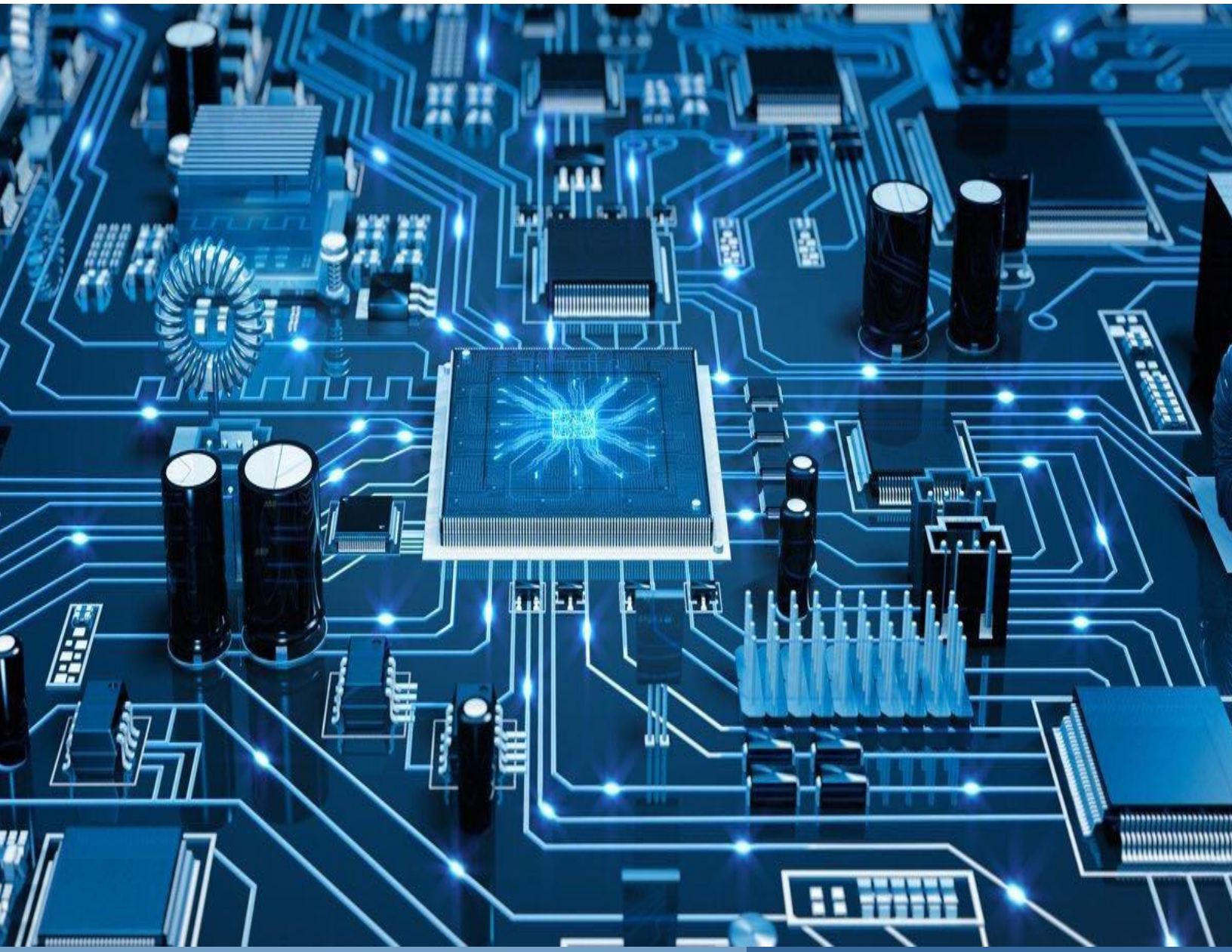
Cairo University – Faculty of Engineering
Credit Hours System – Senior 1 Level
Electrical Energy Engineering Department – EEE
Microprocessors Electric Energy Applications – EPMN201



- ***Team:***
  Mohamed Ahmed Hassan - 1190170
  Youssef Aly Wahid – 1190077
  Hossameldin Khaled-1190184
  Karim Mohamed -

- ***Submitted to:***
  Dr. Ahmed Lotfy

Motor Control Speed

Table Contents

List of Figures:

Introduction

This report provides an in-depth analysis of the methodologies, tools, wiring, and modules used for motor speed control using an AVR microcontroller. It explores the concepts of pulse-width modulation (PWM), analog-to-digital conversion (ADC), and external interrupts for motor speed and direction control. The report includes detailed explanations of AVR tools such as timers and interrupts, along with the wiring connections required. Additionally, it discusses the ALCD module used for interfacing with an LCD and provides references to online materials for further exploration.

