**0. General Information**

**Student Name:** Ellis Hobby

**Student ID Number:**  01928869

**Team Name:** Tube Disaster

**Team Member Names:** Ellis Hobby

**Date of Completion:** 04/12/23

**Demonstration Method:** Zoom

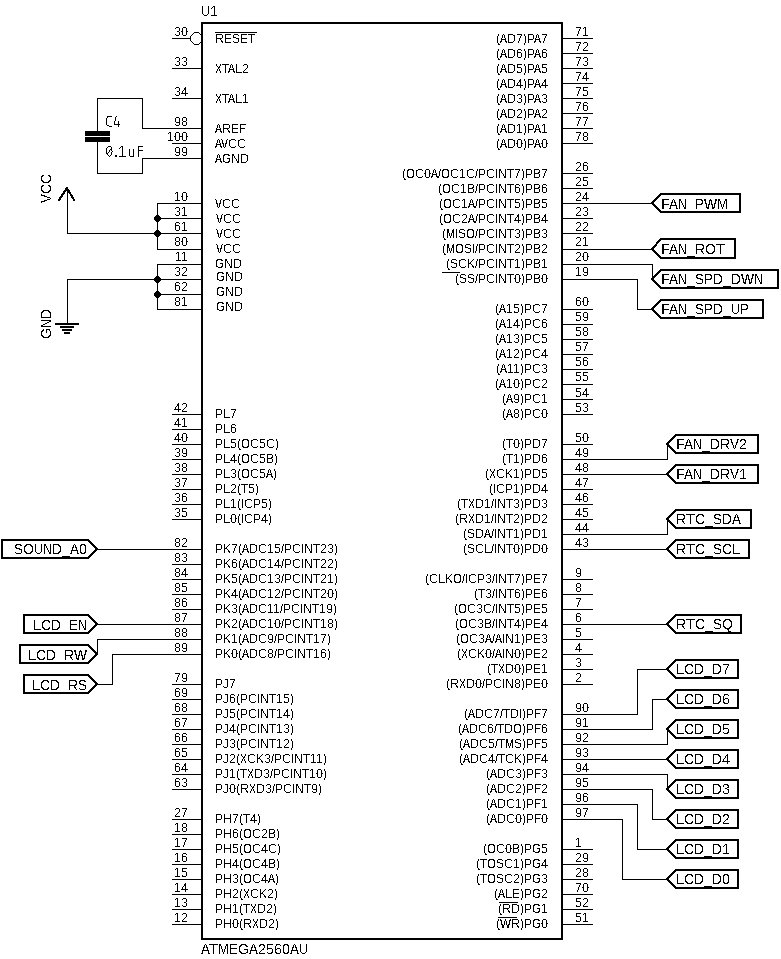
**1. Design**

The current project requires the design of an embedded system that can be used to control a fan driven by a direct current (DC) motor. The fan shall run in either direction; clockwise (CW) or counterclockwise (CCW), and have four discrete speed selections; off, 1/2, 3/4, and full. The system shall include a real time clock (RTC) for keeping track of time passed since system startup. A 16x2 liquid crystal display (LCD) shall be used to indicate the current time, fan speed, and fan rotation. Buttons shall be used to change the fan speed and direction. Additionally, a sound sensor shall be used to detect frequency input and change the fan speed according to the following stimulus: consecutive inputs of C4 (262 Hz) to A4 (440 Hz) will cause the fan to speed up. Similarly, a reversed input of A4 to C4 shall cause the fan to decrease in speed. The frequency detection shall be allowed a 2% margin of error, where C4 will then be recognized between the 256.76 to 267.24 Hz frequency range and A4 within the 431.2 to 448.8 Hz range [1]. Full system schematics can be found in the Appendix.

