

**Namal University**

**Mianwali**

**Department of Electrical Engineering**

EE-252L: Introduction to Embedded Systems

**Lab Manual: 12**

**DC Motor Control Using AVR**

|  |  |
| --- | --- |
| **Students Name** |  |
| **Roll Number** |  |
| **Submission Date** |  |
| **Marks Obtained** |  |

**Instructors: Dr. Hamza Zad Gul**

# Objectives

# In this lab, the student will learn about sensor interfacing and motor control programming using atmega328p.

# Course Learning Outcomes

CLO1: Practice the correct use of programming constructs of assembly language

CLO2: Construct systems by interfacing AVR peripherals

CLO3: Perform the assigned task individually/as a team effectively

CLO4: Report the outcomes of task performed effectively in oral and written form

# Software

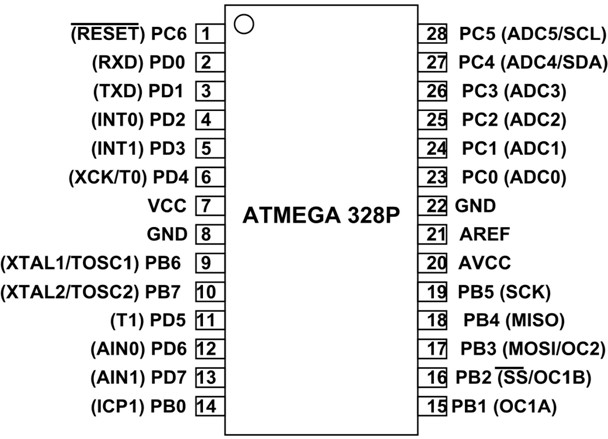
* Microchip studio

# Hardware

* Atmega 328p
* Atmega328p USBasp programmer Board
* Breadboard
* Connecting wires
* LEDs
* Resistors
* Capacitors
* Crystal oscillator
* Push buttons
* Oscilloscope
* IR Sensor
* 9V DC Motor
* 3WD Robot Chasis
* Motor Driver (L293D)

# Instructions

* You must submit the lab report complete within given deadline.
* Plagiarism or any hint thereof will be dealt with strictly. Any incident where plagiarism is caught, both (or all) students involved will be given zero marks, regardless of who copied whom.
* Multiple such incidents will result in disciplinary action being taken.



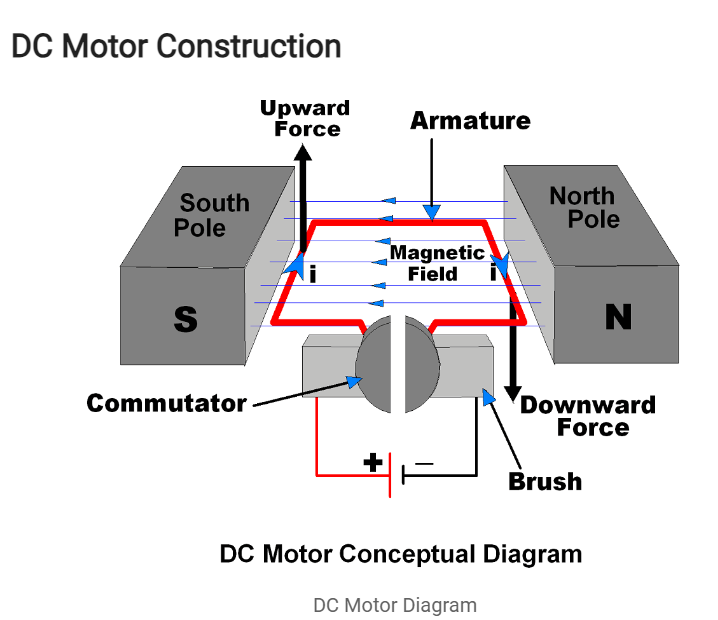
**Introduction:**

**DC Motor:**

DC motor uses Direct Current (electrical energy) to produce mechanical movement i.e. rotational movement. When it converts electrical energy into mechanical energy then it is called as DC motor and when it converts mechanical energy into electrical energy then it is called as DC generator.

The working principle of DC motor is based on the fact that when a current carrying conductor is placed in a magnetic field, it experiences a mechanical force and starts rotating. Its direction of rotation depends upon Fleming’s Left Hand Rule.

