6

Raspberry Pi and Arduino

6.1 Install Arduino IDE on Raspberry Pi

The limitation with Raspberry Pi is the absence of analog ports onboard, which makes it inappropriate for the systems where analog sensors need to be read. To overcome this limitation, the Arduino integrated development environment (IDE) can be installed on Raspberry Pi, as Arduino has analog ports so these ports can be used to interface analog sensors. Installing Arduino IDE on Raspberry Pi is an easy process with simple steps. The Arduino IDE is available for most of the operating systems, but here we will see how to install it on a Raspberry Pi3 model B with running Raspbian Jessie in the graphical user interface (GUI).

1. The first requirement is an active internet connection.
2. A screen, keyboard, and mouse need to be connected with Raspberry Pi.
3. Install the latest version of Arduino IDE using apt:

   sudo apt-get update &&sudo apt-get upgrade
   sudo apt-get install arduino
4. Connect an Arduino board to the Raspberry Pi using the appropriate cable and pull down the Raspbian main Menu and select Arduino IDE under the "Electronics" head. A blank window will open. Figure 6.1 shows the blank window for Arduino IDE.
5. Click on Tools > Board > and select the appropriate board of Arduino.
6. To select the port of the Arduino that is connected, check the serial port under the "Tools" menu. The port name of Arduino is: */dev/ttyUSB0 or /dev/ttyACM0.*

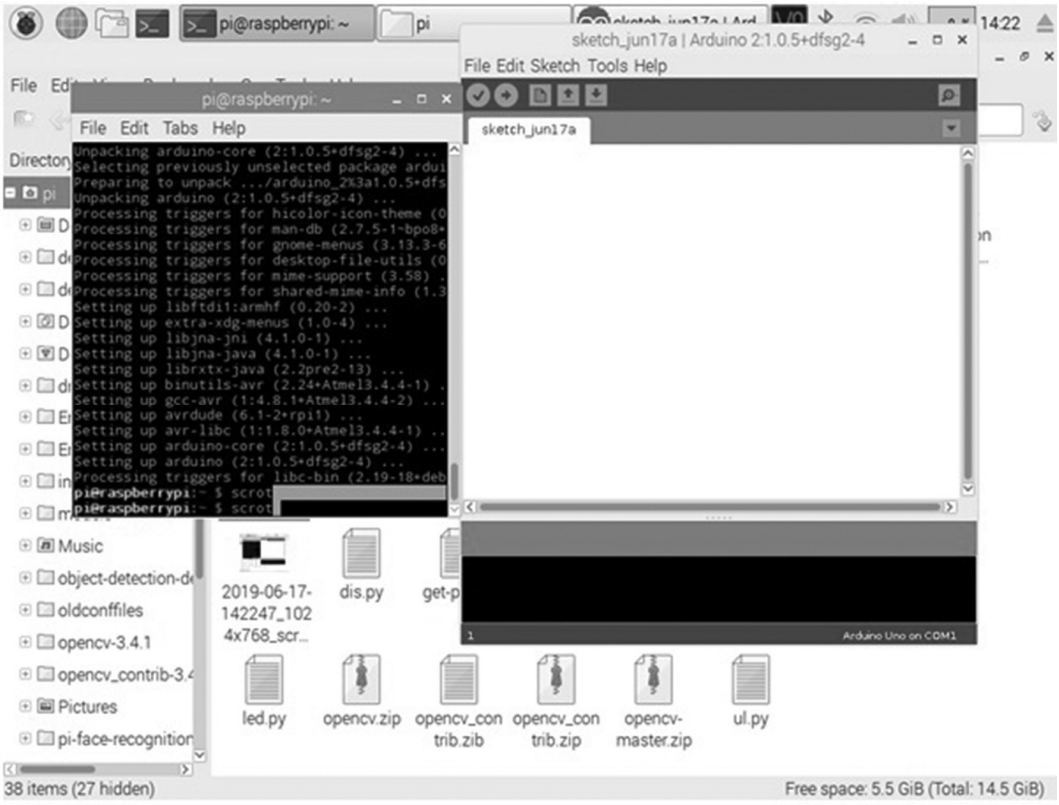

FIGURE 6.1
Blank window of Arduino IDE.

6.2 Play with Digital Sensor

After installing the Arduino IDE on Raspberry Pi, the sensors connected with the Raspberry Pi and Arduino can be read. The Arduino can simply act like the Arduino board, and Raspberry Pi acts as the computer when sensors are connected with Arduino. This can be understood with the help of a few examples.