**1.1 Hardware Design**

The core of the system is built upon the ATmega2560 from Atmel. This 8-bit microcontroller (MCU) has a processing speed of approximately 16 MHz and a multitude of input-output (IO) capabilities that are well suited for the desired system functionality, namely the pulse width modulation (PWM) module, inter-integrated circuit bus (I2C), and analog to digital converter (ADC). The PWM module allows us to create a square wave with a variable pulse width at some arbitrary frequency within the limits of the system clock (16 MHz). The DC voltage that appears at the output of the PWM is directly proportional to the on-time of the pulse within one cycle [2]. This property can be then used to create a variable voltage source for the fan speed control. The I2C bus is used to manage communication with the RTC module for timekeeping data. The ADC permits analog voltage measurements generated by the sound sensor to be utilized for processing by way of digital conversion. The general-purpose digital input and output (GPIO) pins are used for transmitting display data to the LCD. Additionally, where available, the GPIO can be used to asynchronously interrupt the program for important data input such as fan rotation or speed updates. Figure 1 shows a schematic for the MCU circuit block, the net labels were used to indicate how each pin connects to the external peripherals.

Figure 1: Atmega2560 Circuit Block



The DC motor, Figure 2, was configured to operate on a dedicated 5V power supply separate from the digital circuitry. This helped to reduce unwanted noise within the digital power lines that may result from electromagnetic interference (EMI) or transient current spikes that occur while running the motor. To interface the digital circuitry for motor control the L293 H-bridge motor driver integrated circuit (IC) was used. This chip features two power supply inputs, one for the internal logic control circuity (VCC1) and one for the dedicated analog motor supply (VCC2) [3]. Figure 3 shows the fan and motor driver circuit block, here MCU was configured to feed two of the driver control inputs 1A and 2A using pins PH5 and PH6. These pins were used to determine how the driver routes the polarity of the motors output power; this enabled the rotational direction to be changed. Pin PB5 was configured for PWM output and connected to enable 1 (EN1) on the L293. This pin turns the output power to the fan on an off. Feeding it with a PWM signal allows us to vary the relative voltage that appears on the outputs, thereby enabling the motor speed to be changed. The motor leads were connected to outputs 1Y and 2Y on the L293. The enable input (1EN) was connected to the PWM signal which pulsed the driver the outputs (1Y,2Y) and activated the motor. The control inputs (1A,2A). Figure 4 depicts a logic diagram provided in the datasheet. This can help visualize how these mechanisms are implemented within the chip and how they may be applied within the current system.

