

TEMPERATURE BASED FAN CONTROL

Nayankumar K. Patel
B.E., Gujarat University, India, 2005

PROJECT

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL AND ELECTRONIC ENGINEERING

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

FALL
2010

TEMPERATURE BASED FAN CONTROL

A Project

by

Nayankumar K. Patel

Approved by:

_____, Committee Chair
Jing Pang, Ph.D.

_____, Second Reader
Preetham Kumar, Ph.D.

Date

Student: Nayankumar K. Patel

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the Project.

_____, Graduate Coordinator
Preetham Kumar, Ph.D.

Date

Department of Electrical and Electronic Engineering

Abstract
of
TEMPERATURE BASED FAN CONTROL
by
Nayankumar K. Patel

Electric motors are used in many real time applications. Microcontroller controls these motors as programmed to perform in different environment. Microcontroller can store and execute program to control motor and can take feedback from the motor for better performance. The purpose of this project is to design motor controller using microcontroller with temperature sensor. The temperature sensor will sense the temperature and gives data to microcontroller. Microcontroller will control the speed of motor according to the temperature. When temperature reaches above certain value, the microcontroller will turn on the motor. Increase in temperature will result in increase of speed of motor. The expected functional project will store program in microcontroller and execute it in order to control the speed of motor according to data read from temperature sensor.

_____, Committee Chair
Jing Pang, Ph.D.

Date

ACKNOWLEDGMENTS

I would like to thank Dr. Jing Pang for providing me an opportunity to work on this project, which has been a great learning experience. I thank her for providing all necessary resources and help to finish this project. She always encouraged me to work hard and provided all help for the completion of this project. Without her guidance, help and support, this project would not have completed successfully. I really appreciate her help during different phases of project.

I would like to thank Dr. Preetham Kumar, Graduate Coordinator of Electrical and Electronic Engineering for his valuable guidance in project and for reviewing my report and providing valuable suggestions. I would like to thank Dr. Suresh Vadhva, Department Chair of Electrical and Electronic Engineering for his suggestions and support in project.

I would like to thank my family members for providing me inspiration, encouragement and strength throughout my master's program. I would like to thank my all friends especially Pratik Patel, Manan Shah, Dhrumil Jariwala, Chandrasekhar Chukka, Siddharth Shah, Jigar Bhatt, Hardik Shah and engineers' team at Nerdkits for providing all technical help to complete this project.

I am thankful to all faculty members of Electrical Engineering and Computer Science Department for helping me to finish my graduation at California State University, Sacramento.

TABLE OF CONTENTS

	Page
Acknowledgments.....	v
List of Tables	ix
List of Figures.....	x
Chapter	
1. INTRODUCTION	1
1.1 Introduction.....	1
1.2 Purpose of the Project.....	1
1.3 Significance of the Project.....	2
1.4 Organization of Report.....	2
2. ATMEGA168 MICROCONTROLLER ARCHITECTURE.....	4
2.1 Pin Configuration of ATmega168.....	4
2.2 Block Diagram of ATmega168.....	6
2.3 16-bit Timer/Counter1 with PWM.....	7
2.3.1 Output Compare Unit.....	8
2.3.2 Fast PWM Mode.....	9
2.3.3 Register Description.....	11
2.3.3.1 TCCR1A – Timer/Counter 1 Control Register A.....	11
2.3.3.2 TCCR1B – Timer/Counter Control Register B.....	11
2.4 Analog-to-Digital Converter.....	12

2.4.1	ADC Prescaler.....	13
2.4.2	ADMUX – ADC Multiplexer Selection Register.....	15
2.4.3	ADCSRA – ADC Control and Status Register A.....	16
3.	MOTOR CONTROL WITH MICROCONTROLLER.....	17
3.1	Temperature Sensor.....	17
3.2	Crystal Oscillator.....	18
3.3	Voltage Regulator L7805.....	18
3.4	LCD.....	19
3.5	MOSFET 2N7000.....	20
3.6	Brushless DC Motor (BLDC).....	21
3.7	Pulse Width Modulation (PWM).....	22
3.8	Variable Speed Control in BLDC Fan.....	24
4.	DESIGN OF FAN CONTROL SYSTEM.....	26
4.1	Configuring ADC in ATmega168.....	26
4.2	PWM Initialization.....	28
4.3	Implementation of Fan Control System.....	28
5.	MICROCONTROLLER PROGRAMMING.....	32
5.1	WinAVR.....	32
5.1.1	Installing WinAVR.....	32
5.1.2	Makefiles.....	32
5.1.3	Programmer’s Notepad.....	33
5.2	AVRDUDE.....	34

5.3 Programming and Results.....	36
5.3.1 Programming of ATmega168.....	36
5.3.2 Results.....	39
6. CONCLUSION AND FUTURE WORK.....	45
6.1 Conclusion.....	45
6.2 Future work.....	45
References.....	46

LIST OF TABLES

	Page
1. Table 4.1 Input Channel Selections for ADC.....	27

LIST OF FIGURES

	Page
1. Figure 2.1 Pin diagram of ATmega168.....	4
2. Figure 2.2 Block Diagram of ATmega168.....	6
3. Figure 2.3 16-bit Timer/Counter Block Diagram	7
4. Figure 2.4 Output Compare Unit	9
5. Figure 2.5 Timing Diagram: Fast PWM Mode.....	10
6. Figure 2.6 Timer/Counter 1 Control Register A.....	11
7. Figure 2.7 Timer/Counter Control Register B.....	11
8. Figure 2.8 Block Diagram of ADC ATmega168.....	12
9. Figure 2.9 ADC Prescaler.....	14
10. Figure 2.10 ADC Timing Diagrams.....	15
11. Figure 2.11 ADC Multiplexer Selection Register.....	15
12. Figure 2.12 ADC Control and Status Register A.....	16
13. Figure 3.1 Package LM34 and Basic Fahrenheit Temperature Sensor.....	18
14. Figure 3.2 Pin Configuration of L7805 (Top View).....	19
15. Figure 3.3 16-Pin LCD Module.....	19
16. Figure 3.4 4-bit Transfer between ATmega168 and LCD.....	20
17. Figure 3.5 Pin Description of 2N7000 MOSFET.....	21
18. Figure 3.6 BLDC Motor Transverse Section.....	21
19. Figure 3.7 BLDC Fan.....	22

20.	Figure 3.8 PWM Signals of Varying Duty Cycles.....	23
21.	Figure 3.9 BLDC Fan Speed Control with PWM.....	24
22.	Figure 4.1 Block Diagram of Fan Control System.....	27
23.	Figure 4.2 Circuit Diagram of Fan Control System.....	29
24.	Figure 4.3 Flow Chart Diagram of Fan Control System.....	30
25.	Figure 5.1 Makefile Example.....	33
26.	Figure 5.2 Programmer's Notepad.....	34
27.	Figure 5.3 Programming the ATmega168 with Command Prompt.....	37
28.	Figure 5.4 USB-to-Serial Communication Port 1.....	38
29.	Figure 5.5 Fan Control System.....	39
30.	Figure 5.6 Output Voltage on Multimeter 1.....	40
31.	Figure 5.7 Output Voltage on Multimeter 2.....	41
32.	Figure 5.8 Output Voltage on Multimeter 3.....	42
33.	Figure 5.9 Fan Control System with Temperature in Degree Celsius 1.....	43
34.	Figure 5.10 Fan Control System with Temperature in Degree Celsius 2.....	44

Chapter 1

INTRODUCTION

1.1 Introduction

A microcontroller is a general-purpose integrated circuit, which contains microprocessor, memory and input/output devices. Microcontrollers are used in microwaves, remote controls, cruise control system, digital cameras, cell phones, laser printers, answering machines, telephones, pagers, refrigerators, dishwashers and in other home appliances. Microcontrollers are programmable devices. Depending on application, they may contain different type of memory, flash memory that can be reprogrammed at user end or read-only memory which is programmed at company. Compiler compiles the code written in high-level language and converts it in machine-level language that can be written to microcontroller with the help of programmer. Microcontroller executes this program, takes input from the peripheral devices, makes some calculations and take decisions according to the algorithms written in code and gives data to output devices. Microcontrollers are classified in 8-bit, 16-bit or 32-bit microcontroller as per the data bits they handle in a single operation [1].

1.2 Purpose of the Project

The main purpose of this project is to design control system to control the speed of motor with change in temperature of the system. In this project, temperature sensor senses the lab temperature and generates analog output that is given to microcontroller. Microcontroller converts the data in digital using analog-to-digital converter and controls speed of motor that is actually a fan for this project. Microcontroller increases the fan

speed with rise in temperature. Using ATmega168 microcontroller control different peripherals attached to its I/O pins and display measured temperature on LCD (Liquid Crystal Display). Use WinAVR compiler to compile the code written in programming language C, and generate hex file, which is burned to microcontroller's flash memory with programmer.

1.3 Significance of the Project

Motor control with microcontroller is very interesting field of study. Motor control is used in many applications in industry. Particularly this type of approach is used in CPU (Central Processing Unit) fan of desktop and laptop computers for cooling purpose. For these applications, it is important to maintain minimum power consumption, so turning off the motor while not needed. A complete control system saves power by using temperature sensor to measure temperature and control the motor according to the temperature.

1.4 Organization of Report

Chapter two of the report explains architecture of ATmega168 microcontroller. It describes all pins and architectural block diagram of ATmega168. It also describes 16-bit timer/counter1 module with PWM and analog-to-digital converter and its registers.

Chapter three contains information about different components used for the project. It includes brief description of temperature sensor, crystal oscillator, voltage regulator, LCD, MOSFET, brushless DC motor and PWM.

Chapter four explains about how to configure ADC in ATmega168, PWM initialization, implementation of fan control system. It includes block diagram, circuit diagram and flow chart for this project.

Chapter five provides information on WinAVR compiler, makefiles and programmer's notepad. It includes explanation about command line program AVRDUDE. It contains programming of ATmega168 and results for this project.

Chapter six contains conclusion and future work.

Chapter 2

ATMEGA168 MICROCONTROLLER ARCHITECTURE

ATmega168 is an 8-bit reduced instruction set computing (RISC) microcontroller developed by Atmel Corporation. It has 32 8-bit general-purpose registers and 131 powerful instructions. ATmega168 has 16K bytes non-volatile self-programmable flash memory, 1K bytes of static random access memory (SRAM) and 512 bytes of electrically erasable programmable read-only memory (EEPROM). In its peripheral features, it has 6 pulse-width modulation (PWM) channels, 6-channel 10-bit analog-to-digital (ADC), programmable universal synchronous asynchronous receiver transmitter (USART) interface, master/slave serial peripheral interface (SPI) and 2-wire serial inter-integrated circuit (I2C) interface. In my project, I used 28-pin plastic dual in-line package (PDIP) type of ATmega168 [2].

2.1 Pin Configuration of ATmega168

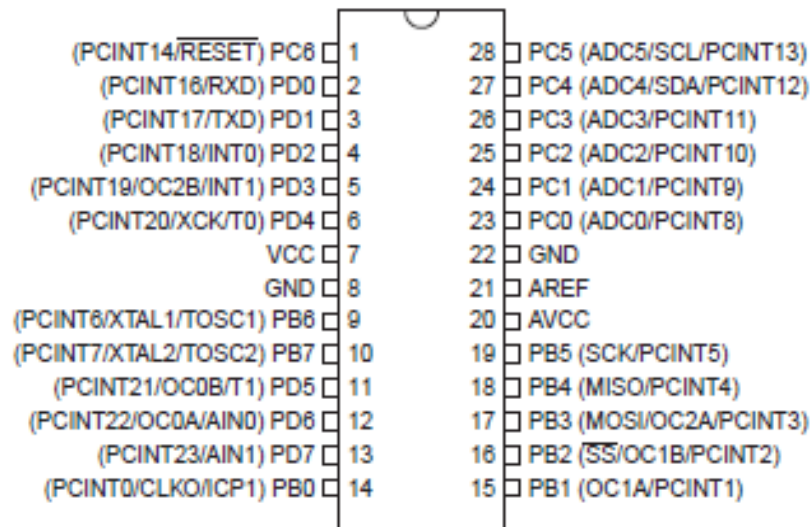


Figure 2.1 Pin diagram of ATmega168 [2]

VCC

VCC is digital supply voltage that is +5 V for this project.

GND

GND is ground pin of ATmega168.

Port B (PB7:0)

It is 8-bit bi-directional input-output port. All ports have multifunction and can be used according to application. PB6 is external clock input to ATmega168 or it can work as chip clock oscillator pin 1. PB7 is chip clock oscillator pin 2. PB0 is used as program/run mode switch in this project. PB2, PB3, PB4, PB5 are used for SPI bus and works as slave select (SS), master out slave in (MOSI), master in slave out (MISO) and serial clock (SCK) respectively. This port has internal pull-up registers [2].

Port C (PC5:0)

“Port C is 7-bit bi-directional I/O port with internal pull-up registers.” This all pins work as analog to digital converter input pins. For other applications PC4 and PC5 can be used for 2-wire serial bus (I2C). For this PC4 is I2C input/output data line and PC5 is I2C clock line [2].

