Arduino and Its Libraries
A Comprehensive Overview with Case Studies

Johar M. Ashfaque

April 8, 2025

Abstract

Arduino is an open-source electronics platform based on easy-to-use hardware and software. This document explores the Arduino ecosystem, focusing on the rich library infrastructure that simplifies development and enables powerful functionality for users of all levels. Libraries in Arduino encapsulate reusable code for controlling sensors, displays, motors, communication protocols, and more. This paper surveys common and advanced libraries and illustrates their use through real-world case studies.

1 Introduction to Arduino

Arduino is both a physical programmable circuit board (often referred to as a microcontroller) and a development environment for writing software that runs on it. Designed for makers, educators, and professionals alike, Arduino simplifies embedded system development through an accessible syntax and a vast ecosystem.

The Arduino IDE supports C/C++ and offers a simplified API for interacting with digital and analog I/O, serial communication, and timing. Libraries in Arduino further abstract complex functionalities into reusable modules.

2 Why Libraries Matter in Arduino

Libraries are collections of code that simplify complex tasks. Instead of writing low-level code to communicate with a sensor or perform data parsing, developers can import libraries that handle the intricacies internally.

Advantages include:

- Reusability and modularity
- Simplified syntax for complex tasks
- Faster prototyping
- Support for a wide variety of hardware components

3 Using Libraries in Arduino

To include a library in your Arduino sketch, you use the `#include` directive. Most libraries require initialization and configuration within the `setup()` function.

```
// Example: Using the Servo library
#include <Servo.h>

Servo myServo;

void setup() {
    myServo.attach(9); // attaches the servo to pin 9
}

void loop() {
    myServo.write(90); // sets the servo to 90 degrees
    delay(1000);
}
```

4 Popular Arduino Libraries

4.1 Servo Library

Controls hobby servo motors with simple commands.

- **Functions**: `attach()`, `write()`, `read()`
- **Applications**: Robotics, animatronics, automated systems

4.2 Wire Library

Handles I^2C communication.

```
#include <Wire.h>
Wire.begin();
Wire.requestFrom(8, 6); // request 6 bytes from slave device #8
```

4.3 SPI Library

Used for high-speed serial communication with peripherals like SD cards and displays.

```
#include <SPI.h>
SPI.beginTransaction(SPISettings(14000000, MSBFIRST, SPI_MODE0));
```

4.4 Adafruit Sensor Libraries

- `Adafruit_GFX` and `Adafruit_SSD1306` for OLED displays
- `Adafruit_BME280` for temperature, humidity, pressure
- `Adafruit_NeoPixel` for RGB LEDs

4.5 LiquidCrystal

For character LCDs.

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
lcd.begin(16, 2);
lcd.print("Hello, Arduino!");
```

4.6 SoftwareSerial

Enables serial communication on other digital pins.

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // RX, TX
```

4.7 EEPROM

Stores data in non-volatile memory.

```
#include <EEPROM.h>
EEPROM.write(0, 42);
int value = EEPROM.read(0);
```

4.8 MFRC522 (RFID)

Reads RFID tags using MFRC522 readers.

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 9
MFRC522 rfid(SS_PIN, RST_PIN);

void setup() {
    SPI.begin();
    rfid.PCD_Init();
    Serial.begin(9600);
}

void loop() {
```

```
    if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial())
        ↪ return;
    Serial.print("UID:");
    for (byte i = 0; i < rfid.uid.size; i++) {
        Serial.print(rfid.uid.uidByte[i], HEX);
    }
    Serial.println();
}
```

4.9 Keypad

Interfaces matrix keypads with ease.

```
#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 3;
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};

byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

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

void loop() {
  char key = keypad.getKey();
  if (key) {
    Serial.println(key);
  }
}
```

5 Advanced Libraries and Applications

5.1 FastLED

Efficiently controls LED strips.

5.2 PubSubClient

MQTT client for IoT systems.

5.3 WiFi and Ethernet Libraries

For web-based communication, especially on ESP32 and MKR boards.

5.4 RTClib

For interfacing with real-time clock modules.

5.5 AccelStepper

Advanced stepper motor control.

5.6 Firmata

Allows Arduino to be controlled by external software like Python or MATLAB.

6 Creating Your Own Arduino Library

You can create custom Arduino libraries using object-oriented principles. A typical library includes:

- A header file (.h) with function declarations
- A source file (.cpp) with implementations
- A `library.properties` file for metadata
- An `examples/` folder

7 Case Studies: Real-World Applications of Arduino Libraries

7.1 Case Study 1: Automated Plant Watering System

Objective: Automatically water plants based on soil moisture.

Libraries: `LiquidCrystal`, `EEPROM`

