4

Basics of Arduino

4.1 Introduction to Arduino

Arduino was invented at the Ivrea Interaction Design Institute. It was designed for fast prototyping, targeting the hobbyist without any programming background. Soon the user-friendly platform attracted an audience covering a wider community and started changing to adapt the latest trends in the market, from an 8-bit board to IoT products, wearable devices, and an embedded environment. Arduino boards are completely open source and can be used for application development with particular requirements. The Arduino software is user-friendly and easy to begin with a flexible environment for advanced users. It can be operated on Mac, Linux, and Window platforms. New things can be learned with Arduino.

Advantages of Arduino:

Cost: Arduino boards are less expensive compared to other microcontroller boards.

Platform: The Arduino Software (IDE) is compatible with most of the operating systems like Macintosh OSX, Windows, and Linux.

User friendly: The Arduino Software (IDE) is user-friendly, easy to begin, and has flexibility for the skilled programmers.

Open source: The Arduino is an open-source software that can be programmed with C, C++, or AVR-C languages. So a variety of modules can be designed by users.

4.1.1 Arduino Uno

The Arduino/Genuino Uno has an onboard ATmega328 microcontroller. It has 6 analog input ports (A0–A5) and 14 digital I/O ports, out of which 6 are PWM pins. Each pin can operate on 0–5 V of voltage. It operates at 16 MHz of frequency. Figure 4.1 shows the Arduino Uno board (Table 4.1).

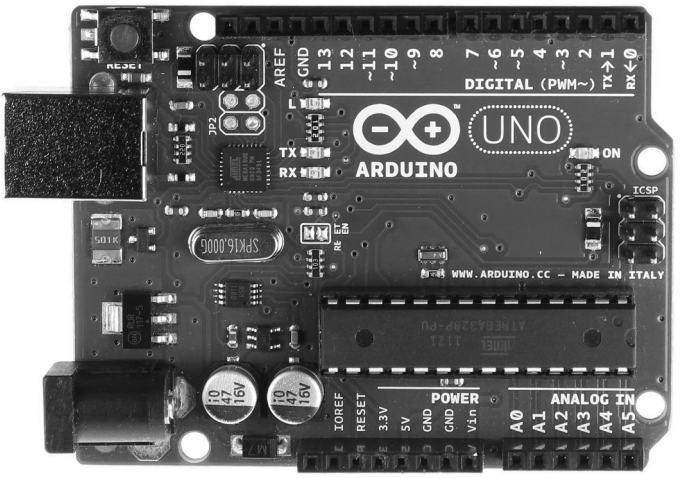

FIGURE 4.1
Arduino Uno board.

TABLE 4.1

Pin Description of Arduino UNO

Pin	Description
Vin	It is the external voltage to the board
3.3 V	3.3 V supply, on board
+5 V	Output voltage +5 V
GND	Ground
IOREF	It is to select the appropriate power source by providing the voltage reference
Serial	It can transmit and receive serial data with 0(Rx) 1(Tx)
External Interrupts	Trigger an interrupt on low value (pins 2 and 3)
PWM	8 bit six PWM (3, 5, 6, 9, 10, 11)
SPI	It supports SPI communication [10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)]
LED	Inbuilt LED driven
TWI	TWI communication [A4 (SDA), and A5 (SCL)]
AREF	Reference voltage with the analog inputs
Reset	It is used to reset the onboard microcontroller

4.1.2 Arduino Mega

The Arduino Mega has an onboard ATmega2560 microcontroller. It has 16 analog inputs, 54 digital I/Os, USB connection, 4 UART, a power jack, and a reset button. It operates on 16 MHz frequency. Figure 4.2 shows the Arduino Mega board (Table 4.2).