PC6/RESET

PC6 pin is used as reset when RSTDISBL fuse is not programmed otherwise it works as I/O pin. This pin is active low reset [2].

Port D (PD7:0)

“Port D is an 8-bit bi-directional I/O port with internal pull-up resistors.” PD0 and PD1 are USART input, output pins respectively. PD2 to PD7 are used to display output on LCD for this project [2].

AVcc

This is supply voltage for A/D converter [2].

AREf

This is analog reference for ADC [2].

2.2 Block Diagram of ATmega168

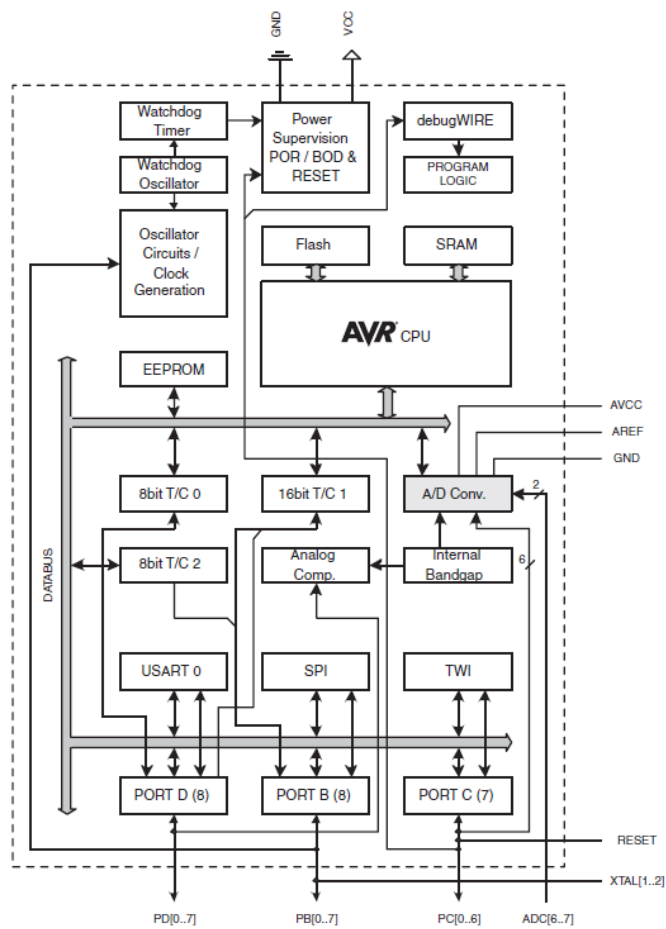


Figure 2.2 Block Diagram of ATmega168 [2]

ATmega168 microcontroller is really flexible to use in embedded control systems. This device can be programmed in two ways. In-system programming through serial peripheral interface (SPI) or programming via boot loader. This boot loader file can be downloaded on chip via any interface [2].

2.3 16-bit Timer/Counter1 with PWM

The 16-bit timer/counter unit generates pulse width modulation (PWM) output for motor control. A simplified block diagram of 16-bit timer/counter is shown in Figure 2.3. An “n” in block represents timer/counter number and “x” represents output compare unit channel [2].

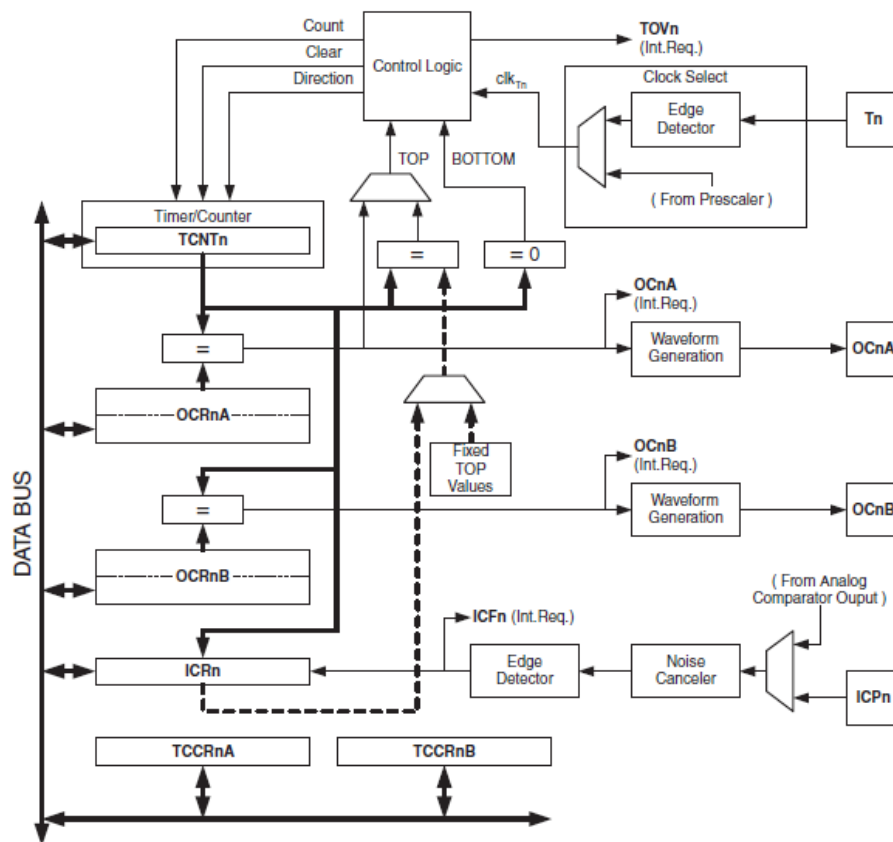


Figure 2.3 16-bit Timer/Counter Block Diagram [2]

This mode includes 16-bit register called Output Compare Register (OCR1A/B) requires special procedure while accessing. Timer/counter control registers are 8-bit. Timer/counter can be clocked externally via clock signal on T1 pin or by internal clock. The clock select bits (CS12:0) in TCCR1B (Timer/Counter Control Register B) defines which clock to use internal or external. OCR1A/B are compared with timer/counter and waveform generator and uses that result is used to generate PWM output on the OC1A/B (Output Compare Pin). The maximum timer/counter (TOP) value is assigned to OCR1A register, so this register can not be used for generating a PWM output for motor control. This maximum counter value can be 0x00FF, 0x01FF or 0x03FF depending on application. Byte addressable 16-bit register OCR1A/B can be accessed by AVR CPU via 8-bit data bus [2].

2.3.1 Output Compare Unit

The block diagram of output compare unit is shown in Figure 2.4 with waveform generator. The 16-bit comparator compares OCR1x (Output Compare Register) with TCN1 (Timer Counter 1) and if both are equal than generated match signal is given to waveform generator. It generates waveform depending on compare output mode bits (COM1x1:0), waveform generation mode bits (WGM13:0) bits and TOP and BOTTOM values assigned to Output Compare Register (OCR1x). The OCR1x register is double buffered register in PWM mode and TOP value assigned describes the period time for waveform generation. So, OCR1x is compared to top or bottom of the counting sequence to generate glitch free symmetrical PWM pulses. In double buffering, “the CPU has access to OCR1x buffer registers.” While reading, it reads low byte first and then high

byte. Writing is done by writing high byte (OCR1xH) first and then writing low byte second [2].

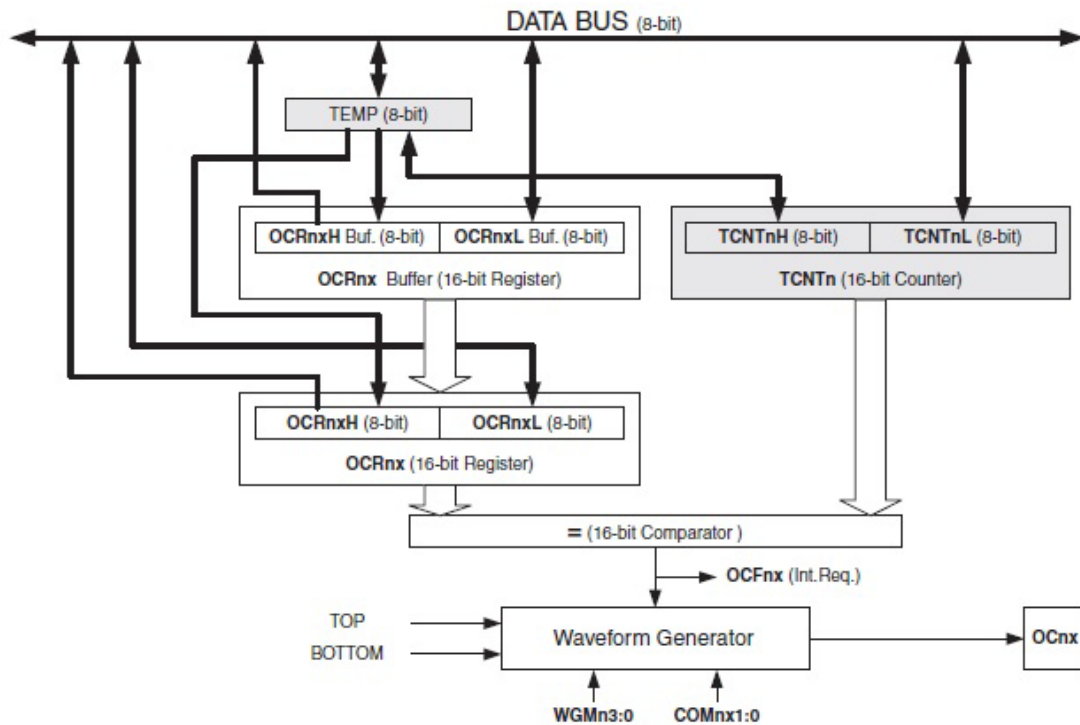


Figure 2.4 Output Compare Unit [2]

2.3.2 Fast PWM Mode

The fast pulse width modulation mode provides high frequency waveforms and has single-slope operation. This single slope operation allows PWM mode to be used with digital to analog converter (DAC) applications, rectification and power regulation. The counter TCNT1 counts from BOTTOM to TOP value assigned to OCR1x register [2].

The PWM resolution in bits is given by the equation (1),

$$R_{FPWM} = \frac{\log(TOP + 1)}{\log(2)} \dots\dots\dots(1)$$

This PWM resolution in bits can be given from 2-bit to 16-bit value assigned as TOP value to OCR1A. In this mode counter is incremented until it reaches the value assigned to OCR1A (WGM13:0=15). Then it is cleared in the following clock. Figure 2.5 shows fast PWM mode timing diagram [2].

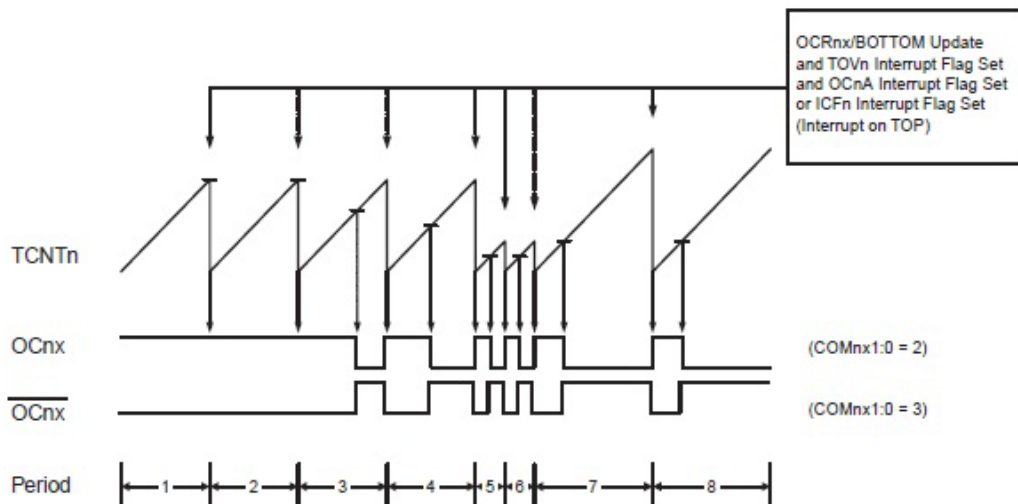


Figure 2.5 Timing Diagram: Fast PWM Mode [2]

OCR1A is used to define TOP in the timing diagram. TCN1 signal shows single-slope operation. The diagram shows marks on signal TCNT1 when OCR1x and TCNT1 are equal and shows change in PWM output on OCnx. This port will generate non-inverted PWM if COM1x1:0 bits are set to “2” or will generate inverted PWM if COM1x1:0 bits are set to “3”. The PWM frequency on OC1x can be calculated by the following equation [2]:

$$f_{OCnxPWM} = \frac{f_{clk IO}}{N \cdot (1 + TOP)} \dots\dots\dots(2)$$

Where N is a variable represents prescaler divider (1, 8, 64, 256 or 1024) [2]. By assigning TOP value to OCR1x will result in a constant high or low output [2].

2.3.3 Register Description

2.3.3.1 TCCR1A – Timer/Counter 1 Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.6 Timer/Counter 1 Control Register A [2]

Bit [7:6] – COM1A1:0: Compare Output Mode for Channel A

Bit [5:4] – COM1B1:0: Compare Output Mode for Channel B

For my project both COM1B1 bits are written to “1”, so it will set OC1A/OC1B on compare match. DDR (Data Direction Register) must be set to enable pin OC1A or OC1B as output [2].

