Arduino and Processing 4 Case Studies

Johar M. Ashfaque

April 9, 2025

Contents

1. Introduction

1.1. Arduino (.ino Files)

Arduino sketches are saved with the `.ino` extension. They contain C/C++ code that interfaces with hardware via the Arduino platform. These files typically consist of two main functions:

- `setup()` – runs once
- `loop()` – runs continuously

1.2. Processing (.pde Files)

Processing is a creative coding environment based on Java, used primarily for visual output. Sketches are saved as `.pde` files and can receive serial data from Arduino to visualize in real-time.

This document explores four case studies using Arduino + Processing, demonstrating how INO and PDE files complement each other in interactive projects.

2. Using the Serial Library in Processing 4

The `processing.serial.*` package provides access to serial ports for communication with Arduino. To use it:

1. Import the library: `import processing.serial.*;`
2. Initialize the port:

```
Serial myPort;

void setup() {
  myPort = new Serial(this, Serial.list()[0], 9600);
}
```

3. Read data using:

```
void serialEvent(Serial p) {
  String data = trim(p.readStringUntil(’\n’));
  float value = float(data);
}
```

Ensure the baud rate matches the Arduino sketch. The `Serial.list()` function retrieves all available serial ports. Use it to identify the port connected to Arduino.

3. Case Study 1: Voltmeter

3.1. Arduino Code (.ino)

Listing 1: voltmeter.ino

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

void loop() {
  int sensorValue = analogRead(A0);
  float voltage = sensorValue * (5.0 / 1023.0);
  Serial.println(voltage);
  delay(200);
}
```

3.2. Processing Code (.pde)

Listing 2: voltmeter.pde

```
import processing.serial.*;

Serial myPort;
float voltage;

void setup() {
  size(400, 200);
  myPort = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
  background(255);
  fill(0);
  textSize(32);
  text("Voltage:␣" + nf(voltage, 1, 2) + "␣V", 50, 100);
}

void serialEvent(Serial p) {
  voltage = float(trim(p.readStringUntil(’\n’)));
}
```

3.3. Schematic

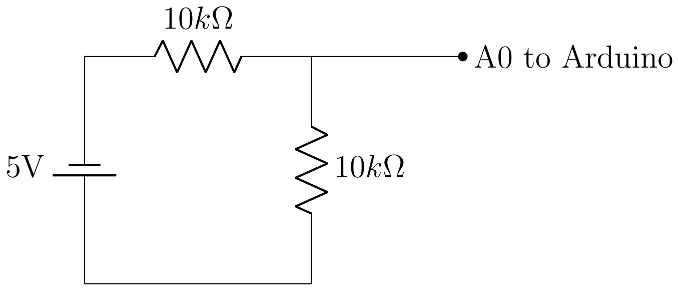

Figure 1: Voltage divider circuit for voltmeter.

4. Case Study 2: Oscilloscope

4.1. Arduino Code (.ino)

Listing 3: oscilloscope.ino

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

void loop() {
  int val = analogRead(A0);
  Serial.println(val);
}
```

4.2. Processing Code (.pde)

Listing 4: oscilloscope.pde

```
import processing.serial.*;

Serial myPort;
int[] values = new int[500];
int index = 0;

void setup() {
  size(800, 400);
  myPort = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
  background(0);
  stroke(0, 255, 0);
  for (int i = 1; i < values.length; i++) {
    line(i-1, height - values[i-1], i, height - values[i]);
  }
}

void serialEvent(Serial p) {
  String data = trim(p.readStringUntil('\n'));
  if (data != null) {
    values[index] = int(data) / 4; // Scale to fit screen
    index = (index + 1) % values.length;
  }
}
```

4.3. Schematic

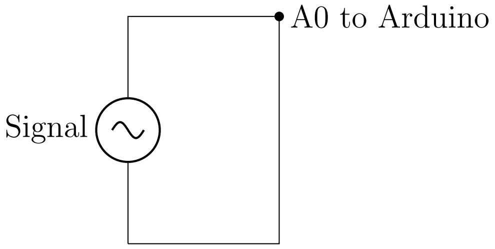

Figure 2: Basic oscilloscope connection using signal input to Arduino A0.

5. Case Study 3: Snake Game (Processing Only)

Listing 5: snake.pde

```
int scl = 20;
int cols, rows;
ArrayList<PVector> snake = new ArrayList<PVector>();
PVector dir;
PVector food;

void setup() {
  size(600, 600);
  cols = width / scl;
  rows = height / scl;
  snake.add(new PVector(5, 5));
  dir = new PVector(1, 0);
  placeFood();
  frameRate(10);
}

void draw() {
  background(51);
  updateSnake();
  drawSnake();
  drawFood();
  checkCollision();
}

void keyPressed() {
  if (keyCode == UP && dir.y == 0) dir = new PVector(0, -1);
  if (keyCode == DOWN && dir.y == 0) dir = new PVector(0, 1);
  if (keyCode == LEFT && dir.x == 0) dir = new PVector(-1, 0);
  if (keyCode == RIGHT && dir.x == 0) dir = new PVector(1, 0);
}

void updateSnake() {
  PVector head = snake.get(snake.size() - 1).copy();
  head.add(dir);
  snake.add(head);
  if (!head.equals(food)) {
    snake.remove(0);
  } else {
    placeFood();
  }
}

void drawSnake() {
  fill(0, 255, 0);
  for (PVector s : snake) {
    rect(s.x * scl, s.y * scl, scl, scl);
  }
}

void drawFood() {
  fill(255, 0, 0);
  rect(food.x * scl, food.y * scl, scl, scl);
}
```

```
void placeFood() {
  food = new PVector(int(random(cols)), int(random(rows)));
}

void checkCollision() {
  PVector head = snake.get(snake.size() - 1);
  for (int i = 0; i < snake.size() - 1; i++) {
    if (head.equals(snake.get(i))) {
      snake.clear();
      snake.add(new PVector(5, 5));
      dir = new PVector(1, 0);
    }
  }
}
```

6. Case Study 4: Ping Pong Game (Processing Only)

Listing 6: pingpong.pde

```
int ballX = 300, ballY = 200;
int ballSpeedX = 4, ballSpeedY = 4;
int paddleWidth = 10, paddleHeight = 80;
int leftPaddleY = 150, rightPaddleY = 150;
int paddleSpeed = 5;

void setup() {
  size(600, 400);
}

void draw() {
  background(0);
  moveBall();
  drawBall();
  drawPaddles();
  checkCollision();
}

void moveBall() {
  ballX += ballSpeedX;
  ballY += ballSpeedY;
  if (ballY <= 0 || ballY >= height) ballSpeedY *= -1;
}

void drawBall() {
  fill(255);
  ellipse(ballX, ballY, 20, 20);
}

void drawPaddles() {
  fill(255);
  rect(10, leftPaddleY, paddleWidth, paddleHeight);
  rect(width - 20, rightPaddleY, paddleWidth, paddleHeight);
}

void keyPressed() {
  if (key == ’w’) leftPaddleY -= paddleSpeed;
  if (key == ’s’) leftPaddleY += paddleSpeed;
  if (keyCode == UP) rightPaddleY -= paddleSpeed;
  if (keyCode == DOWN) rightPaddleY += paddleSpeed;
}

void checkCollision() {
  if (ballX <= 20 && ballY > leftPaddleY && ballY < leftPaddleY +
     paddleHeight) {
    ballSpeedX *= -1;
  }
  if (ballX >= width - 30 && ballY > rightPaddleY && ballY <
     rightPaddleY + paddleHeight) {
    ballSpeedX *= -1;
  }
}
```