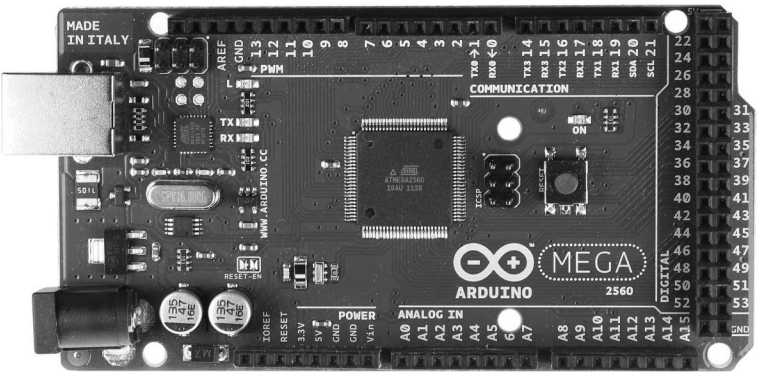

FIGURE 4.2
Arduino Mega board.

TABLE 4.2
Pin Description

Pin	Description
Vin	The external voltage to the Arduino board
+5 V	Output a regulated 5 V
3.3 V	Onboard 3.3 V supply
GND	Ground
IOREF	It is to select the appropriate power source by providing the voltage reference
Serial0	It can transmit and receive serial data with 0(Rx) and 1(Tx)
Serial1	It can transmit and receive serial data with 19(Rx) and 18(Tx)
Serial2	It can transmit and receive serial data with 14(Rx) and 16(Tx)
External Interrupts	It triggers an external interrupt at low value with 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), and 20 (interrupt 2)
PWM	8 bit PWM (pins: 2–13 and 44–46)
SPI	It supports SPI communication [10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK)]
LED	LED driven at pin 13
TWI	Supports TWI communication [pins: 20 (SDA), 21 (SCL)]
AREF	It is a reference voltage for analog inputs
Reset	It is used to reset the microcontroller on board

4.1.3 Arduino Nano

The Arduino/Genuino Nano has an onboard ATmega328 microcontroller. It has 8 analog inputs, 14 digital I/O ports, and 6 PWM. It has onboard 32 KB flash memory, 1 KB EEPROM, 2 KB SRAM, and operates at 16 MHz of frequency. Figure 4.3 shows the Arduino Nano (Tables 4.3 and 4.4).

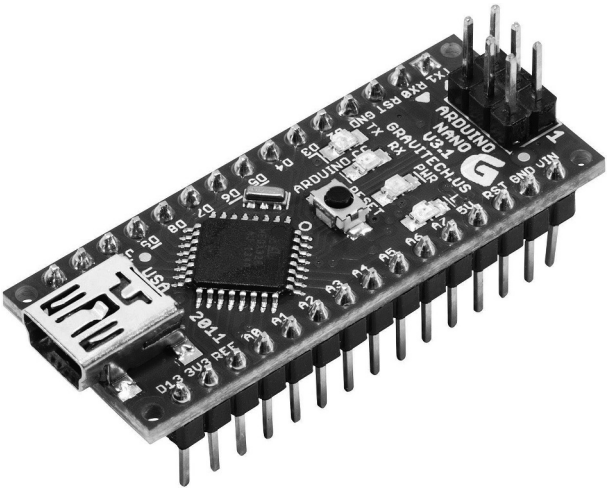

FIGURE 4.3
Arduino Nano board.

TABLE 4.3
Pin Description of Arduino NANO

PIN	Description
Vin	The external voltage to the board
+5 V	Output as +5 V
3.3 V	3.3 V supply on board
GND	Ground
IOREF	It helps to select the appropriate power source by providing a voltage reference
Serial	It can transmit and receive serial data with 0(Rx) and 1(Tx)
External Interrupts	Trigger an interrupt on low value (pins 2 and 3)
PWM	8 bit PWM (3, 5, 6, 9, 10, 11)
SPI	It supports SPI communication with [10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)]
LED	LED driven at pin 13
I2C	Supports two wires interfacing [A4 (SDA) and A5 (SCL)]
AREF	It is a reference voltage for analog inputs
Reset	It is used to reset the microcontroller on board