Bit [1:0] – WGM11:0: Waveform Generation Mode

The combination of WGM13, WGM12, WGM11, and WGM10 from both TCCR1A and TCCR1B will define the counting sequence of counter, what kind of waveform generation used and TOP (maximum counter value). These all bits are set as “1” [2].

2.3.3.2 TCCR1B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.7 Timer/Counter Control Register B [2]

Bit [2:0] – CS12:0: Clock Select

These three bits will select the clock source used by timer/counter. CS/0 bit is set to “1”, so clock used with prescaler divider “1” for this project [2].

2.4 Analog-to-Digital Converter

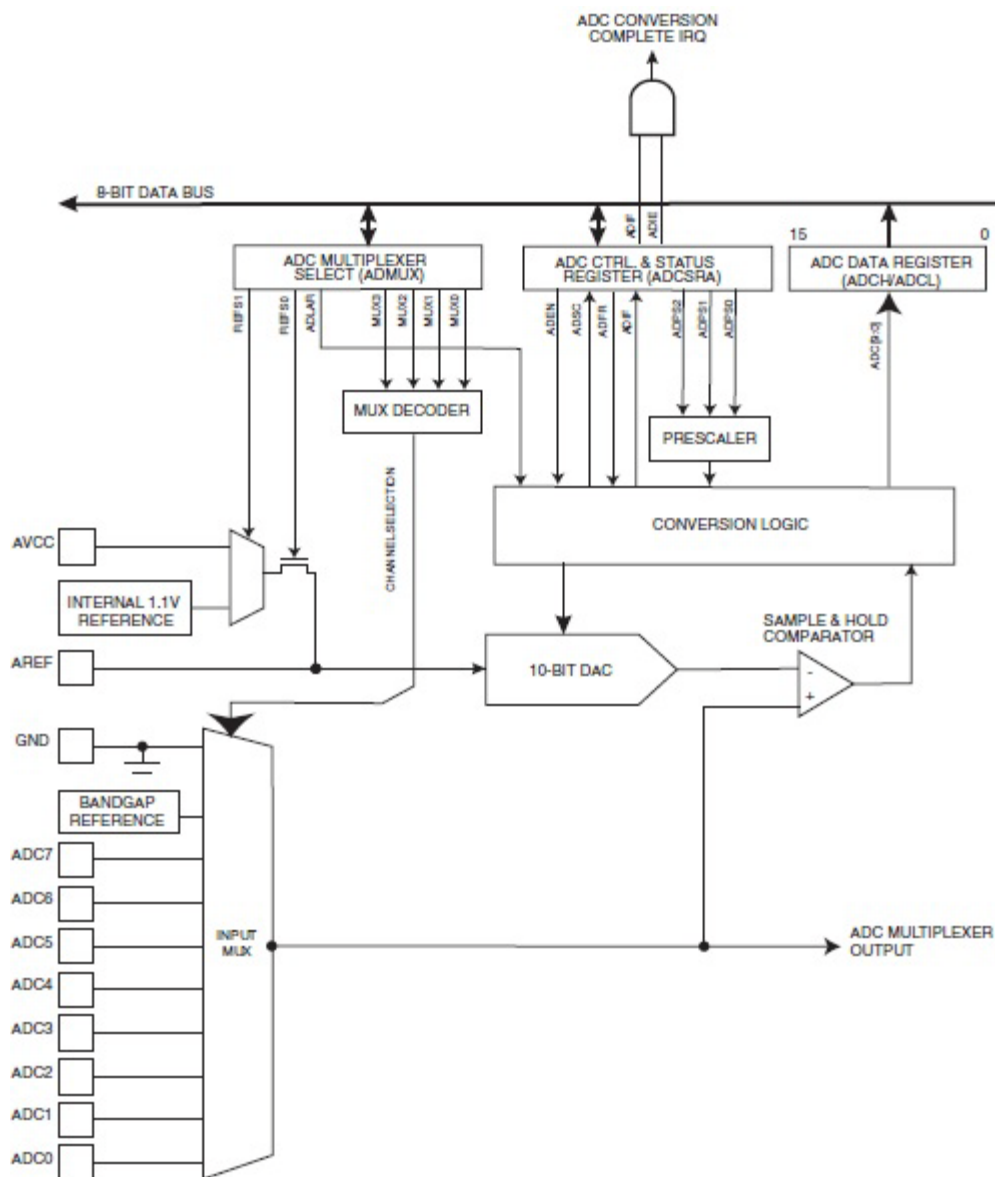


Figure 2.8 Block Diagram of ADC Atmega168 [2]

ATmega168 Analog-to-Digital Converter (ADC) uses 10-bit successive approximation method to convert 8-single ended voltage inputs from the pins of Port C via 8-channel analog multiplexer. Figure 2.8 shows block diagram of ADC [2].

The ADC takes analog input voltage on pins of Port C and converts it in 10-bit digital value, presented as ADCH, and ADCL in ADC data registers. “The minimum value represents GND and maximum value represents the voltage on the AREF pin minus 1 LSB”. The data in registers is right adjusted, so ADCL must be read first. The reference voltage for ADC can be selected as internal 1.1 V, Avcc or external AREF pin that describes the conversion range for the ADC. For this project AREF external reference voltage pin is used [2].

For single ended conversion, the result found in ADC registers (ADCL, ADCH) is described by following equation:

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}} \dots\dots\dots (3)$$

Where V_{IN} is voltage on the selected input pin, V_{REF} is selected reference voltage [2].

2.4.1 ADC Prescaler

For ADC, in order to get optimum resolution we require input frequency between 50 to 200 kHz. A prescaler takes CPU clock frequency as input and generates ADC clock. ADC prescaler is shown in Figure 2.9. Prescaler starts counting and keeps on run mode until ADEN bit is set in ADCSRA register [2].

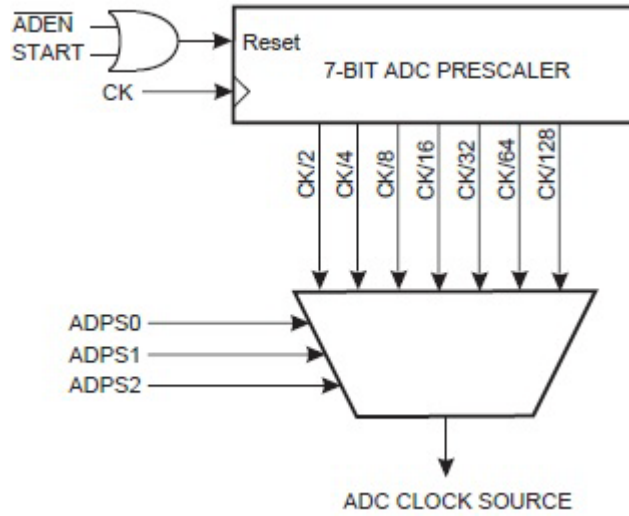
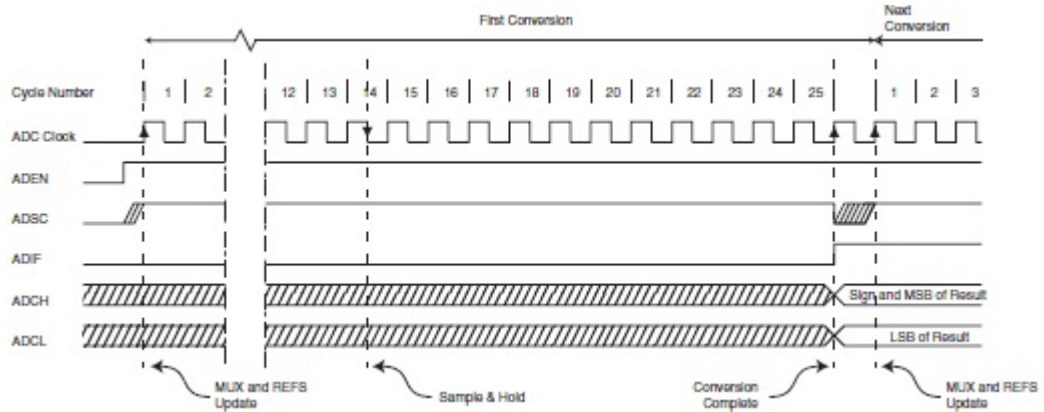
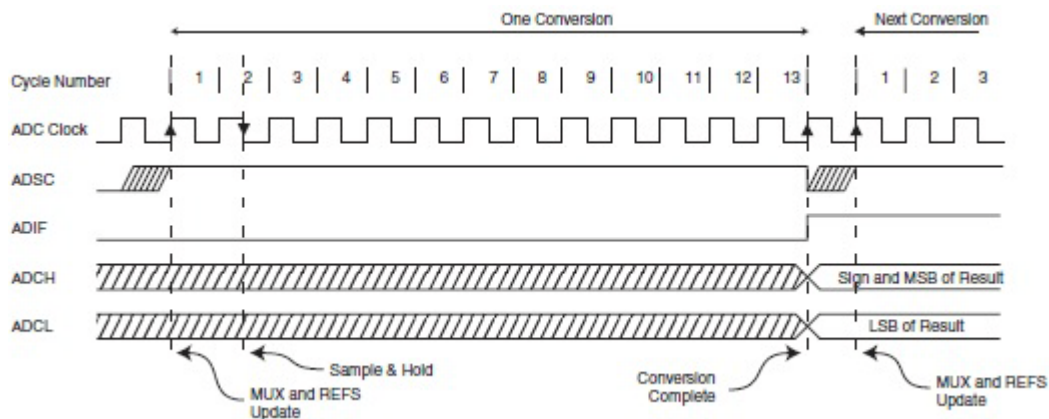


Figure 2.9 ADC Prescaler [2]

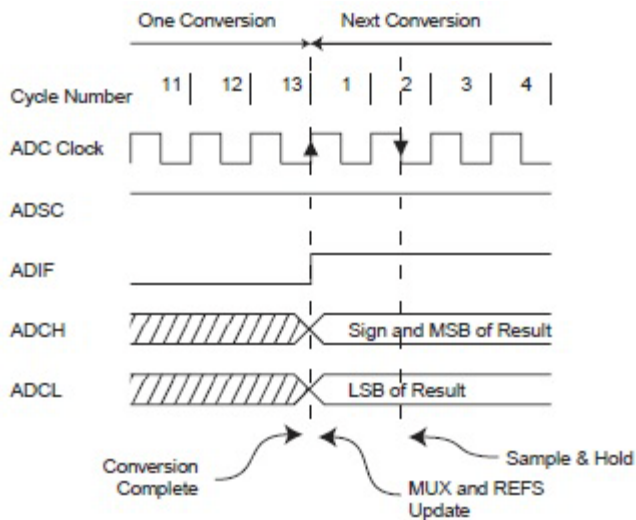
While ADEN is set, for the first conversion, it takes 25 ADC clock cycles and the rest of conversion takes 13 ADC clock cycles. Figure 2.10 shows timing diagram for first conversion, single conversion and free running conversion [2].



(a) For First Conversion



(b) For Single Conversion



(c) For Free Running Conversion

Figure 2.10 ADC Timing Diagrams [2]

2.4.2 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.11 ADC Multiplexer Selection Register [2]

Bit [7:6] – REFS1:0: Reference Selection Bits

These bits select voltage reference for ADC, for my project it is “00” as AREF, external reference voltage is used [2].

Bit 5 – ADLAR: ADC Left Adjust Result

Writing “1” to ADLAR bit adjusts result to left in ADC conversion result. By default, ADC data register conversion is right adjusted [2].

Bit 3:0 – MUX3:0: Analog Channel Selection Bits

For my project these bits are “0000” represents ADC0 is used as an analog input to ADC.

2.4.3 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN ADSC ADATE ADIF ADIE ADPS2 ADPS1 ADPS0								ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.12 ADC Control and Status Register A [2]

Bit 7 – ADEN: ADC Enable

Writing “1” to ADEN will enable ADC of the microcontroller.

Bit 6 – ADSC: ADC Start Conversion

Writing “1” to this bit in free running mode, starts the first conversion. Write “1” to this bit each time to start single conversion mode [2].

Bit 2:0 – ADPS2:0: ADC Prescaler Select Bit

For this project, these bits are “111”, so 128 division factor is selected for ADC prescaler.

Chapter 3

MOTOR CONTROL WITH MICROCONTROLLER

This project is based on cooling fan design of desktop computers and laptops. I have used following components for the implementation of this project:

1. Microcontroller ATmega168
2. Temperature sensor LM34
3. Crystal oscillator
4. Voltage regulator L7805
5. LCD
6. MOSFET 2N7000
7. Diode 1N4003
8. Brushless DC Motor (BLDC)
9. Programmer
10. Resistors, capacitors, breadboard and jumper wires

3.1 Temperature Sensor

Integrated circuit temperature sensors are classified in two categories: analog temperature sensors and digital temperature sensors. Analog temperature sensors are divided in two categories, depending on their output: voltage output temperature sensors and current output temperature sensors [1].

LM 34 is a precision Fahrenheit temperature sensor manufactured by National Semiconductor. Figure 3.1 shows plastic package LM34 and basic Fahrenheit temperature sensor [3].