6.2.1 PIR Sensor

The pyroelectric infrared (PIR) sensor module is used to detect motion. It is compact and easy to use. It has a Fresnel lens and motion detection circuit, which has a wide range of voltages supplied with less current drain. It has a high sensitivity and low noise. The output of the sensor is a transistor–transistor logic (TTL) active low signal. It detects motion by measuring the changes in infrared levels emitted by objects in its surroundings. This module has a detection range of 6 m and can be used in burglar alarms and the control systems. Figure 6.2 shows the block diagram of a system designed to understand the workings of a PIR sensor. It is comprised of an Arduino Uno,

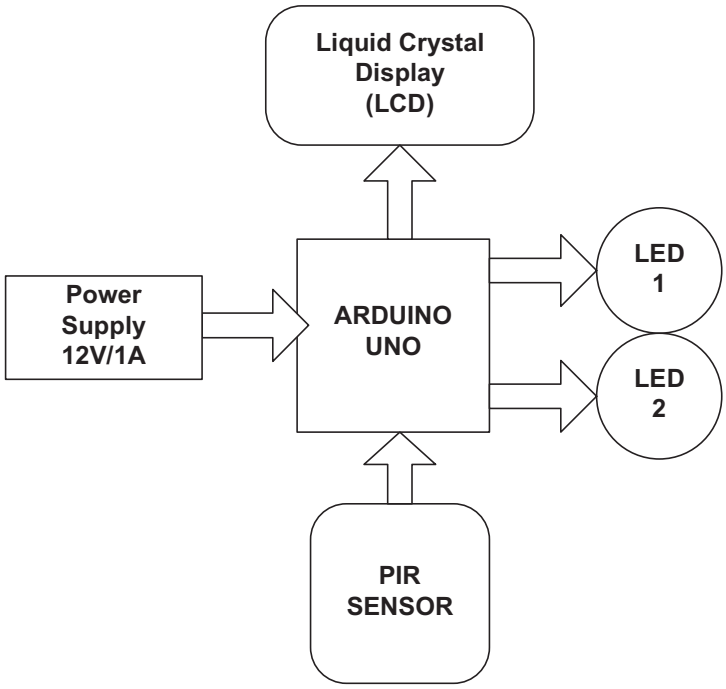

FIGURE 6.2
Block diagram for PIR interfacing with Arduino.

a PIR sensor, a liquid crystal display, and an LED. The system is designed such that "RED LED" will be "ON" if motion is detected; otherwise, "BLUE LED" will be "ON."

6.2.2 Circuit Diagram

Connect the components as shown in Figure 6.3 to check the workings of the PIR sensor. Upload the program described in Section 6.2.2 and check the workings.

PIR sensor connection

- Connect Arduino GND to PIR Module GND.
- Connect Arduino +5 V to PIR Module +.
- Connect Arduino digital pin 2 to PIR Module digital out pin.

LCD connection

- Connect Arduino digital pin (13) to RS pin(4) of LCD.
- Connect Arduino digital pin (GND) to RW pin(5) of LCD.
- Connect Arduino digital pin (12) to E pin(6) of LCD.
- Connect Arduino digital pin (11) to D4 pin(11) of LCD.
- Connect Arduino digital pin (10) to D5 pin(12) of LCD.