TABLE 4.4

Comparison Table for a Few Arduino Boards

Name	Processor	CPU Speed	Operating/ Input Voltage	Digital IO/ PWM	Analog In/Out	UART	Flash [kB]
LilyPad	ATmega168V ATmega328P	8 MHz	2.4–5.5 V/ 2.4–5.5 V	14/6	6/0	—	16
Mega 2560	ATmega2560	16 MHz	5 V/4–12 V	54/15	16/0	4	256
Micro	ATmega32U4	16 MHz	5 V/4–12 V	20/4	12/0	1	32
Uno	ATmega328P	16 MHz	5 V/4–12 V	14/6	6/0	1	32
Leonardo	ATmega32U4	16 MHz	5 V/4–12 V	20/4	12/0	1	32
Yùn	ATmega32U4 AR9331 Linux	16 MHz 400 MHz	5 V	20/4	12/0	1	32
Ethernet	ATmega328P	16 MHz	5 V/4–12 V	14/4	6/0	—	32
Gemma	ATtiny85	8 MHz	3.3 V/ 4–16 V	3/2	1/0	—	8
MKRZero	SAMD21 Cortex-M0+ 32 bit low power ARM MCU	48 MHz	3.3 V	22/12	4 (ADC 8/10/ 12 bit)/1 (DAC 10 bit)	1	256

4.2 Arduino IDE

The Arduino integrated development environment (IDE) is an open-source software, and it makes it easy to write code and upload it to the board.

4.2.1 Steps to Install Arduino IDE

Step 1: Install Arduino IDE and open the window

To begin, install the Arduino IDE. Figure 4.4 shows the window of Arduino IDE.

Step 2: Choose the version of Arduino board

Arduino has many versions like UNO, MEGA, NANO, etc. Before starting the project, find out the suitable version by selecting the parameters according to the requirement. The most common board for beginners is the Arduino UNO. Choose the board and the serial port

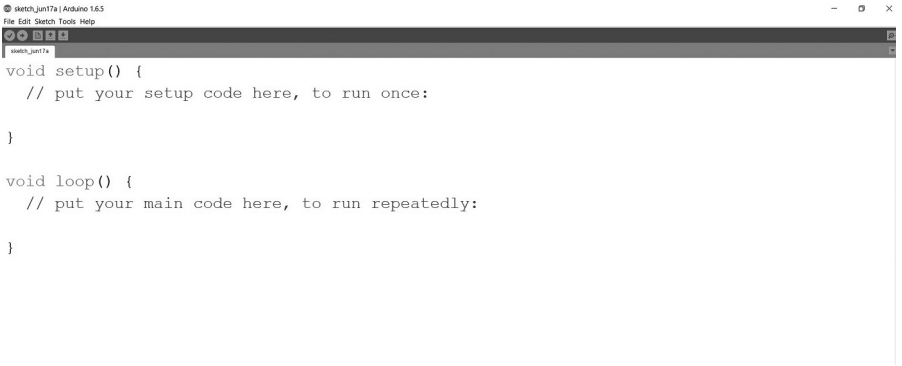

FIGURE 4.4
Window Arduino IDE.