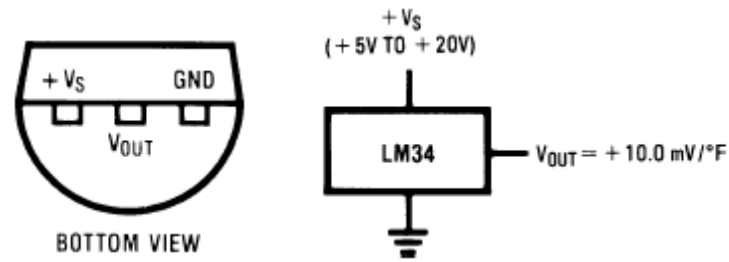


Figure 3.1 Package LM34 and Basic Fahrenheit Temperature Sensor [3]

The output voltage of LM34 is linearly proportional to the Fahrenheit temperature. The linear output is $+10.0 \text{ mV}/^\circ\text{F}$ scale factor. It draws less than $90 \mu\text{A}$ from supply and operates on 5 to 30 volts. This low cost device operates over the temperature range of -50° to $+300^\circ\text{F}$. It has low-impedance output and low self-heating less than 0.2°F in still air. The plastic package TO-92 is better for measuring air temperature than surface temperature [3] [1].

3.2 Crystal Oscillator

A crystal oscillator is an oscillator circuit that uses thin piece of quartz to generate electrical signal with precise frequency. Most of electrical circuits needs precise clock signal to execute instructions on time. I used crystal oscillator with 14.7465 MHz frequency for this project.

3.3 Voltage Regulator L7805

Voltage regulator gives constant regulated voltage to circuit. L7805 voltage regulator is used for to provide regulated output voltage $+5\text{V}$ to microcontroller, temperature sensor and LCD. L7805 comes with short circuit protection and thermal shutdown for reliable operation. Figure 3.2 shows pin configuration of L7805 [4].

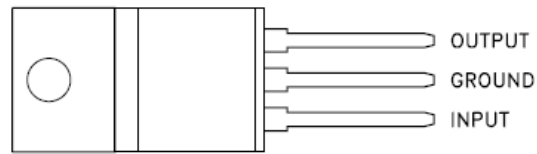


Figure 3.2 Pin Configuration of L7805 (Top View) [4]

3.4 LCD

Liquid Crystal Display (LCD) is used to display temperature of lab. I used 16-pin LCD module, which has HD44780 dot matrix LCD controller. Figure 3.3 shows 16-Pin LCD Module used in this project. It is 4 line 20 character wide LCD.

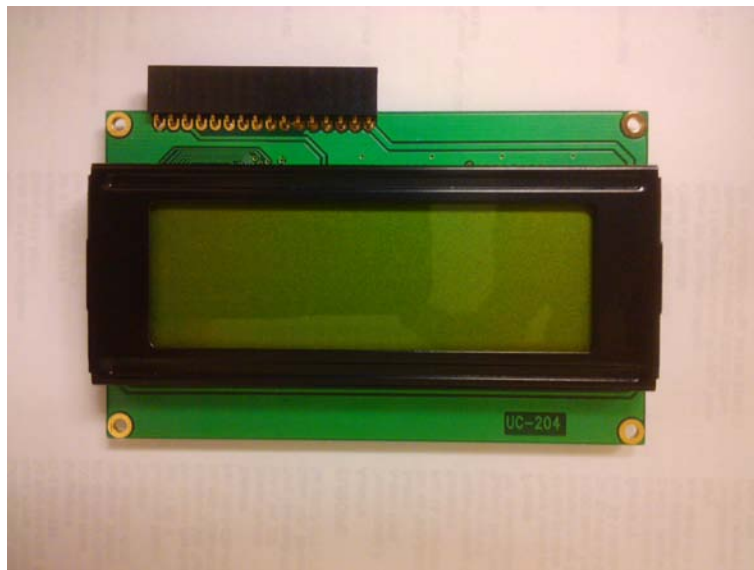


Figure 3.3 16-Pin LCD Module

It allows 4-bit or 8-bit microcontroller interface. DB [7:0], E, R/W and RS pins are used for communication with microcontroller ATmega168. Figure 3.4 shows 4-bit transfer between microcontroller and LCD [5].

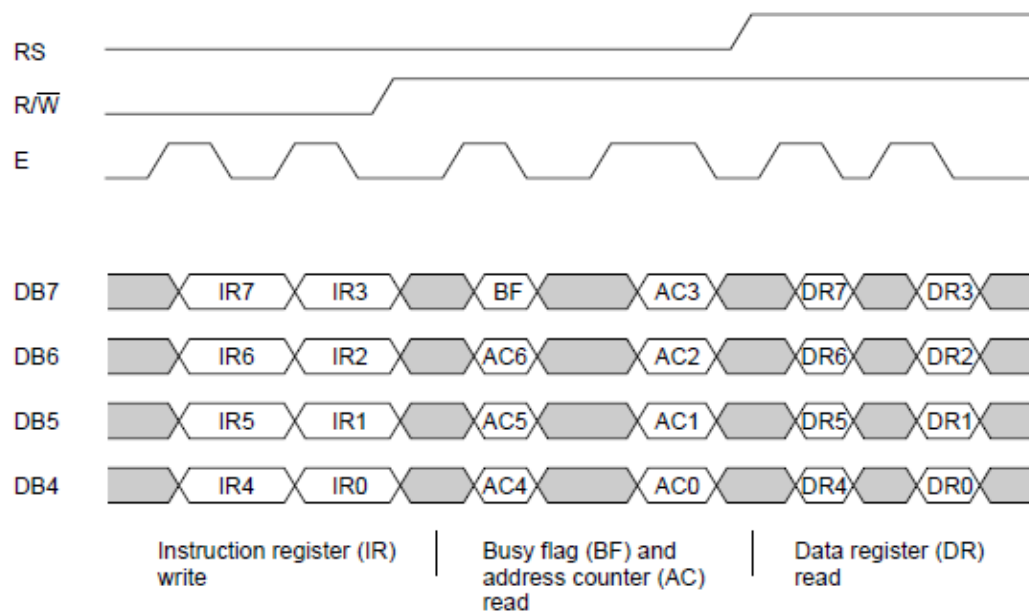


Figure 3.4 4-bit Transfer between ATmega168 and LCD [5]

Pin E tells LCD to start data read/write. Logic “0” on R/W is for write and “1” is for read. Logic “0” on RS pin selects instruction register for write and if flag is busy then it selects address counter for read. Logic “1” on RS pin selects data register for read and write. DB [7:4] is used for data transfer between ATmega168 and LCD. These bits should be transferred first for 8-bit operation then DB [3:0] transfer takes place. DB [3:0] used only for data transfer during 8-bit operation [5].

3.5 MOSFET 2N7000

2N7000 is an N-channel enhancement mode metal oxide semiconductor field effect transistor manufactured by Fairchild. 2N7000 is a reliable and rugged voltage controlled small signal switch, so well suited for project. This device has high saturation capability. Figure 3.5 shows pin description of 2N7000 [6].

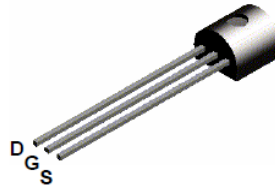


Figure 3.5 Pin Description of 2N7000 MOSFET [6]

3.6 Brushless DC Motor (BLDC)

BLDC electric motors are commutated electronically, they do not use brushes so called brushless DC motor. These motors provide better speed versus torque characteristics, noiseless operation and high efficiency over brushed DC motors and induction motors. Magnetic field generated by stator and rotor have same frequency, so BLDC motors are synchronous motors. Figure 3.6 shows transverse section of BLDC motor. BLDC motors have mainly two parts: stator and rotor. Some of the motors consist of hall sensors. BLDC fans do not have problems related to sparking, wearing of brushes or electromagnetic interference (EMI), as they use electrical commutation. Figure 3.7 shows BLDC fan used for the project [7] [8].

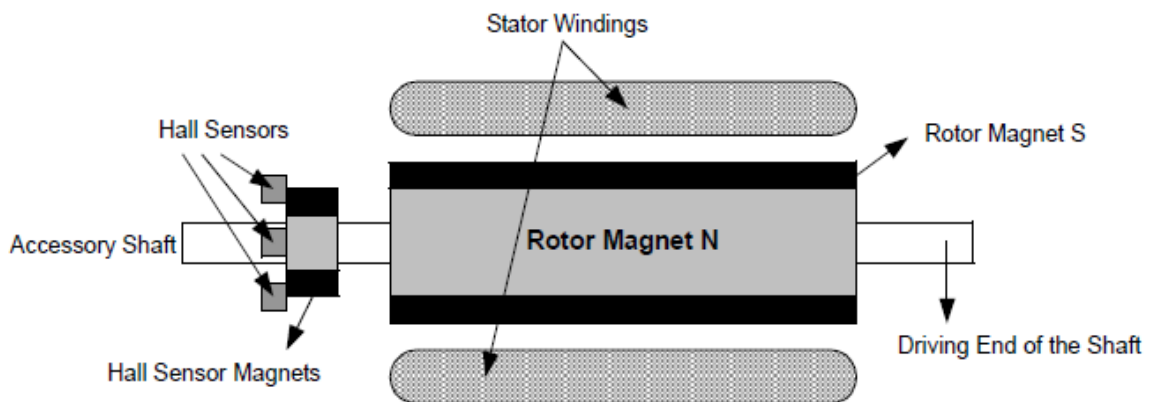


Figure 3.6 BLDC Motor Transverse Section [7]

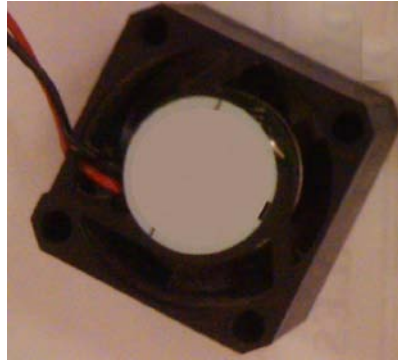


Figure 3.7 BLDC Fan

Stator is the stationary part of motor. Stator have coil, which is placed in the slots around the inner periphery. Stator coil windings are two types depending on the back electromotive force they produce: trapezoidal and sinusoidal [7].

Rotor is the part of motor that rotates. Rotor is made of permanent magnet with alternate North (N) and South (S) poles over the circular core. BLDC fan use permanent magnets, so it has lighter rotor than the conventional DC fan that makes them suitable for the cooling fan application in laptop and desktop computers [7] [8].

Hall sensors detect the rotor position by south and north poles and based on their position, exact commutation sequence is determined. This commutation sequence is important in rotating BLDC motors as they use electronically controlled commutation [7].

3.7 Pulse Width Modulation (PWM)

Pulse width modulation is most commonly used technique to control motor with digital output from ASIC (Application Specific Integrated Circuit) or MCU (Microcontroller Unit). “PWM is a way of digitally encoding analog signal levels”. At

any given time, the DC pulse logic level is either “1” or “0”, so PWM is digital signal. With the use of microcontroller ADC, we can generate perfect duty cycle square wave that completely describes the analog voltage output from the temperature sensor. Any analog value can be described in the series of on and off pulses and voltage or current source can drive motor by these series of pulses. PWM is noise immune because it is digital signal; usually noise is not strong enough to change the signal from logic “0” to logic “1”, or vice versa [9] [12].

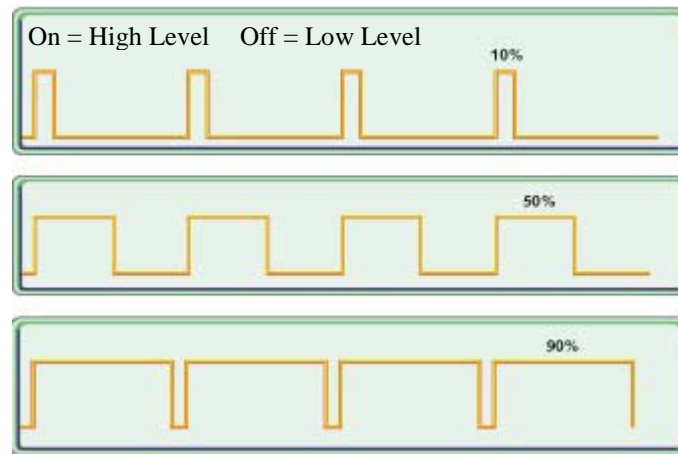


Figure 3.8 PWM Signals of Varying Duty Cycles [9]

Figure 3.8 shows PWM signals with varying duty cycles of 10%, 50% and 90%. The first signal shows 10%, means it is off for 90% of the signal period. Therefore, if I have 9 V supply voltage and 50% duty cycle signal, then analog signal of 4.5 V results [9].

Because of flywheel momentum and natural inertia, motor runs at steady speed with microcontroller if the switching frequency is high. The ratio of the on time to total time period is called duty cycle and time period is inverse of modulating frequency.

ATmega168 have on chip PWM unit, which controls the motor for this project. The on chip 16-bit timer/counter provides modulating square wave [9] [12].

