Spartan 6 FPGA Board

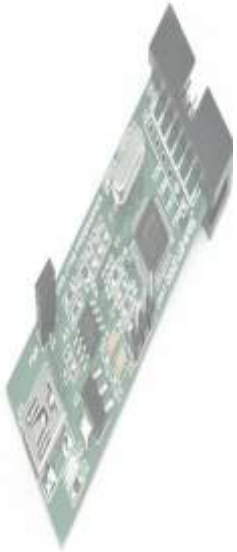
- Xilinx’s Spartan®-6 XC6SLX16-2FTG256C
- MT41J128M16JT-25(DDR3)
- 50MHz Clock
- M25P80 SPI Flash

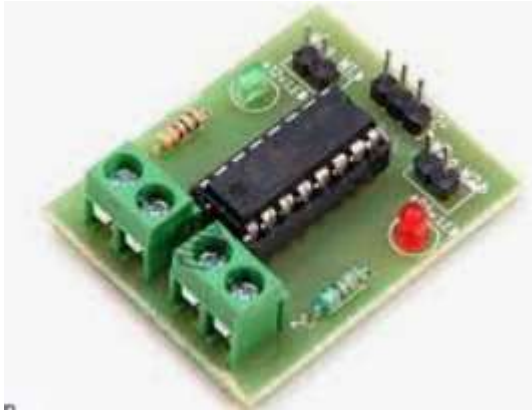
Board Specifications

- On-Board FPGA: XC6SLX16-2FTG256C;
- On-Board FPGA external crystal frequency: 50MHz;
- XC6SLX16-2FTG256C has rich block RAM resource up to 576Kb
- XC6SLX16-2FTG256C has 14,579 logic cells;
- On-Board M25P80 SPI Flash，1M bytes for user configuration code;
- On-Board 256MB Micron DDR3, MT41J128M16JT-125
- On-Board 3.3V power supply for FPGA by using MP2359 wide input range DC/DC
- XC6SLX16 development board has two 64p, 2.54mm pitch headers for extending user IOs.
- All IOs are precisely designed with length matching
- XC6SLX16 development board has 3 user switches
- XC6SLX16 development board has 4 user LEDs;
- XC6SLX16 development board has JTAG interface, by using 6p, 2.54mm pitch header;
- XC6SLX16 development board PCB size is: 6.7cm x 8.4cm;
- Default power source for board is: 1A@5V DC, the DC header type: DC-050, 5.5mmx2.1mm