DC motors are used in many applications like robot for movement control, toys, quadcopters, CD/DVD diskdrive in PCs/Laptops etc.

****

It has mainly two major parts as,

**Stator** – Static part of the motor.

**Rotor** – Rotating part of the motor.

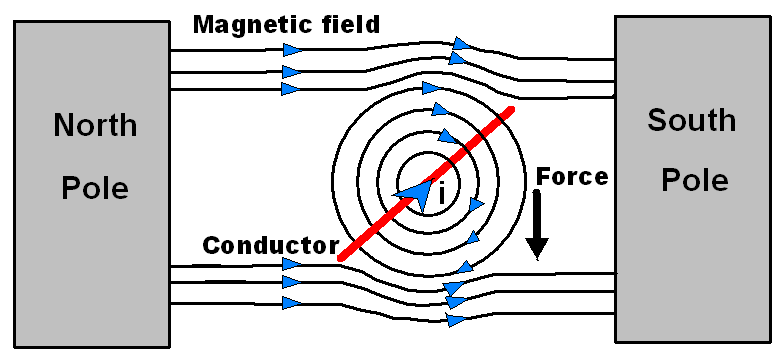
The South and North poles of permanent magnet or Electromagnet are the stator part of the DC motor and armature connected with commutator is rotating part of the DC motor. South and North poles are used to create a magnetic field as shown in figure. The Armature is a conducting material which is placed in between magnetic field produced by North & South Pole.

The current (i) shown in figure is flowing through Armature.

Brushes are used to attach DC supply to the Armature via commutator. Commutators have segments which are attached with each end of conducting Armature. Hence, commutator also rotates with Armature. Brushes are stator part which always keep in contact with commutator**.**

**How DC Motor Works**

As shown in above figure left hand side of armature lifting upward and right hand side of armature going downward. This is because of force, which depends upon direction of magnetic field and direction of current flowing through armature. Fleming’s Left Hand Rule is used to determine direction of force (upward/downward)



Let’s see how force act on armature conductor.

As shown in above figure, the current (i) direction flowing through conductor is inward, hence magnetic field generated around conductor is having direction as per Right Hand Grip Rule shown in figure.

* The magnetic field in between North and South poles having direction from North to South as shown in figure.
* Magnetic lines of forces generated by current carrying conductor and by two poles are shown in figure. These both force lines look in same direction in the above half part of conductor whereas in the below half part of conductor look in opposite direction.
* Hence concentration of magnetic lines of force is more in above part of the conductor which result in the force that moves conductor downward.

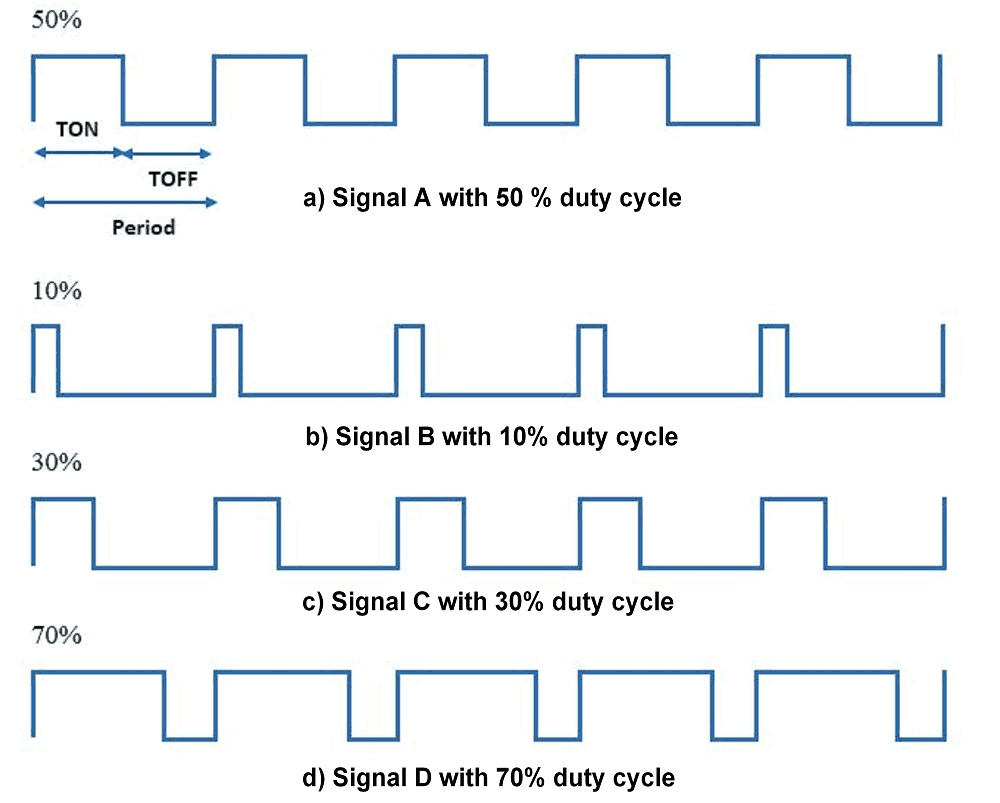
This is for one side of armature conductor whereas on other side of armature conductor current direction will be opposite as shown in constructional diagram and hence force will move the conductor of armature in opposite direction. Hence two side of armature move in upward and downward direction which results in rotation of armature.

