Microprocessors and Microcontrollers Course Project

TITLE OF THE PROJECT:

Speed Control of DC Motor with PWM using PIC18F452

Submitted by

J.SAMANVITHA

REGISTRATION NO: B240791EE

Supervised by

Dr. Korra Balu

Assistant Professor

Department of Electrical Engineering

National Institute of Technology Calicut

1. Project Title

DC Motor Speed Control using Pulse Width Modulation (PWM) with PIC18F452 Microcontroller

3. Objective of the Project

The primary objective of this project is to implement a **variable speed control system** for a **DC motor** using the **PIC18F452 microcontroller**. The control method employed is **Pulse Width Modulation (PWM)**, which allows the effective voltage supplied to the motor to be varied, thereby controlling its rotational speed. A push button switch is used to initiate a speed ramping sequence from a low speed (low frequency PWM) to a higher speed (high frequency PWM).

4. List of Components (with specifications)

The components are required for the microcontroller circuit, the motor driver, and the user interface.

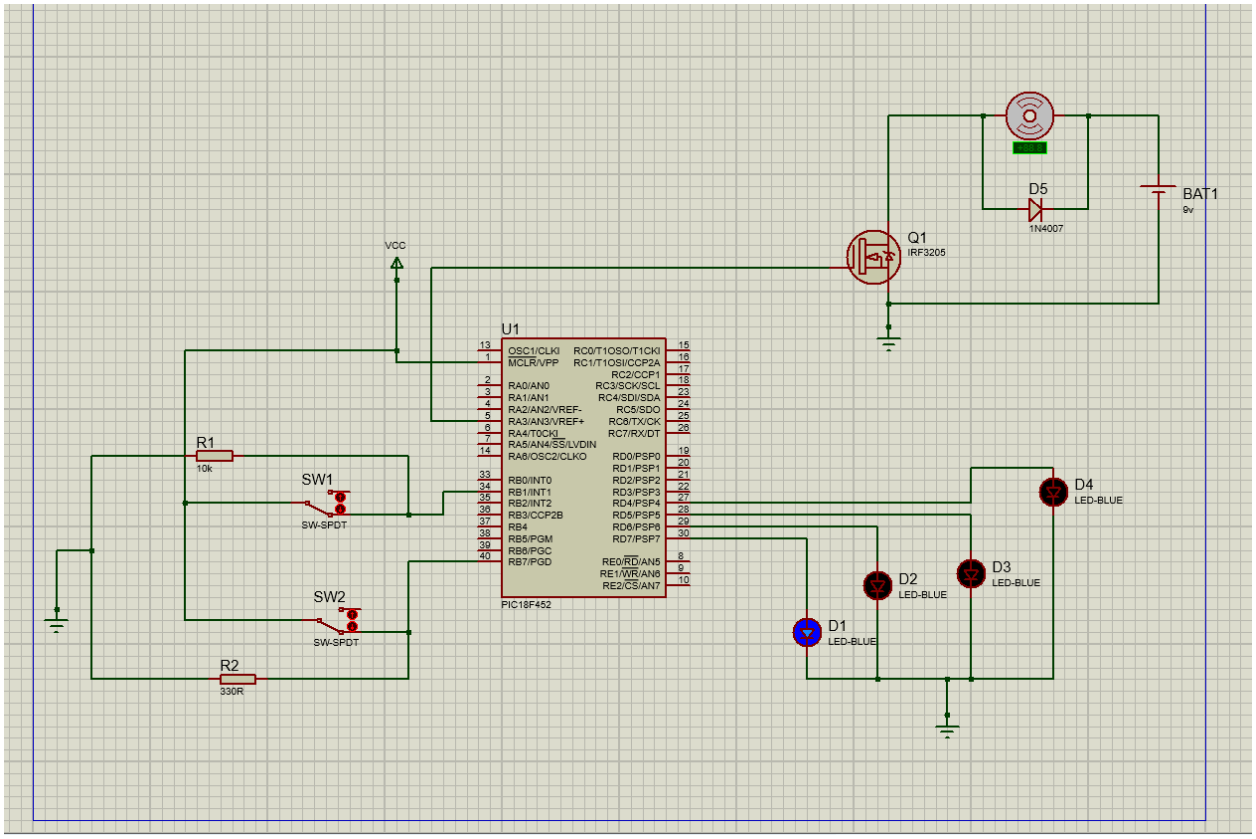
Component	Specification	Quantity	Purpose
Microcontroller	PIC18F452 (40-Pin DIP)	1	Brain of the system; generates the PWM signal.
DC Motor	Typical small-scale DC motor (e.g., 6V or 12V)	1	The load whose speed is to be controlled.
Motor	MOSFET (e.g.,	1	**Q1** in the circuit.

