Ahsanullah University of Science and Technology (AUST)
Department of Computer Science and Engineering

LABORATORY MANUAL

Course No: CSE3216
Course Title: Microcontroller Based System Design Lab

For the students of 3rd Year, 2nd Semester of
B.Sc. in Computer Science and Engineering program

TABLE OF CONTENTS

COURSE OBJECTIVES

This course is designed to provide computer science and engineering undergraduates with basic understanding of the theory and practice of microcontroller based systems. Students will learn about the internal and external structure of microcontroller based embedded systems. Students will learn about the design and construction of microcontroller based automated systems.

PREFFERED TOOL(S)

- Arduino IDE
- Proteus 8.0 (or above)

TEXT/REFERENCE BOOK(S)

- Programming and Interfacing ATMEL's AVRs, by Thomas Grace (1st ed.)
- The AVR Microcontroller and Embedded Systems Using Assembly and C: Using Arduino Uno and Atmel Studio by Sepehr Naimi, Sarmad Naimi, Muhammad Ali Mazidi (2nd ed.)
- Programming Arduino: Getting Started with Sketches, by Simon Monk (2nd ed.)

ADMINISTRATIVE POLICY OF THE LABORATORY

- ✓ Students must perform class assessment tasks individually without help of others.
- ✓ Viva for each program will be taken and considered as a performance.
- ✓ Plagiarism is strictly forbidden and will be dealt with punishment.

Session 1

Experiment 1: Interfacing basic LED blink with Arduino.

OBJECTIVES: Interfacing Arduino analog and digital ports. Implementation of simple Arduino based circuits with LED.

DESCRIPTION: The pins on the Arduino can be configured as either inputs or outputs. Pins configured as OUTPUT with pinMode() are said to be in a low-impedance state. This means that they can provide a substantial amount of current to other circuits. Atmega pins can source (provide positive current) or sink (provide negative current) up to 40 mA (milliamps) of current to other devices/circuits. This is enough current to brightly light up an LED (don't forget the series resistor), or run many sensors, for example, but not enough current to run most relays, solenoids, or motors.

Short circuits on Arduino pins, or attempting to run high current devices from them, can damage or destroy the output transistors in the pin, or damage the entire Atmega chip. Often this will result in a "dead" pin in the microcontroller but the remaining chip will still function adequately. For this reason, it is a good idea to connect OUTPUT pins to other devices with 470Ω or 1k resistors, unless maximum current draw from the pins is required for a particular application.

DESIGN

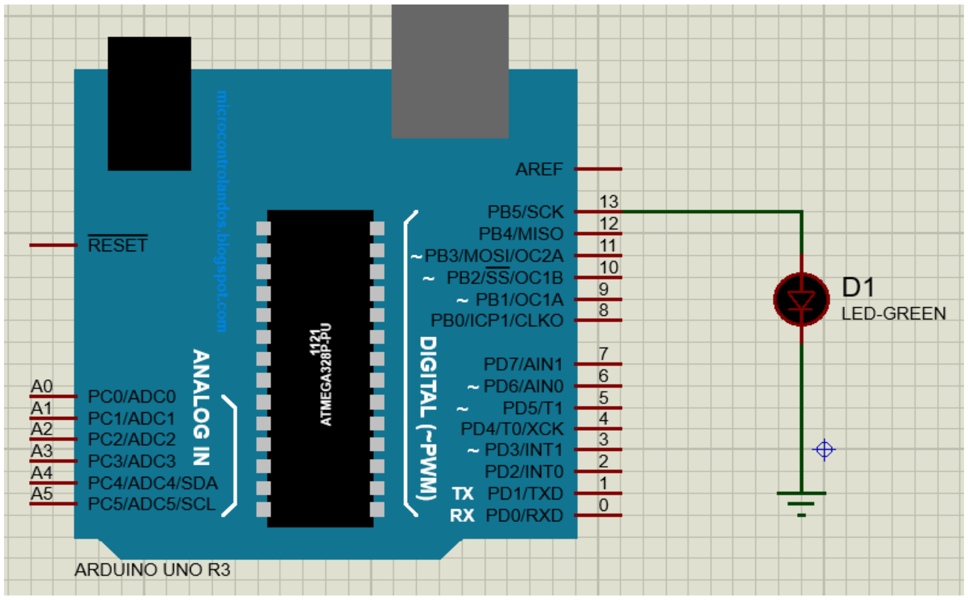
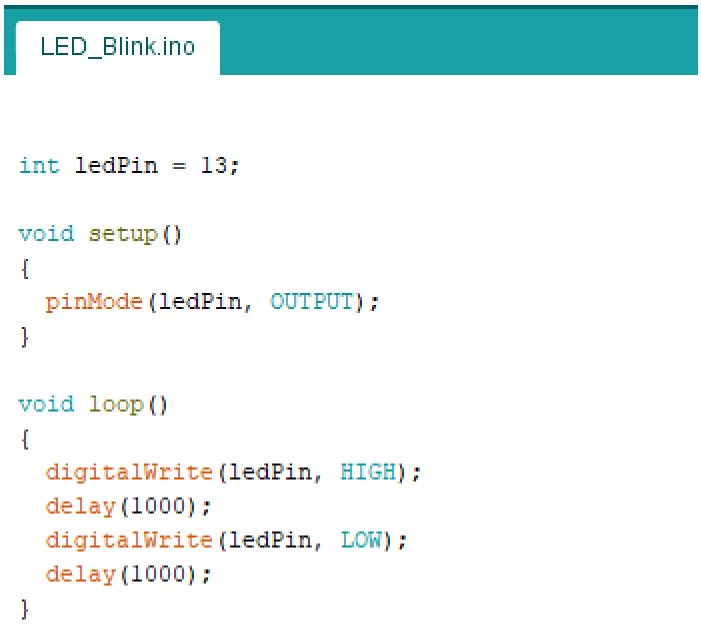