Figure 2: DC Motor with Fan Blade

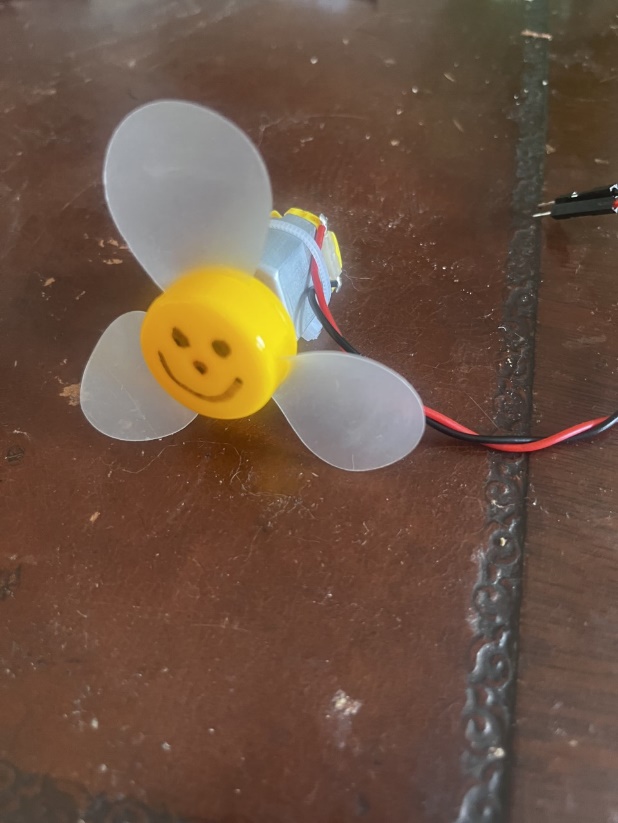


Figure 3: DC Motor with L293 Driver Circuit Block

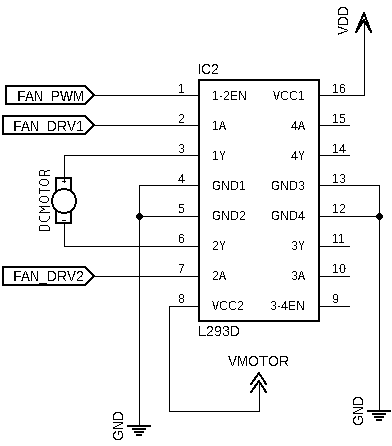
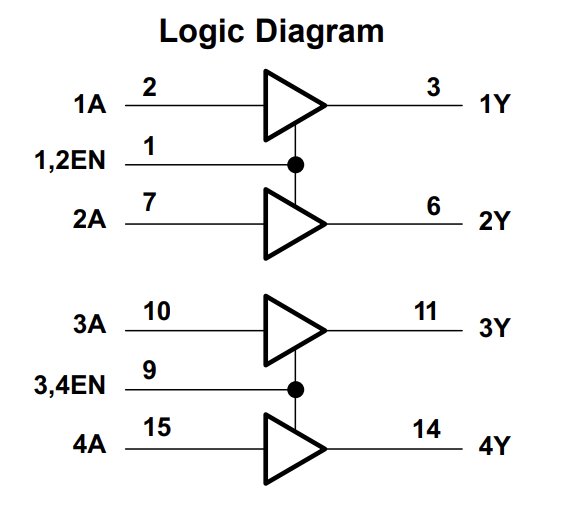


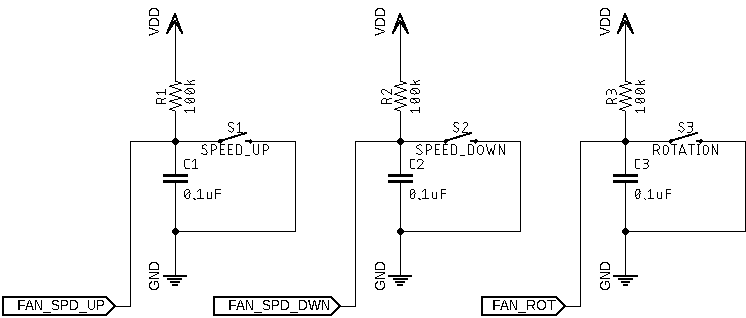
Figure 4: L293 Logic Diagram



Motor control was achieve using three tactile switches for speed up, speed down, and rotation direction, Figure 5. These were connected to MCU pins PB0, PB1, and PB2 respectively, this enabled them to be used as Pin Change Interrupts (PCINT0) for asynchronously updating the motor parameters. To ensure reliable data input each switch line includes a low pass filter for reducing unwanted mechanical noise. Each low pass filter is constructed using a 0.1 μF capacitor and 100 kΩ resistor to create a delay of approximately 10 ms, as described by Equation (1) [4].

**( 1 )**

Figure 5: Fan Switch Control Circuit Block



The Sound Sensor, Figure 6, module was used as an additional control mechanism for motor speed control. As described above a sequence of C4-A4 was used as speed up command and A4-C4 for speeding down. Figure 7 shows the Sound Sensor circuit block, here the analog output of the Sound Sensor (A0) was connected to ADC15 (PK7) on the MCU. This enabled the signal to be converted into the digital domain for frequency analysis. The module provides a trim pot for adjusting the gain of the built-in amplifier. It is important to properly set this so that the sensitivity will not be too high or low. Additionally, understanding the output voltage range determines the proper reference voltage for the ADC. The chosen setting for this was achieve using an oscilloscope to determine peak voltage values, Figure 8.

