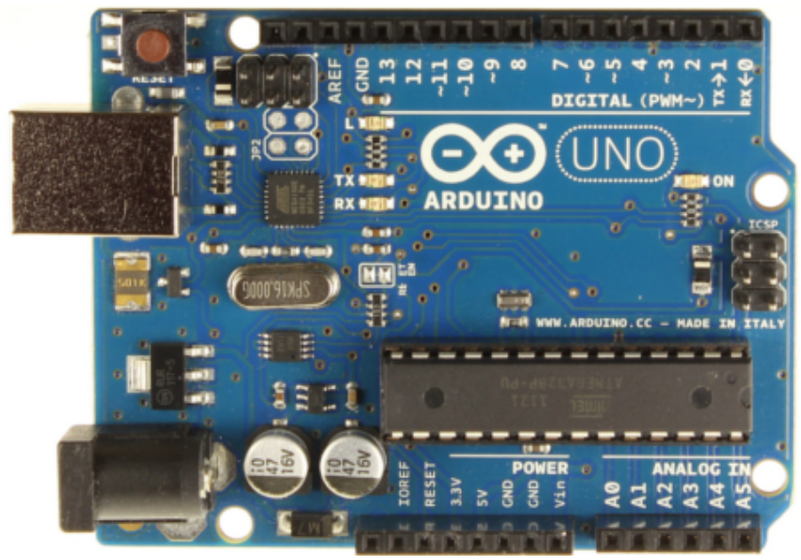
What is a timer?

- ATmega328 has **3 timers**, called timer0, timer1 and timer2.
- Timer0 and timer2 are 8bit timers, while timer1 is a 16bit timer.
- More bits means higher timing resolution.
- Remember, 8bits has 256 different states while 16bit has 65536.
- All timers depends on the system clock of your Arduino system, which is 16MHz for the Arduino UNO.





Timer0

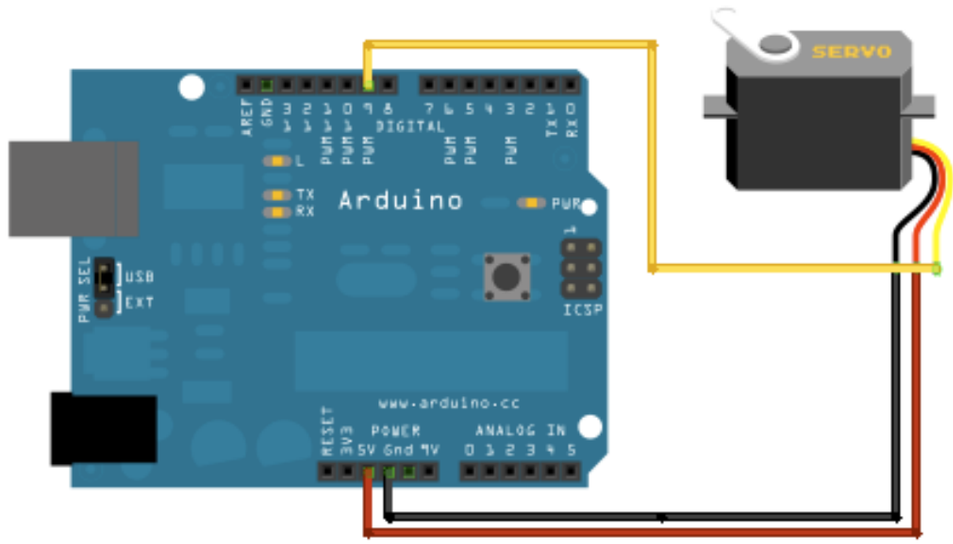


- Timer0 is a 8bit timer.
- In the Wiring libraries, timer0 is used for the commonly used timer functions.
- You know... delay(), millis() and micros().
- **Warning**: If you change the timer0 registers, this may influence the Arduino timings function.
- Changes are not permanent: If accidentally mess up a timer register, just upload a new program.

Timer1

- Timer1 is a 16bit timer.
- In the Wiring libraries, this timer is used for the servo library.

