BLM4021 Gömülü Sistemler

Doç. Dr. Ali Can KARACA

ackaraca@yildiz.edu.tr

Yıldız Teknik Üniversitesi – Bilgisayar Mühendisliği

Sunum 9 – Genel Amaçlı Zamanlıyıcılar ve Motor Kontrolü

- Zamanlayıcılar
- PWM çıkışı
- Motor çeşitleri ve kontrolleri
- Uygulama Örnekleri

Ders Materyalleri

Gerekli Kaynaklar:

- Derek Molloy, Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux, Wiley, 2016.
- M. Wolf, Computers as Components: Principles of Embedded Computing System Design, Elsevier, 2008.

Yardımcı Kaynaklar:

- D. Zhu, T. Sifleet, T. Nunnally, Y. Huang, Analog to Digital Converters, Lecture Notes.
- Philip Koopman, Time and Counters; Watchdog Timers, Embedded System Eng., 2016.
- Farid Farahmand, Chapter- 3, Embedded Systems with ARM Cortex-M, 2018.
- Tolga Ayav, Embedded Control Systems, Data Acquistion and Digital Signal Processing.
- Simon Monk, Raspberry Pi Cookbook, O'Reilly.

Haftalık Konular

Hafta	Teorik	Laboratuvar
1	Giriş ve Uygulamalar, Mikroişlemci, Mikrodenetleyici ve Gömülü sistem kavramlarının açıklanması	Grupların oluşturulması & Kitlerin Testi
2	Bir Tasarım Örneği, Mikroişlemci, Mikrodenetleyici, DSP, FPGA, ASIC kavramları	Kitlerin gruplara dağıtımı + Raspberry Pi Kurulumu
3	16, 32 ve 64 bitlik mikrodenetleyiciler, pipeline, PIC ve MSP430 özellikleri	Raspberry Pi ile Temel Konfigürasyon
4	PIC ve MSP430 özellikleri	---- Resmi Tatil ---
5	ARM tabanlı mikrodenetleyiciler ve özellikleri	Uygulama 1 – Raspberry Pi ile Buzzer Uygulaması
6	ARM Komut setleri ve Assembly Kodları-1	Uygulama 2 – Raspberry Pi ile Ivme ve Gyro Uygulaması
7	ARM Komut setleri ve Assembly Kodları-2, Raspberry Pi vers. ve GPIO'ları	Uygulama 3 – Raspberry Pi ile Motor Kontrol Uygulaması
8	*Vize Sınavı*	
9	Haberleşme Protokolleri (SPI, I2C ve CAN)	Proje Soru-Cevap Saati - 1
10	Sensörlerden Veri Toplama, Algılayıcı, ADC ve DAC	Proje Soru-Cevap Saati - 2
11	Zamanlayıcı, PWM ve Motor Sürme	Proje kontrolü - 1
12	Gerçek Zaman Sistemler ve temel kavramlar	Proje Kontrolü - 2
13	Gerçek zaman İşletim Sistemleri	Mazeret sebepli son proje kontrollerinin yapılması
14	Nesnelerin İnterneti	
15	*Final Sınavı*	

For more details -> Bologna page: http://www.bologna.yildiz.edu.tr/index.php?r=course/view&id=9463&aid=3

Measuring the time is very critical…

Users complain iPhone clock bungles time change

Instead of springing forward, some iPhone clocks fell back

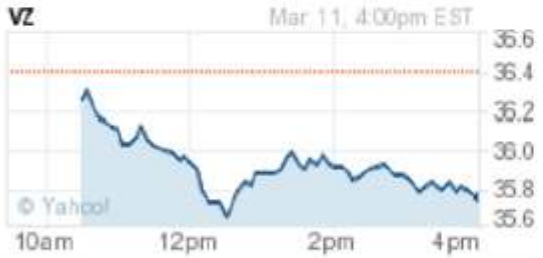
Associated Press

Share 18 retweet 2 Email Print

Companies: Verizon Communications Inc. Com

Related Quotes

Symbol	Price	Change
VZ	35.85	-0.55



On Sunday March 13, 2011, 7:41 pm EDT

NEW YORK (AP) -- It's hard enough to get your bearings when the time changes twice a year. It's all but impossible when your phone starts playing tricks on you, too.

Users of Apple's iPhone peppered Twitter and blogs with complaint Sunday when their phones bungled the one-hour "spring forward" to daylight savings time that went into effect overnight Saturday.

One user complained of missing church, another of almost missing yoga. One called her iPhone stupid and several just asked for help.

It turns out some users' phones fell back one hour instead of springing forward, making the time displayed on the iPhone two hours off.

This is just the latest clock woe for Apple's chic iPhone. A clock glitch prevented alarms from sounding on New Year's Day, causing slumbering revelers to oversleep. The devices also struggled to adjust to the end of daylight savings time back in November.

Measuring the time is very critical…

F-22 Raptor Date Line Incident

- **February 2007**
  - A flight of six F-22 Raptor fighters attempts to deploy to Japan
  - $360 million per aircraft
    - (Perhaps $120M RE, rest is NRE)
  - Crossing the International Date
    - No navigation
    - No communications
    - No fuel management
    - Almost everything gone!
    - Escorted to Hawaii by tankers
    - If bad weather, might have caused loss of aircraft

[DoD]

  - Cause: "It was a computer glitch in the millions of lines of code, somebody made an error in a couple lines of the code and everything goes."

Communication, fuel subsystems, and navigation systems were rendered useless and repeated "reboots" were of no help.

Luckily, the fleet had clear skies and refueling tankers to guide them back to Hawaii.

Credit by P. Koopman

Measuring the time is very critical…

Do Not Set The Date On Your iPhone To Jan. 1, 1970

By Mary Beth Quirk February 12, 2016

Blast from the past.

The original Macintosh introduced the world to computers, forever changing the way people experience technology, and allowing people to do things that were never possible before. With this easter egg, warp back in time with a classic Macintosh theme to relive the magic on your iPhone. Change the date on your iPhone to January 1, 1970, press and hold the power button to reboot your device, and prepare for a wild ride!

Fakety fake fake fake.

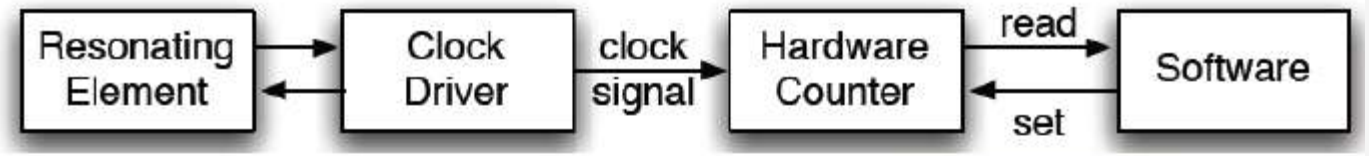
While it might be tempting to take a "wild ride" into the past, do not set the date on your iPhone to Jan. 1, 1970, despite what a hoax image circulating recently says. That is, unless your idea of a wild ride is having a phone you can't use anymore.

Time in Embedded Systems

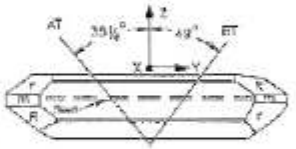
Where do we need accurate time?

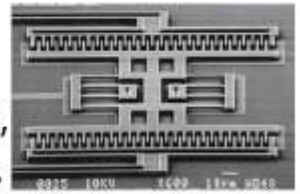
- Scheduling of computation
  - Scheduler in operating systems
  - Real time operating systems
- Signal sampling and generation
  - Audio sampling at 44.1 kHz
  - TV/video generation (sync, vsync)
  - Pulse Width Modulated (PWM) signals
- Communication
  - Media Access Control (MAC) protocols
  - Modulation
- Navigation
  - GPS

Clock Generation



- Resonating element/Driver:
  - Quartz crystal can be made to resonate due to Piezoelectric effect.
    - Resonate frequency depends on length, thickness, and angle of cut.
    - Issues: Very stable (<100ppm) but not all frequencies possible.
  - MEMS Resonator
    - Arbitrary frequency, potentially cheap, susceptible to temperature variations.
  - Others:
    - Inverter Ring, LC/RC circuits, Atomic clock, and many more.





Credit by Farid Farahmand

Drift Rate of An Oscillator - Example

- **A gizmo has a crystal oscillator running at 32,768 Hz + 0.002%**
  - 32,768 is a standard watch crystal frequency (15-bit divider gives you 1 Hz)
    - (.002% is a $2*10^{-5}$ drift rate)
  - The product specification requires accuracy of 2 seconds/day
  - Assume perfect software counting of oscillator clock cycles
  - Will the oscillator meet the specification?

(.00002 sec/sec drift rate * (60 sec * 60 min * 24 hr)

= 1.728 sec drift per day (*so it meets the spec.*)

- How far will it drift over a 2-year battery life?

1.728 sec/day * (365.25 days * 2 years) = 21 minutes drift over 2 years

Counting the Clocks

Time is an integer count of some number of clock "ticks"

- One year @ 2 MHz takes about 47 bits to represent as an integer – too big to be useful for most embedded applications
- But, most applications don't need time to the nearest 1/2,000,000 second
- So, we want time with a bigger granularity than this

Thus, the concept of the timer

- Increment a "timer" once every N CPU clocks (this is a clock "tick")
  - Potentially, tell the CPU to update its software-maintained clock on every timer increment; maybe a 32-bit integer
- Example: Original IBM PC updated time of day 18.2 times/second
  - Windows Forms timer is still that speed (55 msec)
- Many Unix systems have base timers that run at 30 or 60 times/second
  - Why this frequency?