Hardware Required

JTAG Interface –To Program FPGA
Pocket Science Ocilloscope
DC Motor and Driver



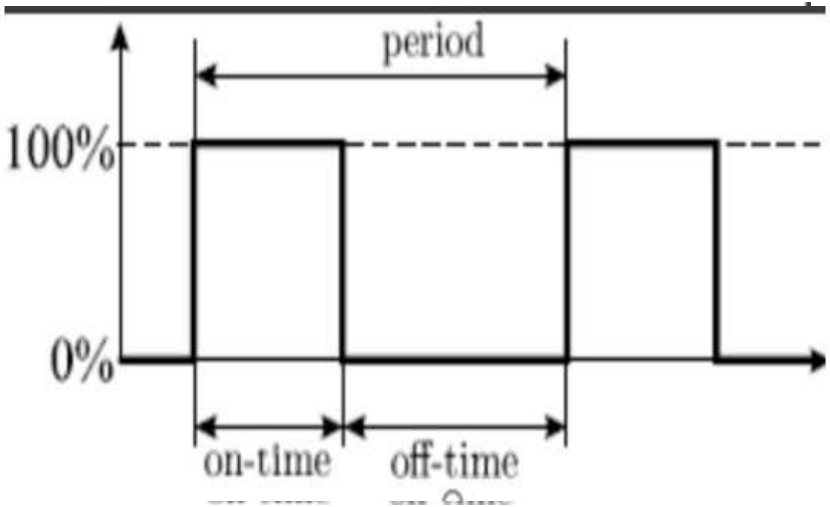


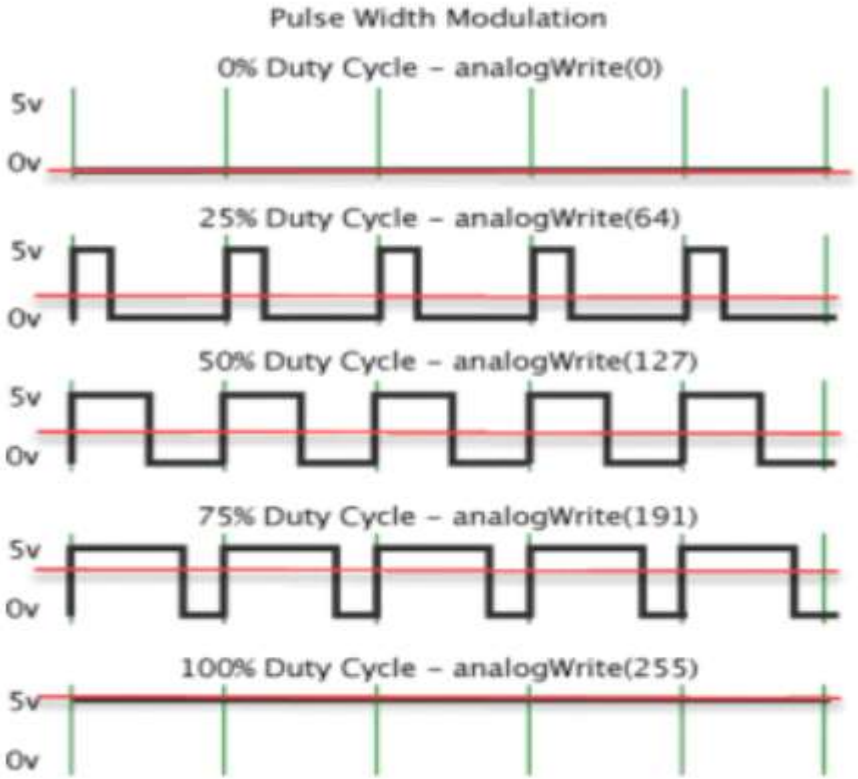
Why FPGA for Motor Control

- Lower System Cost
- Concurrency in the algorithm (Parallel Processing)
- Sufficient I/O s
- Design reuse and scalability
- Extended product lifecycles
- Control multiple DC motors synchronously
- Drive a stepper motor with an analog feedback system
- Use an FPGA to create a closed-loop control system for different motion parameters