Also we can change rotation direction (Clockwise/Anticlockwise) of DC motor by simply changing polarity of applied voltage at motor terminals.

**Pulse Width Modulation Technique**

Pulse Width Modulation is popular technique to control speed of DC motors. It controls average

voltage (V) applied to the DC motor terminals by means of pulse width as shown in below figure.



This is the picture of PWM Waveform

TON is the time for which signal is HIGH and TOFF is the time for which it is LOW. So terminal voltage applies to DC motor is only for TON (ON) time of Period.

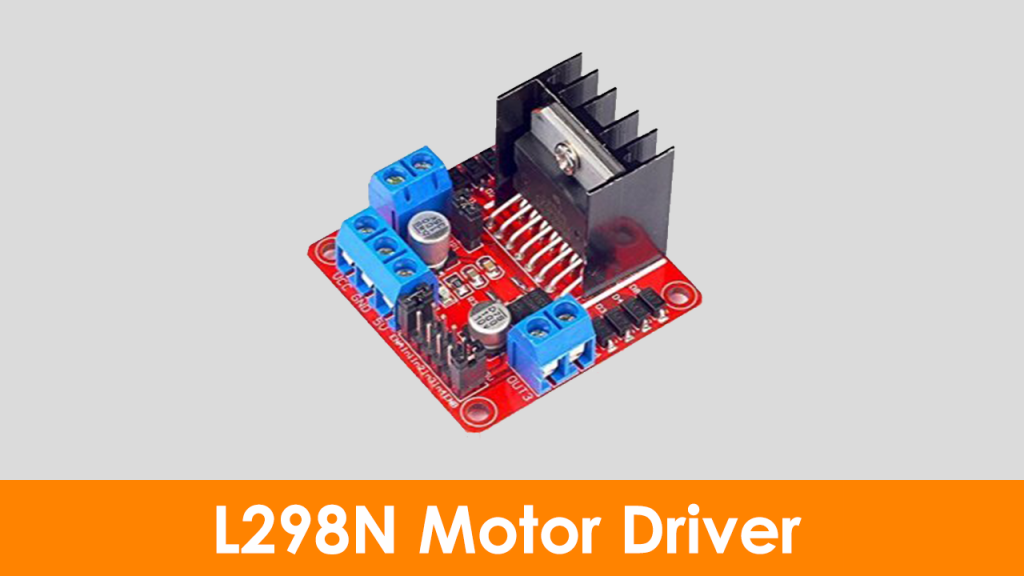
If PWM with 50% duty cycle as shown in above figure it will provide average ≈50% voltage to the motor terminal. So in this way we get simple DC motor speed control using PWM method.

Higher duty cycle gives higher speed and lower duty cycle gives lower speed. We can vary pulse width precisely using Microcontroller to get fine control over DC motor.

**L293D Motor Driver**

L293D is one of the easiest and chipset way to control DC motors. It is the two-channel motor driver that can control the speed and spinning direction of DC motors.

This L293D Motor Driver is a high-power motor driver module. It is used for driving DC and Stepper Motors. This motor driver consists of an L298N motor driver IC and a 78M05 5V voltage regulator, resistors, capacitor, power LED, 5V jumper in an integrated circuit.