Figure 1: Interfacing basic LED blink with Arduino

CODE



```
int ledPin = 13;

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

EXERCISES:

- Design Arduino based switch controlled LED system.
- Design Arduino based traffic light system.
- Design Arduino based emergency light system.

Session 2

Experiment 2: Interfacing basic 4x4 keypad with Arduino.

OBJECTIVES: Interfacing Arduino digital ports with 4x4 keypad and use of serial monitor for value cross checking.

DESCRIPTION: The Keypad library allows your Arduino to read a matrix type keypad. You can scavenge these keypads from old telephones or you can get them from almost any electronics parts store. They come in 3x4, 4x4 and various other configurations with words, letters and numbers written on the keys. This library is capable of supporting all of those.

You can pretty much connect your keypad to any pins you would like. Be careful not to use the serial pins (0 and 1) if you are using them for communication.

If key presses seem to take a long time to show up then you are probably using long delay()'s in your code. The same thing can happen if you use too many small delay()s like delay(10).

Make sure you understand the pin mappings and have the keypad wired up to match. If you wired the pins incorrectly (and already soldered them in) then you may be able to fix it by redefining the pins and/or keymap to make your keypad work.

DESIGN

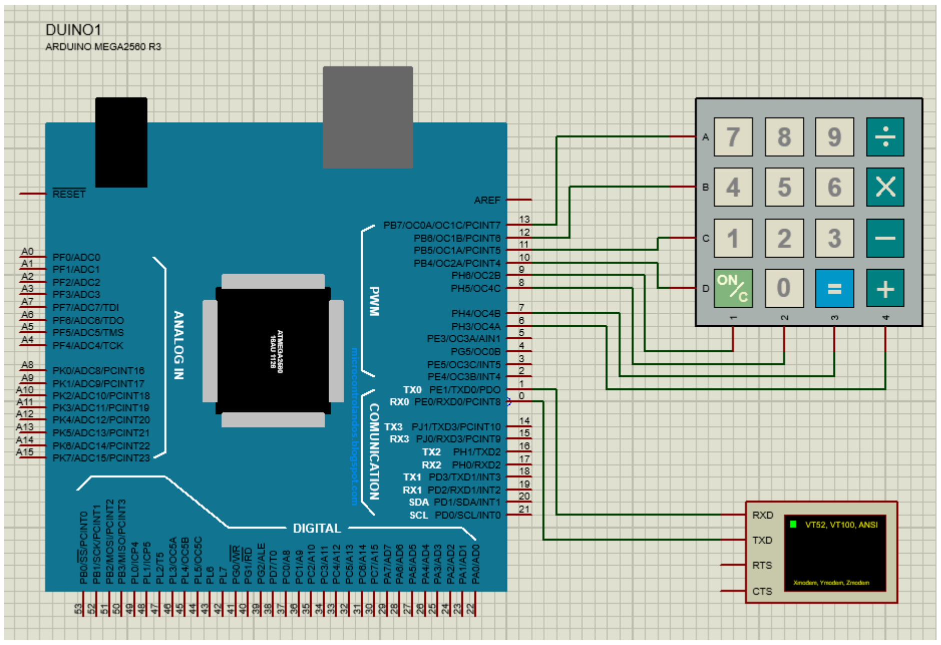
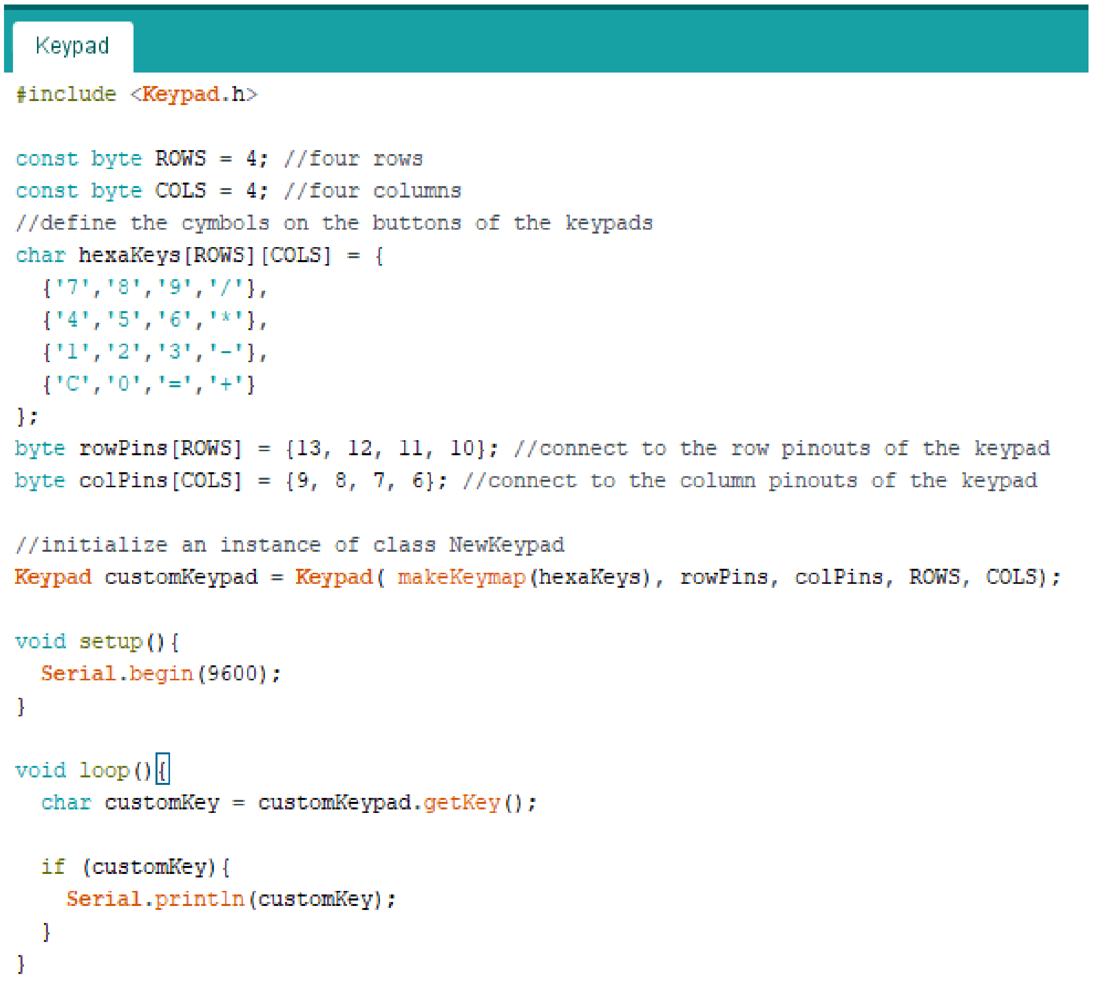


Figure 2: Interfacing basic 4x4 keypad with Arduino

CODE