3.8 Variable Speed Control in BLDC Fan

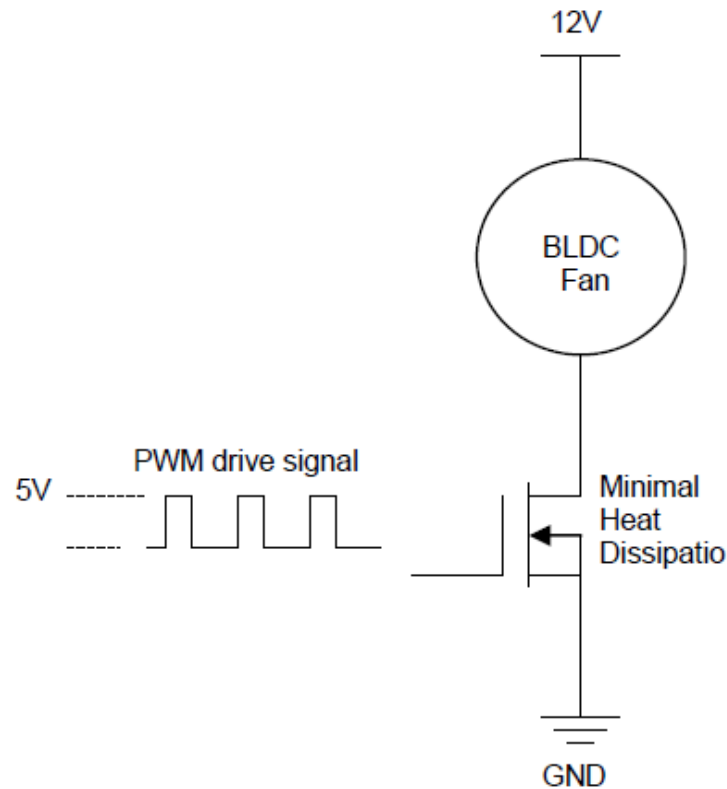


Figure 3.9 BLDC Fan Speed Control with PWM [8]

Figure 3.9 shows BLDC fan speed control using PWM. It shows FET (Field Effect Transistor) is connected in series with the BLDC fan to control supply voltage across it. PWM (Pulse Width Modulation) driving method is used to control that voltage. In this method PWM signal is applied to FET and FET controls switching of BLDC fan. “The speed of the fan is directly proportional with duty cycle of the PWM”. The PWM

frequency is important in controlling motor, as high frequency may malfunction the commutation circuit and low may cause the fan oscillate [8] [11].

Chapter 4

DESIGN OF FAN CONTROL SYSTEM

This chapter describes how the speed of fan is controlled by PWM output from microcontroller with the change in lab temperature. Desktop computers and laptops use CPU fan for the cooling purpose for the microprocessor. Without CPU fan microprocessor may overheat and may cause the computer system to fail or malfunction. CPU fans are used in order to maintain certain temperature level in the computer system to provide optimum operating range of temperature for all components to work properly. To minimize power consumption, these fans work on particular algorithms to run at variable speed. If the temperature is too high, then it runs at maximum speed. Temperature sensor senses the temperature and speed of fan increases with the rise in temperature. Therefore, in this way fan does not run on maximum speed unless required to save power [10].

Figure 4.1 shows block diagram of the fan control system. Block diagram shows microcontroller ATmega168 takes analog input from temperature sensor, gives output on LCD display unit, and controls motor.

4.1 Configuring ADC in ATmega168

Temperature sensor senses the room temperature and generates analog output voltage, which is proportional to the temperature sensed. This voltage can be given to any of ADC pin from ADC0 to ADC5 (PC0-PC5 of ATmega168). ADMUX register value defines which ADC pin is used as input. Table 4.1 shows ADC input channel selection by MUX [3:0] bits [2].

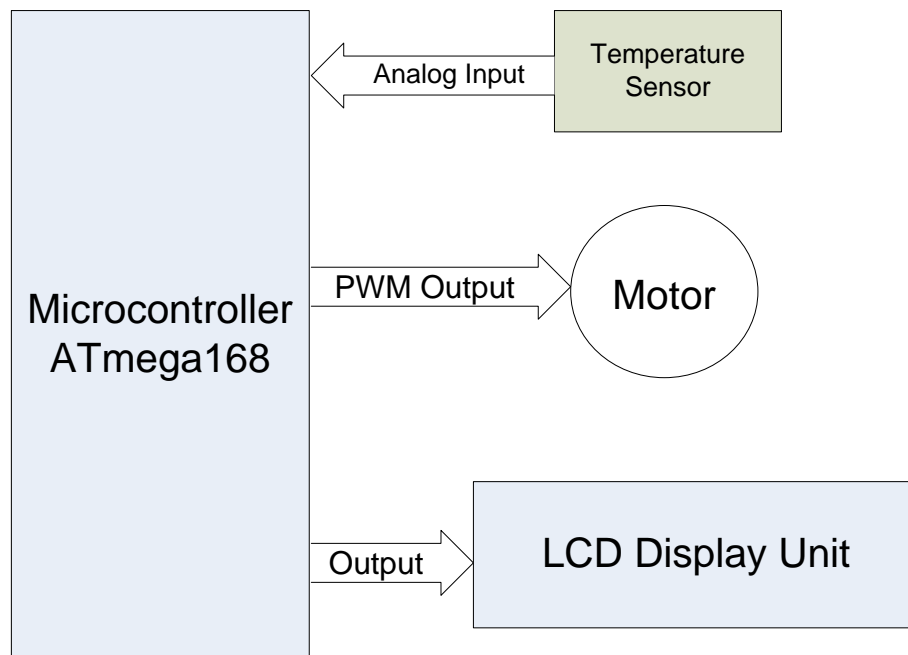


Figure 4.1 Block Diagram of Fan Control System

MUX [3:0]	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5

Table 4.1 Input Channel Selections for ADC [2]

ADCSRA is an 8-bit ADC control and status register, which controls ADC enable (ADCEN), ADC start conversion bit (ADSC) and ADC prescaler bit (ADPS2:0) for the division factor bits. These all bits are set in ADCSRA register for this project. Both ADMUX and ADCSRA values are written to microcontroller with a programming code in language C [2].

4.2 PWM Initialization

This includes enabling PWM controller, configuring 16-bit timer 1 mode and configuring the direction of PWM output. Timer/counter control register A (TCCR1A), Timer/counter control register B (TCCR1B), output compare register A (OCR1A) are configured in C code to make PWM mode work with ATmega168 microcontroller for fan control application [2] [9].

4.3 Implementation of Fan Control System

Figure 4.2 shows circuit diagram of fan control system. Crystal oscillator is connected in between pin 9 (PB6) and pin 10 (PB7) of ATmega168, those are pins if we want to provide external clock to the microcontroller. I used 0.1 μ F bypass capacitor on the output pin +5 V of the voltage regulator L7805 to smooth out the supply voltage to microcontroller and LCD. Vout pin of temperature sensor LM34 is connected on pin PC0 which is ADC0 of all ADC input pins. Pin 3 of LCD is connected to ground via 1Kohm resistor to set the contrast of the LCD to display temperature on LCD. Pins from PD2 to PD7 are connected to remaining LCD pins used for data and control signals between LCD and ATmega168 microcontroller [12] [11].

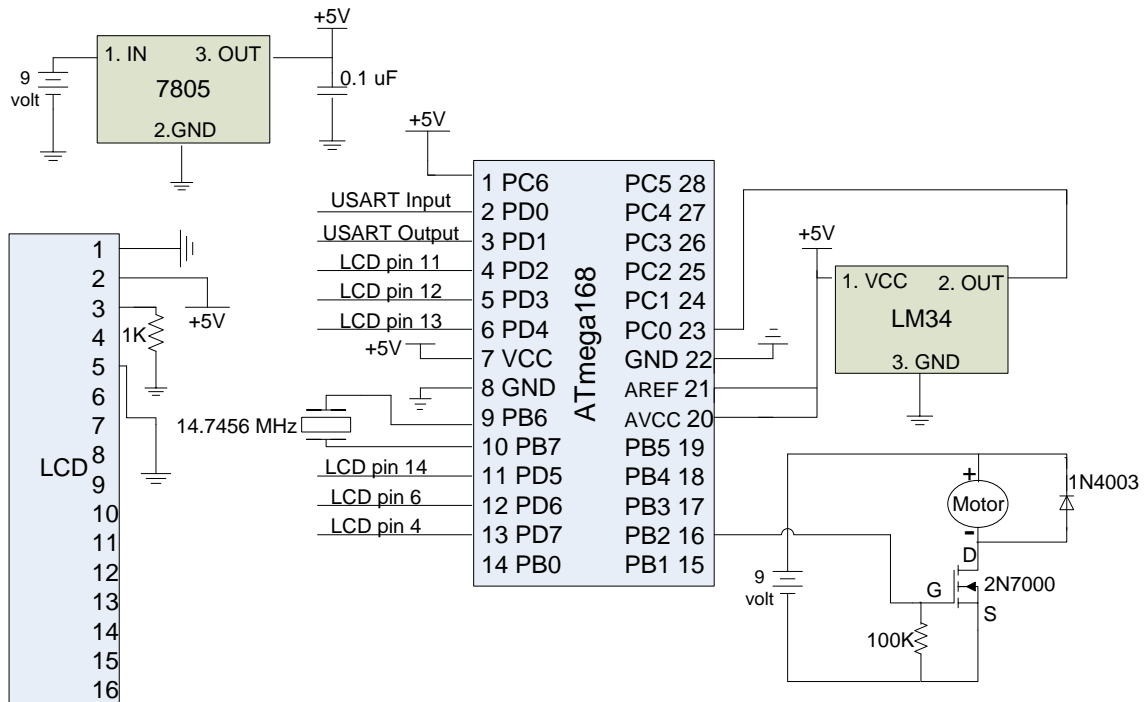


Figure 4.2 Circuit Diagram of Fan Control System [12]

PWM output is given to gate terminal of N-channel MOSFET 2N7000 from microcontroller. MOSFET (Metal Oxide Semiconductor Field Effect Transistor) is high switching speed power switch. This MOSFET switches on and off at PWM frequency and controls the voltage across motor. As shown in diagram positive terminal of motor is connected to +9V and negative terminal is connected to drain of MOSFET. When MOSFET is on, the motor starts to gain speed and off then motor loses speed. 100Kohm resistor is connected from gate of MOSFET to ground to pull the gate to ground. When ATmega168 is not driving the motor, this pull down resistor separates motor from microcontroller and other circuit. Fly back diode 1N4003 is connected in parallel with motor as shown in the circuit diagram. When the MOSFET is off, due to inductive coil of

motor there may be huge voltage that can produce spark and may damage other components. Therefore, fly back diode plays an important role in protecting other circuit [11] [12].

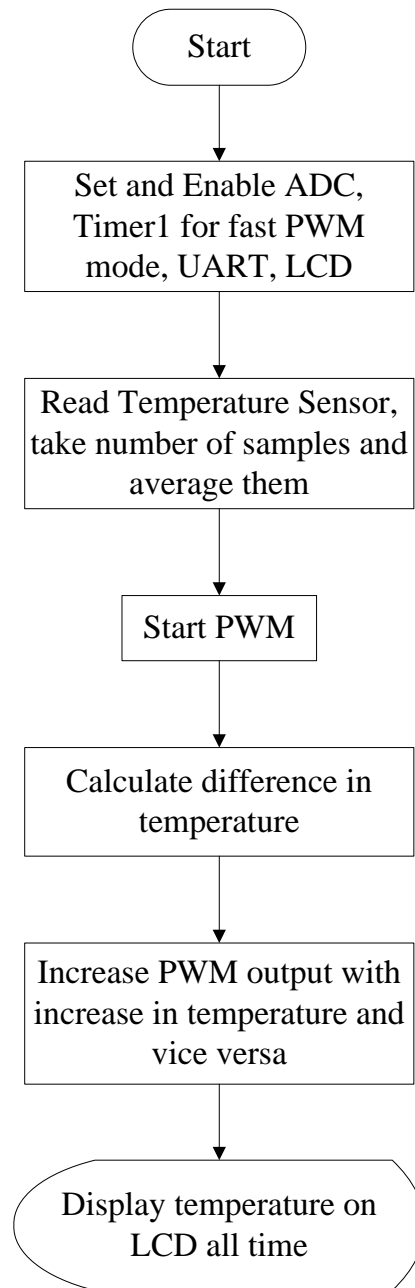


Figure 4.3 Flow Chart Diagram of Fan Control System

Microcontroller continuously monitors the temperature and controls speed of motor by changing the duty cycle of PWM. The speed of motor fan is controlled by the duty cycle of PWM that drives two winding coils of motor fan alternately. Depending on the factors like acoustic noise, controllable speed range and efficiency, the PWM frequency can be chosen in between 18 KHz to 60 KHz. 18 KHz is higher than the audible range of human ear, so no acoustic noise will occur. Higher efficiency can be obtained by choosing high frequency for PWM. Figure 4.3 shows flow chart of fan control system. When microcontroller executes program it sets and enables ADC, Timer 1 for fast PWM, LCD and UART. Microcontroller continuously reads temperature and averages them after sampling. It calculates difference in temperature and by considering this difference, it will increase PWM output, which is given to motor and controls speed of motor fan. Microcontroller continuously displays temperature of lab on LCD [8] [11] [12].