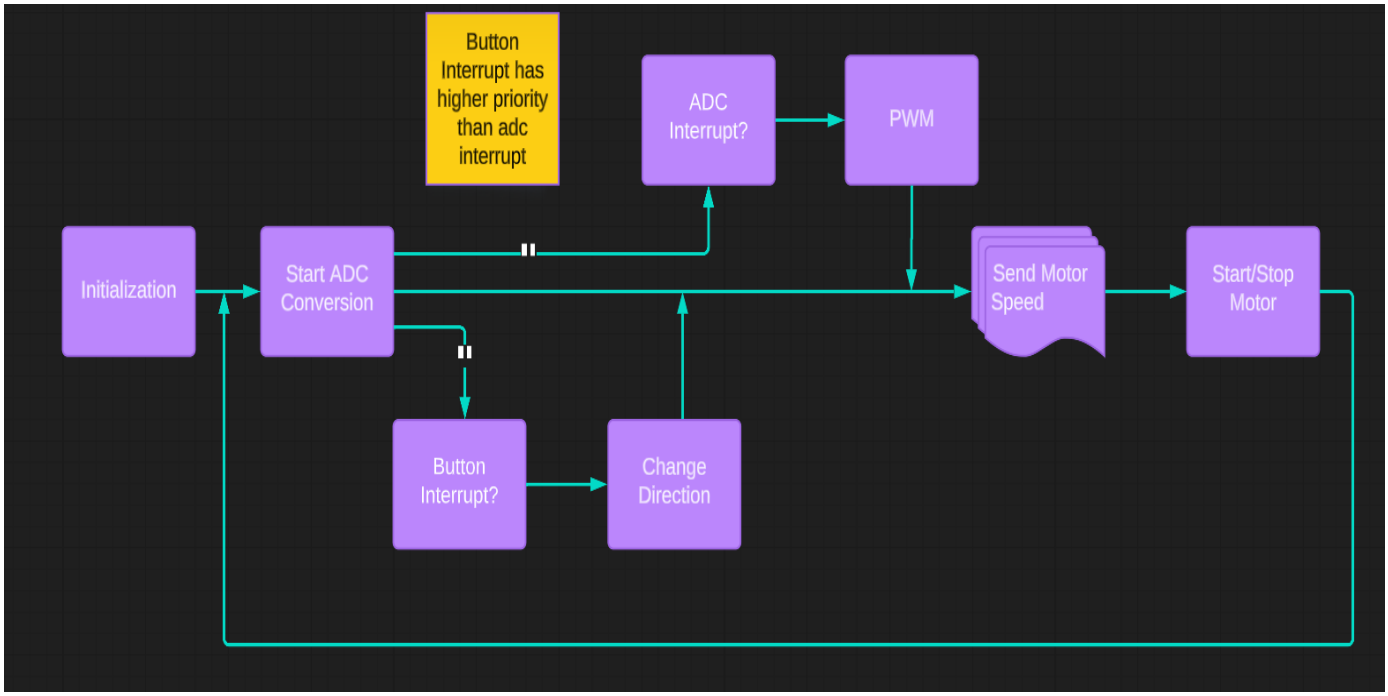


Figure 1:Sequence of Operation

Motor Speed Control Methodology

Analog-to-Digital Conversion (ADC)

ADC (Analog to Digital converter) is the most widely used device in embedded systems which is designed especially for data acquisition. In the AVR AT-mega series normally 10-bit ADC is inbuilt in the controller.

The ADC is used in this project by taking the output voltage of a potentiometer and providing it to the AT-mega and it then converts the analog voltage to a binary number from 0 to 1023 which is then used in PWM.

Pulse-Width Modulation (PWM):

Pulse Width Modulation (PWM) is a square wave with varying low and high amplitude signal. A general PWM signal is given in a figure below:

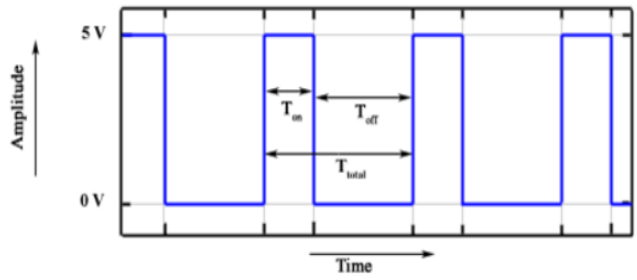


Figure 2:PWM

We used to the returned value from the ADC to initialize our modulated signal Duty Cycle which will change the motor speed.

Duty Cycle:

Calculation of duty cycle is done by calculating the ON-time from total period of time. It is a ratio between ON-time and total time period of the signal using period calculation, duty cycle is calculated as shown in the equation below:

$$D = \frac{T_{on}}{(T_{on} + T_{off})} = \frac{T_{on}}{T_{total}}$$

Serial Communication:

Sending motor relative speed to the two main sources of display LCD and Blynk IOT cloud (Using Node-MCU v3) and we can turn ON/OFF the motor from the Blynk cloud.

