为了让你的 Python 绘图程序 (Plotter_MSM.py) 能够正确实时绘制 Arduino (profcode.ino) 的数据，我们需要解决一个核心逻辑冲突：

当前的问题：

1. **Arduino** 输出格式错误：profcode.ino 在 PID 控制循环中输出的是调试文本（如 "gemessener Spulenstrom ist ..."）[1]，而 Python 程序期望的是纯逗号分隔的数值（CSV格式，如 1.0, 2.5, 0.1）。

2. 数据发送时机错误：profcode.ino 目前只在 2秒的控制循环结束之后才发送 Python 所需的 "Kraft, Spulestrom, Dehnung" 标题和最终数据 [2]。这意味着 Python 绘图程序在 PID 调节过程中收不到任何数据，只能收到结束后的一个点。

你需要修改 **Arduino** 代码，使其在循环过程中实时发送格式化的数据。

---

## 第一步：修改 Arduino 代码 (profcode.ino)

请用下面的 loop 函数替换你原有的 loop 函数。

主要修改点：

1. 将 "Kraft, Spulestrom, Dehnung" 标题移到循环开始前发送。
2. 将激光读取和应变计算 (dehnungMSM) 移到 while 循环内部，以便实时监测。
3. 将 Serial.print 改为输出 数值, 数值, 数值 的格式，去掉了多余的文字描述。

C++

```
void loop() {
 String befehl = Serial.readStringUntil('\n');
 befehl.trim();

 // 接收 "Neu" 命令
 if (status == Normal && befehl == "Neu") {
   i = 0;
   sollStromMSM = 0.5;
   sollStromLA = 0.5;
 }

 if (status == Normal && i == 0) {
```

```
  // 1. 发送 Python 绘图所需的"触发关键词"
  Serial.println("Kraft, Spulestrom, Dehnung");

  // PI-Regler 参数
  long sampleTime_ms = 10;
  unsigned long duration = 2000;
  unsigned long startTime = millis();
  unsigned long lastSampleTime = millis();

  // 进入 2秒 的控制循环
  while (millis() - startTime < duration) {
    unsigned long now = millis();

    if (now - lastSampleTime >= sampleTime_ms) {
      lastSampleTime = now;

      // --- MSM 电流控制 (PI) ---
      istStromSP = (analogRead(analogStromSP) * 5.0) / (1023.0 * 1.5);
      float error = sollStromMSM - istStromSP;
      integralMSM += (KiMSM * error);
      integralMSM = constrain(integralMSM, -80, 80);
      float pwm_out = (KpMSM * error) + integralMSM;
      pwm_out = constrain(pwm_out, 0.0, 100.0);
      AusgabeStromMSM.pulse_perc(pwm_out);

      // --- Lastaktor 电流控制 (PI) ---
      istStromLA = (analogRead(analogStromLA) * 5.0) / (1023.0 * 1.5);
      float errorLA = sollStromLA - istStromLA;
      integralLA += (KiLA * errorLA);
      integralLA = constrain(integralLA, -70, 70);
      float pwm_out_LA = (KpLA * errorLA) + integralLA;
      pwm_out_LA = constrain(pwm_out_LA, 0.0, 100.0);
      AusgabeStromLA.pulse_perc(pwm_out_LA);

      // --- 2. 实时计算应变 (Dehnung) ---
      // 为了绘图，必须在循环内读取
      auslenkung = (analogRead(inLaser) * 5 / 1023.0) * 7.7 / 3.98;
      dehnungMSM = abs(((auslenkung - nullReferenzLaser) / 15) * 100);

      // --- 3. 发送数据给 Python (CSV格式) ---
      // 顺序对应 Python 代码中的: k (Kraft), s (Strom), d (Dehnung)
      // 这里用 istStromLA 代表 Kraft (因为通常电流与力成正比)，你可以根据需要改成 sollKraftLA
      Serial.print(istStromLA);
```

```
      Serial.print(",");
      Serial.print(istStromSP);
      Serial.print(",");
      Serial.println(dehnungMSM);
    }
   }
   i = 1; // 标记测量结束
 }

 // 重置输出
 if (status == Normal && i == 1) {
   AusgabeStromMSM.pulse_perc(0.0f);
   AusgabeStromLA.pulse_perc(0.0f);
 }
}
```

---

## 第二步：检查 Python 代码 (Plotter_MSM.py)

你的 Python 代码逻辑基本是正确的，但需要确认以下几点配置：

1. COM 端口匹配：
   检查 Python 代码第 8 行：
   ```
   Python
   PORT = 'COM8'
   ```
   请确保你的 Arduino 确实连接在 COM8。如果不是，请在设备管理器中查看并修改此行。
2. 数据解析逻辑 (无需修改，但需理解)：
   你的 Python 代码在第 62-64 行期望收到 3 个浮点数：
   ```
   Python
   k = float(parts[0]) # 对应 Arduino 的 istStromLA
   s = float(parts[1]) # 对应 Arduino 的 istStromSP
   d = float(parts[2]) # 对应 Arduino 的 dehnungMSM
   ```
   这也是为什么我在上面的 Arduino 修改中严格按照这个顺序输出的原因。

---

## 第三步：操作流程

按照以下步骤运行系统：

1. 上传代码：将修改后的 profcode.ino 上传到 Arduino 开发板。
2. 关闭 **Arduino IDE** 串口监视器：这是一个常见错误点。必须关闭 Arduino IDE 的 Serial Monitor，否则 Python 脚本会因为端口被占用而报错（Access Denied）。

3. 运行 **Python** 脚本：运行 Plotter_MSM.py。
4. 输入命令：
    - Python 终端会显示：Gib 'Neu' ein, um die Messung zu starten:
    - 输入 Neu 并按回车。
5. 观察绘图：
    - Python 会通过串口发送 "Neu" 给 Arduino。
    - Arduino 收到后，先打印标题，然后开始 2 秒的 PID 循环。
    - 你会看到 matplotlib 窗口开始实时绘制 Kraft (蓝色)、Spulestrom (绿色) 和 Dehnung (红色) 的曲线。

## 总结

核心改动在于让 Arduino 在 while 循环内部，以 数值1, 数值2, 数值3 的格式打印数据，而不是打印文字描述。这样 Python 才能抓取并绘图。

是否需要我为你解释 Python 代码中 update 函数是如何利用双轴 (ax1 和 ax2) 同时绘制电流和应变的？