When the jumper is placed, it enables the 78M05 Voltage regulator. When the power supply is less than or equal to 12V, the voltage regulator will power on the internal circuitry. When the power supply is more than 12v, then the jumper should not place and should give a separate 5v to power the internal circuitry.

Here, ENA & ENB pins are speed control pins for Motor A, and Motor B. IN1& IN2 and IN3 & IN4 are direction control pins for Motor A and Motor B.

**BO Motors**

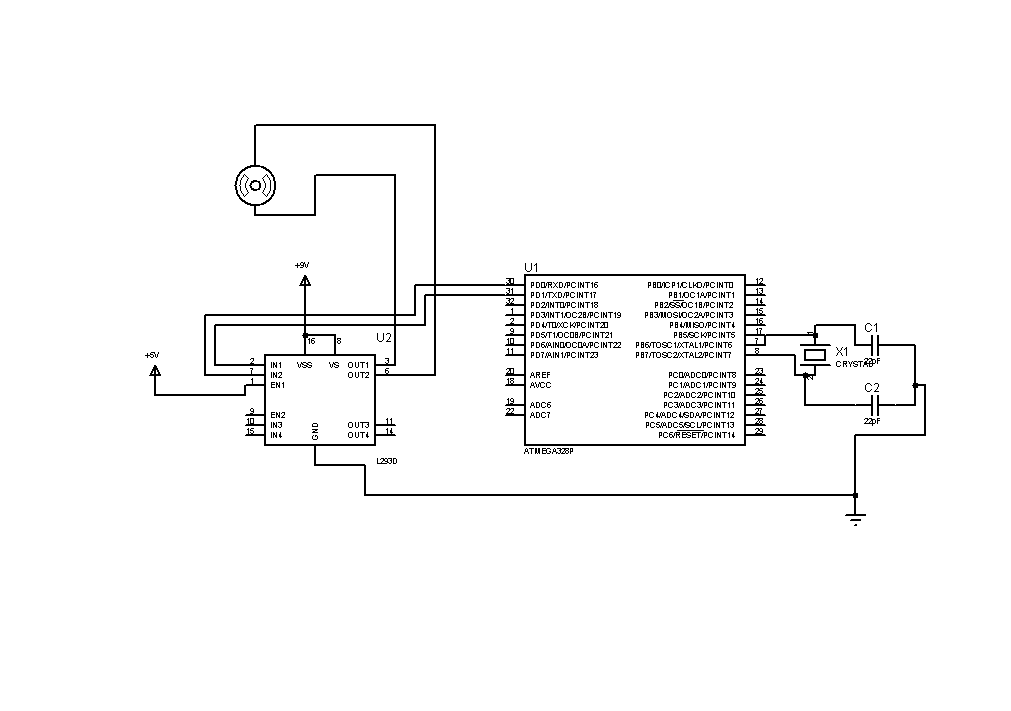
BO Motor is known as Battery Operated motor. These motors are commonly used in hobby-grade projects where the user requires a small DC motor as a simple actuator.



BO series linear motor provides good torque and rpm at lower operating voltages. The BO motors are available in single Shaft, Dual Shaft, and DC Plastic Gear BO. These motors consume low current.

## I: Write the following code in microchip studio and implement the circuit in proteus. Also implement this circuit on hardware

## Write comments in front of each line of code

#include <avr/io.h>

#define *F\_CPU* 16000000UL

#include <util/delay.h>

int main(void)

{

DDRD = 0xFF;

while (1)

{

PORTD = 0x01;

*\_delay\_ms*(2000);

PORTD = 0x00;

*\_delay\_ms*(2000);

PORTD = 0x02;

*\_delay\_ms*(2000);

PORTD = 0x03;

*\_delay\_ms*(2000);

}

}

# Answer the following questions:

# 1. How is the motor direction being controlled? Write down the truth table for direction control.

**II: Assemble your 2WD robot chassis and write code for controlling both DC motors. Make the robot move forward for 10 sec and then move back for 10secs.**

**Implement the circuit in proteus and on hardware.   
Attach proteus screenshot, code and assembled hardware picture below**

# Overview of Ultrasonic Sensor

# Ultrasonic Module HC-SR04 works on the principle of SONAR and RADAR system.

# HC-SR-04 module has an ultrasonic transmitter, receiver, and control circuit on a single board.

# The module has only 4 pins, Vcc, Gnd, Trig, and Echo.

# When a pulse of 10µsec or more is given to the Trig pin, 8 pulses of 40 kHz are generated. After this, the Echo pin is made high by the control circuit in the module.