Timers

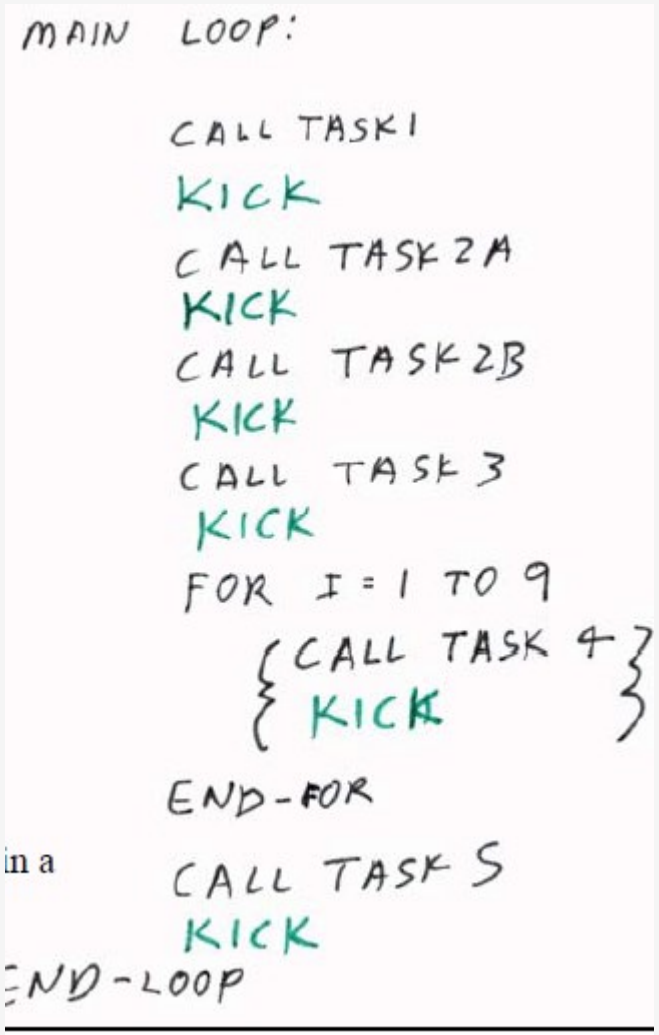
- Microcontrollers may include one or more timers among their peripheral units.
- Timers are generally configurable to greatly enhance their basic functionality.
- They can generate a square signal that can be accommodated to our needs, by adjusting its frequency, duty cycle, or both.
- Timers also have interrupt capabilities, or that of capturing the specific time at which some external event occurs.
- Types of timers: General Purpose timers, PWM, watchdog timers…

WatchDog Timers

- **A common symptom of software problem – system hang**
  - Could be an infinite loop
  - Could be continually chasing a "wild" pointer around
  - Could be corrupted data
  - … but often systems "lock up" or "hang"

- **Good general-purpose remedy – reboot system if it hangs**
  - But, there is no person around to press "ctl-alt-delete"
  - So, let the watchdog timer do it instead
  - BUT realize this ***doesn't*** solve ***all*** problems
    - just some that are nice to address

- **Basic watchdog idea:**
  - Have a hardware timer running all the time (count-down timer)
  - When timer reaches zero, it resets the system
  - Software periodically "kicks" (or "pets") the watchdog, restarting the count
  - If software has "kicked" the watchdog often enough, no reset takes place

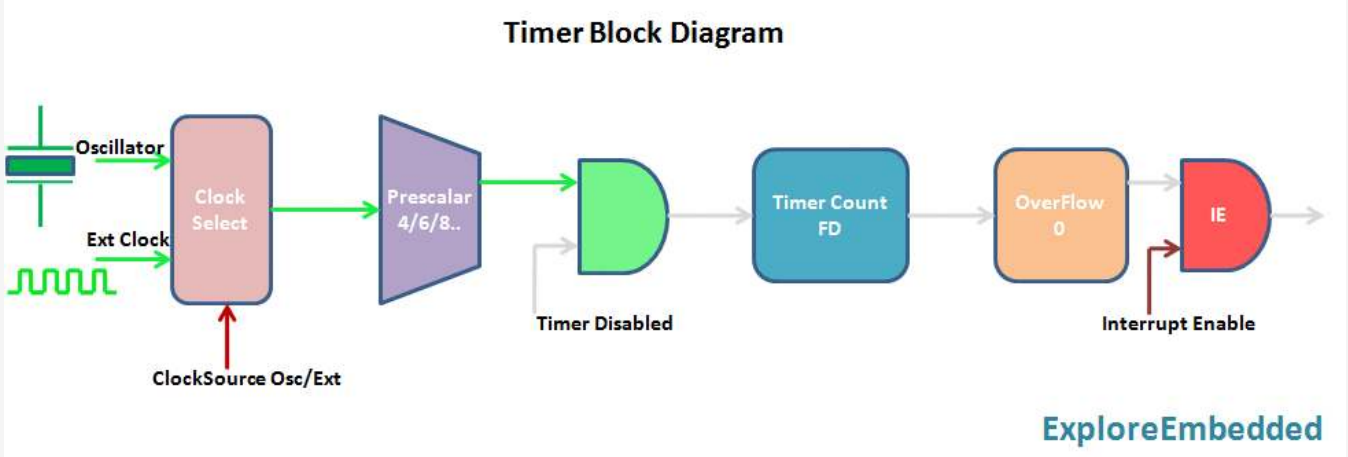


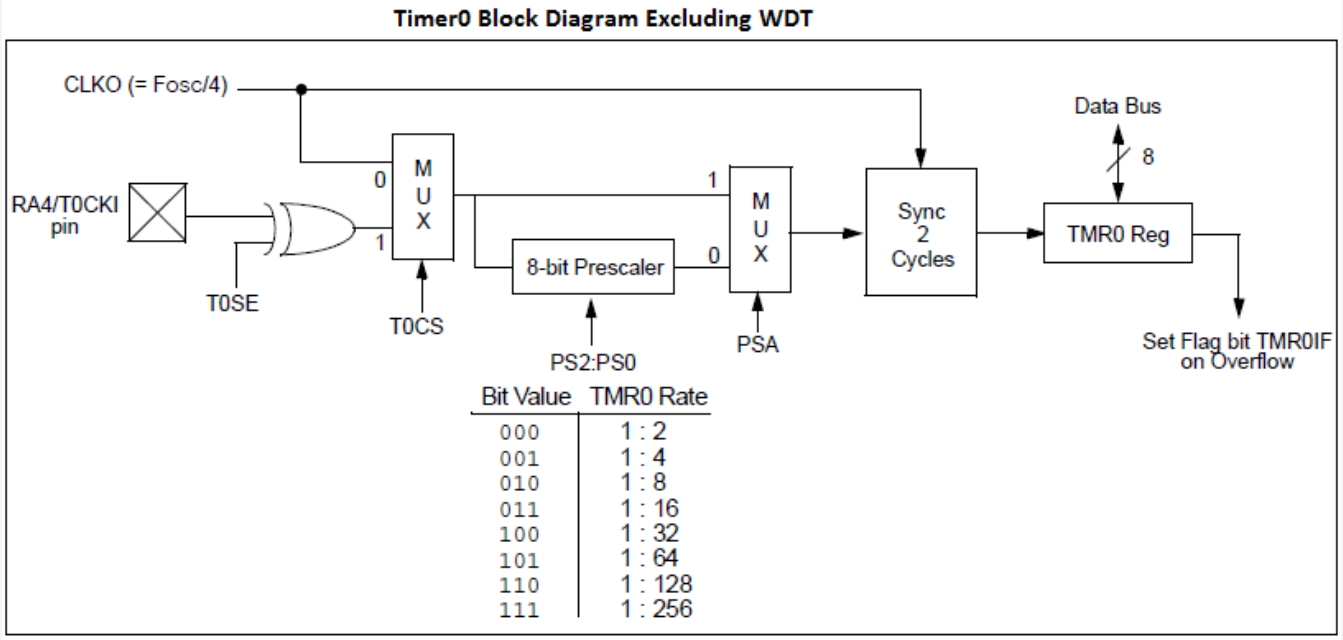
Credit by P. Koopman

WatchDog Timers

- Watchdog timer is a piece of hardware in micro-controller.
- Watchdog timer is used to generates system reset if system gets stuck somewhere i.e. if system goes into endless loop of execution watchdog timer will reset the system to come out of endless loop.
- Watchdog is safety mechanism in embedded system which makes your system reliable