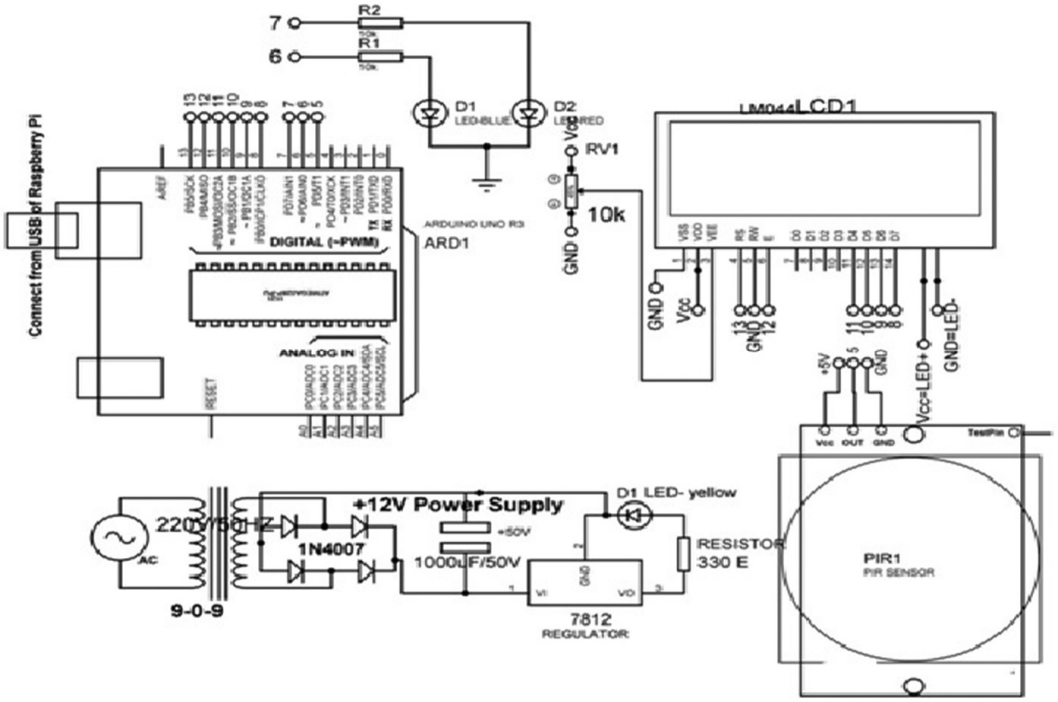

FIGURE 6.3
Circuit diagram for PIR interfacing with Arduino.

- Connect Arduino digital pin (9) to D6 pin(13) of LCD.
- Connect Arduino digital pin (8) to D7 pin(14) of LCD.

LED connection

- Connect Arduino digital pin 7 to anode of RED-LED through 330-ohm resistor.
- Connect Arduino digital pin 6 to anode BLUE-LED through 330-ohm resistor.
- Connect cathode of both LEDs to Ground.

6.2.3 Sketch

```
#include <LiquidCrystal.h> // include library of LCD
LiquidCrystallcd(13, 12, 11, 10,9, 8); // attach LCD pin RS,E,D4,D5,D6,D7
   to the given pins
int PIR_SENSOR_LOW=5; // assign pin 5 as PIR_SENSOR_LOW
int RED_LED=7; // assign pin 7 as RED_LED
int BLUE_LED=6; // // assign pin 6 as BLUE_LED
void setup()
{
```

```
pinMode(PIR_SENSOR_LOW, INPUT_PULLUP); // configure pin5 as
    an input and enable the internal pull-up resistor
pinMode(RED_LED,OUTPUT); // configure pin7 as output
pinMode(BLUE_LED,OUTPUT); // configure pin6 as output
lcd.begin(20, 4); // set up the LCD's number of columns and rows
lcd.setCursor(0, 0); // set cursor to column0 and row1
lcd.print("MOTION SENSOR BASED "); // Print a message to the LCD.
lcd.setCursor(0, 1); // set cursor to column0 and row1
lcd.print("MOTION DETECTION "); // Print a message to the LCD.
lcd.setCursor(0, 2); // set cursor to column0 and row2
lcd.print("SYSTEM AT LPU"); // Print a message to the LCD.
delay(1000);
}
void loop()
{
int PIR_SENSOR_LOW_READ = digitalRead(PIR_SENSOR_LOW);
    // read the PIR value into a variable
if (PIR_SENSOR_LOW_READ == LOW) // Read PIN 5 as LOW PIN
  {
lcd.clear(); // clear the contents of the LCD
lcd.setCursor(0, 3); // set cursor to column0 and row2
lcd.print("MOTION DETECTED "); // Print a message to the LCD.
digitalWrite(RED_LED, HIGH); // Make pin7 to HIGH
digitalWrite(BLUE_LED, LOW); // Make pin6 to LOW
delay(20); // delay of 20 mS
  }
else //otherwise
  {
lcd.clear(); // clear the contents of the LCD
lcd.setCursor(0, 3); // set cursor to column0 and row3
lcd.print("MOTION NOT DETECTED "); // Print a message to the LCD.
digitalWrite(BLUE_LED, HIGH); // Make pin 7 to HIGH
digitalWrite(RED_LED,LOW); // Low pin6 to LOW
delay(20); // delay of 20 mS
  }
}
```

6.3 Play with Analog Sensor

To read the analog sensor with Arduino, simply connect the sensor to any of the analog pins of the board. To understand the workings of the analog sensor, an example of a light-dependent resistor (LDR) is explained here. An LDR has cadmium sulphide (CdS) photoconductive cells with spectral responses. The resistance of the cells decreases with the increase in light intensity. An LDR can be used in many applications, such as smoke detection, automatic light control, batch counting, and burglar alarm systems. Figure 6.4 shows the block diagram to interface the LDR with Arduino. It is comprised of an Arduino Uno, a power supply, a liquid crystal display, and an LDR. The system is designed to display the light intensity of an LCD.

6.3.1 Circuit Diagram

Connect the components as shown in Figure 6.5 to check the workings of the LDR sensor as a simple analog sensor. This LDR has three terminals: Ground, Vcc, OUT. Upload the program described in Section 6.3.2 and check the workings.