# The echo pin remains high till it gets an echo signal of the transmitted pulses back.

# The time for which the echo pin remains high, i.e. the width of the Echo pin gives the time taken for generated ultrasonic sound to travel towards the object and return.

# Using this time and the speed of sound in air, we can find the distance of the object using a simple formula for distance using speed and time.

# HCSR04

# Ultrasonic Module HC-SR04 works on the principle of SONAR and RADAR system.

# HC-SR-04 module has an ultrasonic transmitter, receiver, and control circuit on a single board.

# The module has only 4 pins, Vcc, Gnd, Trig, and Echo.

# When a pulse of 10µsec or more is given to the Trig pin, 8 pulses of 40 kHz are generated. After this, the Echo pin is made high by the control circuit in the module.

# The echo pin remains high till it gets an echo signal of the transmitted pulses back.

# The time for which the echo pin remains high, i.e. the width of the Echo pin gives the time taken for generated ultrasonic sound to travel towards the object and return.

# Using this time and the speed of sound in air, we can find the distance of the object using a simple formula for distance using speed and time.

# Steps of Programming

# ATmega328p microcontroller needs to transmit at least 10 us trigger pulse to the HC-SR04 Trig Pin.

# After getting a trigger pulse, HC-SR04 automatically sends eight 40 kHz sound waves and the microcontroller waits for rising edge output at the Echo pin.

# When the rising edge capture occurs at the Echo pin which is connected to an input of ATmega328p, start Timer of ATmega328p and again wait for a falling edge on the Echo pin.

# As soon as the falling edge is captured at the Echo pin, the microcontroller reads the count of the Timer. This time count is used to calculate the distance to an object.

# Calculation (distance in cm)

# Sound velocity = 343.00 m/s = 34300 cm/s

# The distance of Object (in cm) =

# =

# = 17150 \* TIMER

# Now, here we have selected an 8 MHz oscillator frequency for ATmega328p, with No-presaler for timer frequency.

# Then time to execute 1 instruction is 0.125 us.

# So, the timer gets incremented after 0.125 us time elapse.

# = 17150 x (TIMER value) x 0.125 x 10-6 cm

# = 0.125 x (TIMER value)/58.30 cm

# = (TIMER value) / 466.47 cm

## III: Write the following code in microchip studio and implement the circuit in proteus (attach proteus circuit). Also implement this circuit on hardware and attach a picture of implemented hardware. Write comments in front of each line of code

#define *F\_CPU* 800000UL

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <string.h>

#include <stdlib.h>

#define Trigger\_pin PB1

#define LCD\_Port PORTD

#define LCD\_DPin DDRD

#define RSPIN PD0

#define ENPIN PD1

int TimerOverflow = 0;

ISR(TIMER1\_OVF\_vect)

{

TimerOverflow++;

}

void LCD\_Action(unsigned char cmnd)

{

LCD\_Port = (LCD\_Port & 0x0F) | (cmnd & 0xF0);

LCD\_Port &= ~ (1<<RSPIN);

LCD\_Port |= (1<<ENPIN);

*\_delay\_us*(1);

LCD\_Port &= ~ (1<<ENPIN);

*\_delay\_us*(200);

LCD\_Port = (LCD\_Port & 0x0F) | (cmnd << 4);

LCD\_Port |= (1<<ENPIN);

*\_delay\_us*(1);

LCD\_Port &= ~ (1<<ENPIN);

*\_delay\_ms*(2);

}

void LCD\_Init (void)

{

LCD\_DPin = 0xFF;

*\_delay\_ms*(15);

LCD\_Action(0x33);

LCD\_Action(0x32);

LCD\_Action(0x28);

LCD\_Action(0x02);

LCD\_Action(0x0c);

LCD\_Action(0x06);

LCD\_Action(0x01);

*\_delay\_ms*(2);

}

void LCD\_Clear()

{

LCD\_Action (0x01);

*\_delay\_ms*(2);

LCD\_Action (0x80);

}

void LCD\_Print (char \*str)

{

int i;

for(i=0; str[i]!=0; i++)

{

LCD\_Port = (LCD\_Port & 0x0F) | (str[i] & 0xF0);