Timer Block Diagram for PIC16F877





T0SE: TMRO Source Edge Select Bit

1: High to low / 0: Low to high

T0CS: TMR0 Clock Source Select

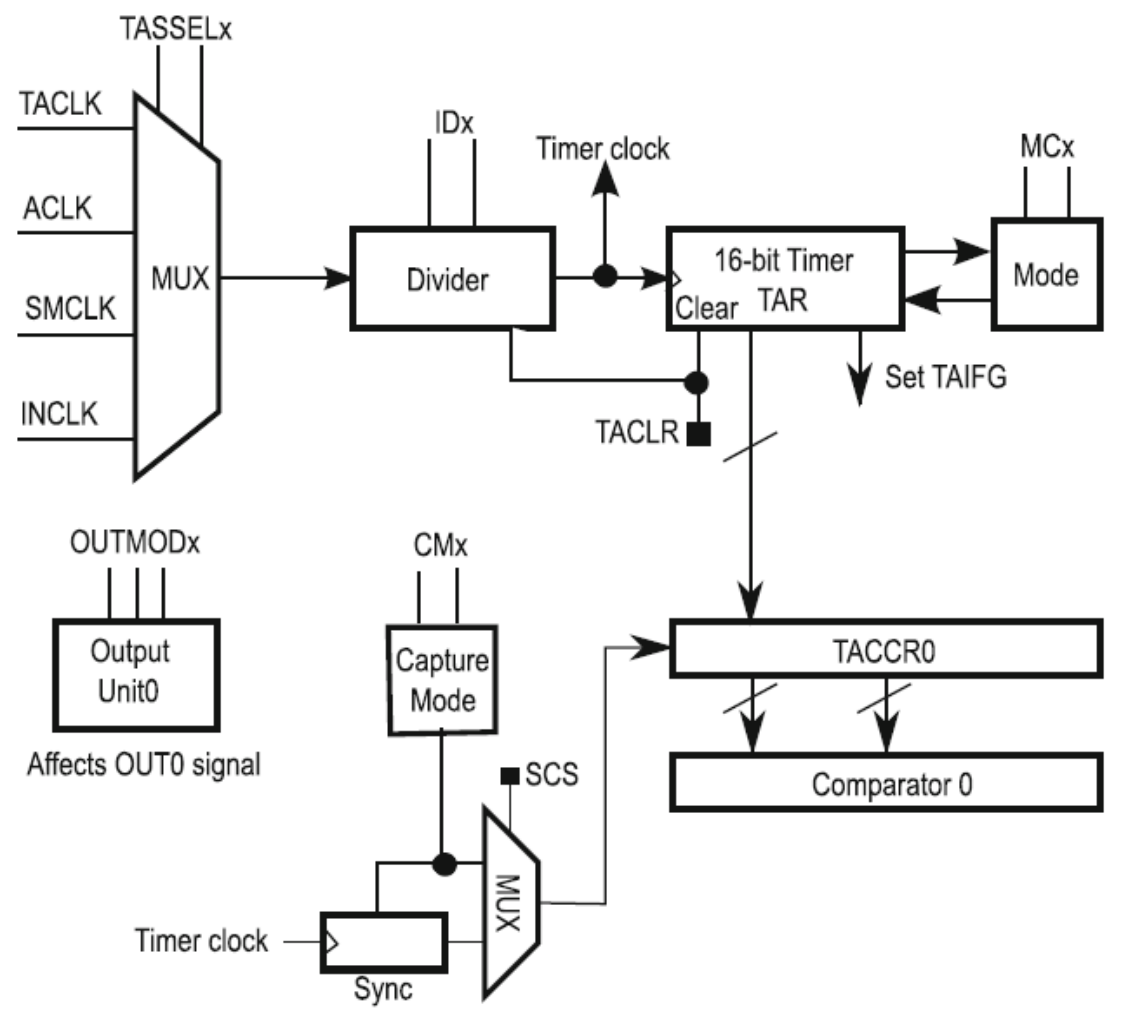
0: internal clock

PSA: Prescaler assignment bit

PS2:PS0: Prescaler rates

TMR0IF: Timer0 Interrupt Flag

Timer Block Diagram for MSP430



TASSELx : Selects clock source

IDx : Determines the frequency division

TACLR: Clears th counter

TAIFG: Timer A Interrupt Flag

TACCR0: Capture&Compare Register

MCx: Counting Up and Down modes

Timer Block Diagram for MSP430

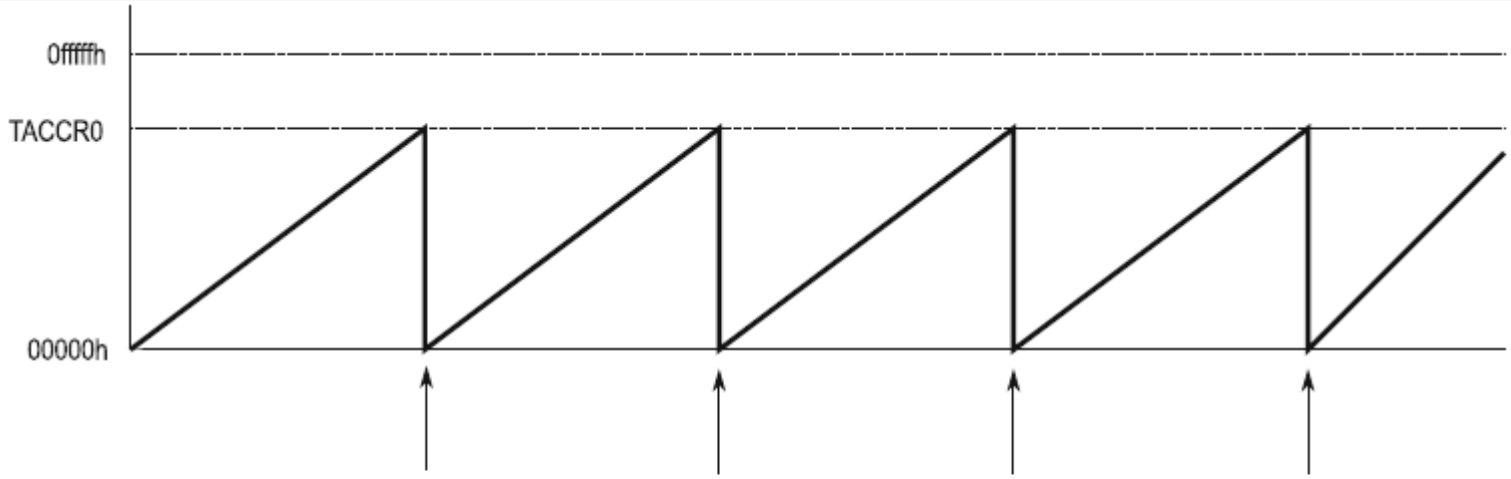
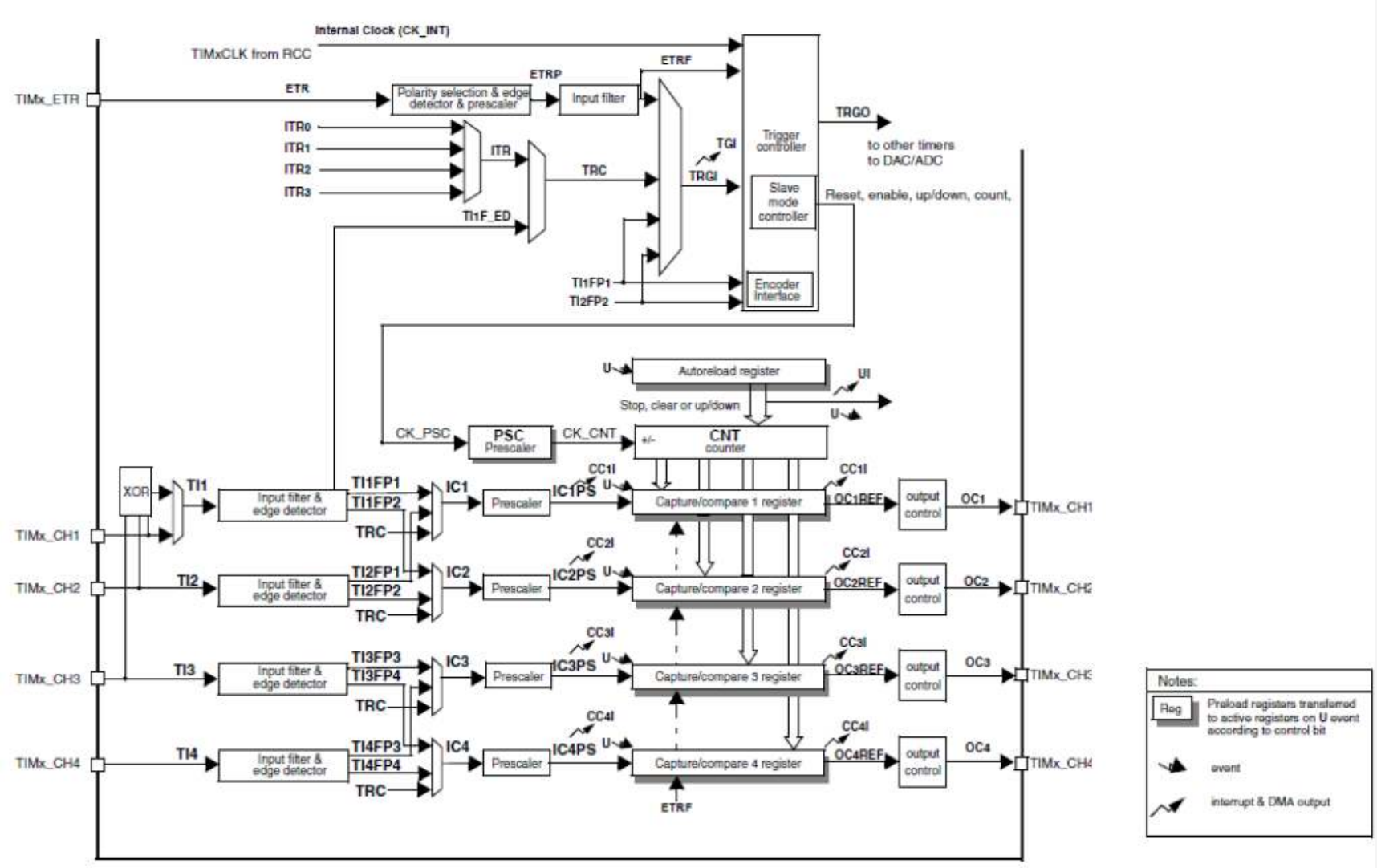
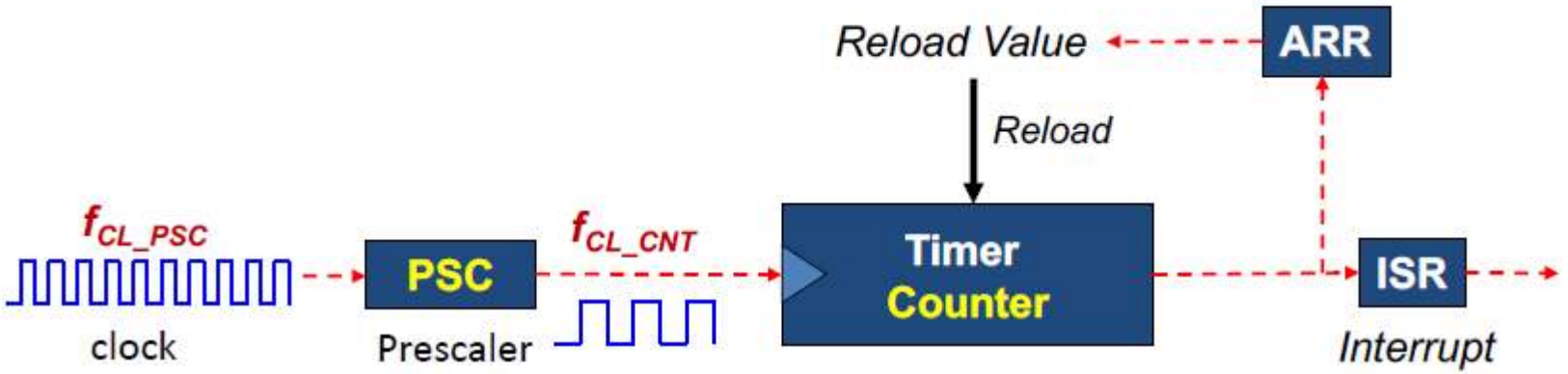