```
#include <Keypad.h>

const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the cymbols on the buttons of the keypads
char hexaKeys[ROWS][COLS] = {
  {'7','8','9','/'},
  {'4','5','6','*'},
  {'1','2','3','-'},
  {'C','0','=','+'}
};
byte rowPins[ROWS] = {13, 12, 11, 10}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {9, 8, 7, 6}; //connect to the column pinouts of the keypad

//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup(){
  Serial.begin(9600);
}

void loop(){
  char customKey = customKeypad.getKey();

  if (customKey){
    Serial.println(customKey);
  }
}
```

EXERCISES:

- Design a simple calculator using Arduino and available keypad.
- Design a system to show values of a POT in Serial Monitor.

Session 3

Experiment 3: Interfacing basic LCD display with Arduino.

OBJECTIVES: Interfacing 2x20 alphanumeric LCD with Arduino digital ports.

DESCRIPTION: The LiquidCrystal library allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface.

The example sketch provided prints "Hello World!" to the LCD and shows the time in seconds since the Arduino was reset.

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

- A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
- A Read/Write (R/W) pin that selects reading mode or writing mode
- An Enable pin that enables writing to the registers
- 8 data pins (D0 -D7). The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.
- There's also a display contrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The LiquidCrystal Library simplifies this for you so you don't need to know the low-level instructions.

The Hitachi-compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode, so example shows how to control a 2x16 LCD in 4-bit mode.

DESIGN

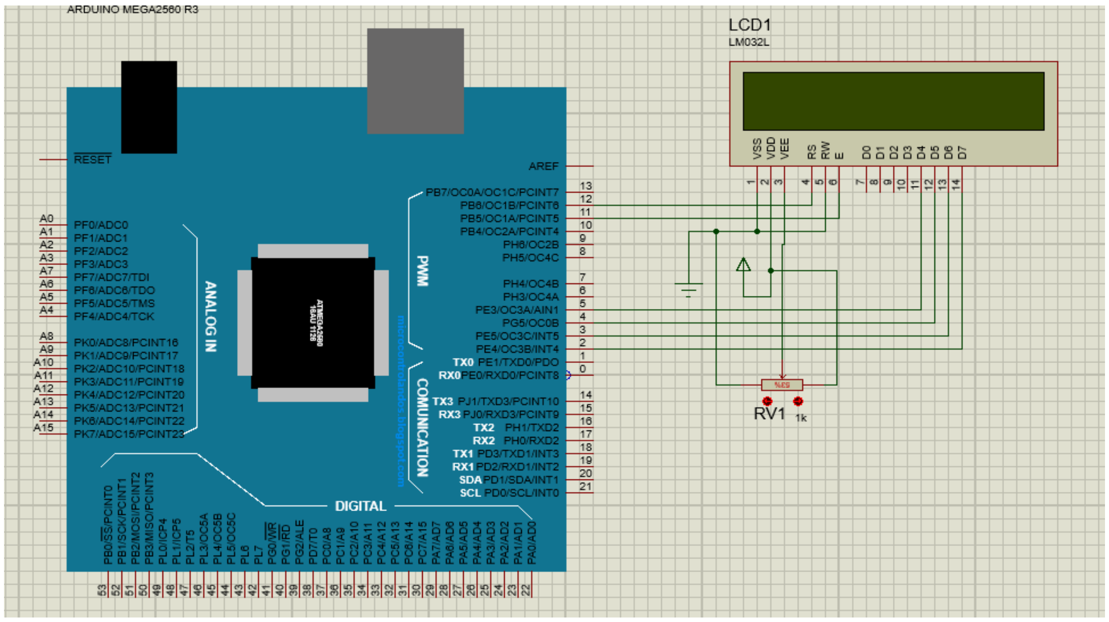
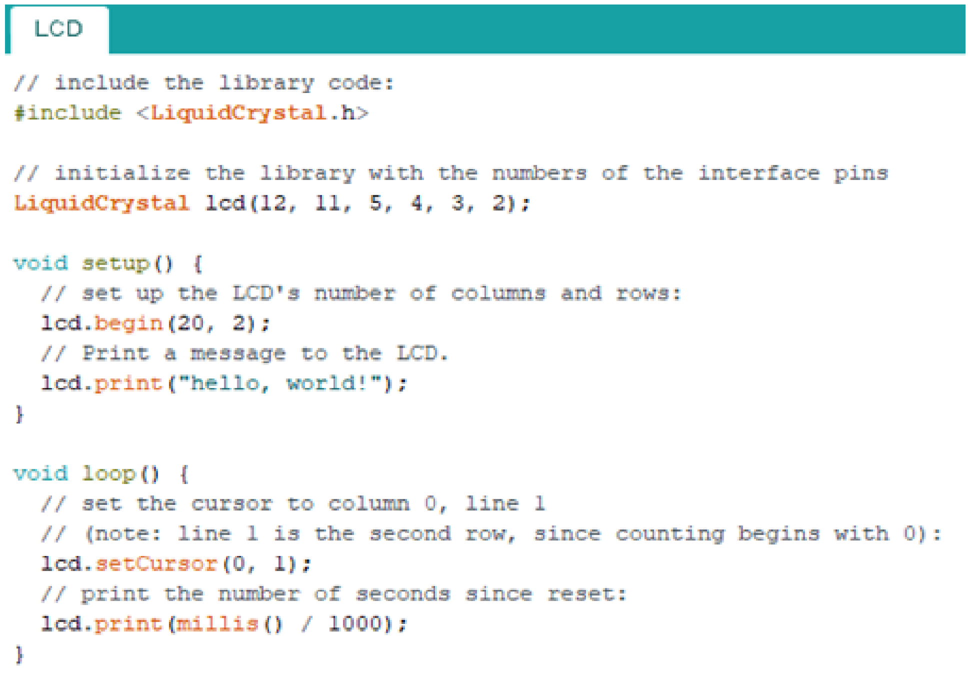


Figure 3: Interfacing basic LCD display with Arduino

CODE



```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(20, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

EXERCISES:

- Design a simple calculator using Arduino, available keypad and 2x20 LCD display.