Light sensor connection

- Connect Arduino GND to LDR module GND.
- Connect Arduino +5 V to LDR Module +.
- Connect Arduino A0 pin to OUT pin of sensor.

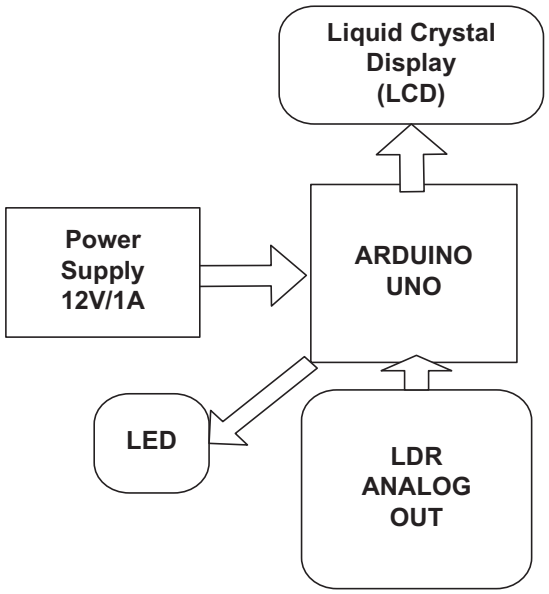

FIGURE 6.4
Block diagram to interface LDR with Arduino.

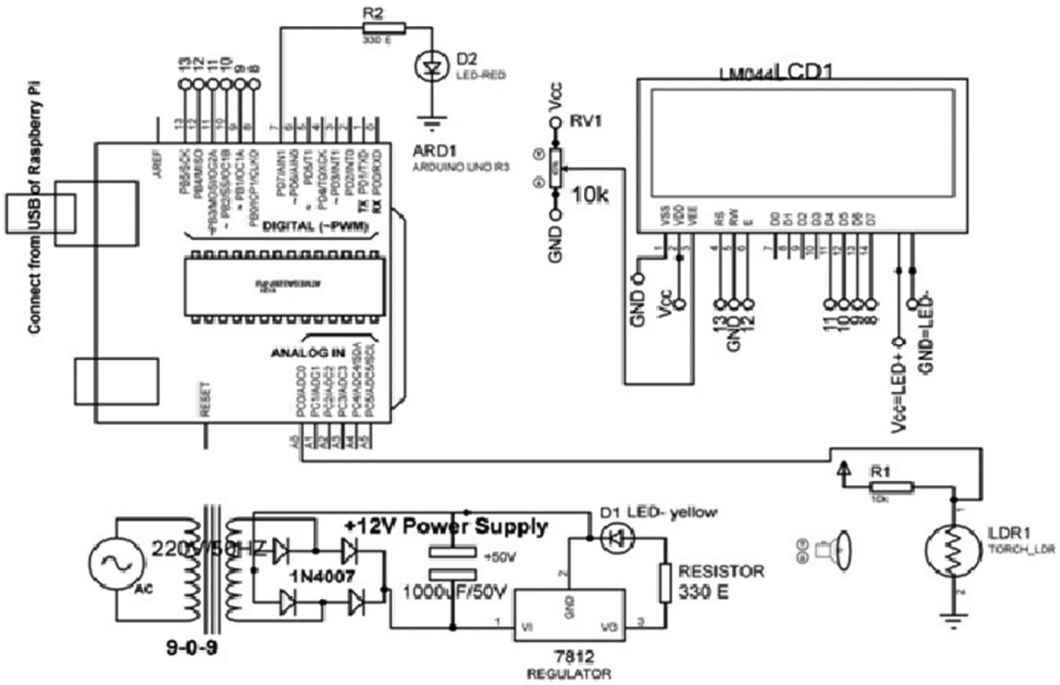

FIGURE 6.5
Circuit diagram for LDR sensor interfacing with Arduino.

LCD connection

- Connect Arduino digital pin 13 to RS pin(4) of LCD.
- Connect Arduino digital pin GND to RW pin(5) of LCD.
- Connect Arduino digital pin 12 to E pin(6) of LCD.
- Connect Arduino digital pin 11 to D4 pin(11) of LCD.
- Connect Arduino digital pin 10 to D5 pin(12) of LCD.
- Connect Arduino digital pin 9 to D6 pin(13) of LCD.
- Connect Arduino digital pin 8 to D7 pin(14) of LCD.