Figure 6: Sound Sensor Board

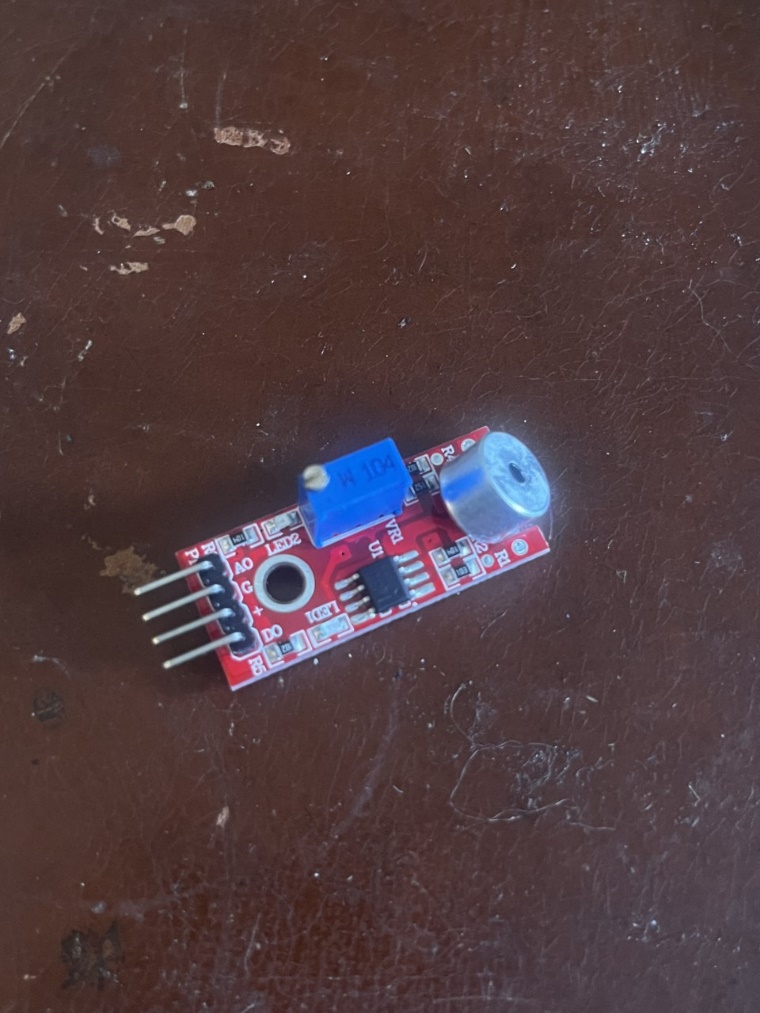


Figure 7: Sound Sensor Circuit Block

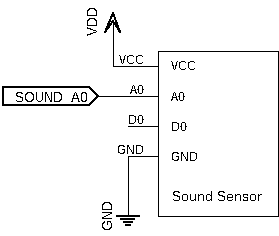
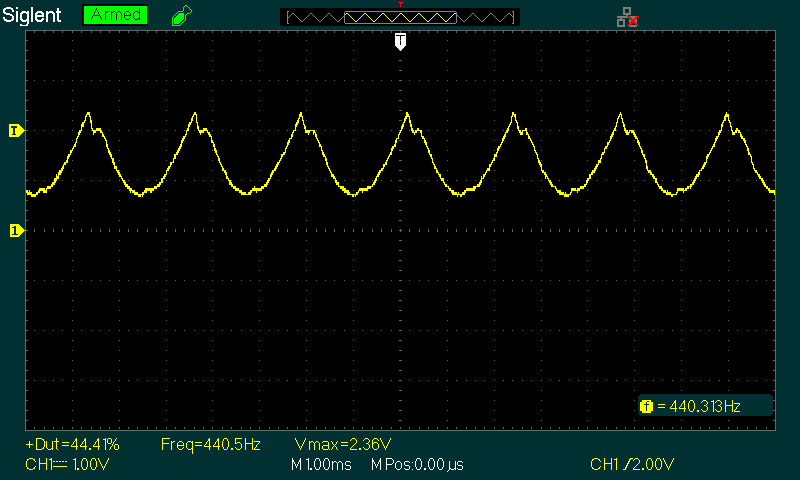