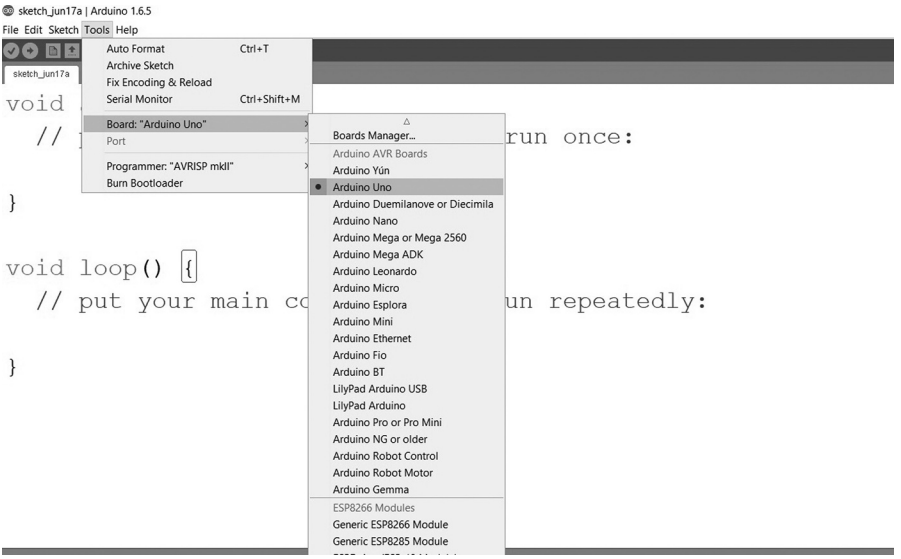

FIGURE 4.5
Selection of Arduino board.

in the Arduino IDE. To select the Arduino board, click on "Tool" and then click on "board." Figure 4.5 shows the selection of "Arduino Uno."

Step 3: Write and compile the program

Write the program in the Arduino IDE window. Then "RUN" the program. Figure 4.6 shows the window to compile the program.

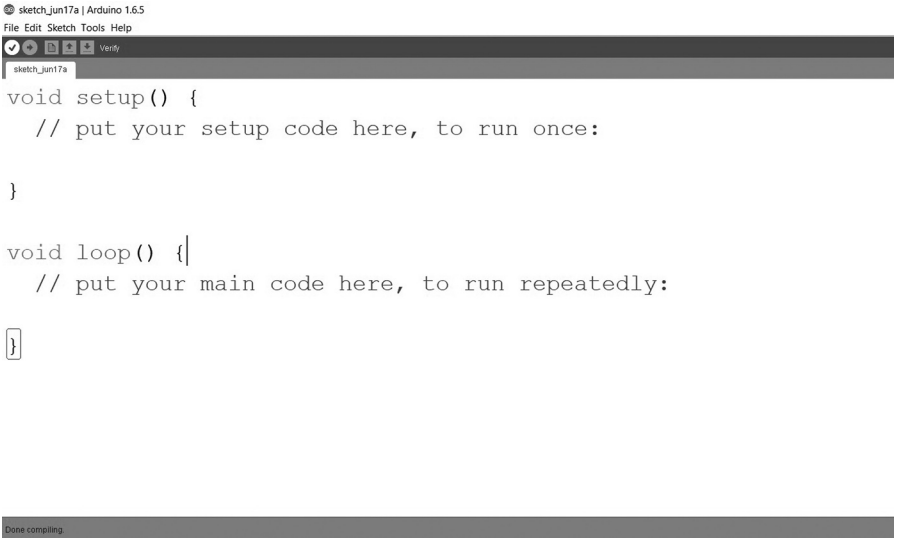

FIGURE 4.6
Compile the program.

Step 4: Connect Arduino with PC

Connect Arduino to the USB port of the PC with a USB cable. Every Arduino board has a different serial-port address (COM2, COM4, etc.), so it is required to reconfigure the port for each Arduino and select it in IDE. To check the port at which the Arduino is connected, right click on "PC" then select "manager"; a window will open. Then double click on "Device Manager." A window as shown in Figure 4.7 will open. Click on ports COM and LPT and port at which device is connected can be found.

Now click on the "Tool" head at the Arduino IDE window. Go to the port and select the same port number, which is found in the device manager (select COM1 or COM2, etc). Figure 4.8 shows the "COM38" as the serial port of the board.

Step 5: Upload program to Arduino board

Upload the program to the Arduino board. Figure 4.9 shows how to upload the program.