6.3.2 Sketch

```
#include <LiquidCrystal.h> // include library of LCD
LiquidCrystallcd(13, 12, 11, 10, 9, 8); // attach LCD pin RS,E,D4,D5,D6,D7
    to the given pins
intLDR_sensor_Pin = A0; // select the input pin for the potentiometer
intLDR_sensor_ADC_Value = 0; // variable to store the value coming
    from the sensor
int RED_LED=7; // assign pin 7 to RED_LED
```

```
void setup()
{
lcd.begin(20, 4); // Initialise 20*4 LCD
pinMode(RED_LED,OUTPUT); // use RED_LED as an output
lcd.setCursor(0, 0); // set cursor of LCD at column0 and Row0
lcd.print("LDR based light"); // print string on LCD
lcd.setCursor(0, 1); // set cursor on LCD
lcd.print("intensity monitoring"); // print string on LCD
lcd.setCursor(0, 2); // set cursor on LCD
lcd.print("system at LPU"); // print string on LCD
delay(1000); // delay of 1000 mS
lcd.clear(); // clear the contents of LCD
}

void loop()
{
LDR_sensor_ADC_Value = analogRead(LDR_sensor_Pin); // read the
   value from the sensor
lcd.setCursor(0,2); // set cursor on LCD
lcd.print("ADC LEVEL+LDR:"); // print string on LCD
lcd.setCursor(17,2); // set cursor on LCD
lcd.print(LDR_sensor_ADC_Value); // // print value on LCD
if(LDR_sensor_ADC_Value>=100)
  {
digitalWrite(RED_LED,HIGH); // make pin7 to HIGH
delay(20); // delay of 20 mS
  }
else
  {
digitalWrite(RED_LED,LOW); // make pin7 to HIGH
delay(20); // delay of 20 mS
  }
}
```

6.4 Play with Actuators

An actuator is a component of a machine that is used for moving and controlling a mechanism or system. A DC motor, stepper motor, and servo motor are commonly used actuators in the systems.

6.4.1 DC Motor

The DC-geared motors with 100 rpm 12 V are generally used for robotics applications. They are very easy to use. They have nuts and threads on the shafts to easily connect and internally threaded shafts for easy connection to the wheel. Figure 6.6 shows the block diagram to interface the DC motor with Arduino. It is comprised of an Arduino Uno, a power supply, a liquid crystal display, a motor driver (L293D), and two DC motors.

6.4.1.1 Circuit Diagram

Connect the components as shown in Figure 6.7 to check the workings of a DC motor. Upload the program described in Section 6.4.1.2 and check the workings.

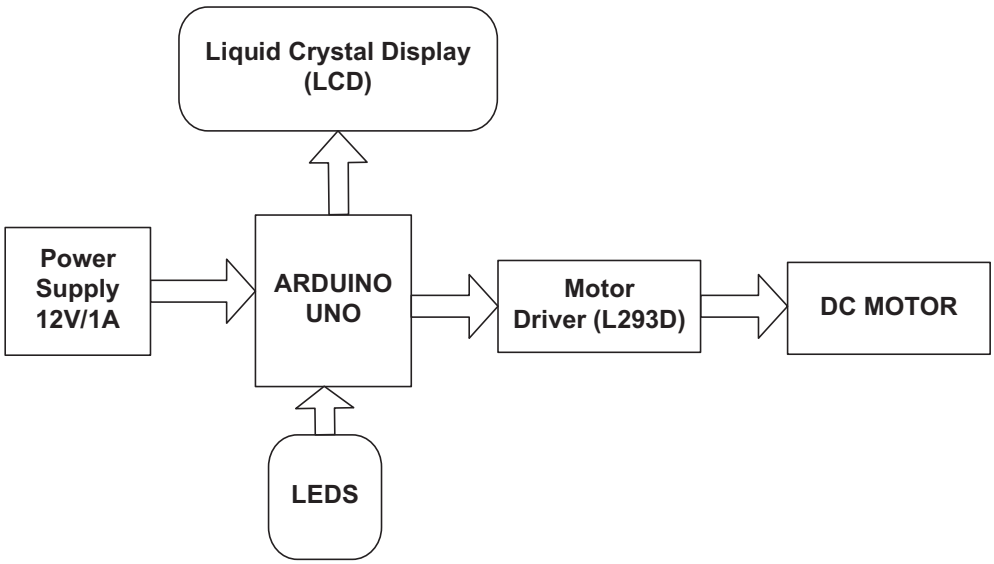

FIGURE 6.6
Block diagram of DC motor interfacing with Arduino.