Figure 8: Sound Sensor Input Oscilloscope – A4 (440Hz) Input

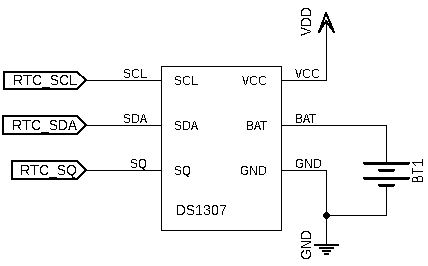


The DS1307 RTC module was used for time keeping, Figure 9. This device includes a battery back up to ensure that the time passage is still tracked when the main power supply is removed. The internal clock/calendar tracks seconds, minutes, hours, day, data, month, and year [5]. This information is accessible by accessing the devices internal storage registers using the I2C communication protocol. The clock is configurable to 24-Hour or 12-Hour with AM/PM format by writing to its internal control registers. The current system uses a 24-Hour formatting scheme. Additionally, the device offers a square wave output pin that is based on the internal clock. This square wave may be set to 1 Hz, 4.096 kHz, 8.192 kHz, or 32.768 kHz which enables external devices to directly synchronize with it. Figure 10 shows the circuit block for the RTC module within this system. The serial clock (SCL) and serial data (SDA) lines were routed to MCU pins PD0 and PD1 respectively. The square wave output was configured to 1 Hz and connected to PE4 which used for external interrupts (INT4) from the RTC.

Figure 9: DS1307 RTC Module



Figure 10: RTC Circuit Block



An LCD screen was used to display the current time, the fan speed, and the fan rotation direction. The module, Figure 11, is in a 16x2 format, where 16 is the number of characters per line and 2 is the number of lines. For interoperating screen control data, the module uses the Hitachi HD44780U LCD driver. This driver accepts parallel data input on pins D0-D7 as bytes (8-bit) or nibbles (4-bit) [6]. The current application uses byte communication, therefore, all of PORTF (PF0-PF7) on the MCU was reserved for sending LCD data messages. Three additional control inputs are required for interpreting values on the data bus; register select (RS), read/write select (RW), and enable (EN). These were connected to PK0, PK1, and PK2 respectively. For control over the screens contrast a 10 kΩ potentiometer was connected with its wiper to pin V0 on the LCD and its inputs connected to +5V and Ground. Figure 12 shows the layout for the LCD circuit block within the system schematic.