LCD\_Port |= (1<<RSPIN);

LCD\_Port|= (1<<ENPIN);

*\_delay\_us*(1);

LCD\_Port &= ~ (1<<ENPIN);

*\_delay\_us*(200);

LCD\_Port = (LCD\_Port & 0x0F) | (str[i] << 4);

LCD\_Port |= (1<<ENPIN);

*\_delay\_us*(1);

LCD\_Port &= ~ (1<<ENPIN);

*\_delay\_ms*(2);

}

}

void LCD\_Printpos (char row, char pos, char \*str)

{

if (row == 0 && pos<16)

LCD\_Action((pos & 0x0F)|0x80);

else if (row == 1 && pos<16)

LCD\_Action((pos & 0x0F)|0xC0);

LCD\_Print(str);

}

int main(void)

{

char string[10];

long count;

double distance;

DDRB |= (1<<PORTB1);

DDRB |= ~(1<<PORTB0);

PORTB |= (1<<PORTB0);

LCD\_Init();

LCD\_Printpos(0, 0, "Ultrasonic");

sei();

TIMSK1 = ((1 << TOIE1));

TCCR1A = 0;

while (1)

{

PORTB |= (1 << Trigger\_pin);

*\_delay\_us*(12);

PORTB &= (~(1 << Trigger\_pin));

TCNT1 = 0;

TCCR1B = 0x41;

TIFR1 = 1<<ICF1;

TIFR1 = 1<<TOV1;

while ((TIFR1 & (1 << ICF1)) == 0);

TCNT1 = 0;

TCCR1B = 0x01;

TIFR1 = 1<<ICF1;

TIFR1 = 1<<TOV1;

TimerOverflow = 0;

while ((TIFR1 & (1 << ICF1)) == 0);

count = ICR1 + (65535 \* TimerOverflow);

distance = (double)count / 466.47;

*dtostrf*(distance, 2, 2, string);

*strcat*(string, " cm ");

LCD\_Printpos(1, 0, "Dist = ");

LCD\_Printpos(1, 7, string);

*\_delay\_ms*(500);

}

# }

# Answer the following questions:

# 1. If the crystal frequency is changed to 16MHz, then derive the formula for distance calculation using HC-SR04.

# 2. What are the “dtostrf” and “strcat” commands used for? Why we include #include <string.h> and #include <stdlib.h> header files?

# 3. What is the difference between long and double data types?

# 4. Explain how the ultrasonic distance sensor works in the above code?

**Introduction to Embedded System Lab Rubrics**

* **Method of Evaluation** Viva Conducted during lab and lab reports submitted by students

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Assessment tool/ weightage/**  **(CLO, PLO)** | **Excellent**  **(10 - 9)** | **Good**  **(8 – 7)** | **Satisfactory**  **(6 – 4)** | **Unsatisfactory**  **(3 – 1)** | **Poor**  **0** | **Marks Obtained** |
| **Programming**  **(CLO1, PLO5)** | Correct Code. Easy to understand with proper comments | Correct Code but without proper indentation or comments | Slightly incorrect code with proper comments | Incorrect code with improper format and no comments | Code not submitted |  |
| **Circuit Design**  **(CLO2: PLO3)** | Circuit is simulated/implemented correctly without any errors | Circuit is simulated but implemented with minor errors | Circuit is simulated & implementation both have errors | Circuit is simulated & implemented however some components are missing/incorrect value | Circuit is simulated/implemented does not work |  |
| **Individual/ Teamwork**  **(CLO3:PLO9)** | The student/s worked effectively throughout lab to perform the assigned tasks | The student/s performed all the assigned lab tasks however one member took lead | The student/s completed all tasks however failed to work effectively | The student/s attempted all the tasks however the one member did most of the work | The student/s did not work together/at all |  |
| **Lab Report**  **(CLO4:PLO10)** | The student was able to effectively answer all questions regarding performed tasks and report provides all information without mistakes | The student was able to effectively answer all questions regarding performed tasks however the report has minor mistakes | The student was able to answer most questions regarding performed tasks and information in report is not communicated effectively | The student was able to answer some questions regarding performed tasks and report is confusing and misleading | The student was not able to answer questions regarding performed tasks and report information is incorrect/irrelevant |  |
| Total | | | | | |  |