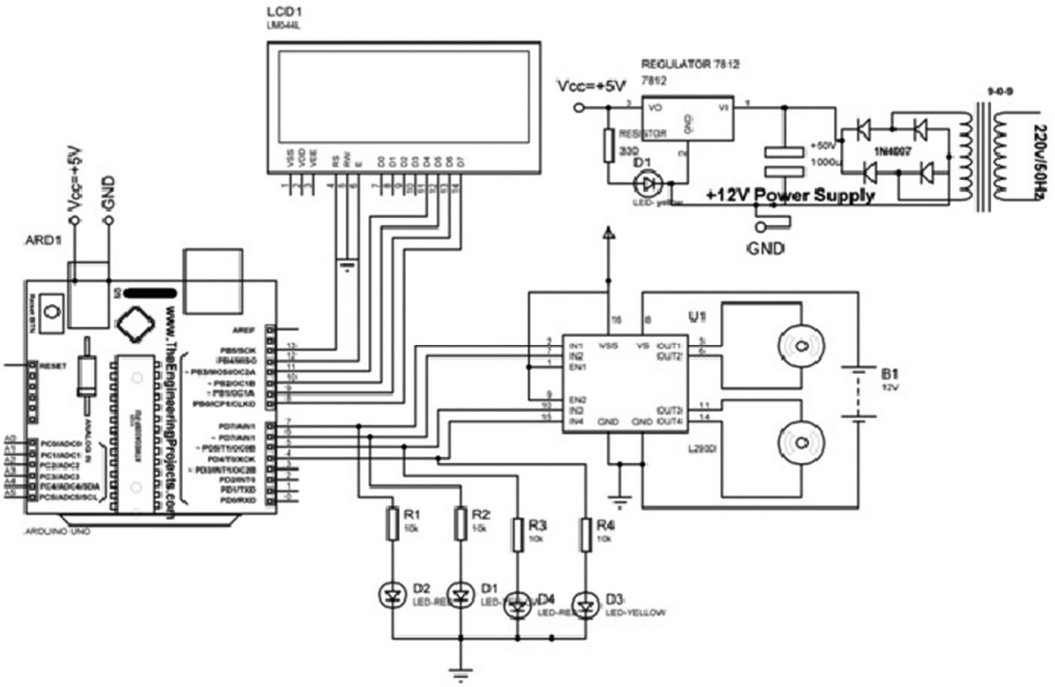

FIGURE 6.7
Circuit diagram of DC motor interfacing with Arduino.

L293D and DC motor connection

- Connect L293D pin 3 to +ve pin of DC motor1.
- Connect L293D pin 6 to −ve pin of DC motor1.
- Connect L293D pin 11 to +ve pin of DC motor2.
- Connect L293D pin 14 to +ve pin of DC motor2.

L293D connection

- Connect Arduino GND to pins 4, 5, 12, 13 of L293D.
- Connect Arduino +5 V to pins 1, 9, 16 of L293D.
- Connect Arduino pin 7 to pin 2 of L293D.
- Connect Arduino pin 6 to pin 7 of L293D.
- Connect Arduino pin 5 to pin 10 of L293D.
- Connect Arduino pin 4 to pin 15 of L293D.
- Connect L293D pin 8 to +ve of 12V battery

LED connection

- Connect Arduino pin 7 to anode of LED1.
- Connect Arduino pin 6 to anode of LED2.
- Connect Arduino pin 5 to anode of LED3.

- Connect Arduino pin 4 to anode of LED4.
- Connect cathode of all LEDs to ground.

LCD connection

- Connect Arduino digital pin 13 to RS pin(4) of LCD.
- Connect Arduino digital pin GND to RW pin(5) of LCD.
- Connect Arduino digital pin 12 to E pin(6) of LCD.
- Connect Arduino digital pin 11 to D4 pin(11) of LCD.
- Connect Arduino digital pin 10 to D5 pin(12) of LCD.
- Connect Arduino digital pin 9 to D6 pin(13) of LCD.
- Connect Arduino digital pin 8 to D7 pin(14) of LCD.