Fig. 7.24 Timer operating in up mode

Timer in ARM microcontrollers

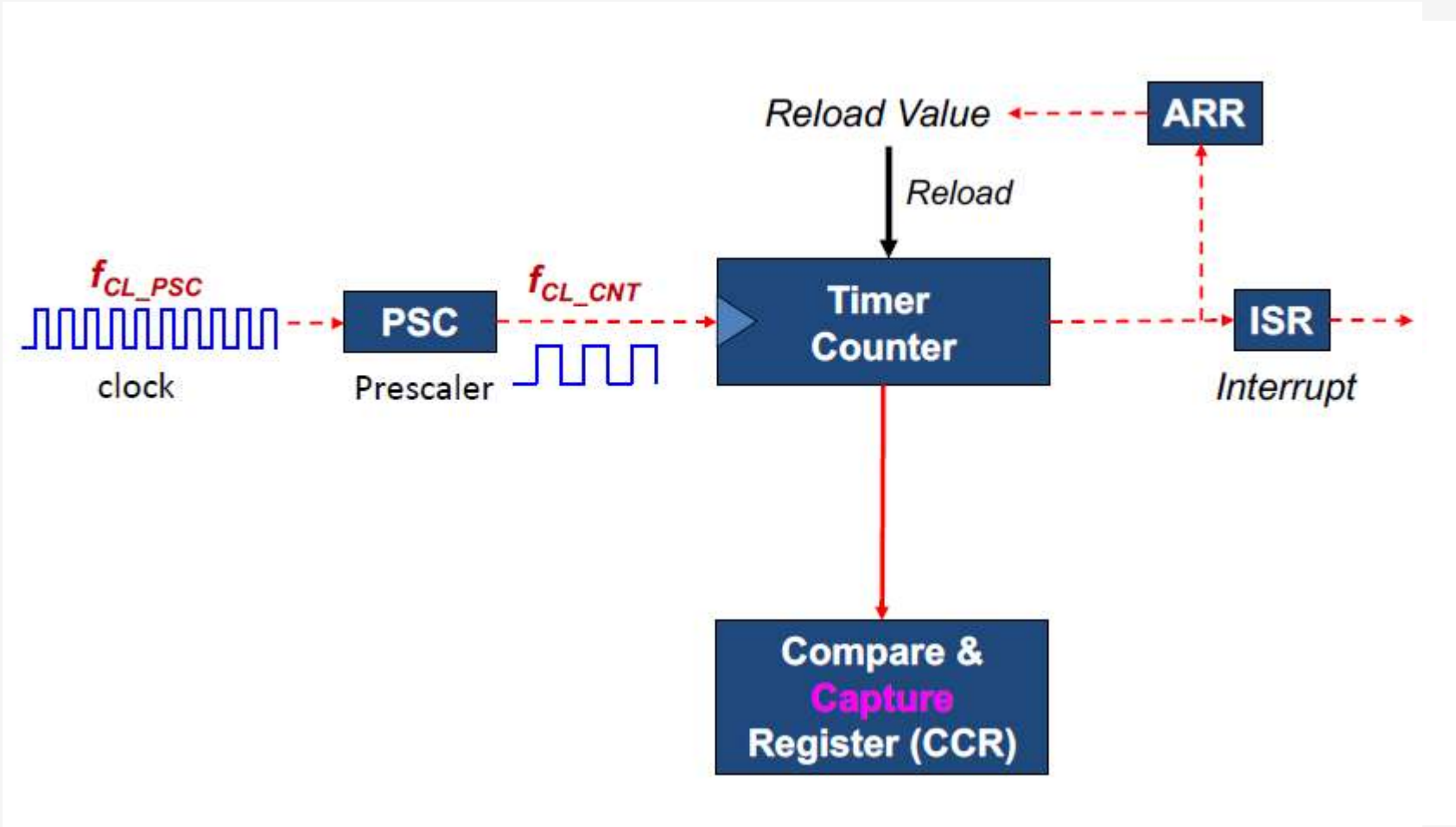


Timer – Clock Calculations



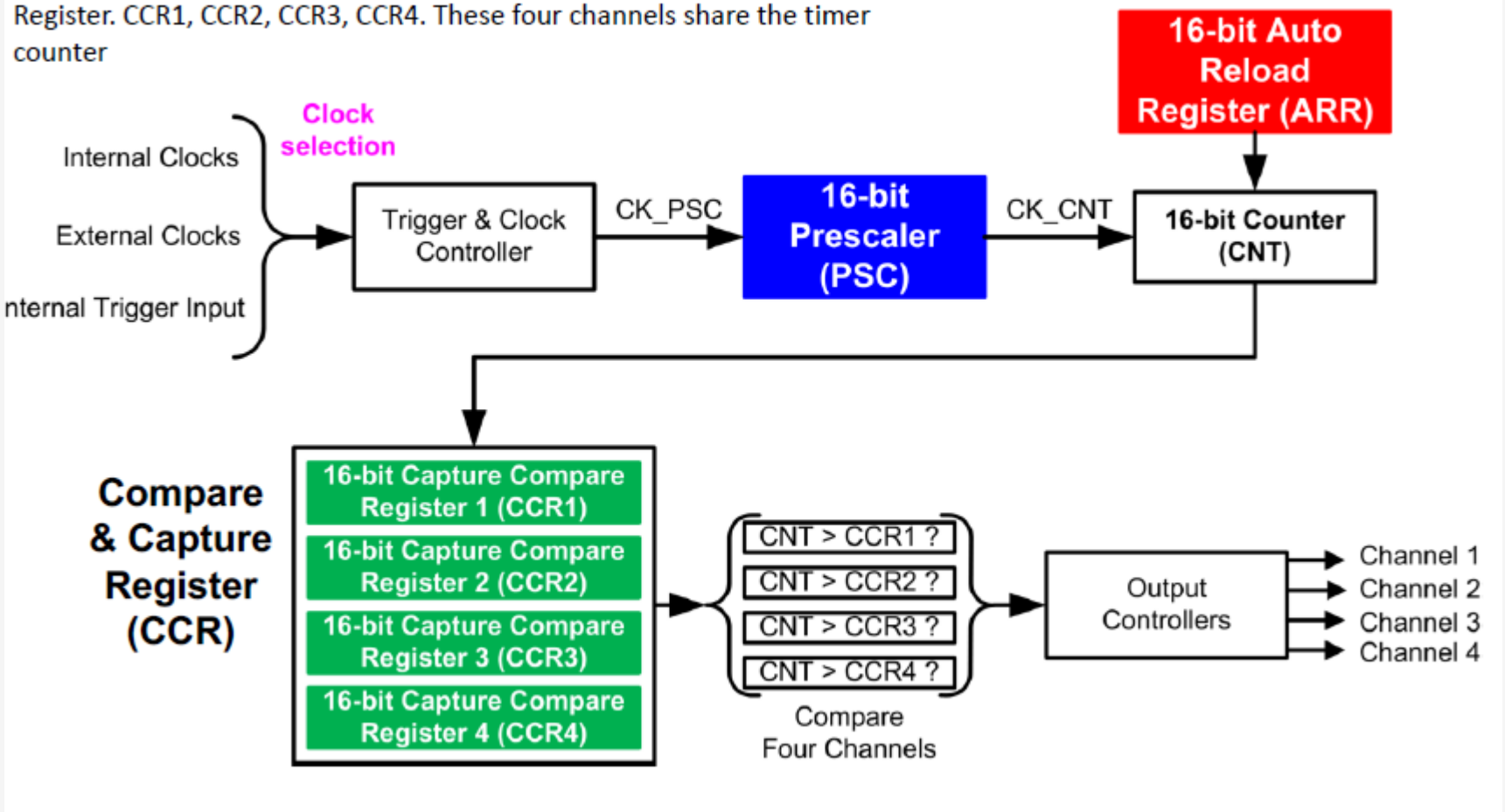
$$f_{CK\_CNT} = \frac{f_{CL\_PSC}}{PSC + 1}$$

Timer – Clock Calculations

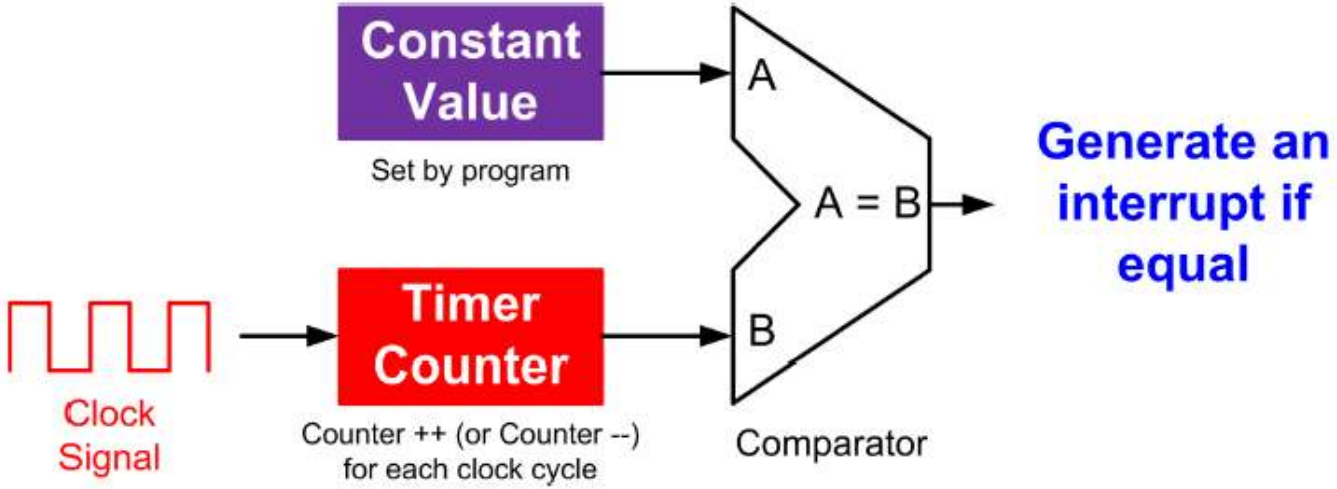


Multi-Channel Outputs

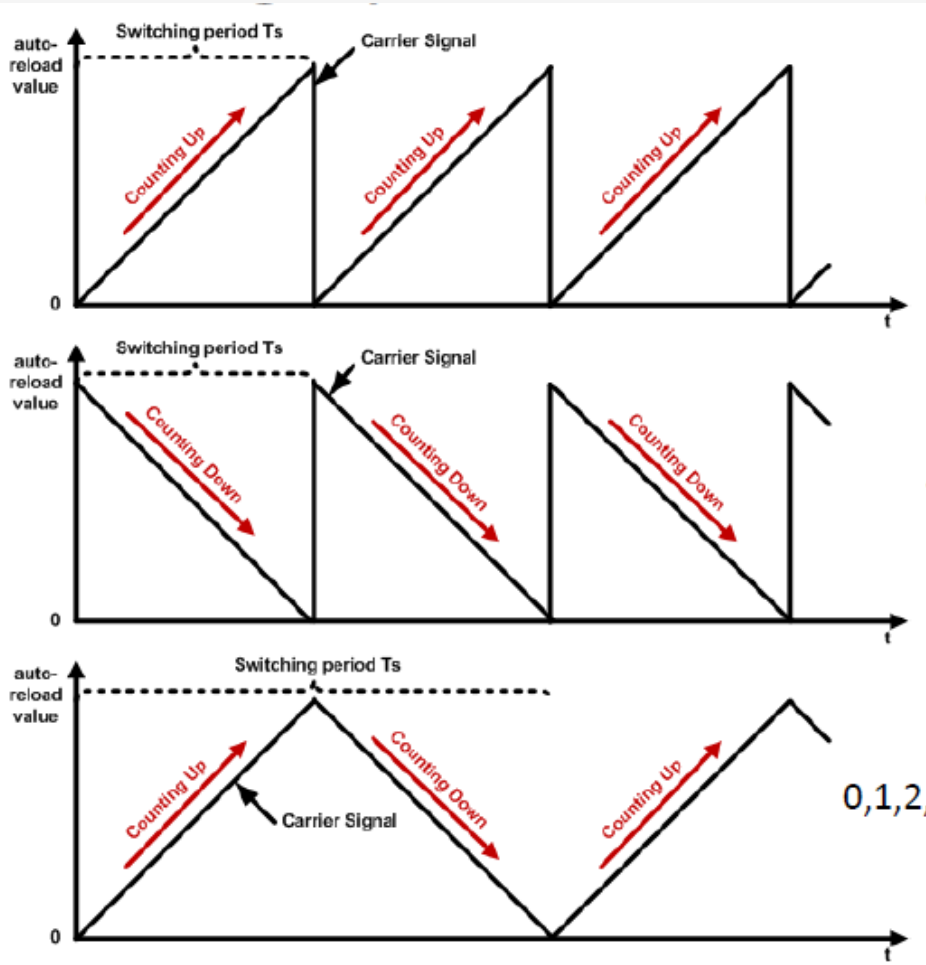
Each timer has four channels. Each channel has its own Compare and Capture Register. CCR1, CCR2, CCR3, CCR4. These four channels share the timer counter



