PROCESSING FINAL CODE

```
// Import the Serial library for communication with external devices.
import processing.serial.*;

// Declare global variables.
Serial port;
static String val;
int sensorVal0 = 0;
int sensorVal1 = 0;
int sensorVal2 = 0;
int numFireflies = 200;

// Create an array to store instances of the Firefly class.
Firefly[] fireflies = new Firefly[numFireflies];

// Setup function runs once at the beginning.
void setup() {
 // Set the canvas size and frame rate.
 size(1090, 720);
 frameRate(60);
 fullScreen();
 // Establish a connection to the serial port (adjust "COM5" as needed) at a baud rate of 9600.
 port = new Serial(this, "COM4", 9600);

 // Initialize the array of fireflies.
 for (int i = 0; i < numFireflies; i++) {
  fireflies[i] = new Firefly();
 }
```

```
}

// Draw function runs continuously in a loop.
void draw() {
  // Check if there is data available on the serial port.
  if (port.available() > 0) {
    // Read the data from the serial port until a newline character is encountered.
    val = port.readStringUntil('\n');
    // Print the received data to the console for debugging.
    println("Received data: " + val);

    // Split the received data into three parts based on the "-" delimiter.
    String[] sensorValues = split(val, '-');

    // Ensure that there are three parts in the split data.
    if (sensorValues.length == 3) {
      try {
        // Attempt to convert each part to an integer and store them in sensorVal0, sensorVal1, sensorVal2.
        sensorVal0 = Integer.parseInt(sensorValues[0].trim());
        sensorVal1 = Integer.parseInt(sensorValues[1].trim());
        sensorVal2 = Integer.parseInt(sensorValues[2].trim());

        // Print the converted sensor values to the console for debugging.
        println("Converted sensor values: " + sensorVal0 + ", " + sensorVal1 + ", " + sensorVal2);
      }
      catch (NumberFormatException e) {
        // Handle any exceptions that may occur during the conversion.
        println("Error converting to integer");
      }
```

```
  }
  else {
    // Print a warning if the data does not have the expected format.
    println("Invalid data format");
  }
 }

 // Set the background color to black.
 background(0);

 // Iterate through the array of fireflies.
 for (int i = 0; i < numFireflies; i++) {
  // Move and display each firefly if it is alive.
  fireflies[i].move();
  fireflies[i].display();
 }
}


// Firefly class definition.
class Firefly {
 // Declare instance variables for the firefly's position, movement, and status.
 float x, y;
 float angle;
 float speed;

 // Constructor function called when a new Firefly object is created.
 Firefly() {
  // Call the respawn function to initialize the firefly's properties.
```

```
  respawn();
 }

 // Respawn function resets the firefly's position and status.
 void respawn() {
  // Set the firefly's position to a random location within the canvas.
  x = random(width);
  y = random(height);
  // Set the firefly's movement properties to random values.
  angle =  map(sensorVal1,  0, 1023, 0, TWO_PI);;

  speed = random(1, 3);  // speed
 }

// Move function updates the firefly's position based on its movement properties.
void move() {
 float scatteringSpeed = 5; // Adjust this value for scattering effect
 float slowSpeed = 0.5; // Adjust this value for slow movement effect

 // Check if any sensor value is outside the desired range
 if (sensorVal0 > 200 || sensorVal0 < 0) {
  // If any sensor value is outside the range, scatter by increasing the speed
  x += cos(angle) * scatteringSpeed;
  y += sin(angle)  * scatteringSpeed;
 } else {
  // If sensor value is within the range, move slowly
  x += cos(angle) * slowSpeed;
  y += sin(angle)  * slowSpeed;
 }
```

```
  // Check if the firefly is outside the canvas boundaries.
  if (x < 0 || x > width || y < 0 || y > height) {
    // If so, call the respawn function to reset the firefly's properties.
    respawn();
  }
}
```

```
  // Display function draws the firefly on the canvas.
  void display() {
    // Draw a glowing effect around the firefly.
    for (int i = 3; i > 0; i--) {
      float glowSize = i * 6;
      int glowAlpha = 20 - i * 5;
      fill(255, 255, 0, glowAlpha);
      ellipse(x, y, 20 + glowSize, 20 + glowSize);
    }

    // Draw the central part of the firefly.
    noStroke();
    fill(255, 255, 0);
    ellipse(x, y, 2, 2);
  }
}
```