6.4.1.2 Sketch

```
#include <LiquidCrystal.h> // include library of LCD
LiquidCrystallcd(13, 12, 11, 10,9, 8); // attach LCD pin RS, E, D4, D5, D6,
   D7 to the given pins
int MPIN1= 7; // assign pin 7 as MPIN1
int MPIN2= 6; // assign pin 6 as MPIN2
int MPIN3= 5; // assign pin 5 as MPIN3
int MPIN4= 4; // assign pin 4 as MPIN4

void setup()
{
pinMode(MPIN1, OUTPUT); // make MPIN1 as an output
pinMode(MPIN2, OUTPUT); // make MPIN2 as an output
pinMode(MPIN3, OUTPUT); // make MPIN3 as an output
pinMode(MPIN4, OUTPUT); // make MPIN4 as an output
lcd.begin(20,4); // initialise LCD
lcd.setCursor(0, 0); // set cursor on LCD
lcd.print("DC Motor direction"); // print string on LCD
lcd.setCursor(0, 1); // set cursor on LCD
lcd.print("control system..."); // print string on LCD
delay(1000); // delay of 1000 mS
lcd.clear(); // clear the contents of LCD
}
void loop() // infinite loop
{
digitalWrite(MPIN1, HIGH); // make MPIN1 to HIGH
```

```
digitalWrite(MPIN2, LOW); // make MPIN2 to LOW
digitalWrite(MPIN3, HIGH); // make MPIN3 to HIGH
digitalWrite(MPIN4, LOW); // make MPIN4 to LOW
lcd.setCursor(0, 2); // set cursor on LCD
lcd.print("CLOCKWISE"); // print string on LCD
delay(2000); // delay of 2 sec
lcd.clear(); // clear the contents of LCD
digitalWrite(MPIN1, LOW); //make MPIN1 to LOW
digitalWrite(MPIN2, HIGH); //make MPIN2 to HIGH
digitalWrite(MPIN3, LOW); //make MPIN3 to LOW
digitalWrite(MPIN4, HIGH); //make MPIN4 to HIGH
lcd.setCursor(0, 2); // set cursor on LCD
lcd.print("ANTI-CLOCKWISE"); // print string on LCD
delay(2000); // delay of 2 Sec
lcd.clear(); // clear the contents of LCD
digitalWrite(MPIN1, LOW); // make MPIN1 to LOW
digitalWrite(MPIN2, LOW); //make MPIN2 to LOW
digitalWrite(MPIN3, HIGH); //make MPIN3 to HIGH
digitalWrite(MPIN4, LOW); //make MPIN4 to LOW
lcd.setCursor(0, 2); // set cursor on LCD
lcd.print("LEFT"); // print string on LCD
delay(2000); // delay of 2 Sec
lcd.clear(); // clear the contents of LCD
digitalWrite(MPIN1, HIGH); //make MPIN1 to HIGH
digitalWrite(MPIN2, LOW); //make MPIN2 to LOW
digitalWrite(MPIN3, LOW); //make MPIN3 to LOW
digitalWrite(MPIN4, LOW); //make MPIN4 to LOW
lcd.setCursor(0, 2); // set cursor on LCD
lcd.print("RIGHT"); // print string on LCD
delay(2000); // delay of 2 Sec
lcd.clear(); // clear the contents of LCD
  }
```

6.4.2 Servo Motor

A servo motor is a rotary actuator used for precise control of the angular position. It is comprised of a motor coupled with a sensor for

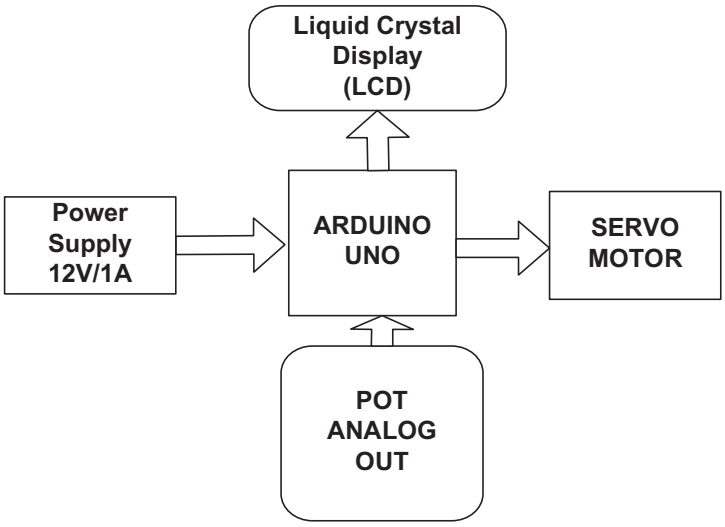

FIGURE 6.8
Block diagram to interface servo motor with Arduino.

position feedback. It also requires a servo drive. The drive uses the feedback sensor to control the rotary position of the motor precisely. This is called a closed-loop operation. The high torque standard servo motor with metal gears and 360° rotation can provide 11 kg/cm at 4.8 V, 13.5 kg/cm at 6 V, and 16 kg/cm at 7.2 V. Figure 6.8 shows the block diagram to interface the servo motor with Arduino. It is comprised of an Arduino Uno, a power supply, a liquid crystal display, a potentiometer (POT), and a servo motor. The system is designed to control the angle of the servo motor with the potentiometer.

6.4.2.1 Circuit Diagram

Connect the components as shown in Figure 6.9 to check the workings of a servo motor. Upload the program described in Section 6.4.2.2 and check the workings.

Servo connection

- Connect Arduino GND to GND pin of servo motor.
- Connect Arduino +5 V to "+" terminal of servo motor.
- Connect Arduino pin(3) to PWM pin of servo motor.