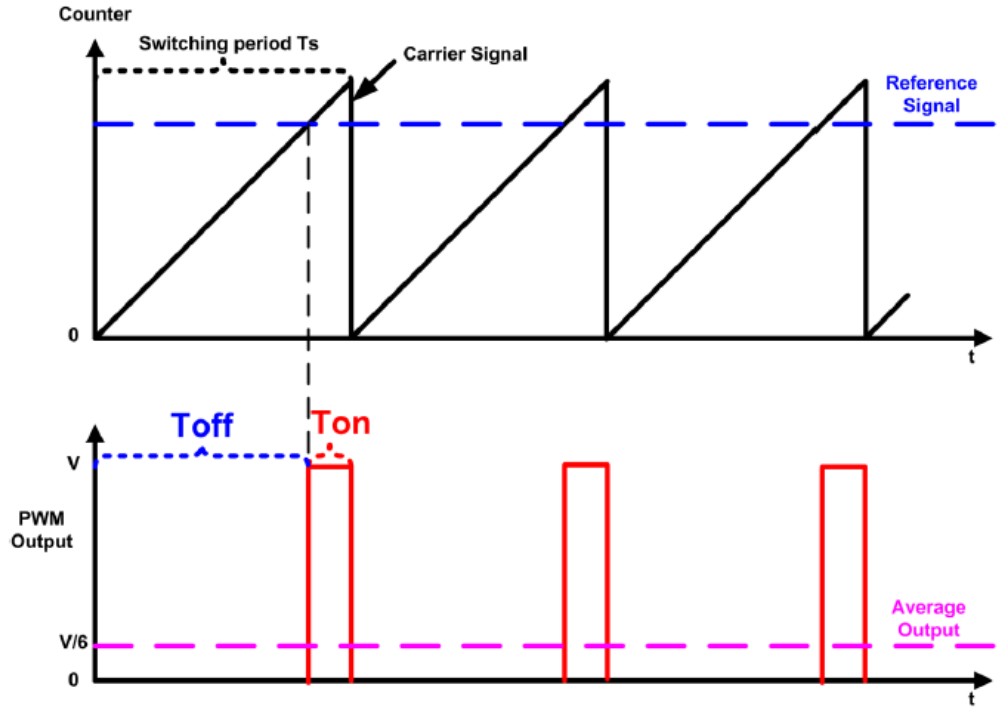
Compare Mode



Output Compare Mode (OCM)	Timer Output (OCREF)
000	Frozen
001	High if CNT == CCR
010	Low if CNT == CCR
011	Toggle if CNT == CCR
100	Forced low (always low)
101	Forced high (always high)



PWM Mode



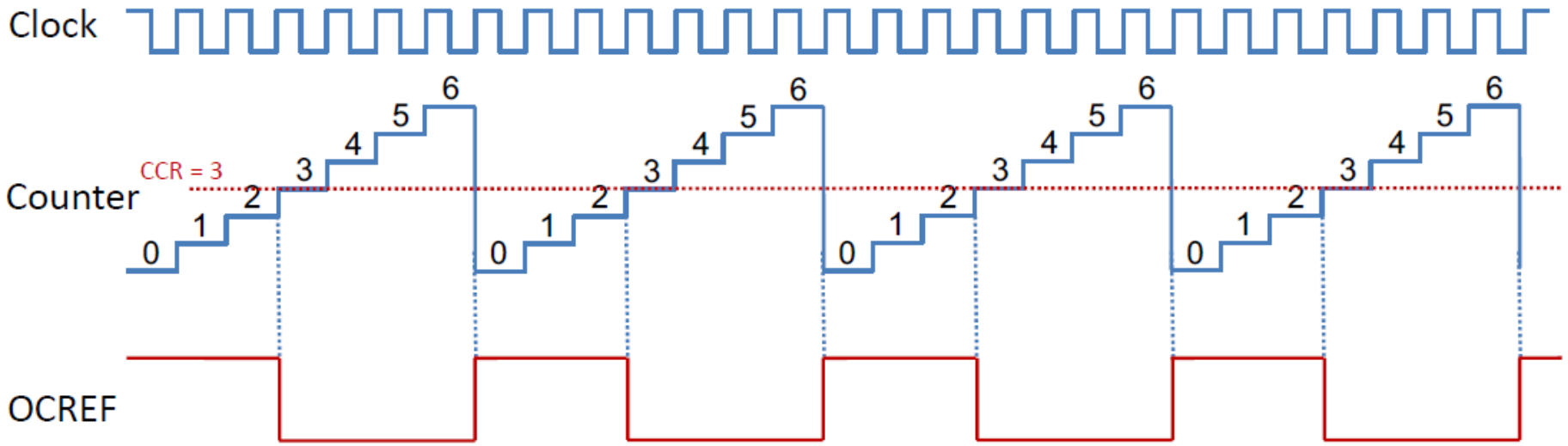
Mode	Counter < Reference	Counter ≥ Reference
PWM mode 1 (Low True)	Active	Inactive
PWM mode 2 (High True)	Inactive	Active

PWM Mode 1 (Low-True)

Mode 1

Timer Output =
- High if counter < CCR
- Low if counter ≥ CCR

Upcounting mode, ARR = 6, CCR = 3, RCR = 0



CCR: Compare&Capture Register

ARR: Auto Reload Reg.

$$\text{Duty Cycle} = \frac{CCR}{ARR + 1} = \frac{3}{7}$$

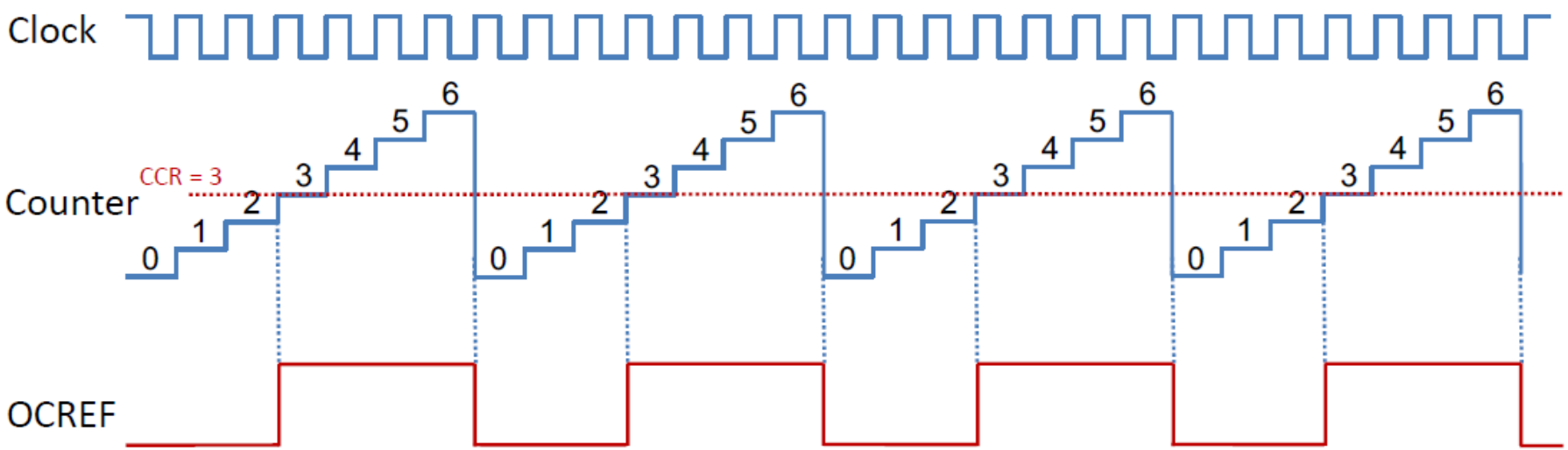
$$\text{Period} = (1 + ARR) * \text{Clock Period} = 7 * \text{Clock Period}$$

PWM Mode 2 (High-True)

Mode 2

Timer Output =
- Low if counter < CCR
- High if counter ≥ CCR

Upcounting mode, ARR = 6, CCR = 3, RCR = 0



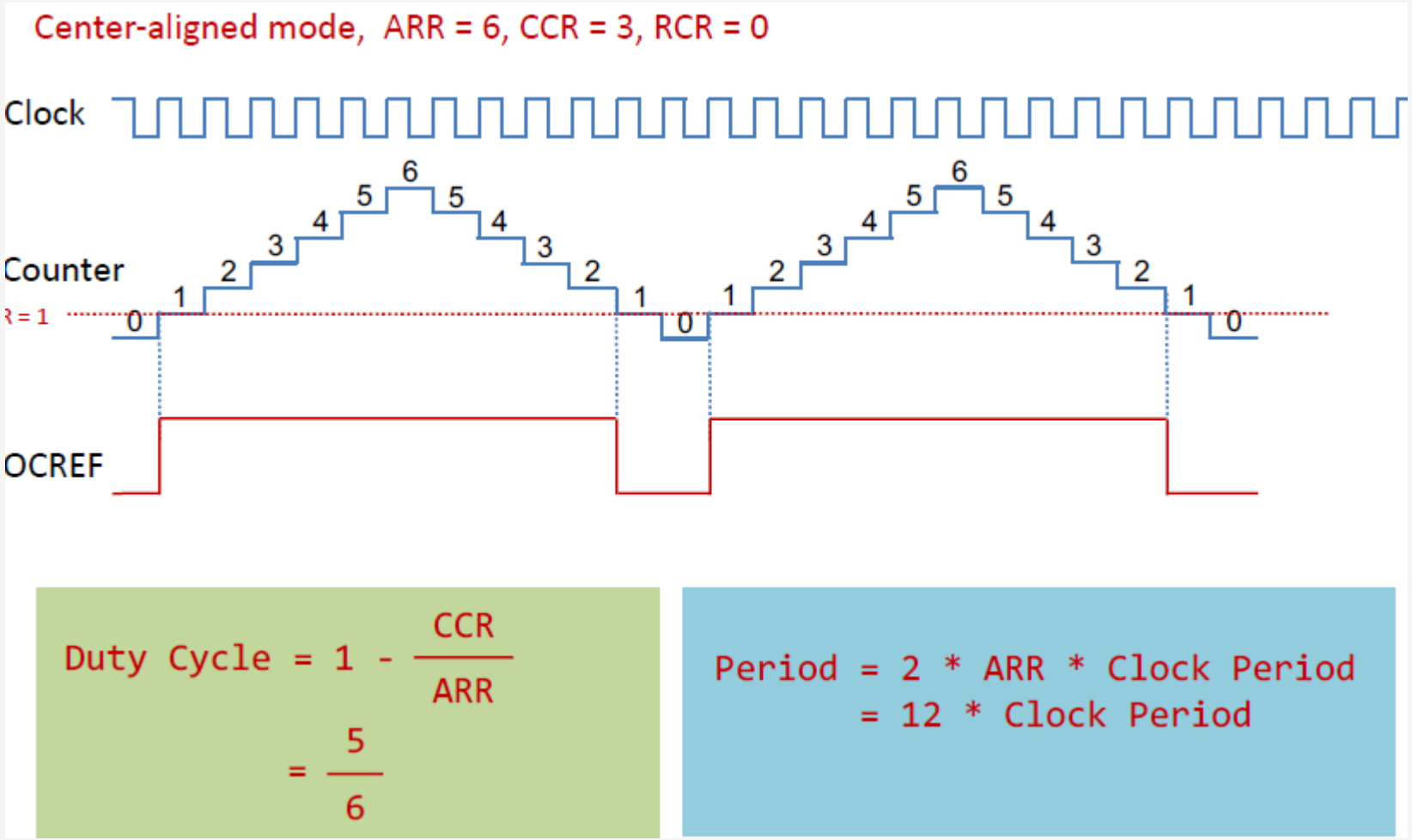
$$\text{Duty Cycle} = 1 - \frac{CCR}{ARR + 1} = \frac{4}{7}$$

$$\text{Period} = (1 + ARR) * \text{Clock Period} = 7 * \text{Clock Period}$$

CCR: Compare&Capture Register

ARR: Auto Reload Reg.