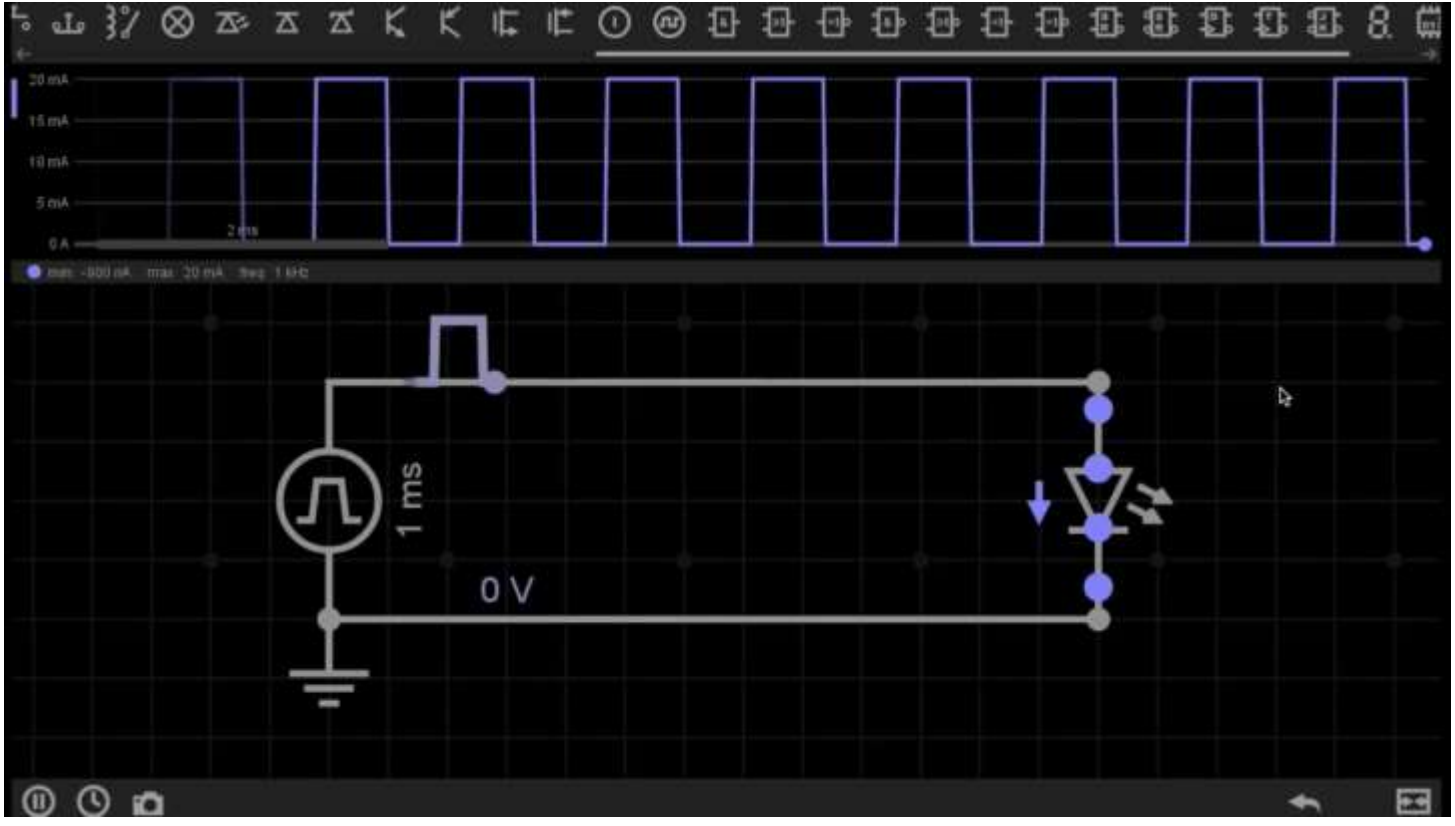
What is PWM

PWM is a way to control analog devices with a digital output.