Session 4

Experiment 4: Interfacing DC motors with motor driver and Arduino.

OBJECTIVES: Interfacing simple DC motors with motor driver L293D and Arduino. Create the working procedures of two wheeler car.

DESCRIPTION: A direct current, or DC, motor is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.

The L293D has two +V pins (8 and 16). The pin '+Vmotor (8) provides the power for the motors, and +V (16) for the chip's logic. We have connected both of these to the Arduino 5V pin. However, if you were using a more powerful motor, or a higher voltage motor, you would provide the motor with a separate power supply using pin 8 connected to the positive power supply and the ground of the second power supply is connected to the ground of the Arduino.

DESIGN

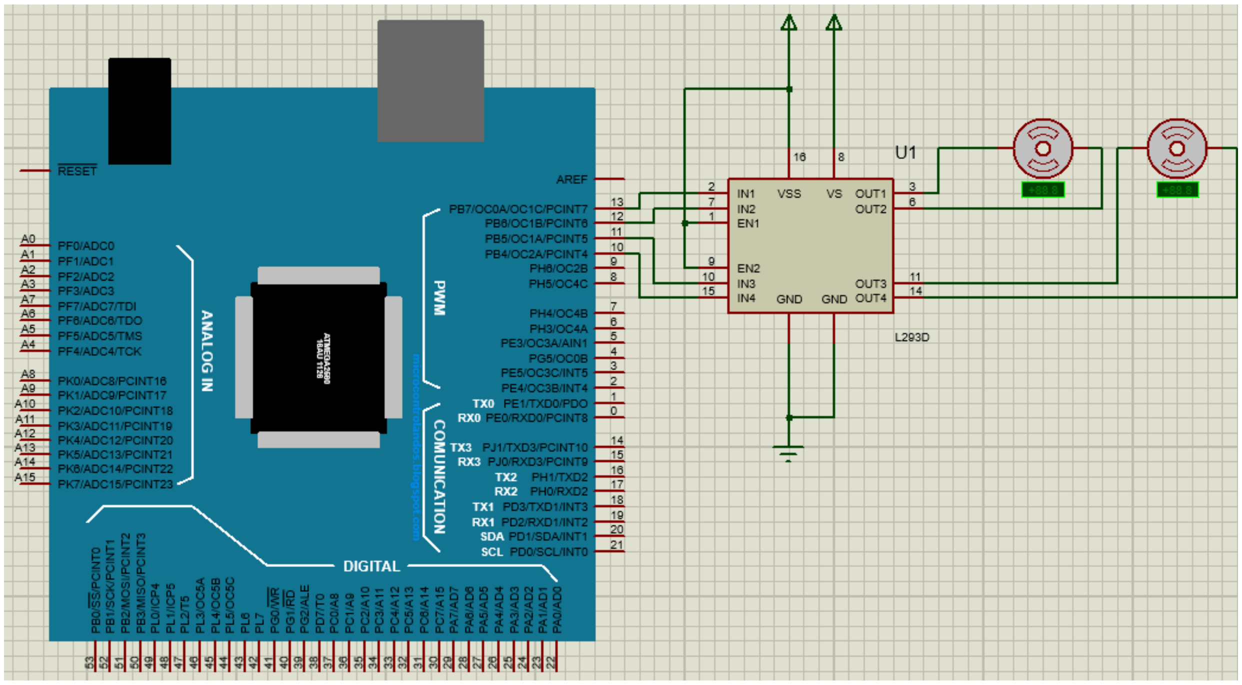
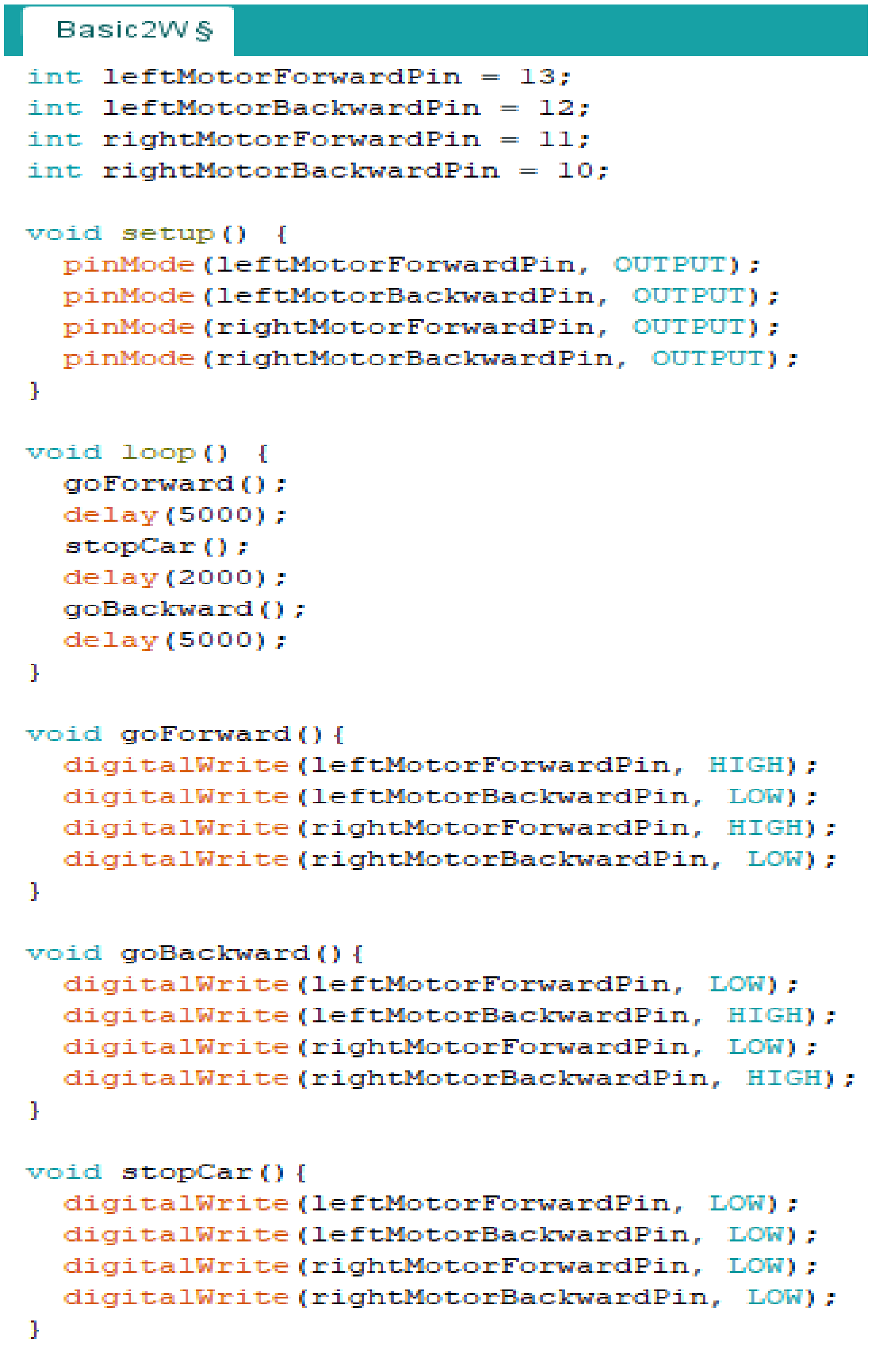