Driver/Switch	IRF540N, suitable for DC motor load)		Used as a high-side or low-side switch to drive the motor with the PWM signal.
Flyback Diode	1N4001 or equivalent	1	**D1** in the circuit. Protects the MOSFET from the motor's inductive kickback (back-EMF).
Push Button	Momentary Push Button Switch	2	**SW1, SW2** (Implied by R1, R2 on the circuit). Used to initiate the speed control/ramping.
Resistors	10kOmega (Pull-down for switches)	2	**R1, R2** in the circuit. Ensures input pins are at a defined low state when the button is not pressed.
Crystal Oscillator	20MHz or similar	1	Provides the clock signal for the PIC microcontroller.
Capacitors	22pF (for crystal)	2	Stabilizes the crystal oscillator.
LEDs	Blue(with current-limiting resistors)	4	**D2, D3, D4, D5** in the circuit. Used as indicators for status/debugging (e.g., motor ON, ramping mode).

Power Supply	5V for PIC, 6V-12V for Motor	1/2	Supplies operating power to the microcontroller and the motor.

5. Circuit Connection Diagram

The provided image shows a Proteus-based schematic for the DC motor speed control system.



Key Connections:

- **Microcontroller (PIC18F452):**
  - The PWM signal (likely **RD1** as per the Assembly code's BTG PORTD,1) is connected to the gate of the **MOSFET (Q1)**, which acts as the main switching element for the motor.
  - Input switches (**SW1, SW2** implied by R1, R2) are connected to **PORTB** pins (e.g., **RB5** as per the Assembly code) to trigger the speed change.

    - Indicator LEDs (**D2-D5**) are connected to an output port, likely **PORTD**.
    - The main power and ground are connected.
- **Motor Driver Stage:**
    - The **DC Motor** is connected between the motor power supply (**BAT1**) and the drain of the **MOSFET (Q1)**.
    - The source of the MOSFET is connected to **Ground**.
    - The **Flyback Diode (D1)** is connected in parallel across the DC motor to suppress voltage spikes.

6. Assembly Language Code

The provided Assembly code implements a software PWM technique by repeatedly toggling an output pin (**RD1**) and using a variable delay loop to control the frequency (and thus the effective duty cycle for a fixed on-time/off-time relationship in this simple toggle implementation) of the signal. The speed is ramped up (frequency increases, delay decreases) upon an external interrupt.

```
LIST    P=18F452
    #include <P18F452.INC>

    ; Configuration bits
    CONFIG  OSC = HS        ; High-Speed Crystal/Resonator
    CONFIG  WDT = OFF       ; Watchdog Timer disabled
    CONFIG  LVP = OFF       ; Low Voltage Programming disabled

    CBLOCK  0x20
        DUTY                ; PWM duty cycle (0-255)
    ENDC

    ORG     0x0000
    GOTO    MAIN            ; Reset vector

MAIN
    MOVLW   0x80                ; Default PWM duty cycle = 50%
    MOVWF   DUTY

    ; Set RD0 and RD1 as inputs for switches (others output for LEDs)
    MOVLW   b'00000011'         ; RD0/RD1 input, RD2-RD7 outputs
    MOVWF   TRISD
```

```
    ; RC2 as output for PWM (CCP1)
    BCF     TRISC,2

    ; Configure CCP1 (RC2) for PWM
    MOVLW   0x0C                ; CCP1M<3:0> = 1100 (PWM mode)
    MOVWF   CCP1CON

    ; Set PWM period (PR2), and start Timer2
    MOVLW   0xFF                ; PR2 = 255. Determines PWM frequency.
    MOVWF   PR2
    MOVLW   0x04                ; T2 prescaler 1:1, TMR2ON bit 2 = 0,
T2CKPS<1:0> = 00
    MOVWF   T2CON
    BSF     T2CON, TMR2ON       ; Turn on Timer2

MAIN_LOOP
    ; Check "speed up" switch on RD0 (pressed = logic HIGH)
    BTFSC   PORTD,0             ; Skip next instruction if RD0 is LOW
(not pressed)
    INCF    DUTY, F             ; Increment duty cycle

    ; Check "speed down" switch on RD1 (pressed = logic HIGH)
    BTFSC   PORTD,1             ; Skip next instruction if RD1 is LOW
(not pressed)
    DECF    DUTY, F             ; Decrement duty cycle

    ; Clamp DUTY value between 0 and 255
    MOVLW   0x00
    CPFSLT  DUTY                ; Compare DUTY with 0x00. Skip if DUTY
< 0x00 (shouldn't happen with DECF)
    MOVLW   0x00
    MOVWF   DUTY                ; Clamp to 0
    MOVLW   0xFF
    CPFSGT  DUTY                ; Compare DUTY with 0xFF. Skip if DUTY
> 0xFF
    MOVLW   0xFF
    MOVWF   DUTY                ; Clamp to 255

    ; Set PWM duty cycle (RC2 pin)
    MOVF    DUTY,W
    MOVWF   CCPR1L              ; Load 8-bit duty cycle value
(DCXB<1:0> bits remain 0)

    ; Indicate approximate speed via LEDs (RD2-RD5)
    MOVF    DUTY,W
    SWAPF   WREG,F              ; Use upper nibble of duty cycle
(0-15)
```

```
    ANDLW   0x0F                ; Mask to get only 4 bits (0-15)
    IORLW   0x00                ; (Adjust this constant based on how
many LEDs you want ON)
    MOVWF   LATD                ; Output to PORTD, which drives the
LEDs D2-D5.

    GOTO    MAIN_LOOP

    END
```

7. Procedure

The project involves hardware assembly, coding, and testing:

A. PWM Theory and DC Motor Speed Control:

Pulse Width Modulation (PWM) is an effective digital technique used to control the **average power** delivered to a load, such as a DC motor, by varying the width of a digital pulse.