AVR Tools

Timers:

Timers are used to generate PWM. As we know that the frequency is number of cycles per second that the timer runs at. So the higher frequency will give us a faster timer. In generating PWM, a faster PWM frequency will give finer control over the output because it can respond faster to new PWM duty cycles.

We setup the timer0 which is 8 bits register to be in fast PWM mode and initialized OCR0A with the value read (desired speed) from the potentiometer.

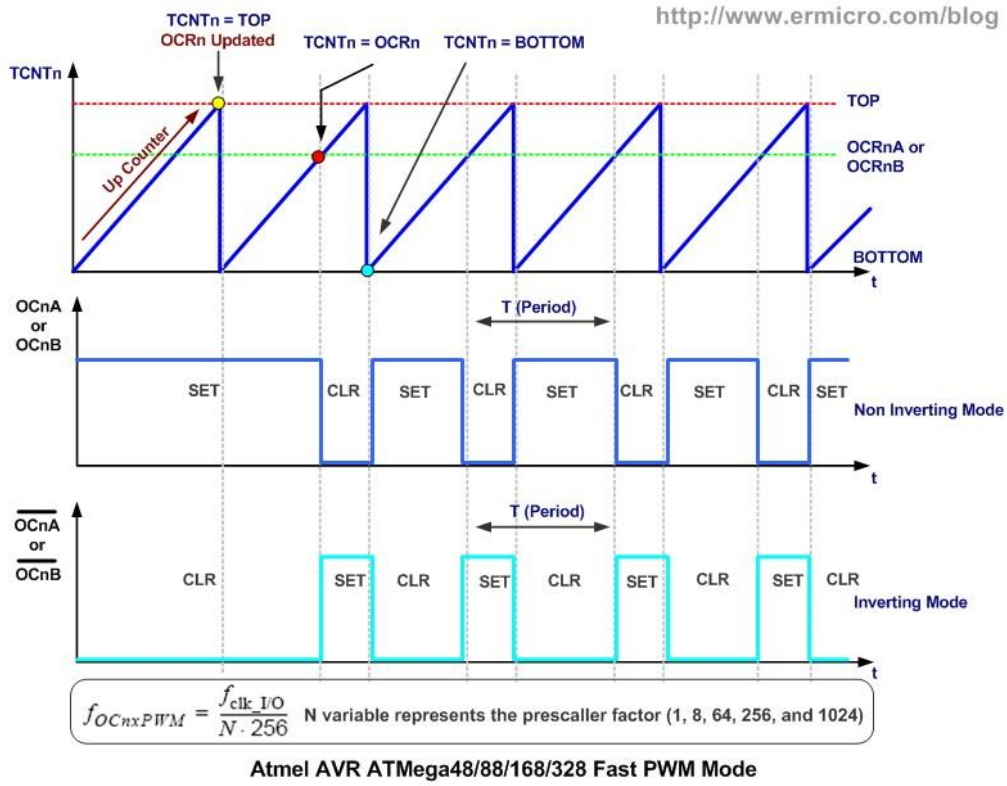


Figure 3:Fast mode PWM

Interrupts:

An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. Whenever an interrupt occurs, the controller completes the execution of the current instruction and starts the execution of an **Interrupt Service Routine** (ISR) or **Interrupt Handler**. ISR tells the processor or controller what to do when the interrupt occurs. The interrupts can be either hardware interrupts or software interrupts.

Hardware Interrupt:

A hardware interrupt is an electronic alerting signal sent to the processor from an external device, like a disk controller or an external peripheral. For example, when we press a key on the keyboard or move the mouse, they trigger hardware interrupts which cause the processor to read the keystroke or mouse position.

Software Interrupt:

A software interrupt is caused either by an exceptional condition or a special instruction in the instruction set which causes an interrupt when it is executed by the processor. For example, if the processor's arithmetic logic unit runs a command to divide a number by zero, to cause a divide-by-zero exception, thus causing the computer to abandon the calculation or display an error message. Software interrupt instructions work similar to subroutine calls.

This project used both interrupts as the hardware interrupt (INT0) was used to detect the sudden falling edge of a push button which leads to the change of the motor direction.
The Software interrupt was used by the ADC (ADC_vect) to initialize the value of the OCR0A which is used in declaring the Duty Cycle.

Analog-to-Digital Converter (ADC):

We used ADC0 and setup its clock to be 125KHZ (Take sample every 8μ sec) which is the maximum frequency that can be taken as the ADC takes a long time in conversion.

GPIO:

PORTD: We initialized PD-4,5,6,7 As Outputs.4 and 5 are used for the direction of the motor,6 for controlling the speed of motor and 7 for the Start and Stop of the motor.
PD-2 was initialized as an input with a pull-up resistor to control the motor direction.

PORTB: Connection: 1) lcd Pin 1 (GND) connected to Arduino GND 2) lcd Pin 2 (Vcc) connected to Arduino Vcc 3) lcd Pin 3 (Vo) connected to Potentiometer output 4) lcd Pin 4 (Rs) connected to Arduino PB4 5) lcd Pin 5 (R/W) connected to Arduino GND 6) lcd Pin 5 (E) connected to Arduino PB5 7) lcd Pin 11 (DB4) connected to Arduino PB0 8) lcd Pin 12 (DB5) connected to Arduino PB1 9) lcd Pin 13 (DB6) connected to Arduino PB2 10) lcd Pin 14 (DB7) connected to Arduino PB3.

UART:

The universal asynchronous receiver-transmitter (UART) takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Serial transmission of digital information (bits) through a single wire or other medium is less costly than parallel transmission through multiple wires.

The UART communication is configured to use the UART0 peripheral of the AVR microcontroller. We initialized the baud-rate to be 9600 which is set by The UBRR register .if the clock frequency (F_CPU) is 16 MHz and you want to set the baud rate to 9600 bps, the calculation would be: UBRR = (16,000,000 / (16 * 9600)) - 1 = 103 and enabled RX and TX and connected it the TX and RX of the Node-MCU in order to communicate with it which connects to the Blynk cloud using its Wifi-module.

Wiring

Overview of Wiring Requirements:

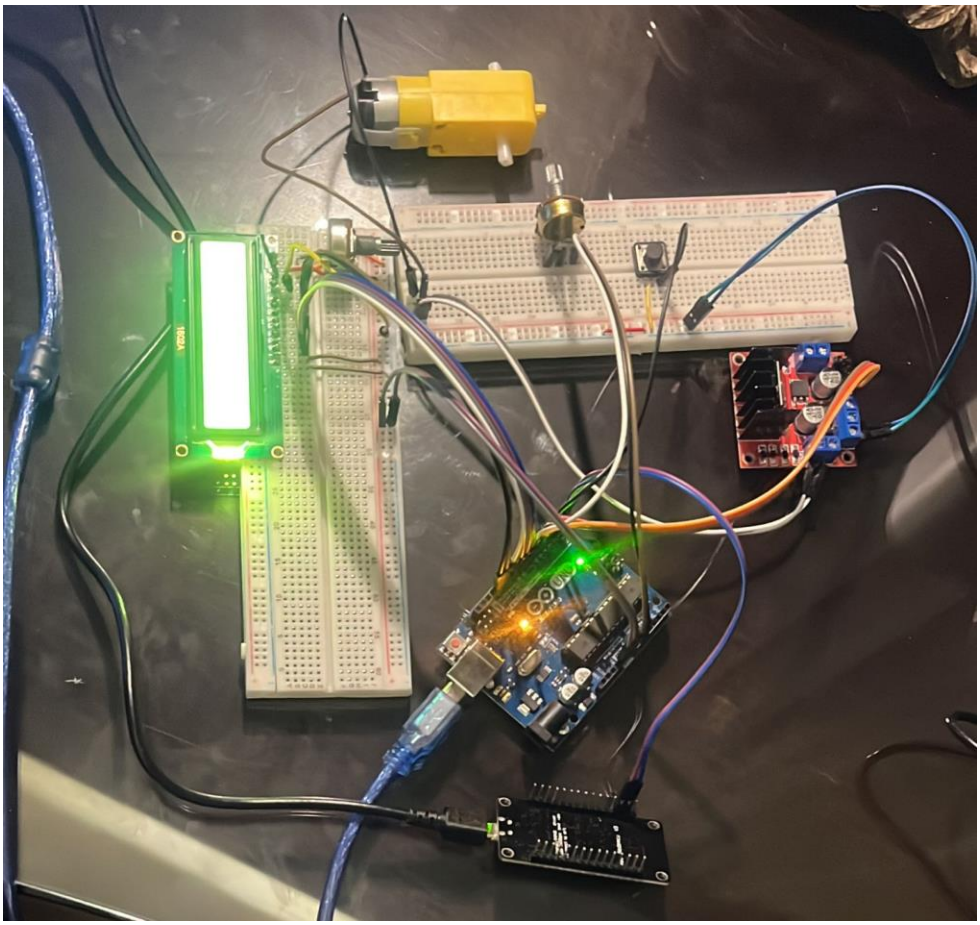


Figure 4:Wiring of the Circuit

Modules Used

1. AT-Mega328p Microcontroller.
2. LCD.
3. H-BRIDGE (L298N).
4. DC Motor.
5. Wires.
6. Board.
7. Potentiometer.
8. Node-MCU.
9. Blynk-Cloud.