Figure 3: Interfacing DC motors with motor driver and Arduino

CODE



```
int leftMotorForwardPin = 13;
int leftMotorBackwardPin = 12;
int rightMotorForwardPin = 11;
int rightMotorBackwardPin = 10;

void setup() {
  pinMode(leftMotorForwardPin, OUTPUT);
  pinMode(leftMotorBackwardPin, OUTPUT);
  pinMode(rightMotorForwardPin, OUTPUT);
  pinMode(rightMotorBackwardPin, OUTPUT);
}

void loop() {
  goForward();
  delay(5000);
  stopCar();
  delay(2000);
  goBackward();
  delay(5000);
}

void goForward(){
  digitalWrite(leftMotorForwardPin, HIGH);
  digitalWrite(leftMotorBackwardPin, LOW);
  digitalWrite(rightMotorForwardPin, HIGH);
  digitalWrite(rightMotorBackwardPin, LOW);
}

void goBackward(){
  digitalWrite(leftMotorForwardPin, LOW);
  digitalWrite(leftMotorBackwardPin, HIGH);
  digitalWrite(rightMotorForwardPin, LOW);
  digitalWrite(rightMotorBackwardPin, HIGH);
}

void stopCar(){
  digitalWrite(leftMotorForwardPin, LOW);
  digitalWrite(leftMotorBackwardPin, LOW);
  digitalWrite(rightMotorForwardPin, LOW);
  digitalWrite(rightMotorBackwardPin, LOW);
}
```

Exercise:

- Design working scenario of four wheeler car using Arduino with all movements.
- Design a remote controlled car system.

Session 5

Experiment 5: Interfacing servo motors with Arduino.

Objective: Interfacing servo motor with Arduino. Show clockwise and counter clock wise rotations with servo motor.

Description: Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.
The Servo library supports up to 12 motors on most Arduino boards and 48 on the Arduino Mega. On boards other than the Mega, use of the library disables analogWrite() (PWM) functionality on pins 9 and 10, whether or not there is a Servo on those pins. On the Mega, up to 12 servos can be used without interfering with PWM functionality; use of 12 to 23 motors will disable PWM on pins 11 and 12.

DESIGN

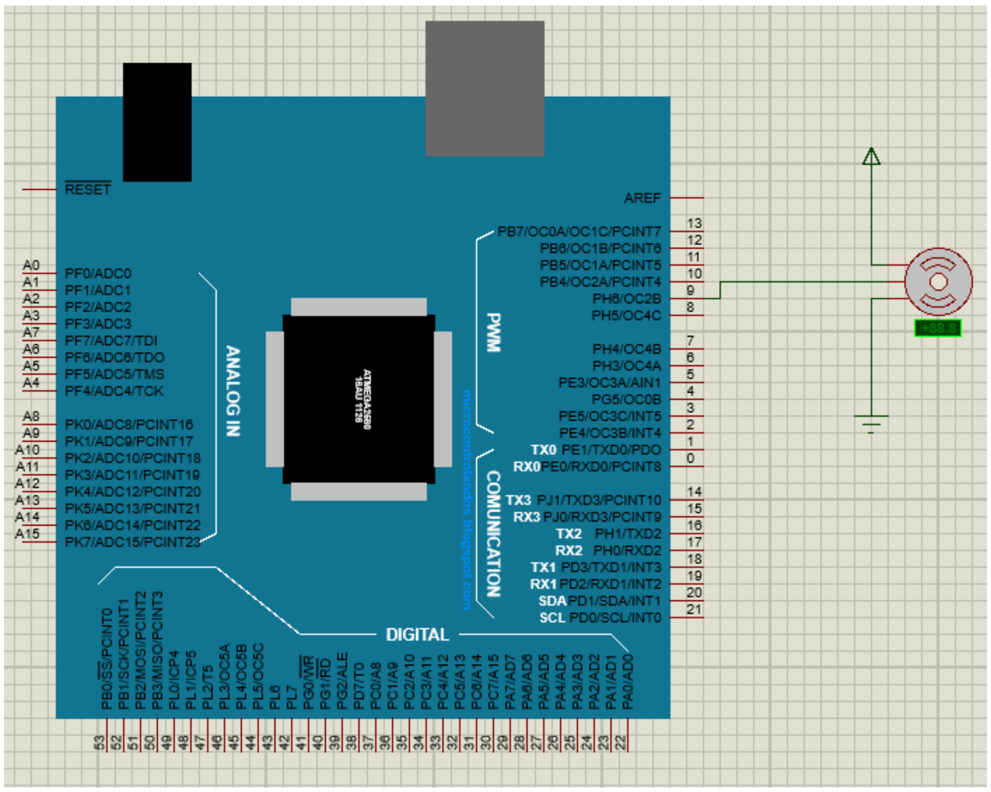
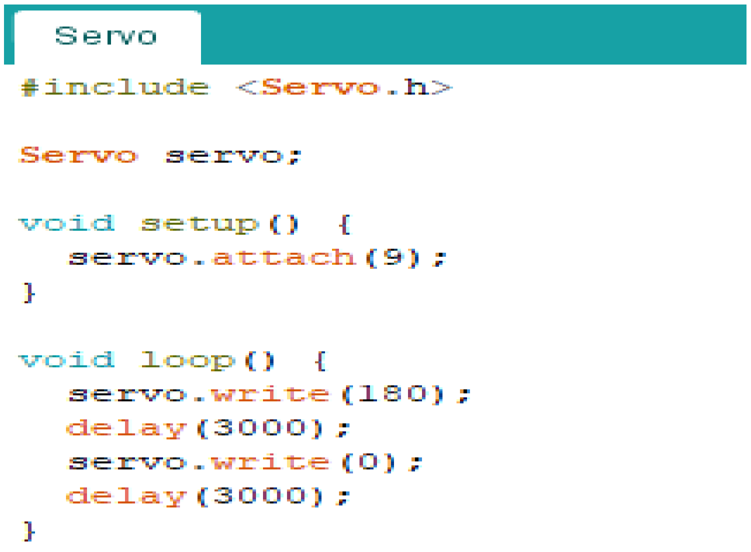