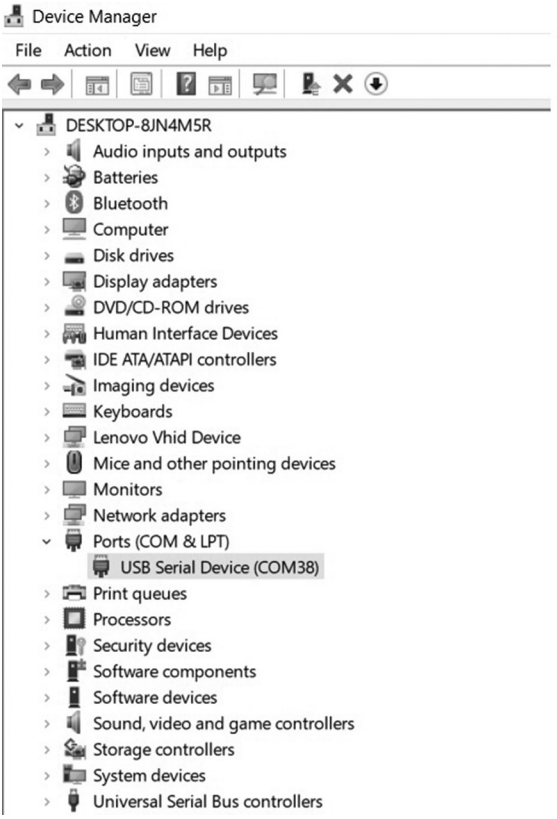

FIGURE 4.7
Window to check port of Arduino.

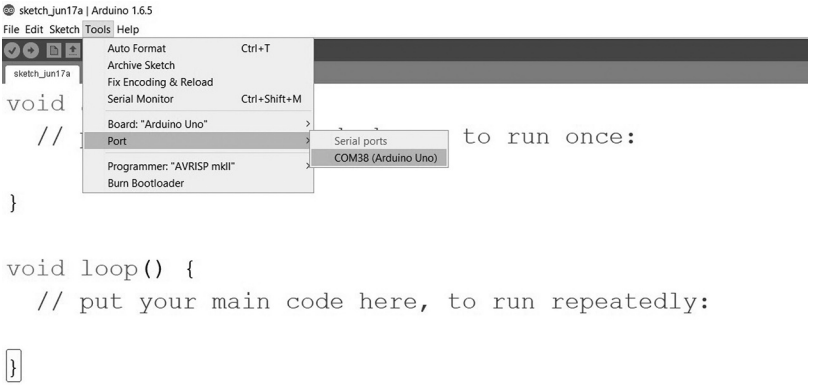

FIGURE 4.8
The serial port of the board.

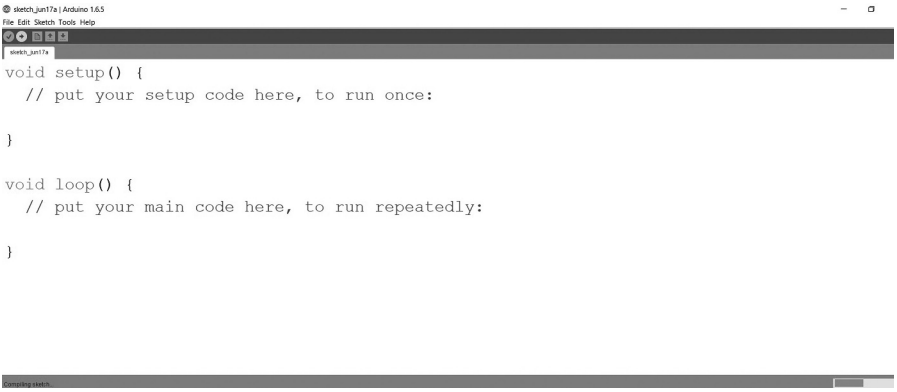

FIGURE 4.9
Window to upload the program.