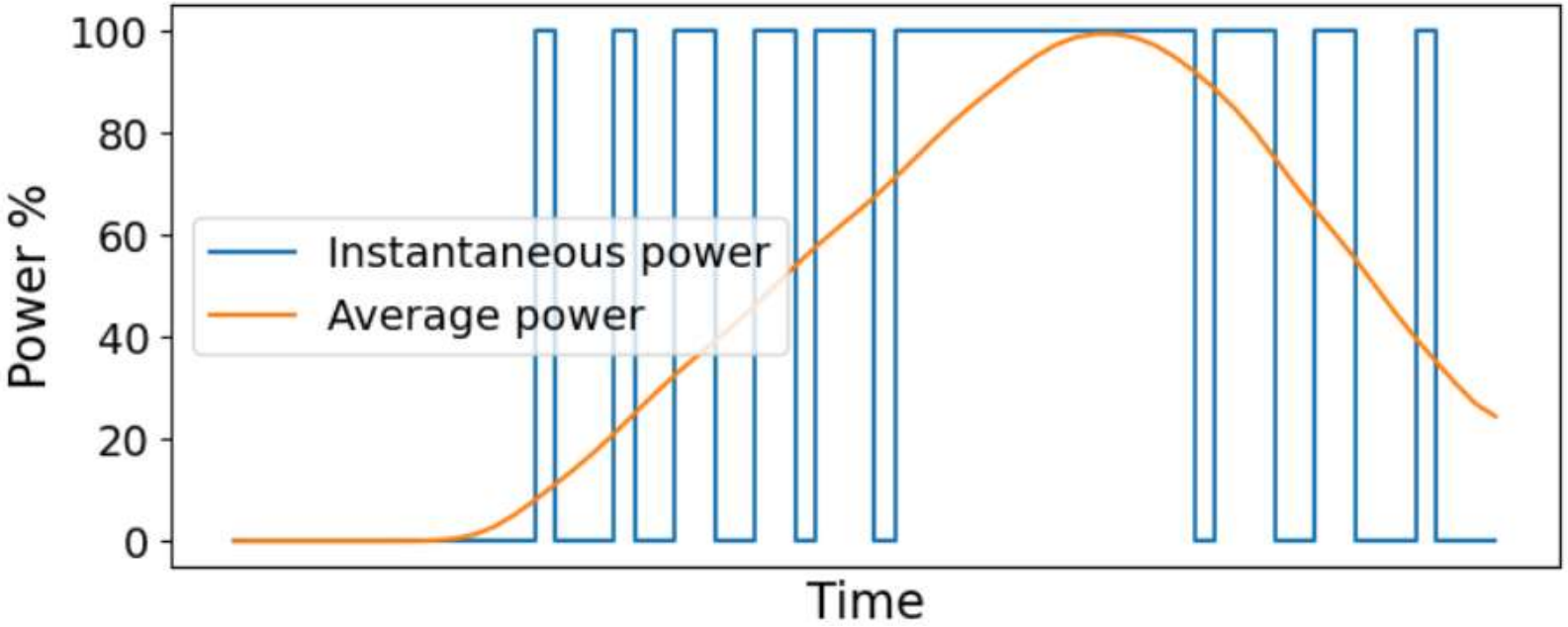
HOW PWM WORKS



- Courtesy : Simply electronics

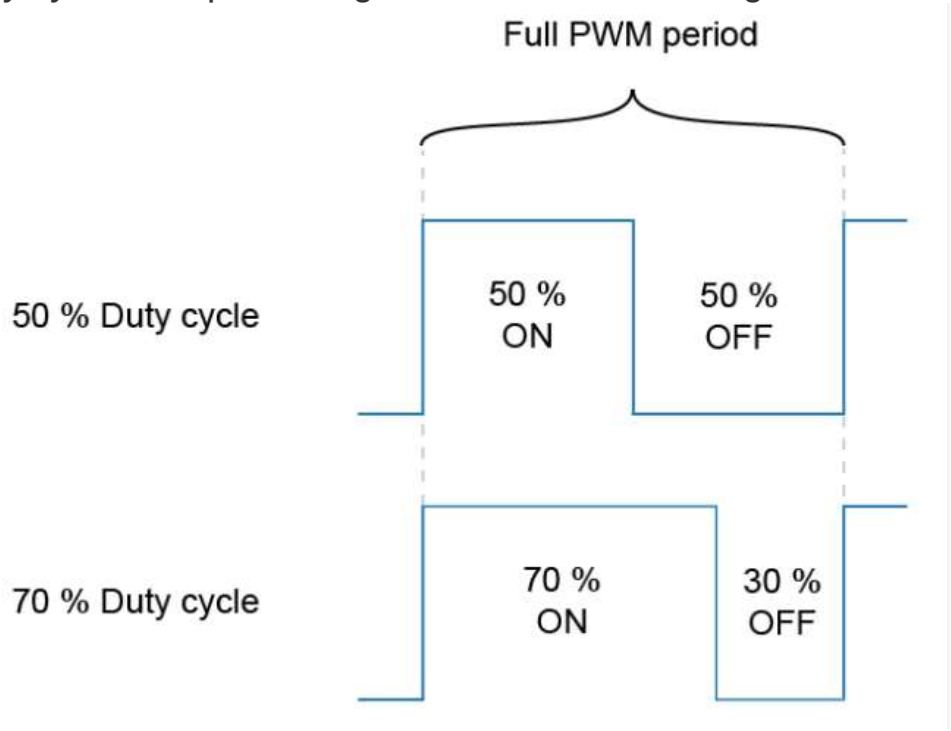
HOW PWM WORKS

By switching on and off the power supply to a device with a high frequency, we can accurately control the average current flowing through it. The illustration below shows the basics of how PWM works. The PWM output controls a binary switch that can either set the power to 100% or 0%. By quickly