Figure 11: LCD Module

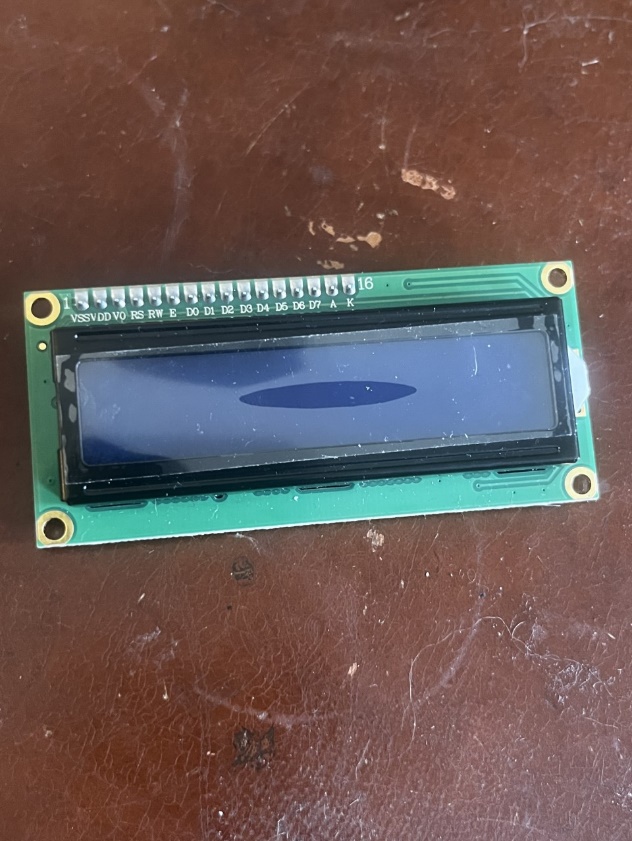
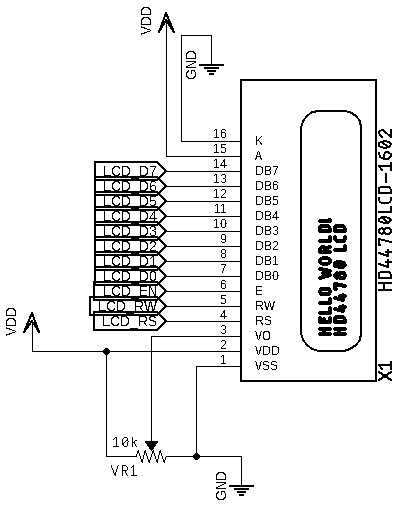


Figure 12: LCD Circuit Block



The system was built using three separate power sources with a common ground: one for the MCU, one for the digital peripherals, and on for the motor. This became necessary due to transient noise artifacts introduced onto the power lines while running the motor. Digital circuitry can be very sensitive to noisy power lines, so separation of analog and digital supplies was necessary. The MCU receives power from the universal serial bus (USB) port. The digital peripherals and motor received power from an Elenco XP-720 bench top power supply, Figure 13. This supply offers three output sources: +5V at 3A, + 0 – 15V at 1A, and - 0 – 15V at 1A. The +5V source was connected to the digital peripherals. The variable +15V source was set to +5V and used for the motor. It should be noted that the MCU was given separate power solely due to extra sources being available, typically it would be fine to connect it with other digital circuity.

Figure 13: Elenco XP-720 Power Supply



**1.2 Software Design**

The software design implements four custom data structures dedicated to performing the necessary functionality for each of the main peripherals (Fan, Sound Sensor, RTC, LCD), and one additional data structure for managing I2C transactions with the RTC module. All programming was done using low level AVR register manipulation in an effort to optimize processing time. Each modules class implementation was tested on it own with hardware to verify the expected behavior. This also enabled class specific bugs to be caught and fixed early on before complexity was increased during the integration stage. The test programs are included along with the main program in the projects GitHub repository. This is currently private with collaborator access only, to request access please email the author at: [ellis\_hobby@student.uml.edu](mailto:ellis_hobby@student.uml.edu).

**1.2.1 Fan**

The class in charge of managing fan/motor operation was name *PWM1*. The header file for this class defines the necessary pins and port access for all GPIO related to the fan functionality. During class initialization the two motor driver pins and PWM pin are configured as outputs. To generate the PWM signal *Timer1* was configured to *Fast PWM Mode* with 9-bit resolution. Its frequency is dependent on the clock prescaler value (*N*) and resolution (*TOP*) as determined by Equation (2) [7]. Using a value on 64 for the prescaler (*N*) and given 9-bit resolution (*TOP=511*) results in a PWM frequency of approximately 480 Hz. The three fan control switches are set digital inputs with internal pull-up resistors enabled and PCINT0 interrupts enabled.