Since the PIC18F452 microcontroller operates digitally (producing ON/OFF or HIGH/LOW signals), PWM is essential for providing the motor with a controllable "analog"voltage effect.

- Any duty cycle between **0% and 100%** controls the average voltage, thereby controlling the motor's speed.

▪ PIC18F452 and PWM Generation

The **PIC18F452** microcontroller contains built-in peripherals for generating PWM signals, specifically the **Capture/Compare/PWM (CCP) modules** (usually CCP1 and CCP2).

CCP Module Configuration for PWM

The PIC microcontroller uses one of its on-chip timers, typically **Timer2 (TMR2)**, to set the **PWM frequency** and control the internal clocking for the PWM period.

1. The frequency is generally kept constant to avoid audible noise or mechanical issues.
2. **Setting the Duty Cycle:** The duty cycle is set by a 10-bit value composed of the **CCPRxL** register (8 MSbs) and two bits from the **CCPxCON** register (2 LSbs). This 10-bit value determines the duration of the HIGH pulse (T-ON). Changing this value varies the motor speed.

DC Motor Speed Control Implementation

Since a microcontroller's output pin cannot supply the required voltage and current to directly drive a DC motor, a **Motor Driver Circuit** (like an **L293D** H-bridge or a MOSFET/transistor circuit) is necessary.

1. **Microcontroller Output:** The PWM signal is generated on a designated CCP pin (e.g., **RC2/CCP1** on the PIC18F452).
2. **Motor Driver Interface:** This PWM signal is connected to the **Enable (EN)** pin of the motor driver IC.
3. The driver's logic control pins (for direction) are connected to other general-purpose I/O pins of the PIC.
4. **Motor Operation:** The motor driver uses the low-power PWM signal to rapidly switch the **high-power** voltage supply to the DC motor ON and OFF.This results in the motor receiving the average voltage (V-AVG) as determined by the duty cycle, thus controlling its speed.

7.1 Hardware Setup

1. Assemble the basic PIC18F452 minimum system (power supply, crystal oscillator).
2. Connect the input switches (with pull-down resistors) to the designated input pins (**RB5**).
3. Connect the indicator LEDs (**D2-D5**) to the output pins.
4. Connect the MOSFET switch (**Q1**) gate to the PWM output pin (**RD1**) via a suitable current-limiting resistor (not shown in the provided schematic but advisable).
5. Connect the DC motor in series with the motor power supply (**BAT1**), and the MOSFET, ensuring the flyback diode (**D1**) is correctly placed across the motor terminals.

7.2 Software Implementation

1. Enter the Assembly code into the development environment (e.g., MPLAB IDE).
2. Set the Configuration Bits as specified (OSC=HS, WDT=OFF, LVP=OFF).
3. Compile the code to generate the .hex file.
4. Program the PIC18F452 microcontroller with the compiled code using a programmer (e.g., PICkit).

7.3 Testing and Operation

1. Apply power to the circuit. The DC motor should start rotating at the initial **low speed (1 kHz PWM)**.
2. Press the input push button connected to **RB5**. This should trigger the **Port B change interrupt**.
3. The microcontroller enters the ramping mode (**FLAG = 1**). The **FREQ COUNT** is progressively decreased in the main loop, causing the PWM frequency to **increase** and the motor speed to gradually **accelerate**.
4. The acceleration continues until the **RAMP COUNT** reaches its limit or the minimum delay (**FREQ COUNT = 10** for 5kHz) is achieved.
5. The motor should settle at the **final high speed (5kHz PWM)**, and the ramping flag is cleared (**FLAG = 0**).

8. Expected Outcome / Observation

- Upon power-on, the DC motor should rotate at a **slow, constant speed** corresponding to the initial 1kHz PWM signal.
- When the push button (RB5) is pressed, the motor speed should not change instantaneously but should **gradually increase** (ramp up) over a few seconds.
- The motor should eventually reach a **higher, constant speed** corresponding to the final 5kHz PWM signal.
- Pressing the button again will reset the motor speed to the low speed and restart the ramping process.
- The indicator LEDs can be observed to confirm the state of the system (e.g., an LED turns ON during the ramping phase).

9. Result / Conclusion

The project successfully demonstrated the **speed control of a DC motor** using a **software-implemented PWM technique** on the **PIC18F452 microcontroller**. The implementation of an **interrupt-driven ramping mechanism** provided a smooth and controlled acceleration of the motor from an initial low speed (1kHz PWM) to a higher final speed (5kHz PWM), showcasing the effectiveness of PWM in motor control applications.

10. References

- Microchip PIC18F452 Data Sheet.

- Proteus ISIS/ARES User Manual.