4.3 Basic Commands for Arduino

1. **pinMode(x, OUTPUT);** // assigned pin number x as output pin where x is number of digital pin
2. **digitalWrite(x, HIGH);** // turn ON the pin number x as HIGH or ON where x is number of digital pin
3. **pinMode(x, INPUT);** // assigned pin number x as input pin where x is number of digital pin
4. **digitalRead(digital Pin);** // read the digital pin like 13 or 12 or 11 etc.
5. **analogRead(analog pin);** // read the analog pin like A0 or A1 or A2 etc.

4.4 LCD Commands

1. **lcd.begin(16, 2);** // initialize LCD 16*2 or 20*4
2. **lcd.print("RAJESH");** // print a string "RAJESH" on LCD
3. **lcd.setCursor(x, y);** // set the cursor of LCD at desired location where x is number of COLUMN and y
4. **lcd.print(LPU);** // print a LPU as integer on LCD
5. **lcd.Clear();** // clear the contents of LCD

4.5 Serial Communication Commands

1. **Serial.begin(baudrate);** // initialize serial communication to set baud rate to 600/1200/2400/4800/9600
2. **Serial.print("RAJESH");** // serial print fixed string with define baud rate on Tx line
3. **Serial.println("RAJESH");** // serial print fixed string with define baud rate and enter command on Tx line
4. **Serial.print("LPU");** // serial print int string with define baud rate on Tx line
5. **Serial.print("LPU");** // serial print int string with define baud rate and enter command on Tx line
6. **Serial.Write(BYTE);** // serial transfer the one byte on Tx line
7. **Serial.read();** // read one byte serial from Rx line

4.6 Play with LED and Arduino

A light-emitting diode (LED) is a device that can be used as an indicator. An LED has two terminals, anode and cathode. LEDs are available in a different colors. Figure 4.10 shows the LED.

FIGURE 4.10
Light-emitting diode.

Different colors can be used to represent different conditions. The color of the LED is due to the emission of light in specific regions of the visible light spectrum by different compounds.

To understand the working of LED, connect the anode of the LED to pin 4 of Arduino and the cathode to ground. Upload the sketch described in Section 4.4.1 to Arduino and observe the blinking of the LED.

Figure 4.11 shows the circuit diagram of the Arduino interfacing with the LED.