Figure 4: Interfacing servo motors with Arduino

CODE



```
#include <Servo.h>

Servo servo;

void setup() {
  servo.attach(9);
}

void loop() {
  servo.write(180);
  delay(3000);
  servo.write(0);
  delay(3000);
}
```

EXERCISES:

- Design security door system with keypad and servo motor with Arduino.
- Design pinball shooting system with buttons and servo motors.

Session 6

Experiment 6: Interfacing some sensors with Arduino.

Objective: Interfacing LDR, LM-35 and ultrasonic sensor Arduino and displaying the values in serial monitor.

Description: The HC-SR04 Ultrasonic Sensor is a very affordable proximity/distance sensor that has been used mainly for object avoidance in various robotics projects. It essentially gives your Arduino eyes / spacial awareness and can prevent your robot from crashing or falling off a table. It has also been used in turret applications, water level sensing, and even as a parking sensor. The LM35 is a common TO-92 temperature sensor. It is often used with the equation:

*temp = (5.0 * analogRead(tempPin) * 100.0) / 1024;*

However, this does not yield high resolution. This can easily be avoided, however. The LM35 only produces voltages from 0 to +1V. The ADC uses 5V as the highest possible value. This is wasting 80% of the possible range. If you change aRef to 1.1V, you will get almost the highest resolution possible.

A LDR (Light Dependent Resistor) or a photo resistor is a photo conductive sensor. It is a variable resistor and changes its resistance in a proportion to the light exposed to it. It's resistance decreases with the intensity of light.LDR is connected to a 10 Resistance in series. +5 Voltage is applied to this arrangement. As the light intensity changes LDR value changes thus the voltage drop on LDR will change and we are going to measure that voltage change.

DESIGN

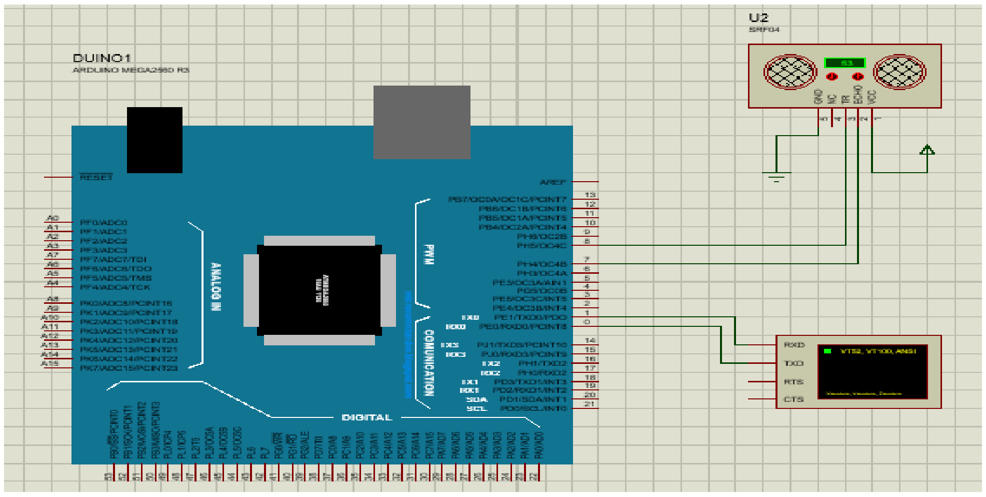
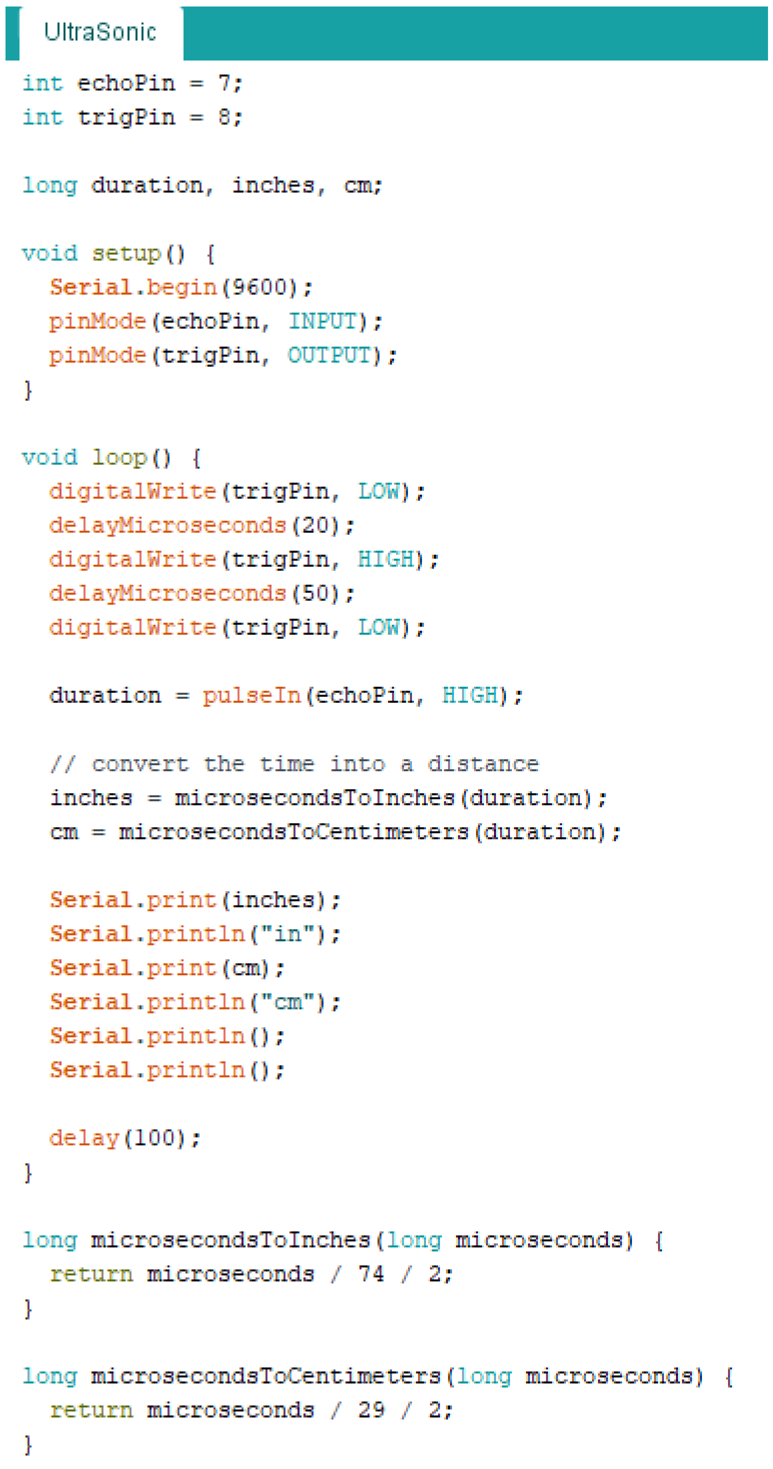