Chapter 5

MICROCONTROLLER PROGRAMMING

I used general-purpose programming language “C” for the implementation of this project. One main advantage of C over assembly language is that you can easily switch between microcontrollers. Assembly language is specific for particularly that microcontroller, so if you happen to change microcontroller then you have to start the code from beginning. I used WinAVR compiler for this project to compile the project code written in language C.

5.1 WinAVR

WinAVR is not just one tool, it contains many tools like avr-gcc (compiler), avr-gdb (debugger), avrdude (the programming interface), avr-libc (library) and many more. “WinAVR (pronounced “whenever”) is a suite of executable, open source software development tools for the Atmel AVR series of RISC microcontrollers hosted on the Windows platform. It includes the GNU GCC compiler for C and C++” [13] [14].

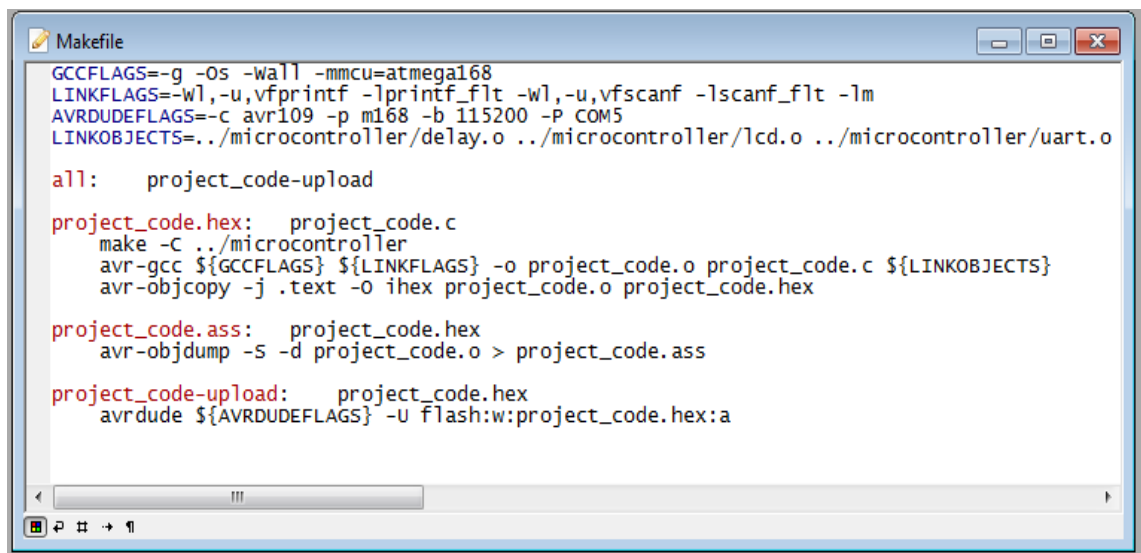
5.1.1 Installing WinAVR

WinAVR is an open source development tools that can be downloaded free from <http://winavr.sourceforge.net/>. After downloading it, run the file for installation. It will download to C:\WinAVR as default. In this folder there is a text file called WinAVR-user-manual, it contains information regarding how to use [13] [14].

5.1.2 Makefiles

A makefile is a text file, which controls flow of the program make. Programmer writes make file, which includes lists and controls about how the design is going to build.

Makefile is very powerful file, which usually have information regarding type of microcontroller, connection of the programming device, type of programming device, source coding files and degree of optimization. Makefile tells compiler what files to compile and link them if there are more, what commands to run and what output file to create [15] [14].



```

GCCFLAGS=-g -Os -Wall -mmcu=atmega168
LINKFLAGS=-Wl,-u,vfprintf -lprintf_flt -Wl,-u,vfscanf -lscanf_flt -lm
AVRDUDEFLAGS=-c avr109 -p m168 -b 115200 -P COM5
LINKOBJECTS=./microcontroller/delay.o ../microcontroller/lcd.o ../microcontroller/uart.o

all:    project_code-upload

project_code.hex:  project_code.c
    make -C ../microcontroller
    avr-gcc ${GCCFLAGS} ${LINKFLAGS} -o project_code.o project_code.c ${LINKOBJECTS}
    avr-objcopy -j .text -O ihex project_code.o project_code.hex

project_code.ass:  project_code.hex
    avr-objdump -S -d project_code.o > project_code.ass

project_code-upload:  project_code.hex
    avrdude ${AVRDUDEFLAGS} -U flash:w:project_code.hex:a

```

Figure 5.1 Makefile Example

Figure 5.1 shows an example of makefile. It shows “-mmcu=atmega168” in the first line, type of microcontroller used. Communication port and programmer are defined in third line. Other files required for the compilation with the code are linked in 4th line from different directory. Last line tells to download hex file in microcontroller flash.

5.1.3 Programmer’s Notepad

Programmer’s notepad is an open source text editor comes with WinAVR. It has many features and supports many programming languages. For my project, I used it for

writing code in C in its text editor. After writing, you can compile code with the compiler WinAVR.

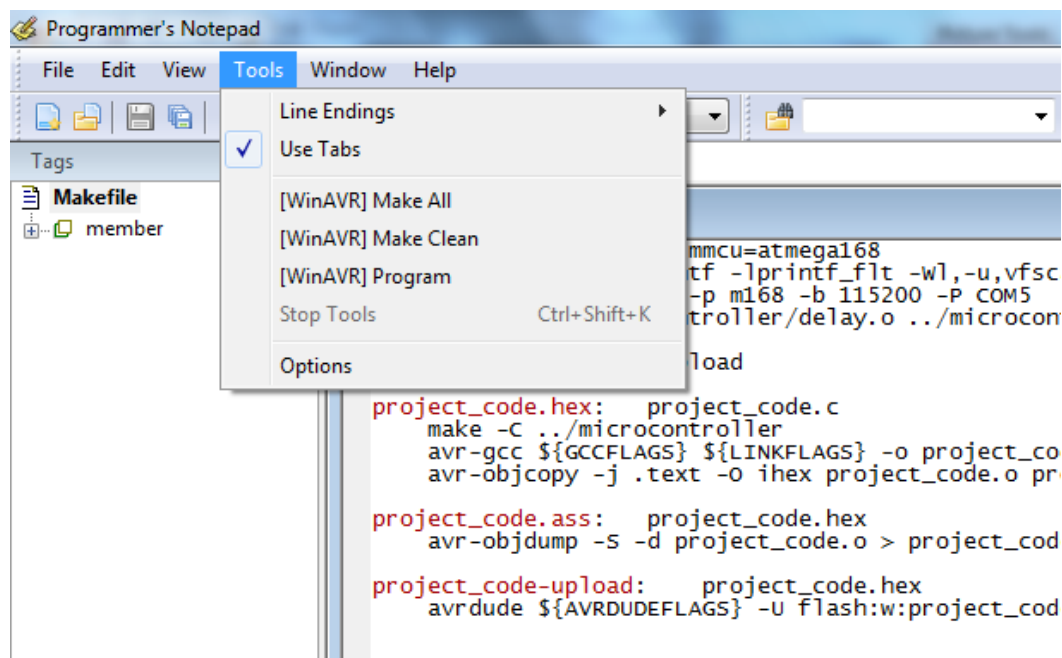


Figure 5.2 Programmer's Notepad

Figure 5.2 shows Programmer's Notepad. For compile purpose, "Make All" generates hex file following the information from makefile. "Make Clean" deletes all temporary files including hex file. "Program" transfers hex file to the AVR microcontroller.

5.2 AVRDUDE

AVRDUDE (AVR Downloader UploADER) is a command line program for programming the flash and EEPROM of Atmel's AVR series of microcontrollers, supported by serial programming protocol using the in-system programming technique (ISP). With AVRDUDE, users can use an interactive terminal mode or command line to

write to fuse bits, lock bits, flash, EEPROM or signature bytes of AVR microcontroller. This program was written by Brian S. Dean as a private project of an in-system programmer for AVR microcontroller [16].

AVRDUDE supports many programmers like STK500, STK600, avr910, avr109 and PPI (Parallel Port Interface) programmers too. AVRDUDE can be configured to work with any of these programmers by entering its specifications directly on command line or by editing configuration file [16].

AVRDUDE options, which should be considered while programming, are as follows:

`-p partno`

This option is mandatory to write in makefile, that tells AVRDUDE program which AVR microcontroller is connected to the programmer. In Figure 5.1, third line shows how AVRDUDE is configured, “-p m168” tells programmer that ATmega168 is connected to programmer [16].

`-b baudrate`

That specifies RS-232 connection baud rate with respect to programmer used. “-b 115200” in Figure 5.1 shows 115200 bps (bits per second) baud rate is specified for the serial interface [16].

`-c programmer-id`

Figure 5.1 shows “-c avr109” specifies avr109 programmer to be used. I also used STK500 to implement this project, so in that case it should be replaced by “-c stk500”.

`-P port`

Specifies on which port the programmer is connected. Figure 5.1 shows “-P COM5”, so programmer is connected on laptop’s COM5 port.

`-u`

It will disable the reading out of fuses before programming and verifies they have not been changed [16].

`-U memtype:OP:filename[:format]`

This is important to specify while you are doing any operation on AVR microcontroller with AVRDUDE program. This specifies memory type (flash, eeprom, fuse, etc.). OP specifies operation type to be performed, read, write or verify by “r”, “w”, “v” respectively. Figure 5.1 last line shows “-U flash:w:project_code.hex:a” specifies project_code.hex file is going to be written to the flash of ATmega168. “:a” is an optional format field [:format] of the file to be read or written, specifies auto detect, so AVRDUDE will detect the file format automatically from that directory [16].

5.3 Programming and Results

5.3.1 Programming of ATmega168

For this project, I programmed ATmega168 with programmer STK500 and avr109. I have made makefile in the same directory where I put the project code for fan control system. Makefile includes all description like how the program should be compiled with linking other files and at the end which file should be generated. I put header files for LCD and UART in other directory and put both of these folders under one directory. Figure 5.3 shows how to call the makefile to program the ATmega168 by writing, “make” on the command prompt.

```

C:\Windows\system32\cmd.exe

avrdude done. Thank you.

C:\Users\Nayan Patel\Desktop\nayan\project>make
make -C ../microcontroller
make[1]: Entering directory `C:/Users/Nayan Patel/Desktop/nayan/microcontroller'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `C:/Users/Nayan Patel/Desktop/nayan/microcontroller'
avr-gcc -g -Os -Wall -mmcu=atmega168 -Wl,-u,vfprintf -lprintf_flt -Wl,-u,vfscanf -lscanf_flt -lm -o project_code.o project_code.c ../microcontroller/delay.o ../microcontroller/lcd.o ../microcontroller/uart.o
avr-objcopy -j .text -O ihex project_code.o project_code.hex
avrdude -c avr109 -p m168 -b 115200 -P COM1 -U flash:w:project_code.hex:a

Connecting to programmer: .
Found programmer: Id = "FDL v02"; type = S
Software Version = 0.2; No Hardware Version given.
Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
Device code: 0x35

avrdude: AVR device initialized and ready to accept instructions

Reading : ##### : 100% 0.01s

avrdude: Device signature = 0x1e9406
avrdude: NOTE: FLASH memory has been specified, an erase cycle will be performed

To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "project_code.hex"
avrdude: input file project_code.hex auto detected as Intel Hex
avrdude: writing flash (11138 bytes):

Writing : ##### : 100% 2.39s

avrdude: 11138 bytes of flash written
avrdude: verifying flash memory against project_code.hex:
avrdude: load data flash data from input file project_code.hex:
avrdude: input file project_code.hex auto detected as Intel Hex
avrdude: input file project_code.hex contains 11138 bytes
avrdude: reading on-chip flash data:

Reading : ##### : 100% 1.98s

avrdude: verifying ...
avrdude: 11138 bytes of flash verified

avrdude done. Thank you.

```

Figure 5.3 Programming the ATmega168 with Command Prompt

Figure 5.4 shows the USB-to-serial communication port “COM1” used to connect to programmer and I described this communication port in makefile to tell the

AVRDUDE program to which communication port the ATmega168 is connected through the programmer.