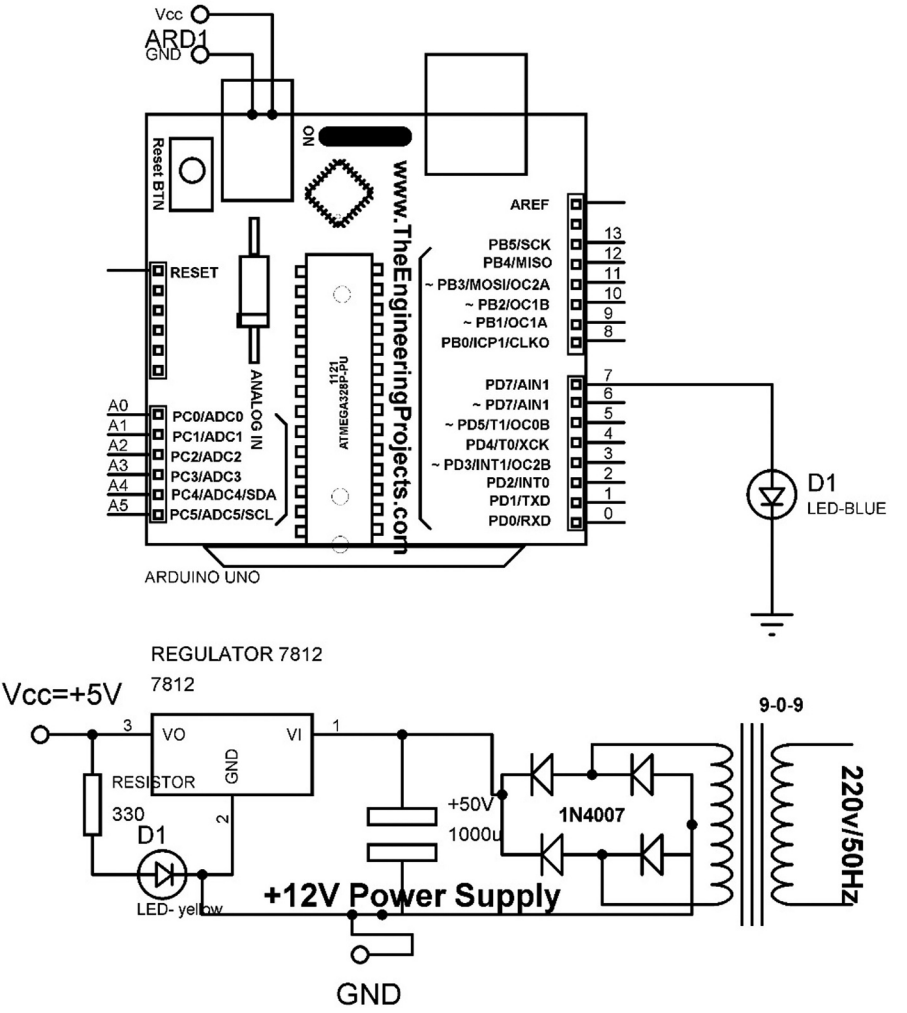

FIGURE 4.11
Circuit diagram to interface LED with Arduino.

4.6.1 Sketch

```
int LED_CONTROL=4;
void setup()
{
  pinMode(LED_CONTROL, OUTPUT); // initialize pin 4 as output pin
}
void loop()
{
  digitalWrite(LED_CONTROL, HIGH); // Make pin 4 HIGH
  delay(1000); // 1000 mS delay
  digitalWrite(LED_CONTROL, LOW); // Make pin 4 HIGH
  delay(1000); // 1000 mS delay
}
```

4.7 Play with LCD with Arduino

The liquid crystal display (LCD) is a commonly used display module. A 16 × 2 LCD display is used as a display device in the circuits. This module is preferred over seven segments because they have no limitation of displaying special, and even custom, characters and are economical.

A 16 × 2 LCD can display 16 characters per row, and there are 2 rows. In this LCD, the 5 × 4 pixel matrix displays the character. It has two registers, namely, data register and command register. Figure 4.12 shows a 16 × 2 LCD.

A 20 × 4 LCD has 4 rows and can display 20 characters per row. A 5 × 4 pixel matrix is used to display characters. Pin description is the same as LCD (16 × 2). Figure 4.13 shows a 20 × 4 LCD (Table 4.5).

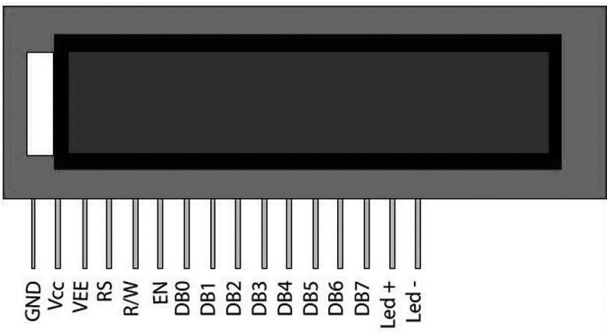

FIGURE 4.12
Liquid crystal display (16 × 2).

FIGURE 4.13
Liquid crystal display (20 × 4).

TABLE 4.5

LCD Pin Description

Pin	Description
Pin (1) Ground	Ground (0 V)
Pin (2) V_{CC}	Power supply (5 V)
Pin (3) V_{EE}	A variable resistor is used to adjust the contrast
Pin (4) Register Select	When low, it selects the command register, and if high, then it selects the data register
Pin (5) Read/Write	High to read the register and low to write on the register
Pin (6) Enable	Send data to data line when high to low pulse is given
Pin (7) DB0	
Pin (8) DB1	
Pin (9) DB2	8 bit data lines
Pin (10) DB3	
Pin (11) DB4	
Pin (12) DB5	
Pin (13) DB6	
Pin (14) DB7	
Pin (15) LED^+	Backlight Vcc (5 V)
Pin (16) LED^-	Backlight ground (0 V)

LCD connection

Connect the components as follows:

- Arduino digital pin (13) to RS pin (4) of LCD.
- Arduino digital pin (GND) to RW pin (5) of LCD.
- Arduino digital pin (12) to E pin (6) of LCD.