PWM Mode 2 (High-True)

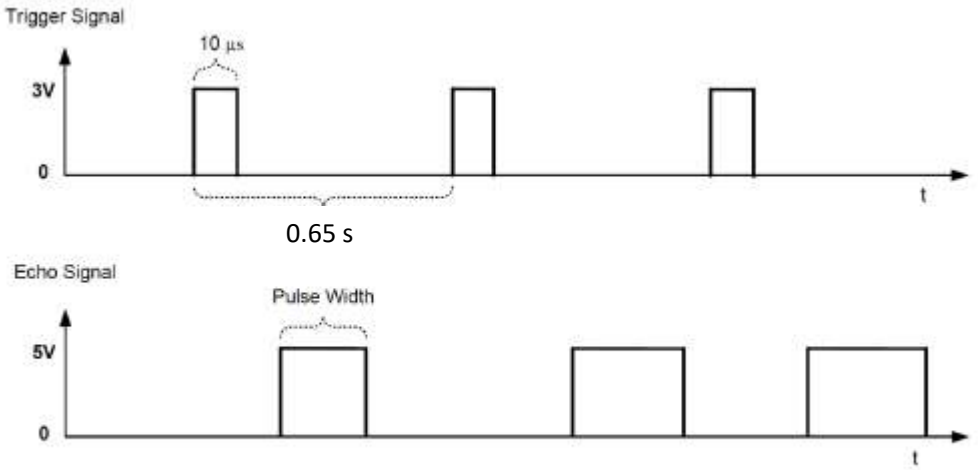


Ultrasonic Distance Sensor

$$Distance = \frac{Round\ Trip\ Time \times Speed\ of\ Sound}{2}$$

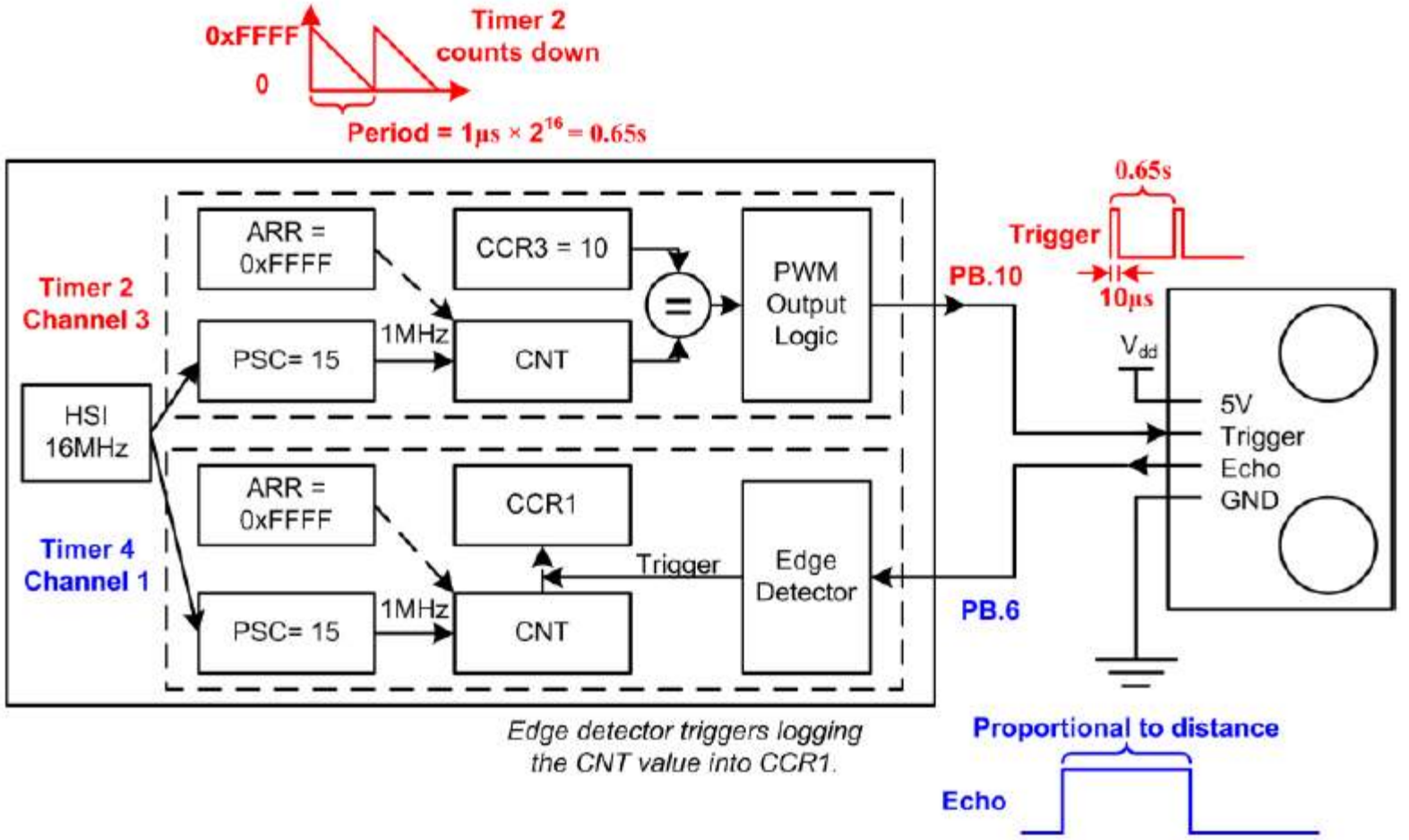
$$= \frac{Round\ Trip\ Time(\mu s) \times 10^{-6} \times 340 m/s}{2}$$

$$= \frac{Round\ Trip\ Time(\mu s)}{58}$$



If pulse width is 38*ms*,
no obstacle is detected

Measuring Parameters



2 Timer oluşturalım.

İlki tetiklemek için PWM modunda çalışacak. (PSC=15) -> 10us

İkincisi de süre ölçümü yapacak. (PSC=15)

Ölçüm yükselen kenarda aktif olup düşen kenarda duracak.

Darbe genişliği CCR1'de tutulacak.

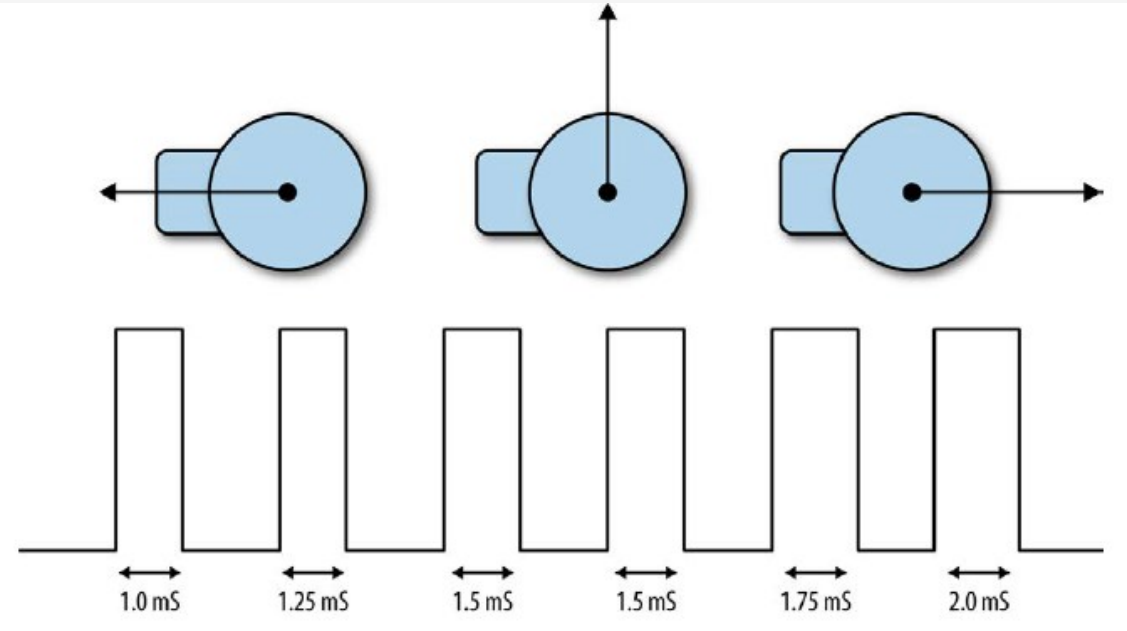
Different Types of Motors for Small Embedded Systems

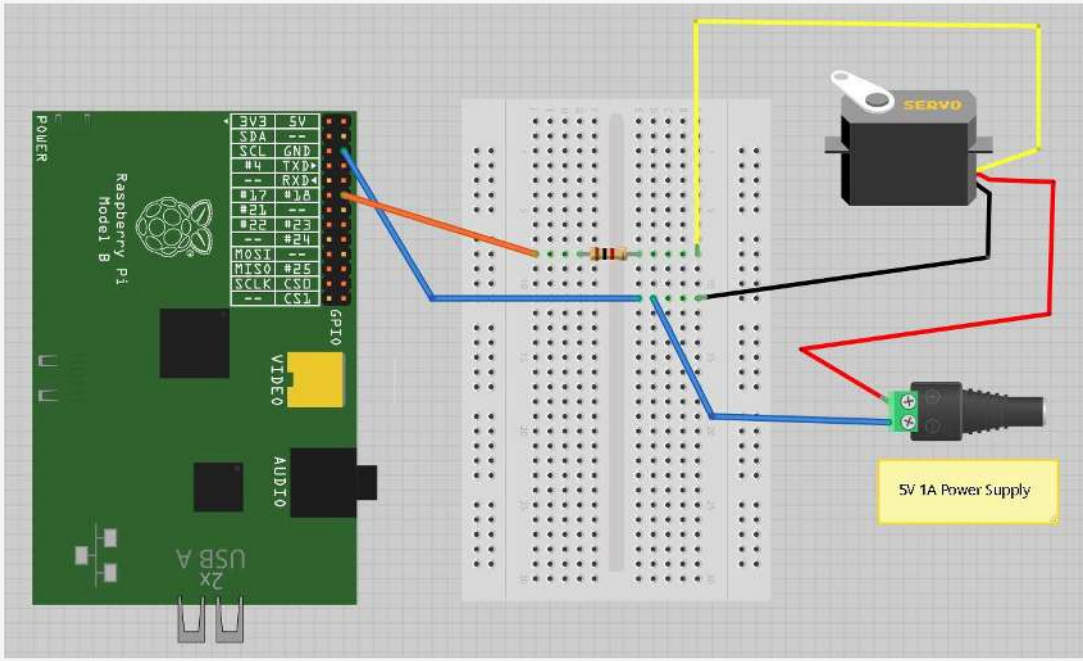
	SERVO MOTOR	DC MOTOR	STEPPER MOTOR
Typical application	When high torque, accurate rotation is required.	When fast, continuous rotation is required.	When slow and accurate rotation is required.
Control hardware	Position is controlled through pulse width modulation (PWM). No controller required. May require PWM tuning.	Speed is often controlled through PWM. Additional circuitry required to manage power requirements.	Typically requires a controller to energize stepper coils. The RPi can perform this role, but an external controller is preferable and safer.