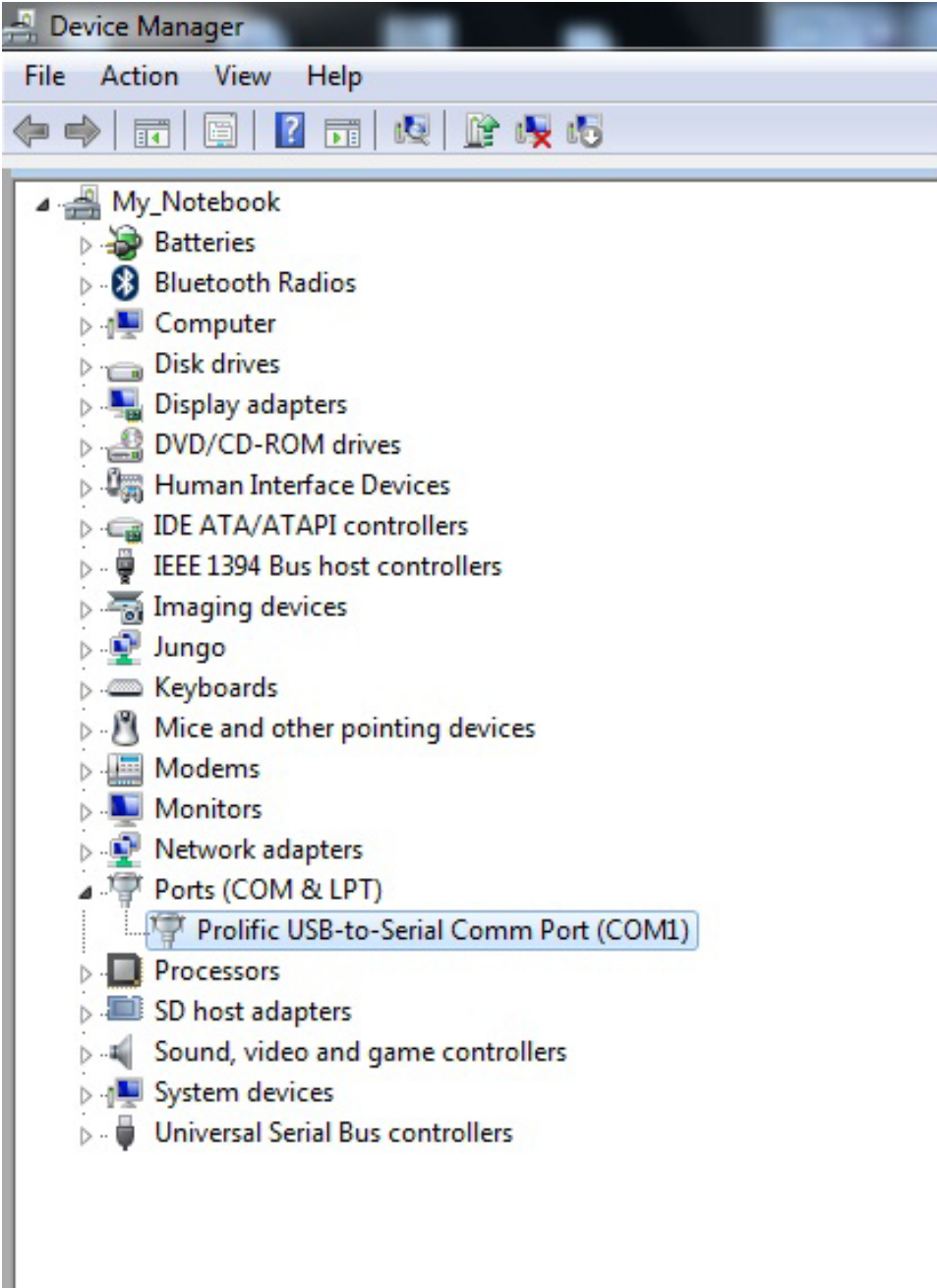


Figure 5.4 USB-to-Serial Communication Port 1

5.3.2 Results

The following images describe results for this project. Figure 5.5 shows fan control system, temperature sensor senses the lab temperature and controls motor fan, temperature in degree Fahrenheit and PWM output is shown on LCD.

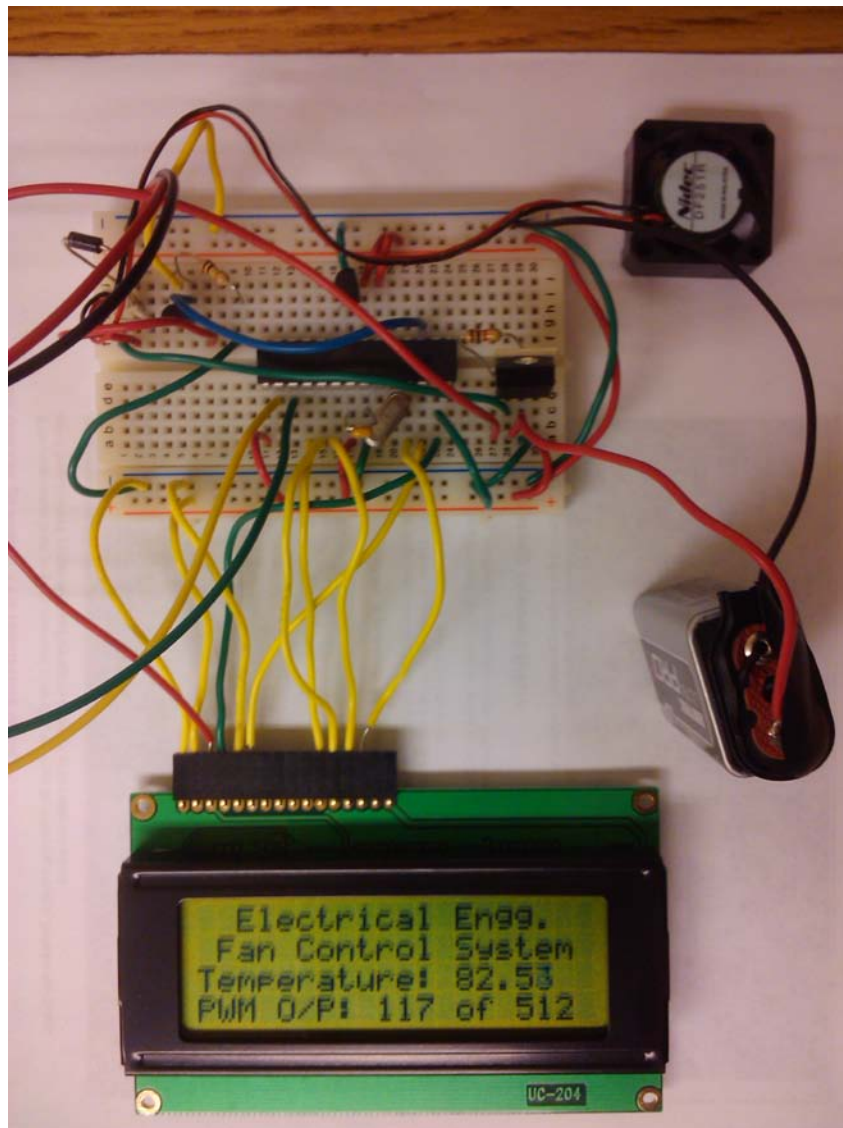


Figure 5.5 Fan Control System

Figure 5.6, Figure 5.7 and Figure 5.8 shows variation in output voltage of temperature sensor in volts on multimeter with respect to lab temperature and LCD shows temperature in degree Fahrenheit and PWM output.