Figure 5: Implementation of Ultra Sonic Sensor

CODE



```
int echoPin = 7;
int trigPin = 8;

long duration, inches, cm;

void setup() {
  Serial.begin(9600);
  pinMode(echoPin, INPUT);
  pinMode(trigPin, OUTPUT);
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(20);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(50);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);

  // convert the time into a distance
  inches = microsecondsToInches(duration);
  cm = microsecondsToCentimeters(duration);

  Serial.print(inches);
  Serial.println("in");
  Serial.print(cm);
  Serial.println("cm");
  Serial.println();
  Serial.println();

  delay(100);
}

long microsecondsToInches(long microseconds) {
  return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29 / 2;
}
```

DESIGN

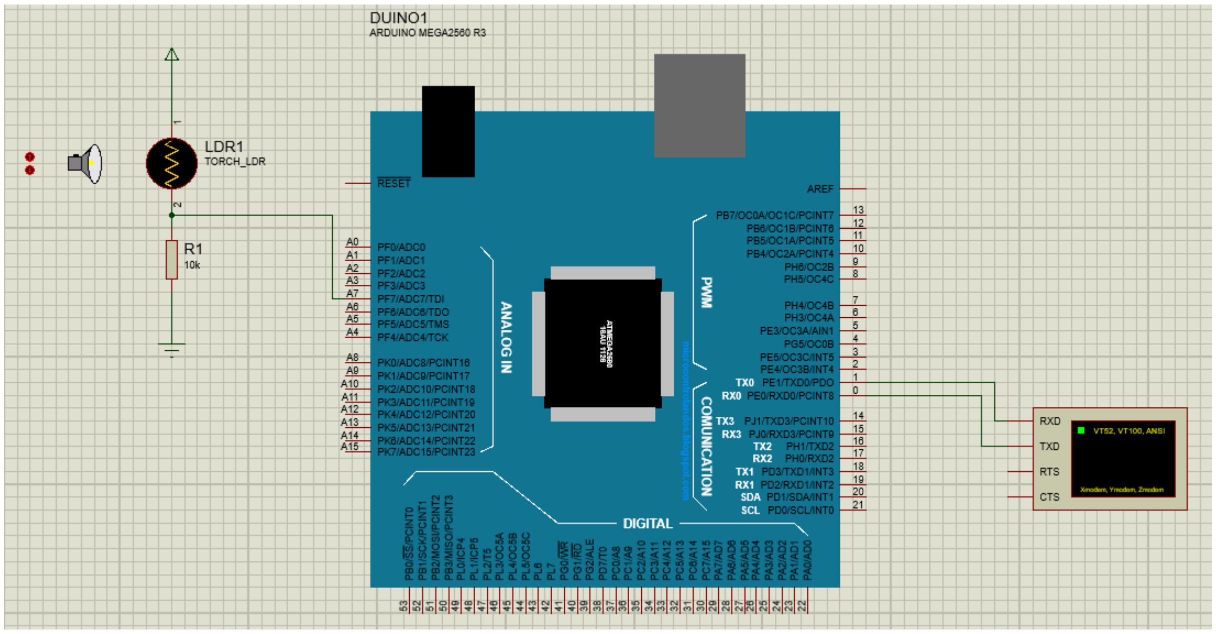
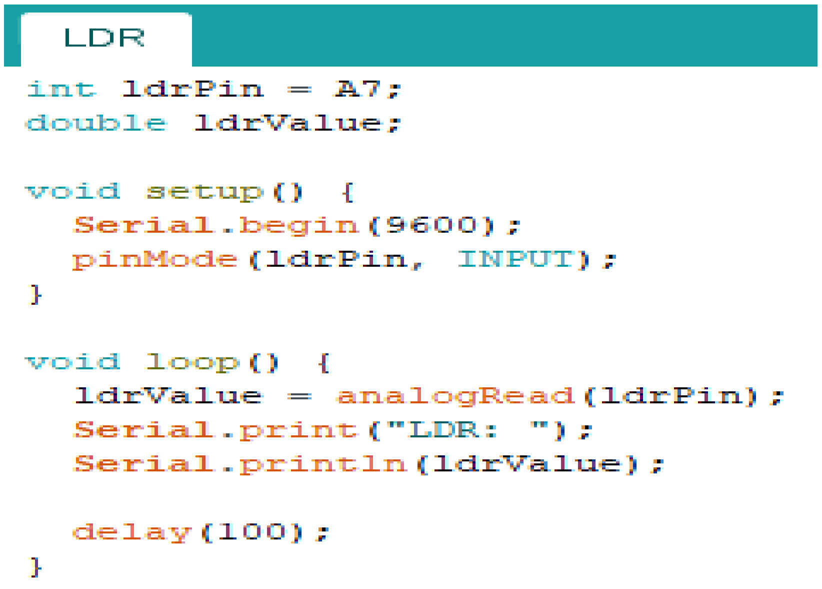