	SERVO MOTOR	DC MOTOR	STEPPER MOTOR
Control type	Closed-loop, using a built-in controller.	Typically closed-loop using feedback from optical encoders.	Typically open-loop, because movement is precise and steps can be counted.
Features	Known absolute position. Typically, limited angle of rotation.	Can drive very large loads. Often geared to provide very high torque.	Full torque at standstill. Can rotate a large load at very low speeds. Tendency to vibrate.
Example applications	Steering controllers, camera control, and small robotic arms.	Mobile robot movement, fans, water pumps, and electric cars.	CNC machines, 3D printers, scanners, linear actuators, and camera lenses.

1- Servo Motor Example

Servomotorun poziyonu, PWM darbesinin genişliği tarafından belirlenir.





The servo expects to receive a pulse at least every 20 milliseconds.

If that pulse is high for 1 millisecond, the servo angle will be zero; if it is 1.5 milliseconds, it will be at its center position; and if it is 2 milliseconds, it will be at 180 degrees

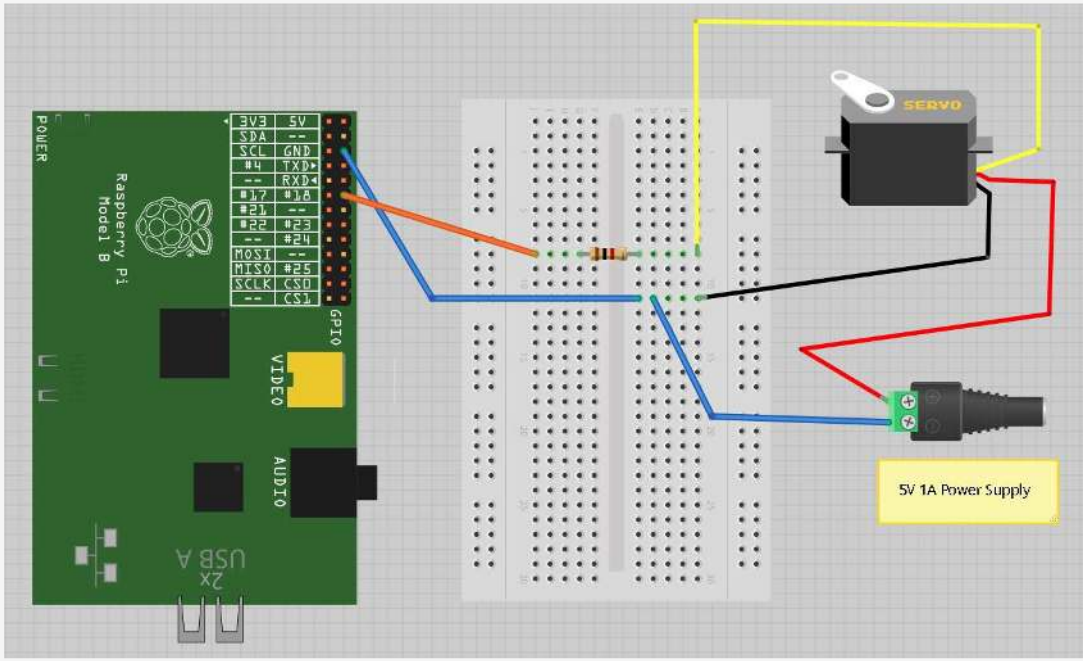
1- Servo Motor Example

```
from Tkinter import *
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
pwm = GPIO.PWM(18, 100)    // T= 10ms
pwm.start(5)               // %5

class App:
        def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        scale = Scale(frame, from_=0, to=180,
        orient=HORIZONTAL, command=self.update)
        scale.grid(row=0)

        def update(self, angle):
        duty = float(angle) / 10.0 + 2.5  // 180 derecede ~%20 olmalı
        pwm.ChangeDutyCycle(duty)

root = Tk()
root.wm_title('Servo Control')
app = App(root)
root.geometry("200x50+0+0")
root.mainloop()
```

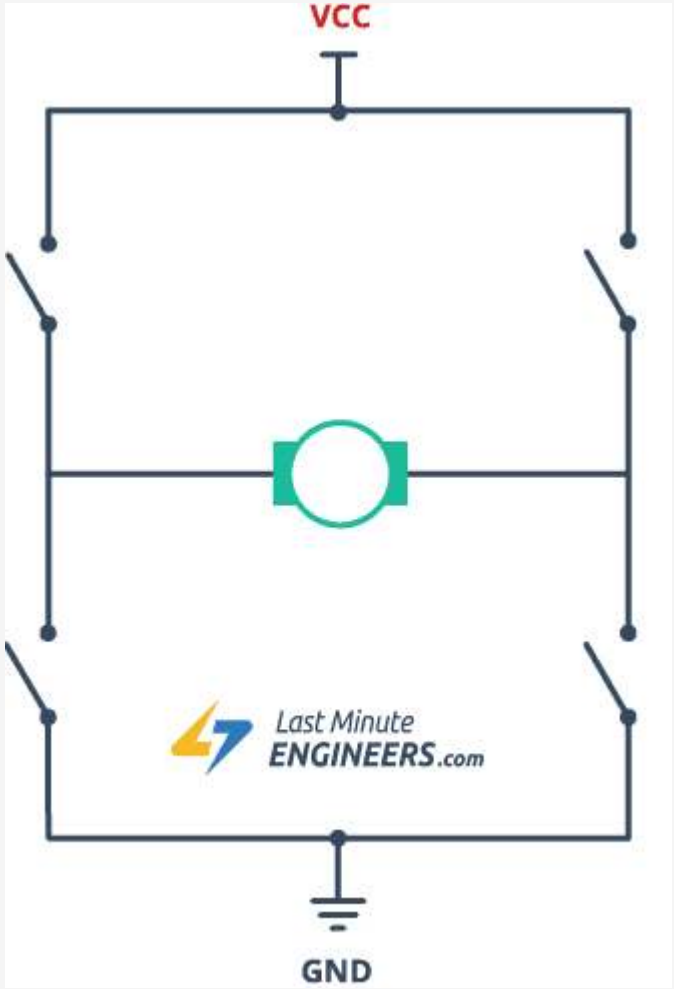


2- Driving DC Motor and Controlling the Direction

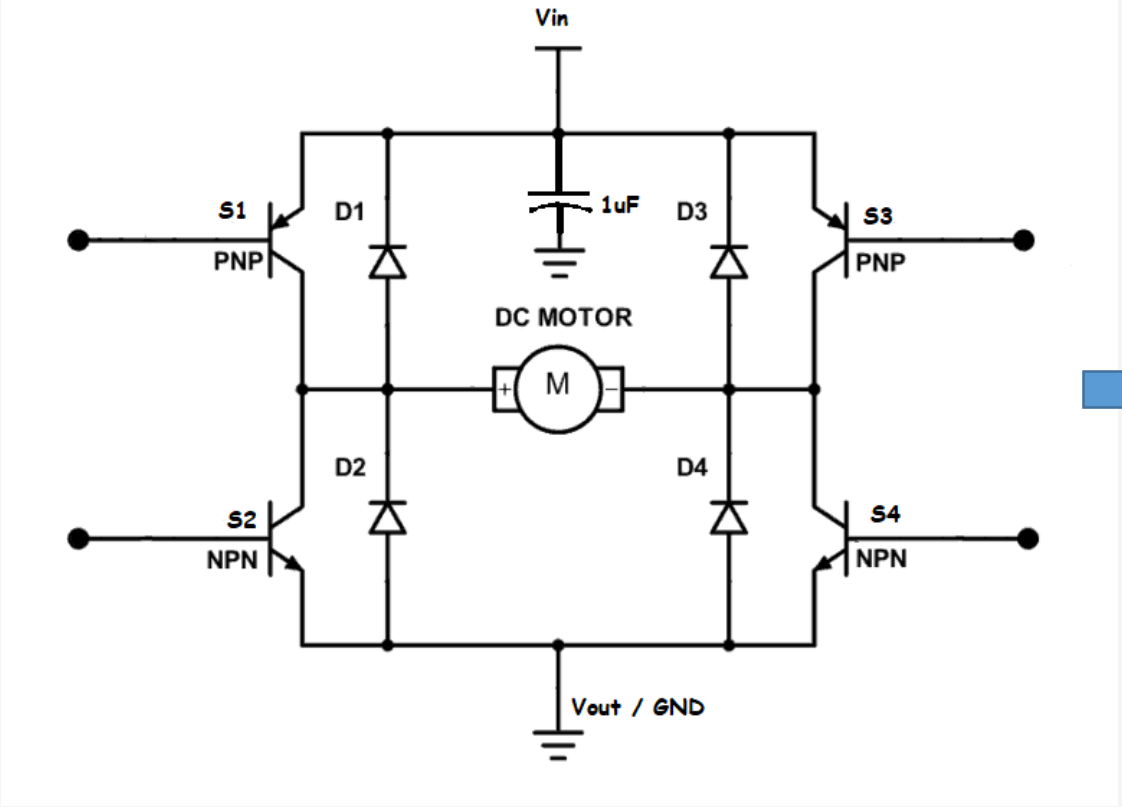
An H-bridge is built with four switches (solid-state or mechanical). When the switches S1 and S4 (according to the first figure) are closed (and S2 and S3 are open) a positive voltage is applied across the motor.

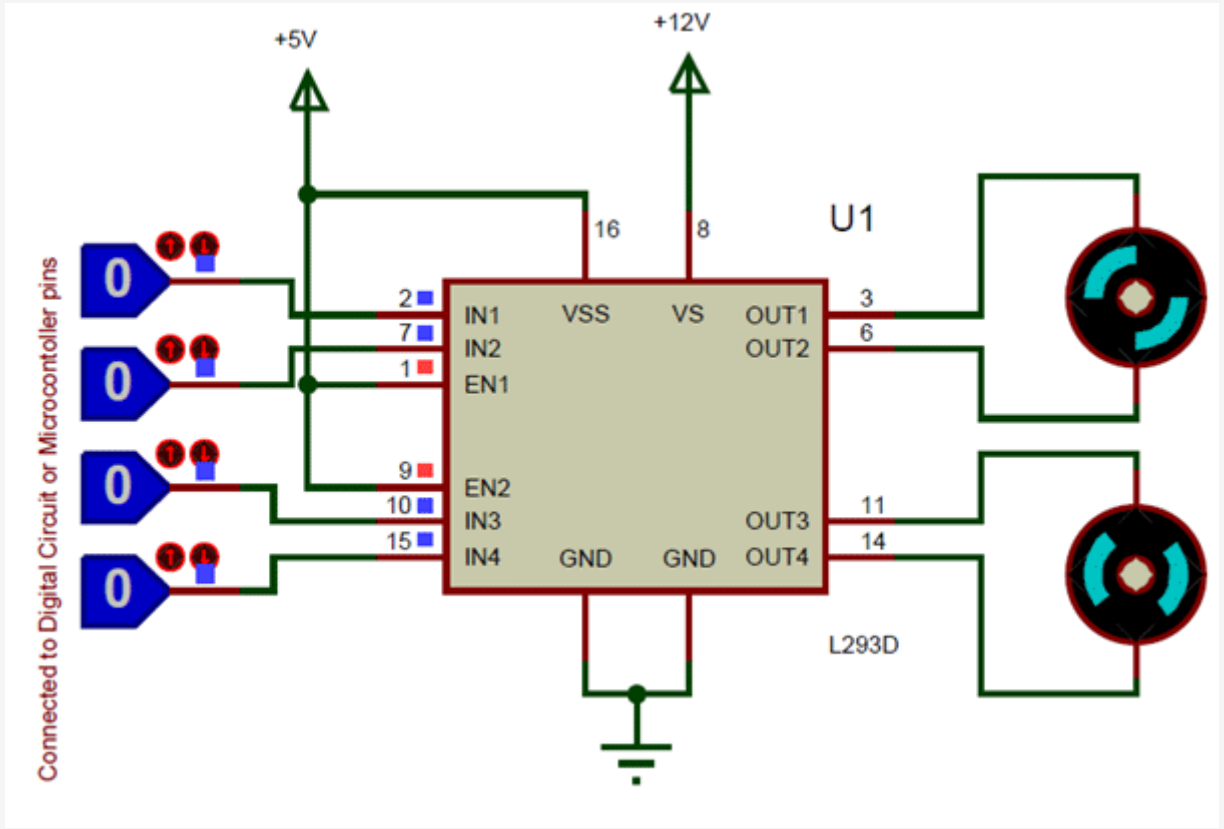
By opening S1 and S4 switches and closing S2 and S3 switches, this voltage is reversed, allowing reverse operation of the motor.