- Arduino digital pin (11) to D4 pin (11) of LCD.
- Arduino digital pin (10) to D5 pin (12) of LCD.
- Arduino digital pin (9) to D6 pin (13) of LCD.
- Arduino digital pin (8) to D7 pin (14) of LCD.

Figure 4.14 shows the circuit diagram of the Arduino interfacing with the LCD.

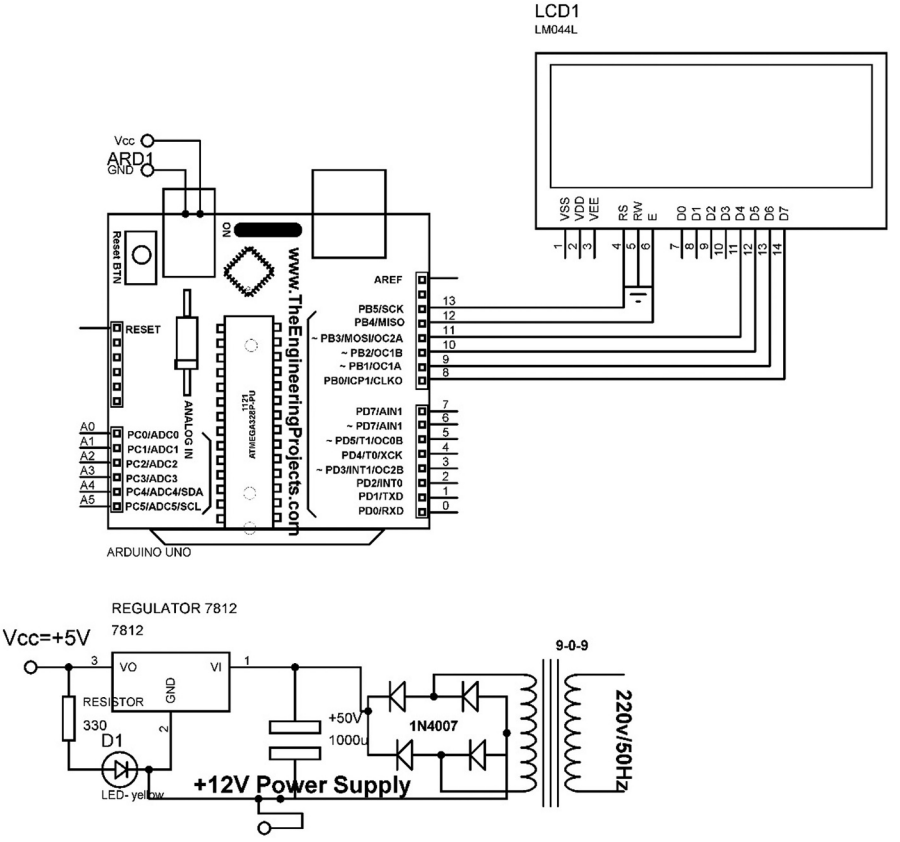

FIGURE 4.14
Circuit diagram to read LCD.

4.7.1 Sketch

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);
void setup()
{
  lcd.begin(20, 4); // Initialize LCD
  lcd.print("WELCOME"); // Print string on LCD
  delay(2000); // Delay 2000mS
  lcd.clear();
}
void loop()
{
  lcd.setCursor(0, 1); // set cursor of LCD
  lcd.print("ECE Department"); // Print string on LCD
  delay(2000); // Delay 2000mS
  lcd.setCursor(0, 2); // set cursor of LCD
  lcd.print("Rajesh Singh"); // Print string on LCD
  delay(2000); // Delay 2000mS
}
```