POT connection

- Connect Arduino GND to GND pin of POT.
- Connect Arduino +5 V to "+" terminal of POT.
- Connect Arduino A0 pin to data out pin of POT.

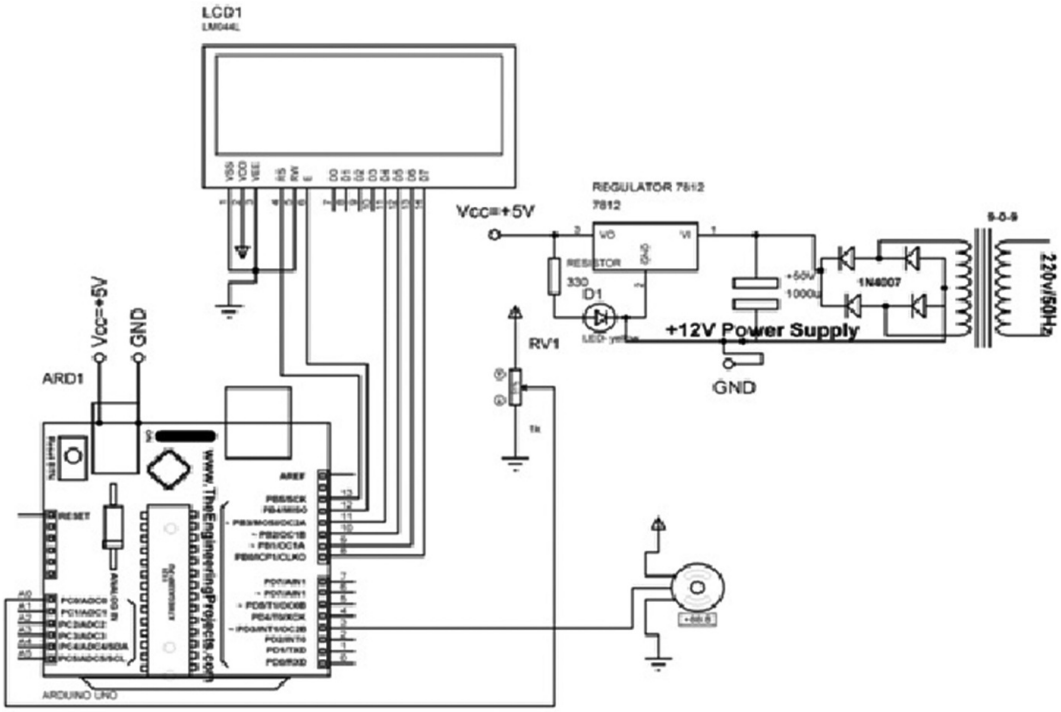

FIGURE 6.9
Circuit diagram to interface servo motor with Arduino.

LCD connection

- Connect Arduino digital pin (13) to RS pin(4) of LCD.
- Connect Arduino digital pin (GND) to RW pin(5) of LCD.
- Connect Arduino digital pin (12) to E pin(6) of LCD.
- Connect Arduino digital pin (11) to D4 pin(11) of LCD.
- Connect Arduino digital pin (10) to D5 pin(12) of LCD.
- Connect Arduino digital pin (9) to D6 pin(13) of LCD.
- Connect Arduino digital pin (8) to D7 pin(14) of LCD.

6.4.2.2 Sketch

```
#include <LiquidCrystal.h> // include library of LCD
LiquidCrystallcd(13, 12, 11, 10, 9, 8); // attach LCD pin RS,E,D4,D5,D6,D7
    to the given pins
Servo myservo; // create servo object to control a servo
int POT_PIN = A0; // analog pin used to connect the potentiometer
int POT_PIN_ADC_LEVEL; // variable to read the value from the
    analog pin
```

```
void setup()
{
myservo.attach(3); // attaches the servo on pin 9 to the servo object
lcd.begin(20,4); // initialise LCD
lcd.setCursor(0, 0); // set cursor on LCD
lcd.print("Servo ANALOG write "); // print string on LCD
lcd.setCursor(0, 1);// set cursor on LCD
lcd.print("system at LPU....");// print string on LCD
}
void loop()
{
POT_PIN_ADC_LEVEL = analogRead(POT_PIN); // reads POT value
   in the form of levels
POT_PIN_ADC_LEVEL = map(POT_PIN_ADC_LEVEL, 0, 1023, 0, 179);
   // map the value //between 0 to 180 degree for servo
myservo.write(POT_PIN_ADC_LEVEL); // sets the servo position
   according to the scaled value
lcd.setCursor(0, 2); // set cursor on LCD
lcd.print("ANGLE:"); // print string on LCD
lcd.print(POT_PIN_ADC_LEVEL); // print value on LCD
delay(15); // delay of 15 mSec
}
```