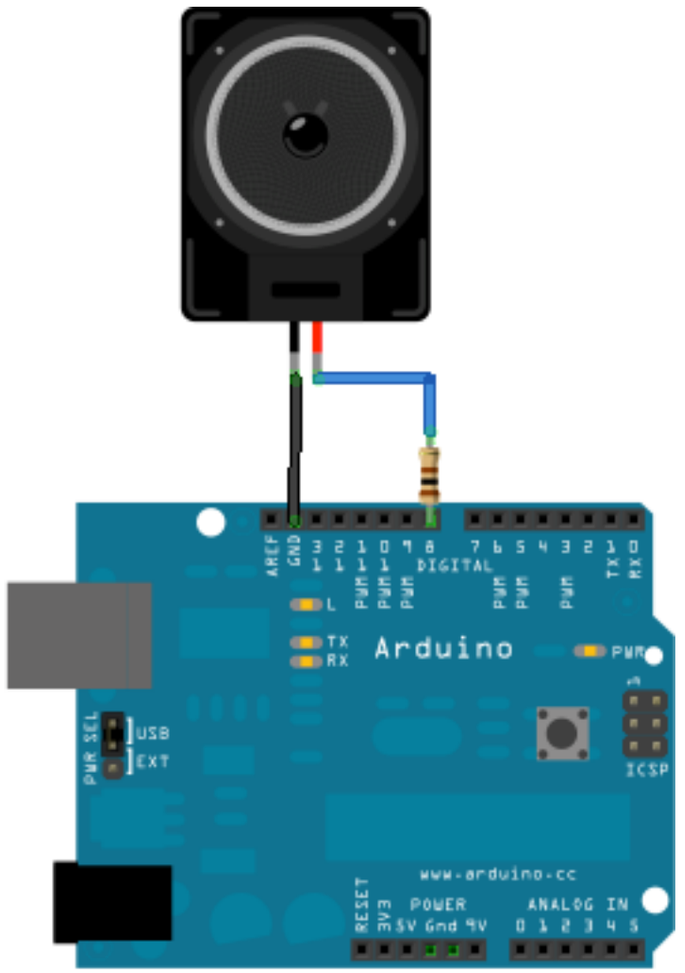
```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
                // a maximum of eight servo objects can be created

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

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


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>=1; 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
  }
}
```



Timer2

- Timer2 is a 8bit timer.
- In the Wiring libraries, this timer is used for the tone() function.

```
// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3,NOTE_G3, NOTE_A3, NOTE_G3,0, NOTE_B3, NOTE_C4};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4,4,4,4,4 };

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000/noteDurations[thisNote];
    tone(8, melody[thisNote],noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(8);
  }
}

void loop() {
  // no need to repeat the melody.
}
```