```
#include <LiquidCrystal.h>
#include <EEPROM.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
const int moisturePin = A0;
const int pumpPin = 7;
int threshold;

void setup() {
    pinMode(pumpPin, OUTPUT);
```

```
    lcd.begin(16, 2);
    threshold = EEPROM.read(0);
    if (threshold == 0xFF) threshold = 300;
}

void loop() {
    int moisture = analogRead(moisturePin);
    lcd.setCursor(0, 0);
    lcd.print("Moisture: ");
    lcd.print(moisture);

    if (moisture < threshold) {
        digitalWrite(pumpPin, HIGH);
        lcd.setCursor(0, 1);
        lcd.print("Watering...     ");
    } else {
        digitalWrite(pumpPin, LOW);
        lcd.setCursor(0, 1);
        lcd.print("Moist OK        ");
    }

    delay(1000);
}
```

7.2 Case Study 2: Smart Home Energy Monitor

Objective: Log power consumption to MQTT.

Libraries: `WiFiNINA`, `PubSubClient`

```
#include <WiFiNINA.h>
#include <PubSubClient.h>

WiFiClient wifiClient;
PubSubClient client(wifiClient);

const char* ssid = "YourSSID";
const char* password = "YourPassword";
const char* mqttServer = "192.168.1.10";
const int sensorPin = A1;

void setup() {
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) delay(500);
    client.setServer(mqttServer, 1883);
}

void loop() {
    if (!client.connected()) {
        client.connect("ArduinoEnergyMonitor");
    }
    int raw = analogRead(sensorPin);
    float current = (raw - 512) * (5.0 / 1023.0) / 0.066;
```

```
    char msg[50];
    dtostrf(current, 4, 2, msg);
    client.publish("home/energy", msg);
    delay(2000);
}
```

7.3 Case Study 3: LED Art Installation with Animations

Objective: Animate a WS2812B LED strip.

Library: `FastLED`

```
#include <FastLED.h>

#define NUM_LEDS 120
#define DATA_PIN 6
CRGB leds[NUM_LEDS];

void setup() {
    FastLED.addLeds<WS2812B, DATA_PIN, GRB>(leds, NUM_LEDS);
    FastLED.setBrightness(100);
}

void loop() {
    static uint8_t hue = 0;
    for (int i = 0; i < NUM_LEDS; i++) {
        leds[i] = CHSV(hue + i * 10, 255, 255);
    }
    FastLED.show();
    hue++;
    delay(50);
}
```

7.4 Case Study 4: Data Logging Weather Station

Objective: Store environmental data to SD card.

Libraries: `Adafruit_BME280`, `SD`, `SPI`

```
#include <Adafruit_BME280.h>
#include <SD.h>
#include <SPI.h>

Adafruit_BME280 bme;
File dataFile;

void setup() {
    bme.begin(0x76);
    SD.begin(10);
}

void loop() {
    float t = bme.readTemperature();
```

```
    float h = bme.readHumidity();
    float p = bme.readPressure() / 100.0F;

    dataFile = SD.open("weather.csv", FILE_WRITE);
    if (dataFile) {
        dataFile.print(t); dataFile.print(",");
        dataFile.print(h); dataFile.print(",");
        dataFile.println(p);
        dataFile.close();
    }
    delay(5000);
}
```

7.5 Case Study 5: Secure Entry System with RFID and Keypad

Objective: Grant access only when a valid RFID tag and correct keypad code are entered.

Libraries: `MFRC522`, `Keypad`

```
#include <SPI.h>
#include <MFRC522.h>
#include <Keypad.h>

#define SS_PIN 10
#define RST_PIN 9
MFRC522 rfid(SS_PIN, RST_PIN);

const byte ROWS = 4, COLS = 3;
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

const String correctUID = "DEADBEEF"; // Replace with your tag UID
const String correctCode = "1234";

void setup() {
    SPI.begin();
    rfid.PCD_Init();
    Serial.begin(9600);
}

void loop() {
    if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
        String uid = "";
        for (byte i = 0; i < rfid.uid.size; i++) {
            uid += String(rfid.uid.uidByte[i], HEX);
        }
```

```
        Serial.println("RFID scanned. Enter PIN:");
        String input = "";
        while (input.length() < 4) {
            char key = keypad.getKey();
            if (key) {
                input += key;
                Serial.print("*");
            }
        }

        if (uid == correctUID && input == correctCode) {
            Serial.println("\nAccess Granted");
        } else {
            Serial.println("\nAccess Denied");
        }

        delay(2000);
    }
}
```

Summary

These case studies reflect the versatility of Arduino and its libraries across agriculture, smart homes, creative arts, access control, and environmental science. Libraries enable developers to build robust, scalable systems while focusing on innovation rather than infrastructure.