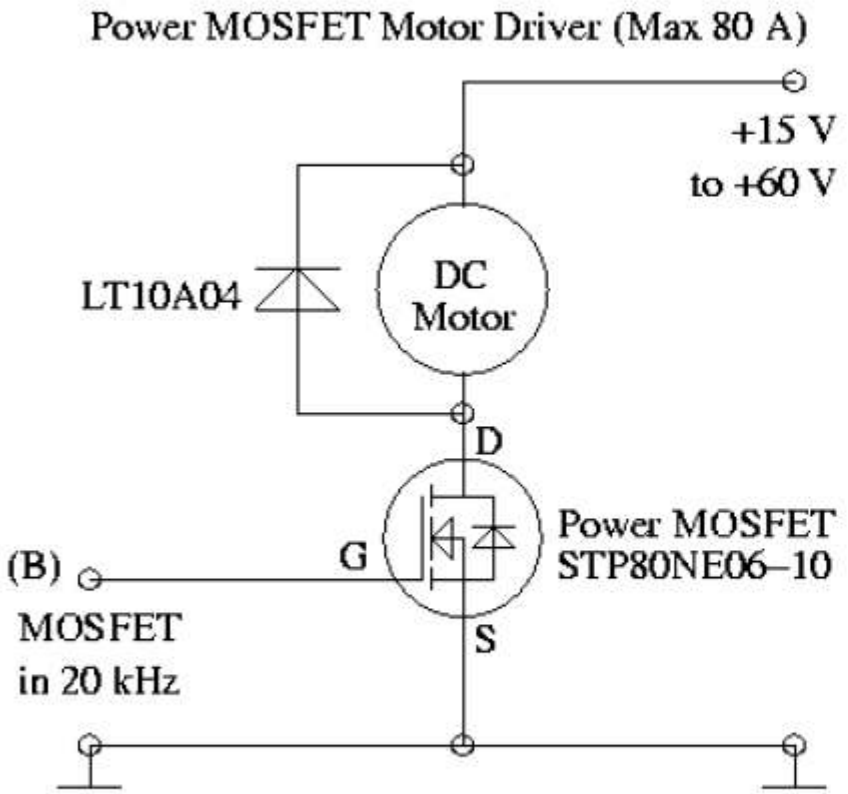
S1	S2	S3	S4	Result
1	0	0	1	Motor moves right
0	1	1	0	Motor moves left
0	0	0	0	Motor coasts
1	0	0	0	
0	1	0	0	
0	0	1	0	
0	0	0	1	
0	1	0	1	Motor brakes
1	0	1	0	
x	x	1	1	Short circuit
1	1	x	x	



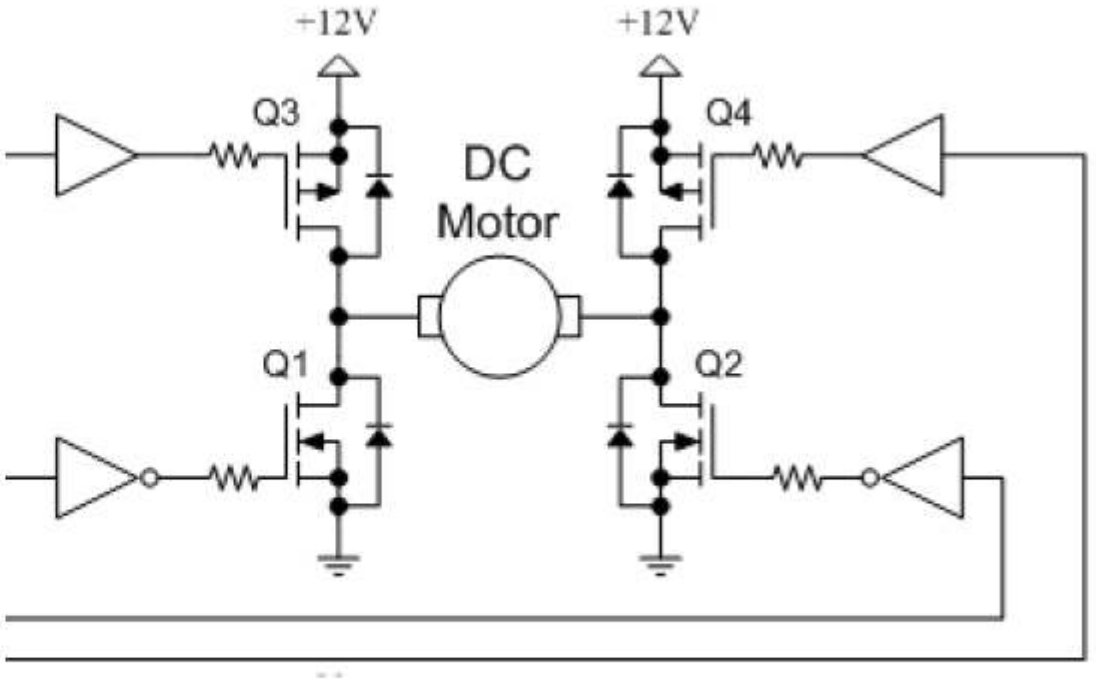
2- Driving DC Motor and Controlling the Direction





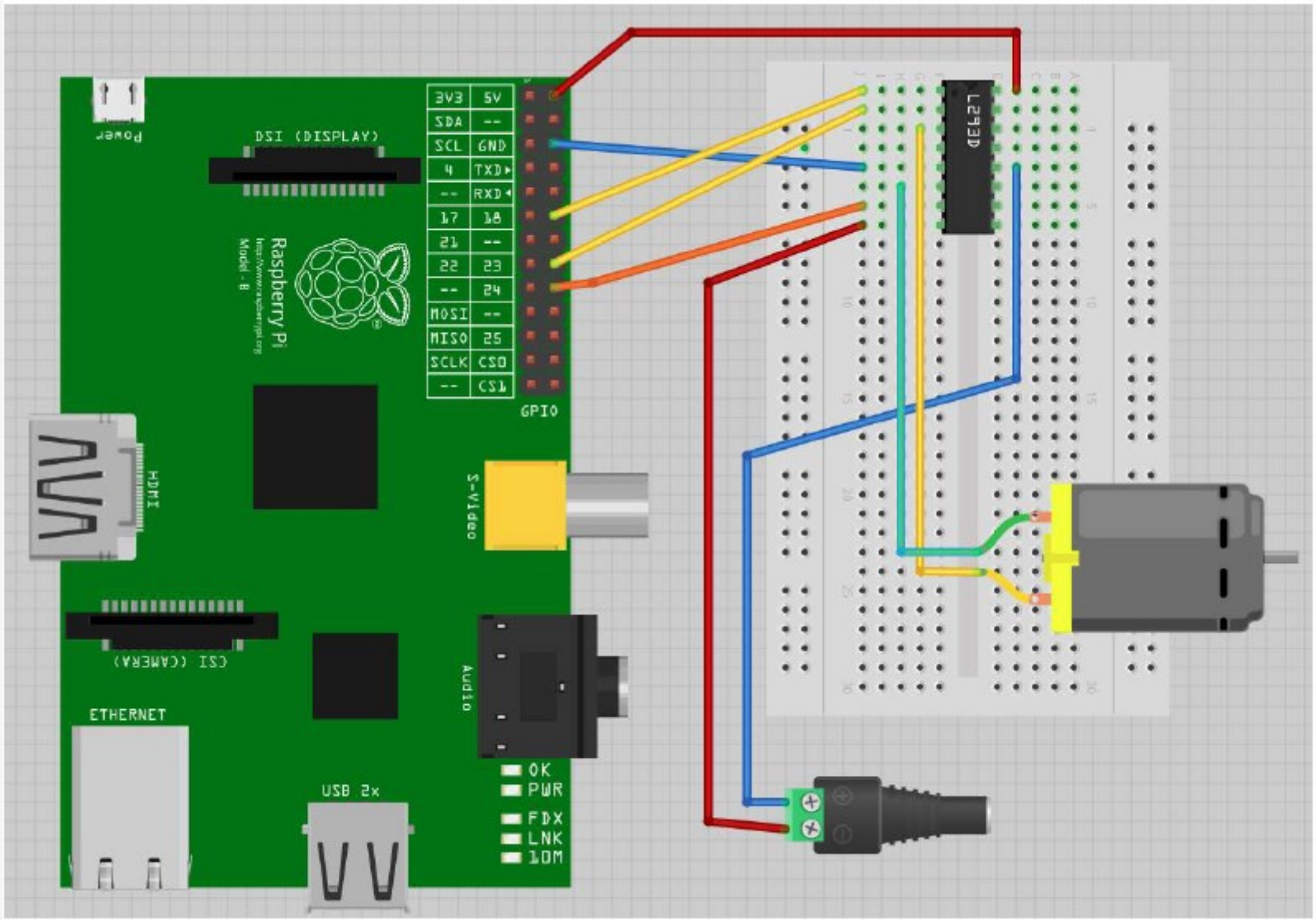


Unidirectional drive



Bidirectional drive

2- Driving DC Motor and Controlling the Direction




```
enable_pin = 18
in1_pin = 23
in2_pin =24
GPIO.setmode(GPIO.BCM)
GPIO.setup(enable_pin, GPIO.OUT)
GPIO.setup(in1_pin, GPIO.OUT)
GPIO.setup(in2_pin, GPIO.OUT)
pwm = GPIO.PWM(enable_pin, 500)
pwm.start(0)

def clockwise():
        GPIO.output(in1_pin, True)
        GPIO.output(in2_pin, False)
def counter_clockwise():
        GPIO.output(in1_pin, False)
        GPIO.output(in2_pin, True)
while True:
    cmd = raw_input("Command, f/r 0..9, E.g. f5 :")
    direction = cmd[0]
    if direction == "f":
    clockwise()
    else:
    counter_clockwise()
    speed = int(cmd[1]) * 10
    pwm.ChangeDutyCycle(speed)
```