alternating between the two extremes, the sliding window average will be a function of the time spent in each of the states

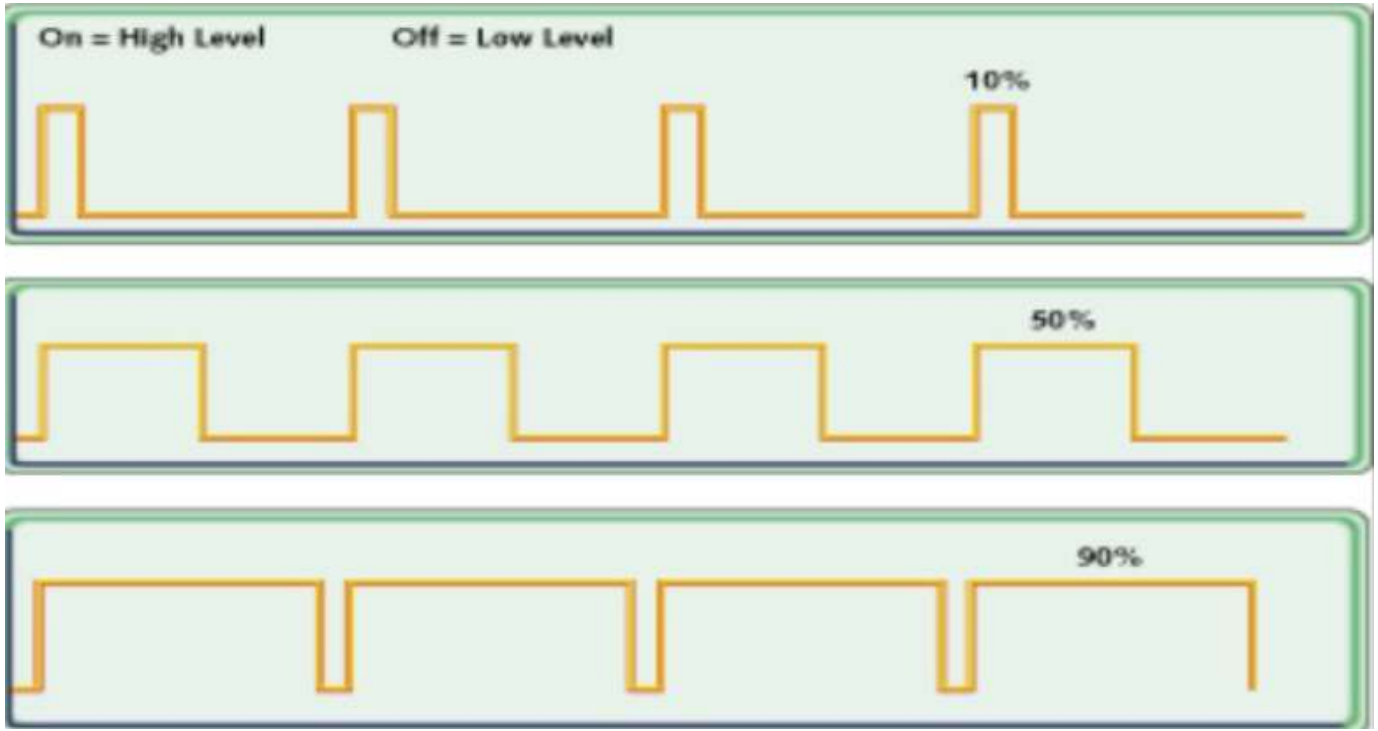


WHAT IS DUTY CYCLE

The duty cycle is key to controlling the power given to the analog device in PWM. The term *duty cycle* means how much time the PWM output spends at the ON position. It's common to describe the duty cycle as a percentage, as shown in the image below.