The PWM signal essentially works in the background once configured correctly. Its theory of operation relates to the value loaded into the output compare registers (OCR1A). Every cycle of the timer clock ( *fcpu / N* ) increments the timer count register (TCNT1). When the count of TCNT1 reaches the value loaded into OCR1A the PWM output pin is set HIGH. The counter keeps incrementing until it reaches the TOP value and rolls over to 0x00 (BOTTOM). Upon reaching BOTTOM the PWM output is set LOW. To vary the pulse-width (on-time) we must load different values into OCR1A. Figure 14 shows a timing diagram included in the ATmega2560 datasheet which helps to visualize this process. For example, if the desired pulse-width was 25% then OCR1A must be loaded with a value of 128 ( (TOP+1) \* 0.25 = 512 \* 0.25 ) . The speed requirements of the fan for this system are off (0%), 1/2 (50%), 3/4 (75%), and max (100%). Knowing this we can then determine that OCR1A must be loaded with either 0, 256, 384, or 512 depending on the requested fan speed.

The four values required for each fan speed were stored into a private class member array called *\_speed*. Putting them in an array like this made constructing functions for speed control much easier as only two would be needed, *speedup( )* and *speedDown( )*. Within these functions a private data member for tracking (*\_speed\_idx*) can simply be incremented or decremented, modded with a value of 4 to stay within array bounds, then used to index the *\_speed* array and set the value of OCR1A. To switch the fan rotation a class method called *switchRotation( )* was created which simply inverts the values of the two motor driver pins. Two additional methods were created to return string values regarding the current state of speed and direction which could then be displayed on the LCD.

The three switches used for setting the fan speed and direction all jump to the same interrupt vector. This is the nature of how Pin Change Interrupts work. This required checks nested within the interrupt service routine (ISR) to determine which switch was pressed and achieved by saving the last state of the relevant port (PORTB). Using the known state allows us to perform an exclusive-or (XOR) with the current state which results in a value indicating which port pin has changed. This value can then be checked against bit masks for each switch. Once a match has occurred the relevant switch function can be dispatched.

**1.2.2 Sound Sensor**

The class in charge of managing acquisition and processing of data from the Sound Sensor was simply named *SoundSensor*. This class determines the fundamental frequency of a sample set using Fast Fourier Transform algorithms (FFT) provided in the *arduinoFFT* library. A total of 128 samples are taken per set at a sample rate of 880 Hz. This conforms to the Nyquist sampling theorem which states that the sampling rate must be double the highest frequency of interest (A4, 440 Hz) to guarantee an accurate discretized representation. The samples are taken using ADC15 (PK7) and the sampling rate is controlled by *Timer4*.

**( 2 )**

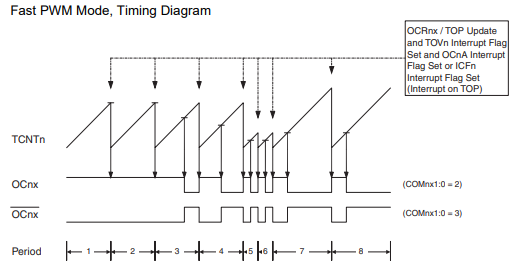


Figure 14: Fast PWM Mode Diagram [7]

# References

|  |  |
| --- | --- |
| [1] | Y. Luo, Lab 3: Controlling a Fan, Lowell: University of Massachusettes Lowell, 2023. |
| [2] | E. Williams, "Chapter 10. Pulse-Width Modulation," in *Make: AVR Programming*, Sebastopol, Maker Media, Inc., 2014. |
| [3] | T. Instruments, L293x Quadruple Half-H Drivers, Dallas: Texas Instruments Incorporated, 2016. |
| [4] | T. Instruments, Debounce a Switch, Dallas: Texas Instruments Incorporated, 2020. |
| [5] | M. Integrated, DS1307 64 x 8, Serial, I2, Sunnyvale: Maxim Integrated Products , 2015. |
| [6] | Hitachi, HD44780U (LCD-II) (Dot Matrix Liquid Crystal Display Controller/Driver), Japan: Hitachi, Ltd, 1998. |
| [7] | Microchip, "ATmega640/V-1280/V-1281/V-2560/V-2561/V - Complete Datasheet," in *16-bit Timer/Counter (Timer/Counter 1, 3, 4, and 5)*, Microchip Technology, 2000, pp. 133-163. |