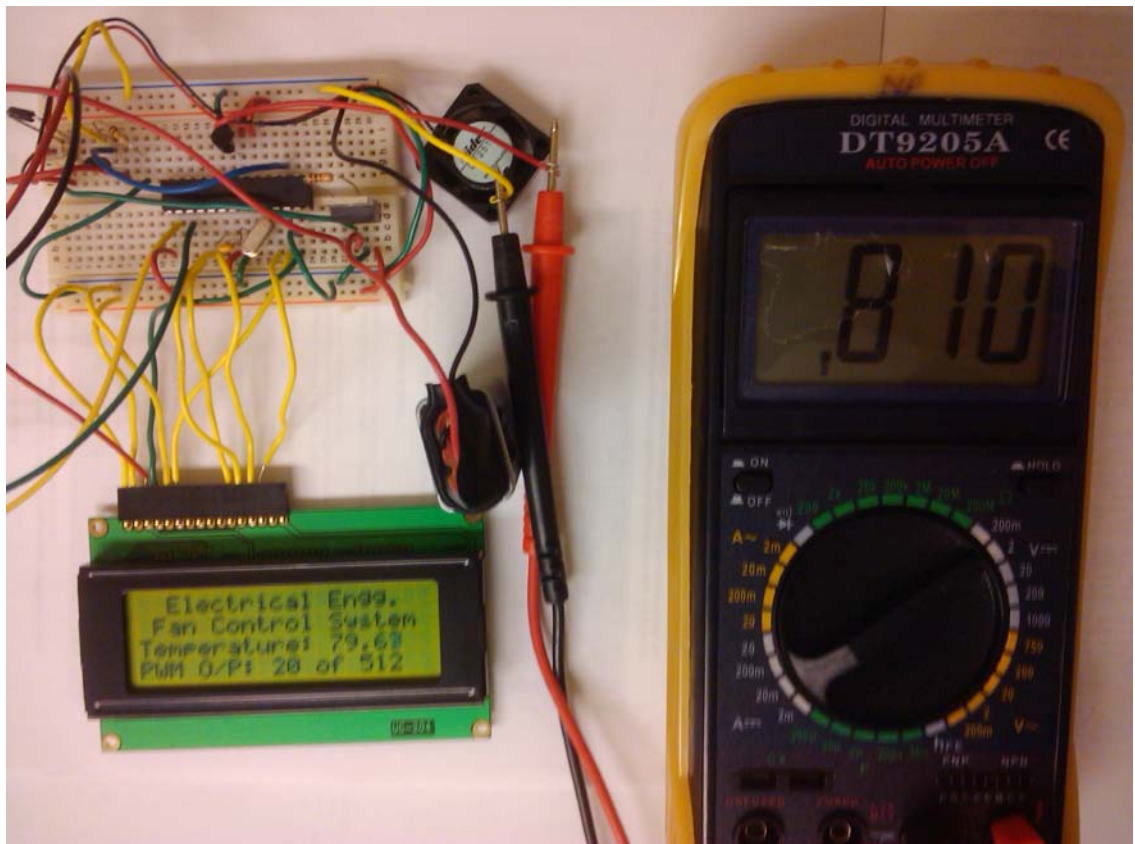


Figure 5.6 Output Voltage on Multimeter 1

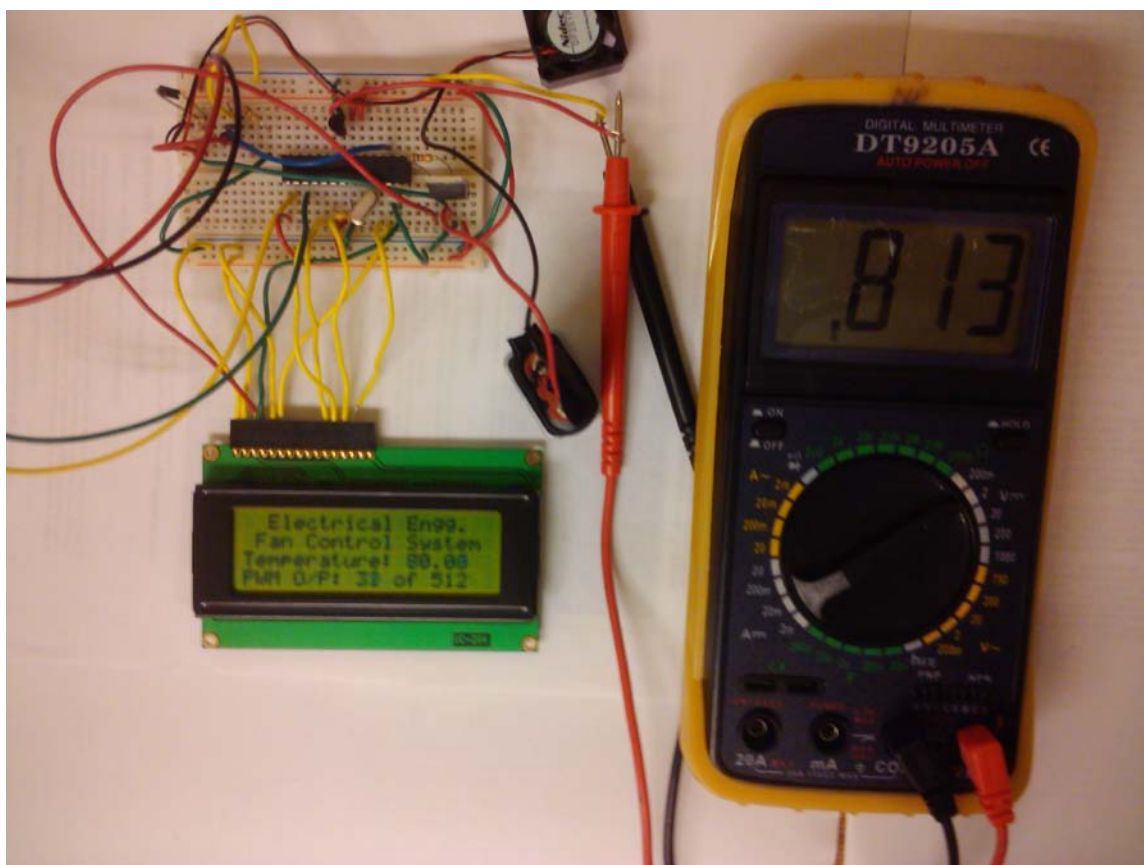


Figure 5.7 Output Voltage on Multimeter 2

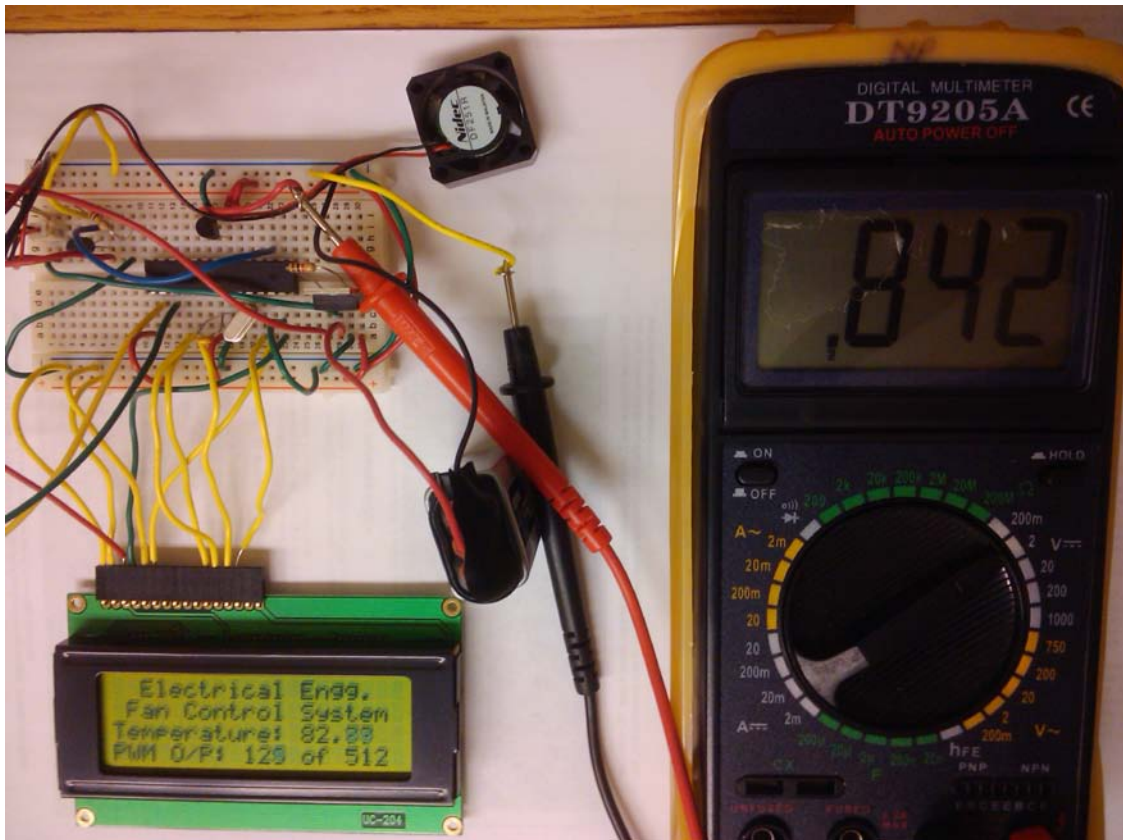


Figure 5.8 Output Voltage on Multimeter 3

Output voltage of temperature sensor changes with temperature and this analog voltage value is given to ADC of microcontroller, microcontroller changes PWM duty cycle output with respect to this voltage differences and gives it to MOSFET to control the motor fan.

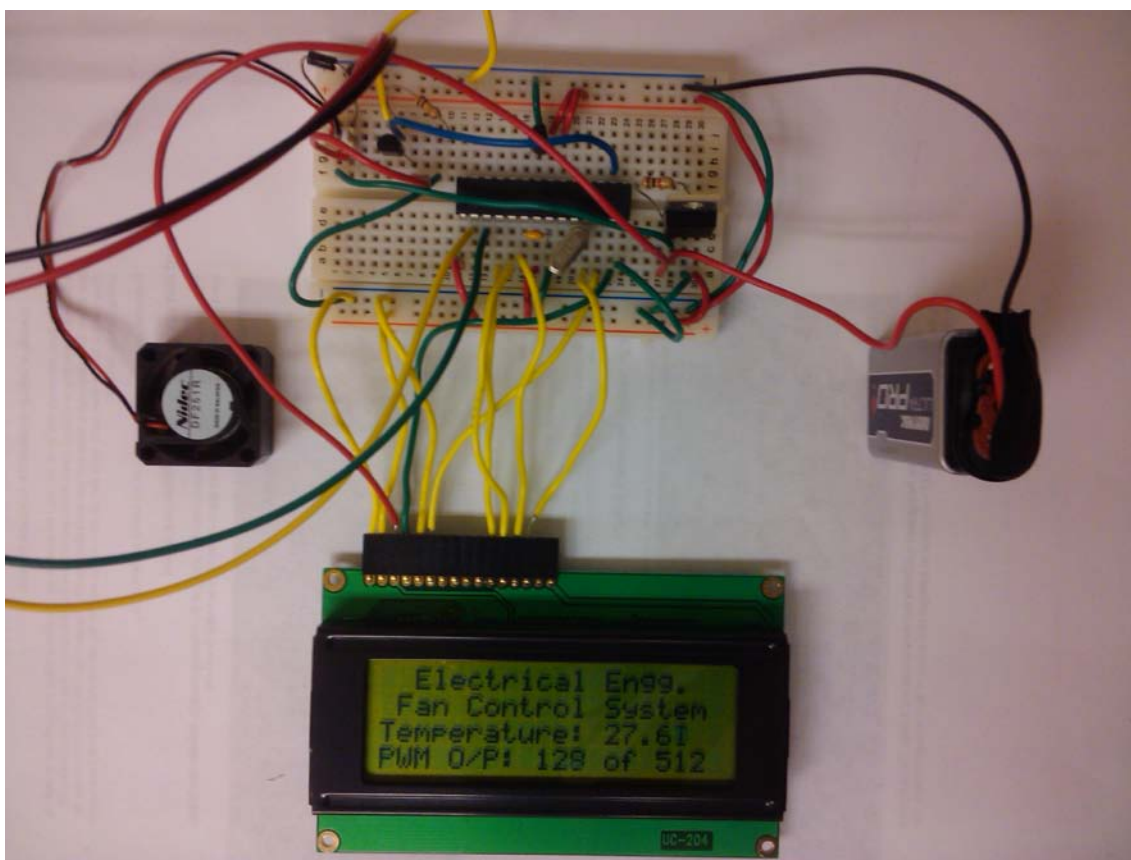


Figure 5.9 Fan Control System with Temperature in Degree Celsius 1

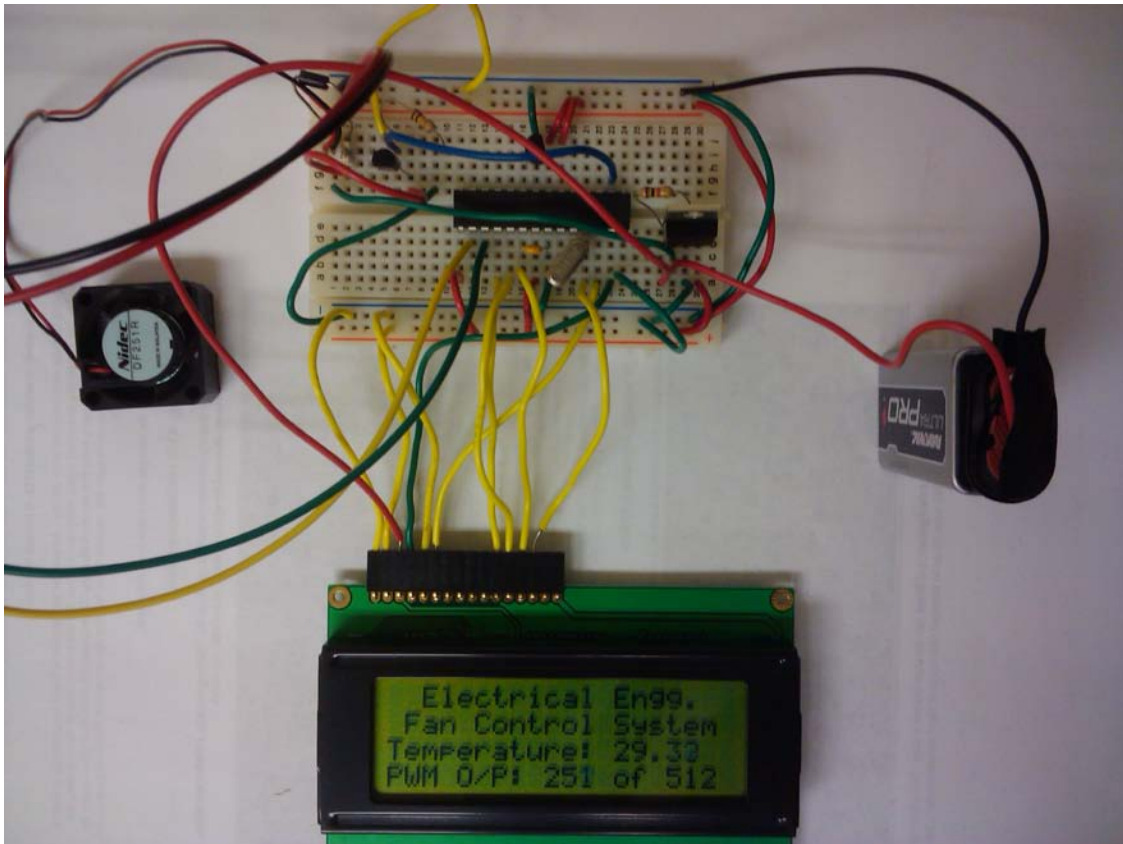


Figure 5.10 Fan Control System with Temperature in Degree Celsius 2

I implemented the whole design to show temperature in degree Celsius too. Figure 5.9 and Figure 5.10 shows the same working project with temperature in degree Celsius on LCD.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Fan control system with temperature sensor and microcontroller was successfully designed and implemented. The working project use temperature sensor to monitor the lab temperature and based on that microcontroller controls speed of motor fan and continuously displays temperature on LCD. This fan control system algorithm provides high efficiency, low noise and low power consumption. Knowledge regarding architecture of microcontroller and how different types of motor work in real time applications were gained. Detailed knowledge of programming language C, WinAVR complier, PWM, ADC and hardware and software interfaces for microcontroller was gained. All goals were achieved and the complete working project can work as a part of cooling system of desktop computers.

6.2 Future work

In the current design, more hardware and software interfaces can be added in such a way that the measured temperature can be transmitted on wireless transmitter or can be sent over internet. This microcontroller can be programmed to control high efficiency motor loads used in industries. ATmega168 microcontroller can be designed to control several motors to make robotic arm. Microcontroller can be used to make line follower – obstacle avoiding robot or light tracer robot with the use of two dc motors and few sensors.

REFERENCES

- [1] Dogan Ibrahim, "Microcontroller Based Temperature Monitoring and Control", Elsevier Science & Technology Books, pp 1-7, pp 129-144, September 2002.
- [2] Atmel Corporation, "8-bit Microcontroller with 8K Bytes In-System Programmable Flash", pp 1- 6, 109-137, 245-260, September 2007.
- [3] National Semiconductor Corporation, "LM34 Precision Fahrenheit Temperature Sensors", pp 1- 6, November 2000.
- [4] STMicroelectronics, "L78xxC-L78xxC Positive voltage regulators", pp 1-7, August 2007.
- [5] HITACHI, "HD44780U Dot Matrix Liquid Crystal Display Controller/Driver", pp 1-9, pp 22-29, 1998.
- [6] Fairchild Semiconductor, "2N7000 N-Channel Enhancement Mode Field Effect Transistor", November 1995.
- [7] Microchip, "Brushless DC (BLDC) Motor Fundamentals", pp 1-18, July 2003.
- [8] T.C.Lun, "Microcontroller for Variable Speed BLDC Fan Control System", Freescale Semiconductor.
- [9] Michael Barr, "Introduction to Pulse Width Modulation (PWM)", July 2003, Retrieved on November 8, 2010 From the World Wide Web:
<http://www.oreillynet.com/pub/a/network/synd/2003/07/02/pwm.html?page=last&x-maxdepth=0>
- [10] Rodney H.G. Tan, Y.H. Goh, Y.Q. Wong and V.H.Mok, "Energy Efficient Cooling Fan for PC Chassis", IEEE conference on Innovative Technologies in Intelligent Systems and Industrial Applications, July 2009
- [11] STMicroelectronics, "AN2680 Application Note: Fan speed controller based on STDS75 or STLM75 digital temperature sensor and ST72651AR6 MCU", February 2008.
- [12] Mike Robbins, "Motors and Microcontrollers 101", NerdKits, Retrieved on October 8, 2010 From the World Wide Web:
http://www.nerdkits.com/videos/motors_and_microcontrollers_101/

[13] Sourceforge.net, “WINAVR”, Retrieved on November 10, 2010 From the World Wide Web:

<http://winavr.sourceforge.net/>

[14] Muhammad Sadiq Bin Sahari, “Programmable Switching Power Supply”, pp 19-24, May 2009.

[15] Eric B. Weddington, “WinAVR User Manual – 20050214”, February 2005, Retrieved on November 10, 2010 From the World Wide Web:

<http://winavr.sourceforge.net/WinAVR-user-manual.html>

[16] Joerg Wunsch, “AVRDUDE”, January 15, 2010, Retrieved on November 16, 2010 From the World Wide Web:

<http://www.nongnu.org/avrdude/user-manual/avrdude.html>