Figure 6: Implementation of LDR

CODE



```
LDR
int ldrPin = A7;
double ldrValue;

void setup() {
  Serial.begin(9600);
  pinMode(ldrPin, INPUT);
}

void loop() {
  ldrValue = analogRead(ldrPin);
  Serial.print("LDR: ");
  Serial.println(ldrValue);

  delay(100);
}
```

DESIGN

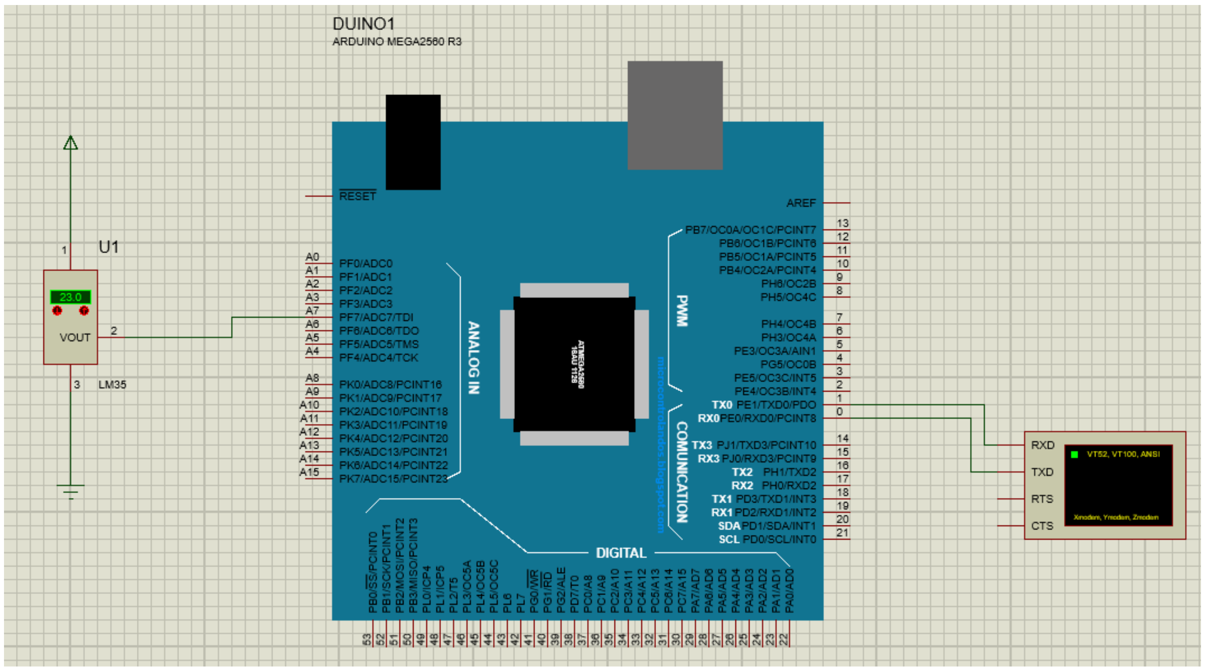
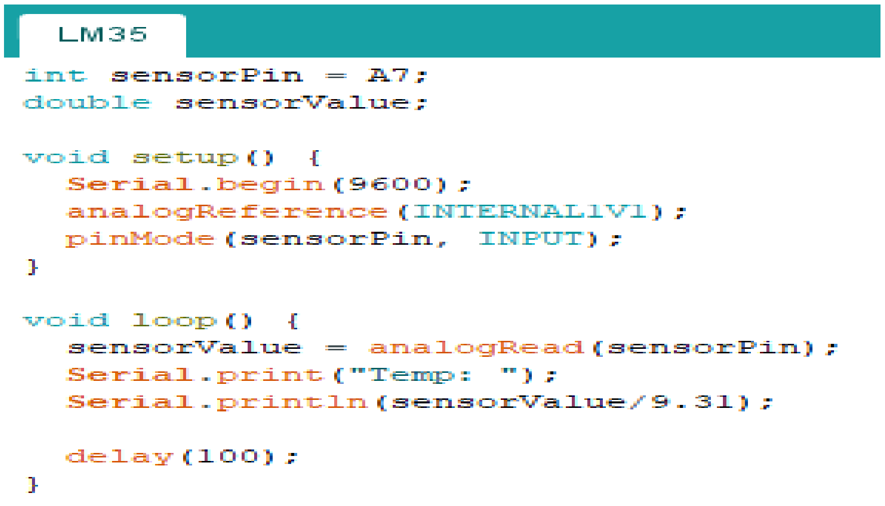


Figure 7: Implementation of LM-35

CODE


LM35

```
int sensorPin = A7;
double sensorValue;

void setup() {
   Serial.begin(9600);
   analogReference(INTERNAL1V1);
   pinMode(sensorPin, INPUT);
}

void loop() {
   sensorValue = analogRead(sensorPin);
   Serial.print("Temp: ");
   Serial.println(sensorValue/9.31);

   delay(100);
}
```

EXERCISES:

- Design a simple home automation system with Arduino and required instruments.
- Design a maze solver robot with Arduino and required sensors.
- Design a Line Follower Robot with Arduino and required sensors.
- Implement Inter Integrated Circuit Protocol between two Microcontrollers.

MID TERM EXAMINATION

There will be a 40-minutes written mid-term examination. Different types of questions will be included such as simulations, writing code fragments etc.

FINAL PROJECT SUBMISSION

Students have to submit their final project with detailed project report.