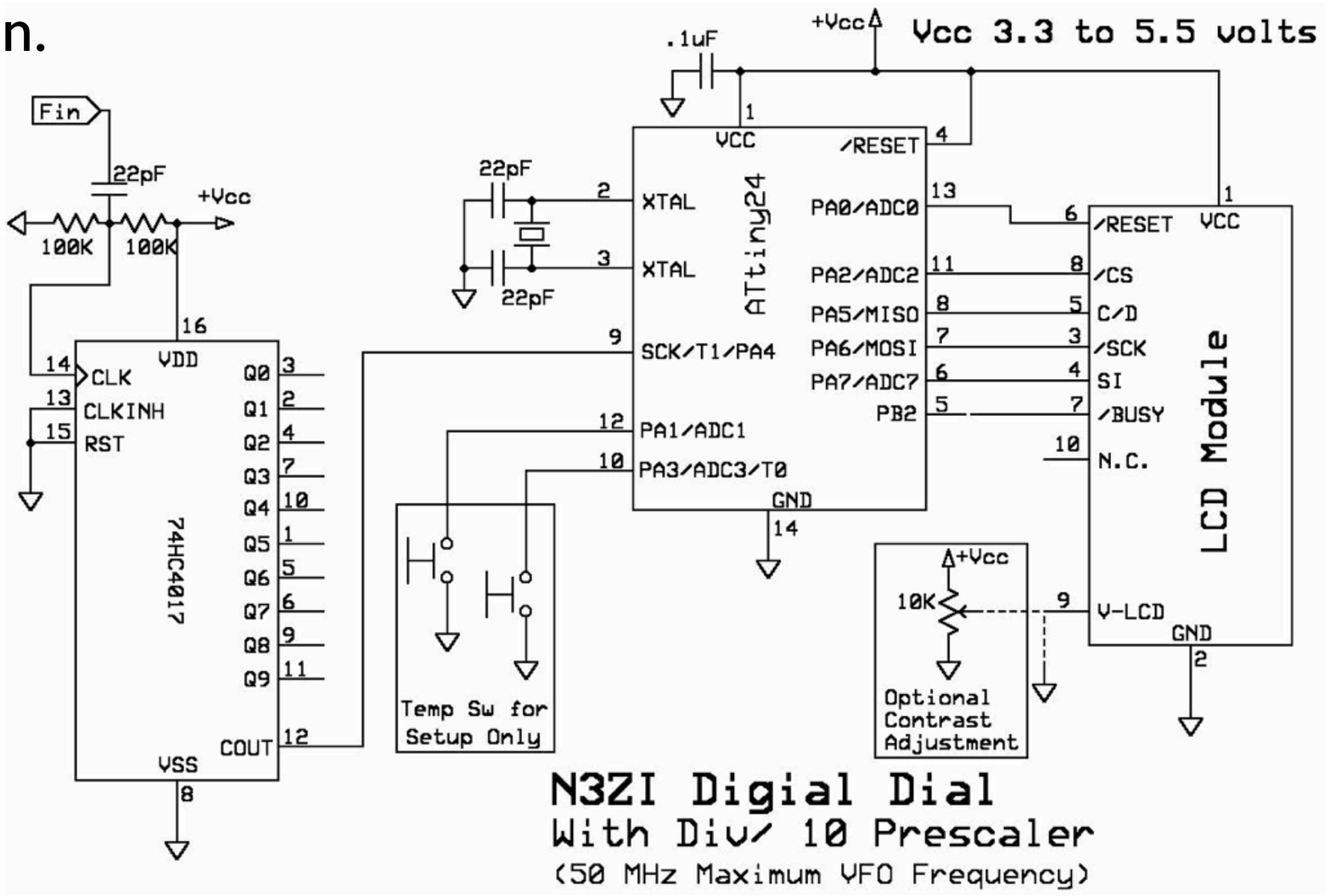


Frequency review (from wikipedia)

- Frequency is the number of occurrences of a repeating event per unit time.
- The period is the duration of one cycle in a repeating event, so the period is the reciprocal of the frequency.
- f = 1 / T
- For example, if a newborn baby's heart beats at a frequency of 120 times a minute, its period (the interval between beats) is half a second.
- Click for a neat frequency animation.

Prescaler or count-divider

A prescaler is an electronic counting circuit used to reduce a high frequency electrical signal to a lower frequency by integer division.



How to define your own timer

- In the ATmega328 there are 6 important timer registers, that can be interfaced with to adjust the Timer behavior.
- **TCCRx** - Timer/Counter Control Register. Prescaler is configured here.
- **TCNTx** - Timer/Counter Register. The actual timer value is stored here.
- **OCRx** - Output Compare Register
- **ICRx** - Input Capture Register (only for 16 bit timer)
- **TIMSKx** - Timer/Counter Interrupt Mask Register. To enable/disable timer interrupts.
- **TIFRx** - Timer/Counter Interrupt Flag Register. Indicates a pending timer interrupt.

Clock select and timer frequency

- Different clock sources can be independently selected for each timer.
- To calculate the timer frequency (e.g. 2Hz using timer1) you need:
  - CPU frequency: 16 MHz for Arduino UNO.
  - Maximum timer counter value: 256 for 8bit timer, 65536 for 16bit.
  - A prescaler value: either 256 or 1024.
  - Divide CPU frequency with a **prescaler** (16000000 / **256** = 62500).
  - Divide result through the desired frequency (62500 / 2Hz = 31250).
  - Verify the result against the maximum timer counter value (31250 < 65536 success). If fail, choose bigger prescaler.

Calculating the prescaler

This can be done on the website: http://www.et06.dk/atmega_timers/

Calculate ATmega Timer/Counter and prescaler values directly on this page.

Timer/Counter0 (8 bit) and Timer/Counter1 (16 bit)

Clocks

f_{clock} = 16 M Hz

f_{timer} = 2 Hz

Factor 8e6

Calculate

Prescalers

Prescaler	1	8	64	256	1024
Decimal	8000000	1000000	125000	31250	7813
Hexadec.	0x7A1200	0xF4240	0x1E848	0x7A12	0x1E85
Error [%]	0	0	0	0	-5.12

Green - Timer/Counter0.
Orange - Timer/Counter1.
Red - Cannot be used.

Registers (prescaler NaN and prescaler 256)

FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
							0xNAN	OCR0
COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
								OCR1AH
							0x7A12	OCR1AL
OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK

As calculated on previous slide.

Blinking LED with compare match interrupt

```
#define ledPin 13

void setup(){
  pinMode(ledPin, OUTPUT);

  // initialize timer1
  noInterrupts();           // disable all interrupts
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1  = 0;

  //compare match register 16MHz/256/2Hz
  OCR1A = 31250;
  TCCR1B |= (1 << WGM12);   // CTC mode
  TCCR1B |= (1 << CS12);    // 256 prescaler
  TIMSK1 |= (1 << OCIE1A);  // enable timer compare interrupt
  interrupts();             // enable all interrupts
}

//timer compare interrupt service routine
// toggle LED pin
ISR(TIMER1_COMPA_vect){
  digitalWrite(ledPin, digitalRead(ledPin) ^ 1);
}

void loop()
{
  // your program here...
}
```

- Timer1 is in CTC mode (**clear timer on compare match**).
- The timer is configured for a frequency of 2Hz.

Prescalers

Prescaler	1	8	64	256	1024
Decimal	8000000	1000000	125000	31250	7813
Hexadec.	0x7A1200	0xF4240	0x1E848	0x7A12	0x1E85
Error [%]	0	0	0	0	-5.12

Registers (prescaler NaN and prescaler 256)

FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
							0xNAN	OCR0
COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
								OCR1AH
							0x7A12	OCR1AL
OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK