IIT Dharwad

Semester-5 Research and Development Project

Embedded Programming using ATMega2560

Driving a motor via 3-phase PWM Generation

Supervisor: Prof Abhijit Kshirsagar
Assistant Professor, Dept. of Electrical Engineering.

Tanish H Talapaneni
200020050
Electrical Engineering

Contents

- What are Embedded Systems and Arduino?
- Aim of this RnD
- About ATMega2560
- Programming Requirements and implementation
- GPIO
- ADC
- Generation of 3-phase PWM using timer modules
- Operation set-up and driving of induction motor, V/f control
- UART
- Conclusion and Further Scope

What are Embedded Systems?

- An embedded system is a combination of computer hardware and software designed for a specific function.
- The term embedded means that these devices always function as a part of a complete device.
- They are generally of low-cost and low-power-consuming. They comprise of processor, power supply, and memory and communication ports.

What's Arduino?

- Arduino is an open-source electronics platform based on easy-to-use hardware and software.
- Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming.
- Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments.
- Arduino boards are relatively inexpensive compared to other microcontroller platforms.
- The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well.

Aim of this RnD

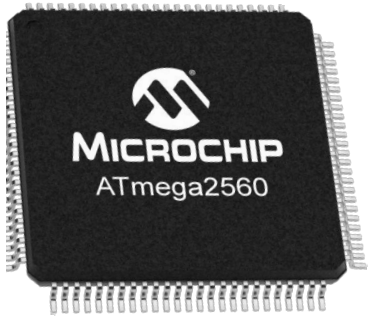
- Understanding how to program the ATMega2560 MCU using low level C Programming. This can be extended to program a wide variety of chip-sets.
- Configuration of GPIO and accessing the required registers.
- Implementation of ADC- Analog-Digital Converter.
- Generation of PWM Signals whose duty cycle varies sinusoidally. Three different waves are generated in three phases, and deadtime is added in order to avoid shoot-through.
- Using these PWM signals in order to drive a motor via a 3-phase inverter. V/f control of the motor.
- Understanding Communication Protocols in general, implementation of USART

What's ATMEL and ATMega?

Atmel Corporation was a creator and manufacturer of semiconductors before being taken over by Microchip Technology in 2016. Atmel was founded in 1984. The company focused on embedded systems built around microcontrollers.

ATMega Microcontrollers belong to the AVR family of microcontrollers and is manufactured by Atmel Corporation. An ATMega Microcontroller is an 8-bit microcontroller with Reduced Instruction Set (RISC) based Harvard Architecture.

AVR refers to Alf and Regards' RISC Processor



Features of ATMega2560

ATMega2560 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. Its features are as follows:

Parametrics:

Name	Value
Program Memory Type	Flash
Program Memory Size (KB)	256
CPU Speed (MIPS/DMIPS)	16
SRAM (KB)	8,192
Data EEPROM/HEF (bytes)	4,096
Digital Communication Peripherals	4-UART, 5-SPI, 1-I2C
Capture/Compare/PWM Peripherals	4 Input Capture, 4 CCP, 16PWM
Timers	2 x 8-bit, 4 x 16-bit
Number of Comparators	1
Temperature Range (°C)	-40 to 85
Operating Voltage Range (V)	1.8 to 5.5
Pin Count	100

Method of implementation: Low-level C Programming

- A low-level programming language provides little or no abstraction from a computer's instruction set architecture.
- The commands or functions in the language map are structurally similar to a processor's instructions. Generally, this refers to either machine code or assembly language.
- In this RnD, we use low-level C Programming as it is both understandable by the AVR processor and is easier to interpret by the user as well.

About the Programming requirements

- gcc-avr : a GNU C cross-compiler designed specifically for AVR.
- avr-libc: it is a package for AVR C library.
- AVR downloader uploader(avrdude): software(utility program) for downloading and uploading on-chip memories of Microchip's AVR microcontrollers.
- Arduino CLI: Arduino Command LIne Interface- Arduino CLI is a command line tool that contains all you need to easily build applications around the Arduino ecosystem.

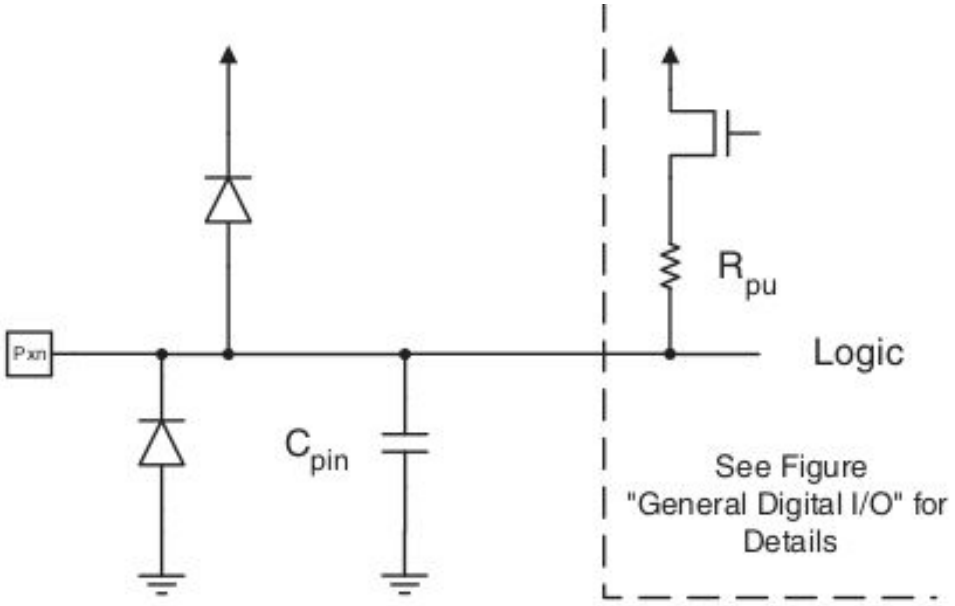
Steps of Implementation

- Compilation which creates object files.
- Linking the object files to create ELF files(Executable and Linkable Format)
- Object-copy(assembly) to create hex files.
- Flashing(dumping) the hex file on the board using avrdude.
- Execution is done using make- GNU make:

  The "make" utility automates the mundane aspects of building executable from source code. "make" uses a so-called makefile, which contains rules on how to build the executables. Make is different from a script as a script shows no intelligence. All instructions run blindly, without depending on the consequences of the previous instructions.

GPIO and Registers involved

- The ports are bi-directional I/O ports with optional internal pull-ups.
- Each port pin consists of three register bits: DDxn, PORTxn, and PINxn.
- The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.
- If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin.
- If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high.
- If PORTxn is written logic zero when the pin is configured as an input pin, the port pin is driven low.
- Most port pins have alternate functions in addition to being general digital I/Os.



I/O Pin Schematic

ADC

- An ADC converts a continuous-time and continuous-amplitude analog signal to a discrete-time and discrete-amplitude digital signal.
- The ATmega2560 features a **10-bit** successive approximation ADC.
- ADC Value=(Vin*1024)/Vref

- **ADCSRA: ADC Control and Status Register**
- This register is used to enable the ADC and start conversion.
- Prescaler(division factor between f-osc and ADC clock) can be set and the status of Interrupt flag (ADIF) can be known

- **ADMUX: ADC Multiplexer**
- To select the reference voltage and the input channel.
- Bit 5 = Result(10-bits-wide) is left-justified. Left-justified is useful for arithmetic operations point of view.

- ADCL: ADC Data Register Lower= stores the lower 8 bits of result of conversion
- ADCH: ADC Data Register Higher= stores the higher 8 bits of result of conversion

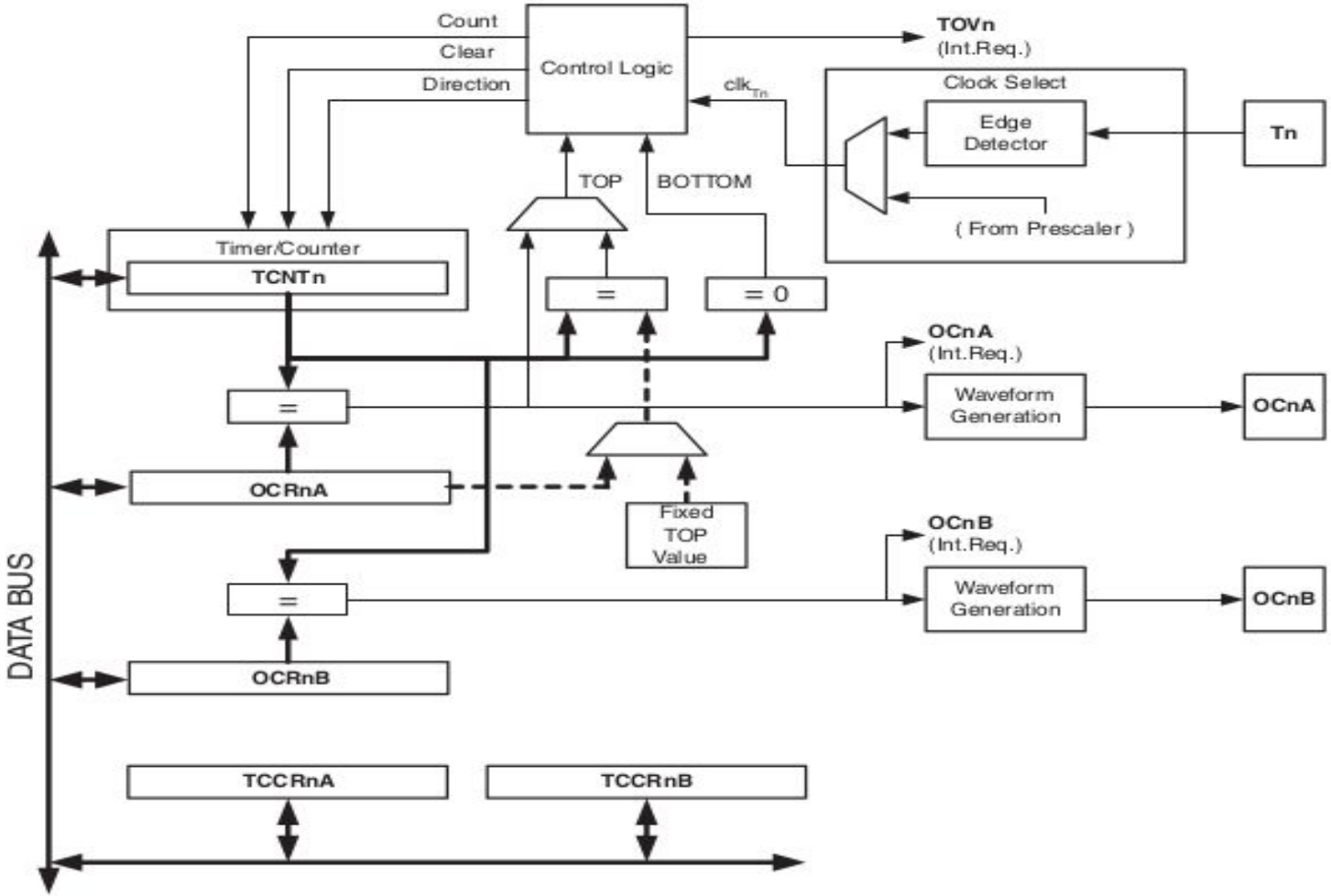
Timer modules in ATMega2560

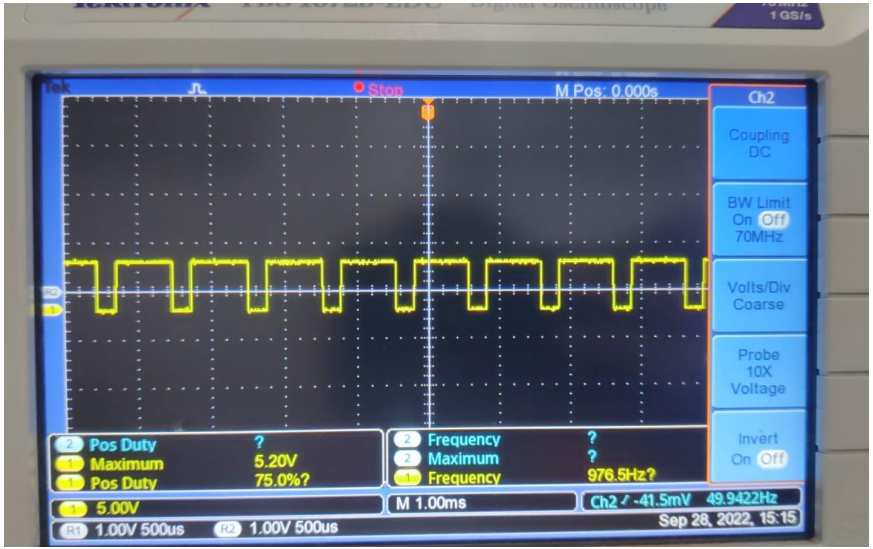
Different types of timers involved:

Timer/Counter0 and 2 are general purpose 8-bit Timer/Counter modules, with two independent Output Compare Units, and with PWM support. They are generally used for accurate timing measurement of events and wave generation. Similarly, Timer/Counter1, 3,4 and 5 are general purpose Timer/Counter modules, but are of 16-bit-type.

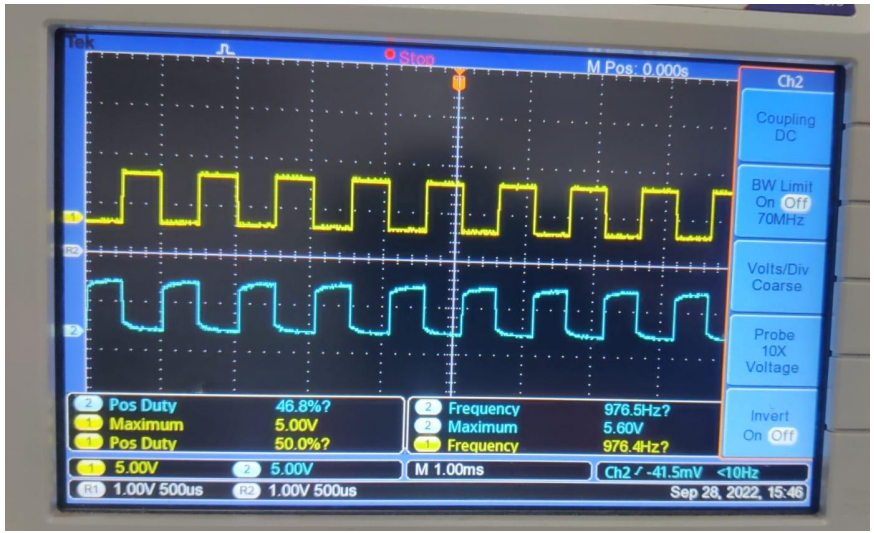
Generating PWM Signals:

Value loaded in Output Compare Register(OCR) is compared with the Timer value,and when the value matches, the specific pin gets toggled. The pin is reset again at the top/bottom value based on the mode selected. The duty cycle of the PWM varies according to the value loaded in the OCR.


Count
Clear
Direction
Control Logic
TOVn
(Int.Req.)
clk Tn
Clock Select
Edge
Detector
Tn
(From Prescaler)
TOP
BOTTOM
Timer/Counter
TCNTn
=
= 0
OCnA
(Int.Req.)
Waveform
Generation
OCnA
OCRnA
Fixed
TOP
Value
OCnB
(Int.Req.)
Waveform
Generation
OCnB
OCRnB
DATA BUS
TCCRnA
TCCRnB



Fast PWM signal of 1 kHz with 75% duty cycle

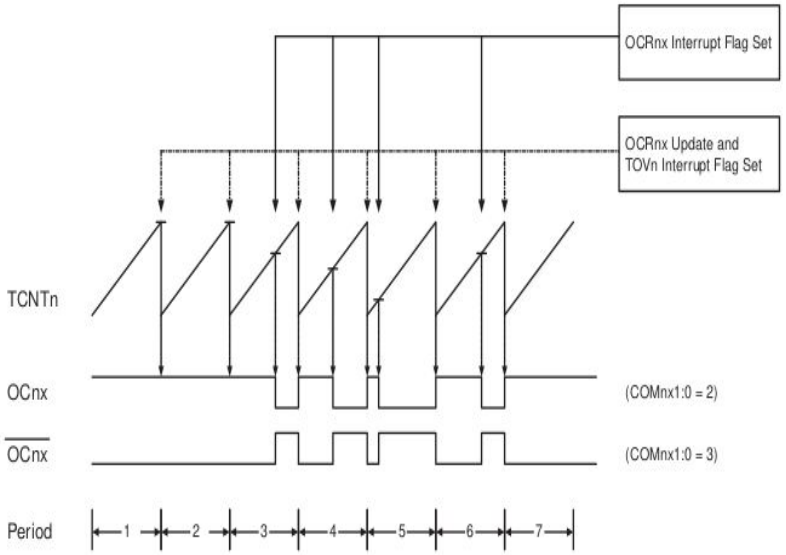


2 complementary PWM Signals

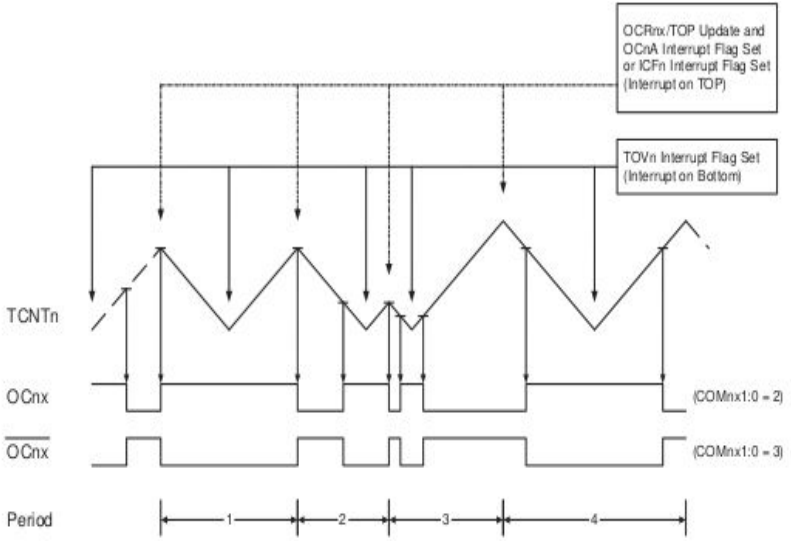
PWM Generation and Look-up-Table

Fast mode and Phase-Correct mode:
Fast PWM refers to the 8-bit mode. The OCRnX and Timer registers are compared only once within a cycle of the PWM signal being generated. In Phase-correct PWM, the OCRnX and Timer registers are compared twice within a cycle of the PWM signal being generated. Phase-correct mode refers to the 16-bit PWM being used. Hence phase-correct mode is helpful in order to induce deatime.

Look-up-Table and phase-control:
The values in this array, resembles the samples of a sine wave. We obtain these look-up-table values from a regular sine-wave, after shifting by 1, and then scaling the amplitude to a size of 10 bits(i.e.max value-1024). The output compare register stores these values one by one at a time, and thus the corresponding duty cycle of the PWM signal is set. The number of values in the look-up-table is 360, which correspond to the 360 samples(1 sample/degree) of 1 period of a sine wave. The index position in the look-up-table from where the values start to get loaded in OCRnX, will indicate the phase of the signal(wrt to a reference). This is how a signal of a any phase can be generated.



Fast-PWM Mode



Phase-Correct-PWM Mode

3-phases of the sinusoidal PWM signal generated

Deadtime

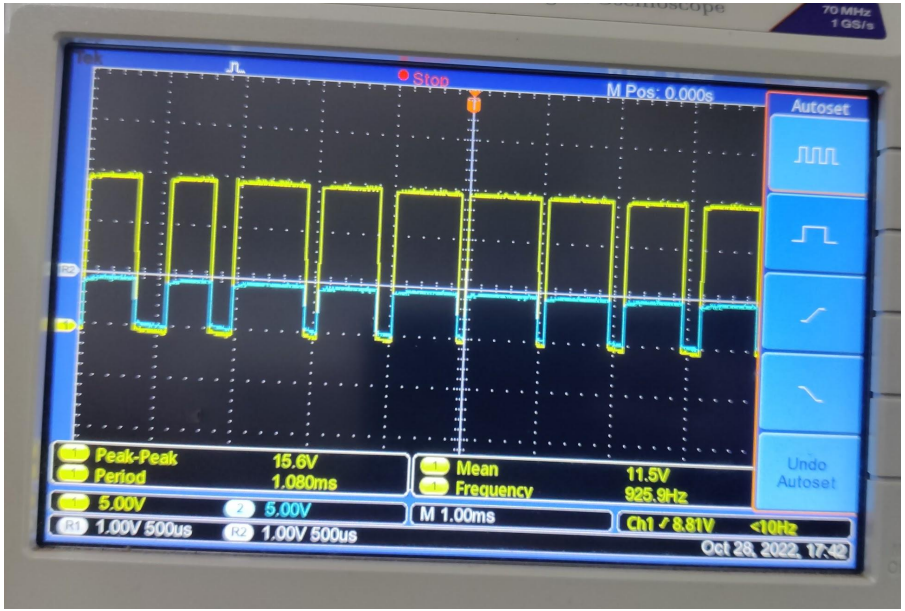
Deadtime refers to the necessary blanking time between top and bottom switch to avoid inverter leg shoot-through of the DC bus circuit. The type of the switch and pre-driver characteristics determine amount of the necessary blank time. We induce a deadtime of approx 4 microseconds.

The deadtime is induced by making a modification in the value after it is referred from the look-up-table. A small amount is subtracted from the value to be loaded in OCR so that duty cycle is reduced to a small extent. This induces a delay between the falling edge of one signal and rising edge of the corresponding complementary signal.

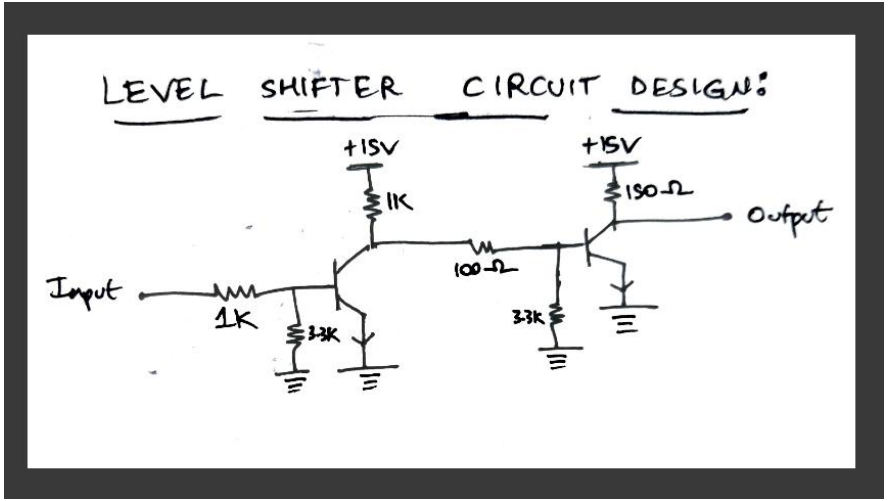
Level Shifter

The output obtained from the Arduino controller is just 5V at max. We intend to drive a 3-phase inverter in order to generate the required 3-phase signals. The voltage required to drive the MOSFETs is around 15V. Hence we need to amplify the signals obtained from the Arduino controller. We achieve this using the transistor level-shifter.

The transistor used is 2N222a



Input and output of the level-shifter circuit

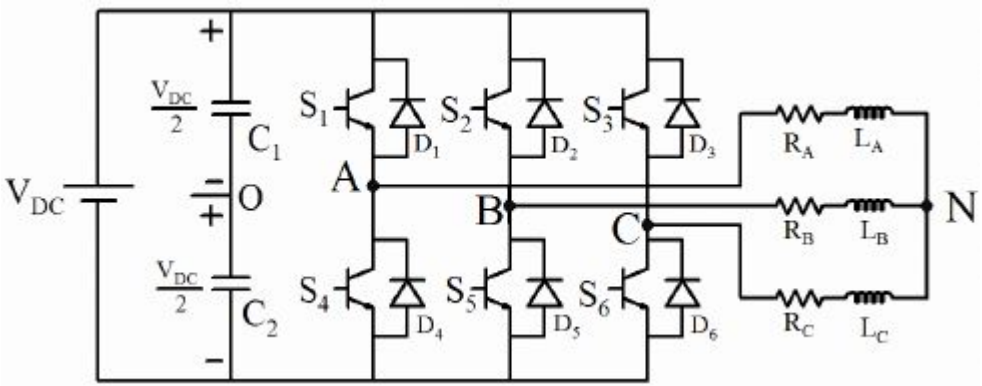


Level Shifter Circuit

3-phase Inverter

The total number of signals required to drive the 3-phase inverter is 6, i.e. 3 signals(with a phase difference of 120-degrees between any pair of them) and their corresponding complementary signals.

The output voltage waveform varies according to the variation in duty cycle. Since our duty cycle varies sinusoidally, the output voltage is also roughly sinusoidal. 3 phases are generated in R,Y,B legs of the 3-phase inverter.

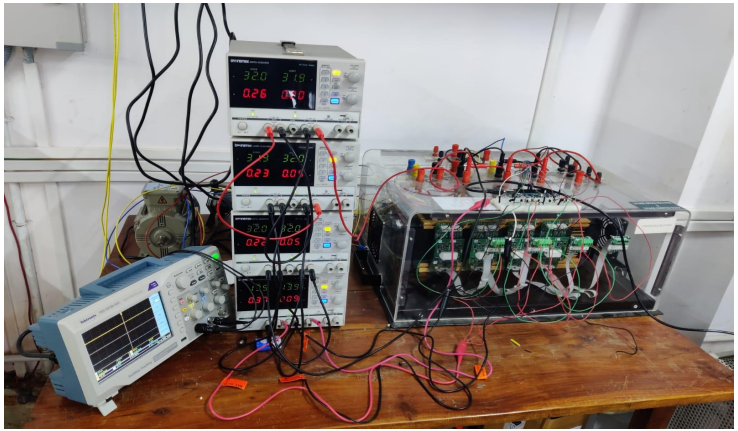


3-phase inverter circuit

Operation Set-up

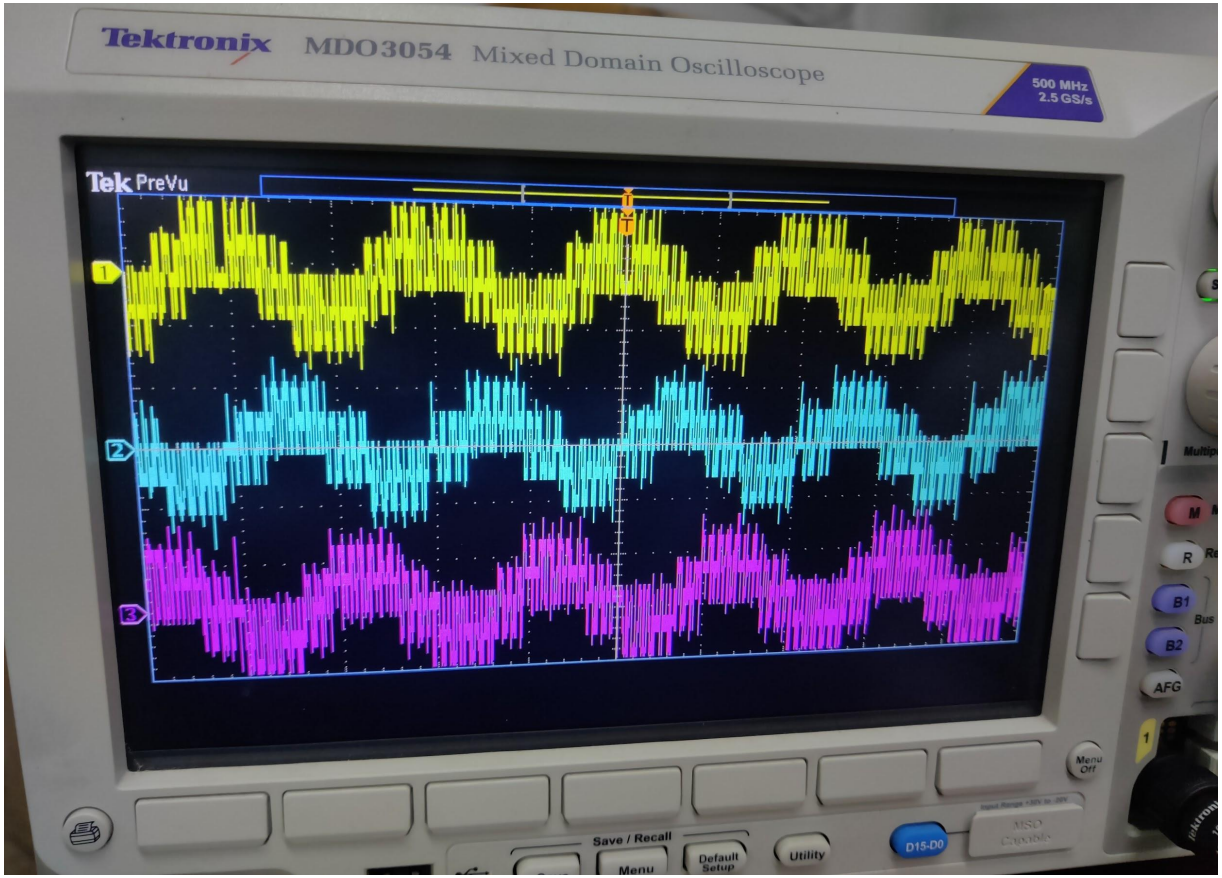
Now, these 3 signals are loaded into an induction motor. The three phases together generates a rotating magnetic field, which thus rotates the rotor.

The induction motor is rated to operate at 415V. The three phase AC mains is used an input source. The input source is rectified into DC and is loaded into the three phase-inverter. An auto-transformer is used in order to vary the input voltage that is fed to the rectifier.



Testing of 3-phase output using
DC Load-bank of 180V

Operation set-up to drive the 3-phase motor



3-phase sinusoidal inverter output

V/f Control Method

The speed of rotation of the motor is directly dependent on the frequency of the sinusoidal signal obtained. The speed is equal to (120*f/p). At the same time, we ensure voltage is also halved so that we can maintain a constant motor magnetic flux. Magnetic flux is proportional to V/f and reducing only the frequency would lead to increase in magnetic flux which causes magnetic saturation. On reducing the duty cycle by half, the overall average voltage is also halved. The result we obtain is the speed of rotation decreasing by half.

This frequency is halved by roughly doubling the delay(from 50us to 100us) involved in between each change of duty-cycle.

Experimental results: (the speed of rotation is measured using tachometer)
Setup1: Speed- 1450 rpm
Setup2: Speed- 715 rpm

UART

Embedded systems use the communication ports to transmit data between the processor and peripheral devices, which are often other embedded systems. This communication is governed by a set of rules and methods, known as a communication protocol.

UART is a simple protocol, where data is transmitted frame by frame serially. The transmitter and receiver are asynchronous to each other.

There are totally 4 UART modules in ATMega 2560 and we check transmission and reception of data using these modules.

Registers involved in UART

UDRn – USART I/O Data Register: Both Tx and Rx buffer registers share the same address. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDRn Register location. Reading the UDRn Register location will return the contents of the Receive Data Buffer Register (RXB).

UCSRnA/B/C – USART Control and Status Registers A,B and C: These registers are used to control the various parameters involved in transmission of data. These include- no. of transmission bits, stop bits, parity bits etc. These registers also indicate the status with the help of flag-bits involved in the transmission.

UBRRnL and UBRRnH – USART Baud Rate Registers: 13 out of these 16 bits in these two registers are used in order to set the baud rate of transmission.
Formula used: f-osc/16(UBRRn+1)

Conclusion and Further Scope

In this RnD project, we have successfully dealt with the configuration of various modules in the ATMega2560 chip. We have understood the functioning of these modules and we have also dealt with low-level embedded programming. Using this knowledge, it is easier now to work with various types of chipsets that are available out there.

The main aspect of this project was the understanding of the generation of 3-phase PWM Signals. When these PWM signals were plugged into the 3-phase inverter in order to drive the motor, the motor ran as expected, but with a lot of harmonics, indicated by the noise. Thus, there is scope for generation of better PWM signals.

The Arduino board can be communicated to via the cloud/Wifi using ESP32 boards. With this we can control the speed of the motor using smart devices and other IoT(Internet-of-things) Technologies that maybe required.

References

- Arduino Schematic to refer to pin configurations on the board.
- Book- AVR Microcontroller and Embedded Systems using Assembly and C, by Muhammad Ali Mazidi, Sarmad Naimi, Sepehr Naimi.
- ATmega640/1280/1281/2560/2561 Datasheet.

Acknowledgement

I would like to express my sincere gratitude to my supervisor Prof Abhijit as well as Mr. Karthik Kumar for helping me throughout this Research and Development Project. I came to know about so many new things and has given me a great hands-on experience in working with embedded systems and electrical machines.

Thank You!!