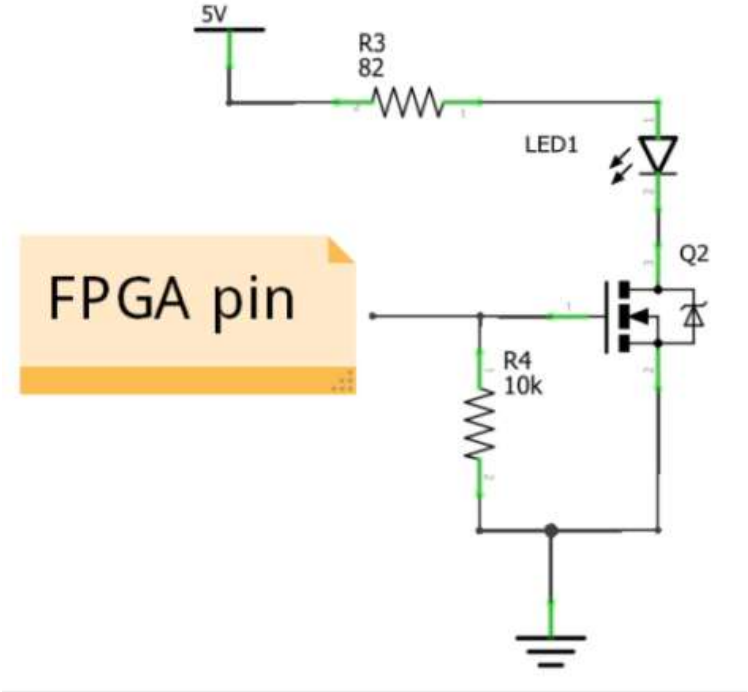
PWM Signals of Varying Duty cycles



Four Basic Motor Controllers

- There are four basic **motor controller** and drive **types**: AC, DC, servo, and stepper.
- Special Motor Control Drivers
  - SRM
  - PMSM

How PWM Controls Motors



Analog power electronics isn't as fast as the digital FPGA pin. A typical PWM setup uses power MOSFETs as switches to control the current flowing through the analog device.

DEMO

- 2 Push buttons for Increment and Decrement of speed
- L293D Driver for Motor

VHDL Code for increment

```
--LIBRARY DECLARATION----------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

--Entity declaration

entity PWMMOTOR is
    Port ( CLK : in  STD_LOGIC;
           INCREMENT : in  STD_LOGIC;-- Pushbutton for increment
           DECREMENT : in  STD_LOGIC;-- Pushbutton to decrement
           PWM_OUTPUT : out  STD_LOGIC);
end PWMMOTOR;
```

- process(clk)

- variable count: integer range 0 to 500;
- variable duty_cycle: integer range 100 to 500;
- variable i : integer range 0 to 500;

- begin

```
if rising_edge(clk) then

count:=count+1;
                    if count =duty_cycle then
                                        PWM_OUTPUT<='1';
                    end if;

                    if count = 500 then

                    PWMOUT <='0';
                    count:=0;
                                        if (INCREMENT ='1') and (duty_cycle <450)  then
                                        i:=i+1;
                                                            if i=500 then

                                                                                duty_cycle:= duty_cycle+1;
                                                                                i:=0;
                                                            end if;

                                        else

                                        duty_cycle:=duty_cycle;

                                        end if;
```

```
if (DECREMENT ='1') and (duty_cycle > 50)  then
                                    i:=i+1;
                                                if i=500 then
                                                                    duty_cycle:= duty_cycle-1;
                                                                    i:=0;
                                                end if;

                                    else

                                    duty_cycle:=duty_cycle;

                                    end if;



                end if;

end if;
end process;
```