Micro: Lock

作者：刘泊岩，杨一帆

摘要：

一、 选题及创意介绍

我们本次的 C1 编程的选题是"Micro: Lock"，因为我们是利用 Micro: bit v2 做了一个锁。具体而言，因为 Micro: bit v2 的传感器非常之多，从而可以感受到外界输入的自由度也非常之多，于是我们便在这些可用的输入手段中挑出了很适合作为密码的一些输入（离散程度高、输入方便），也即：A、B 按键的按压输入；LOGO 的触摸输入；PIN0、1、2 的触摸输入；microphone 接收到的声音输入。通过对上述各种输入次数和种类的组合，我们便可以生出无穷无尽的密码组合。

二、 设计方案

为了实现一个锁的功能，我们首先需要实现并维护几个状态：

① 平常态(A, at rest)，在此状态下，对 Micro: Lock 进行任何操作都不会触发解锁。在此状态下，可以通过触摸 LOGO 进入输入态。

②输入态(I, inputting)，在此状态下，可以利用 Micro: Lock 的各种传感器来实现密码的输入，当输入完成时，可以通过触摸 LOGO 进行密码校验：

2.1 如果输入的密码正确，则 Micro: Lock 所存储的秘密信息将在屏幕上进行展示。展示过后，将进入解锁态

2.2 如果输入的密码错误，则 Micro: Lock 回到平常态

③解锁态(U, unlocked)，在此状态下，如果触摸 LOGO 则 Micro: Lock 会回到平常态。如果发出足够大的声音，达到 microphone 接收音量的一个阈值，则会进入重设态（"发出足够大的声音"这个设计是为了安全性考虑的，可以防止知道密码的坏人在我们的眼皮子底下修改密码）。

④重设态(R, repassword)，在此状态下，可以利用 Micro: Lock 的各种传感器来实现密码的输入，当输入完成时，可以通过触摸 LOGO 进行密码的重设。重设过后 Micro: Lock 将进入平常态

三、 实现方案及代码分析

为了实现上面的功能，我们维护三个变量：password、inputcache 和 status。在输入态或者重设态的时候，我们依据 Micro: Lock 的传感器输入，来分别在 inputcache 或 password 的末尾插入相应的字符，具体的字符如下：

按压 A：+="A"
按压 B：+="B"
PIN0：+="0"
PIN1：+="1"
PIN2：+="2"

于是在输入态结束时，我们只需要比对 password 和 inputcache 就可以知道输入的密码是否

正确

同时我们通过 status 变量来标识当前的状态，从而对各种输入做出回应。

以下是 GPT-4 为我们的代码给出的代码分析(源代码见同文件夹 python 文件):

代码分析：

1. 导入所需库：
首先，代码导入了 microbit 库，speech 库和 music 库，分别用于操作 micro:bit 的硬件、语音合成和播放音乐。

2. 全局变量定义：
接着，定义了四个全局变量：inputcache、password、status 和 message。其中 inputcache 用于临时存储输入的密码，password 存储设定的正确密码，status 代表当前 micro:bit 的状态（atrest、inputing、unlocked 和 resetpassword），message 用于存储解锁后显示的信息。

3. 定义 check() 函数：
check() 函数用于检查输入的密码是否正确。当 inputcache 与 password 相等时，会播放一段音乐、显示一段图片并滚动显示预设的 message。此外，函数还会根据密码检查结果修改状态变量 status。

4. 主循环：
在主循环中，程序根据当前的状态 status 对不同的输入进行响应。具体来说：
- 当 A 键被按下时，如果状态为 "inputing" 或 "resetpassword"，则在 inputcache 或 password 末尾添加 "A"。
- 当 B 键被按下时，如果状态为 "inputing" 或 "resetpassword"，则在 inputcache 或 password 末尾添加 "B"。
- 当 pin0、pin1 或 pin2 被触摸时，如果状态为 "inputing" 或 "resetpassword"，则在 inputcache 或 password 末尾添加对应的 "0"、"1" 或 "2"。

- 当 pin_logo 被触摸时，程序会根据当前的状态进行切换，例如从 "atrest" 切换到 "inputing"，或者从 "unlocked" 切换回 "atrest"。

- 当状态为 "unlocked" 且麦克风检测到大声音时，程序会清空密码并切换到 "resetpassword" 状态。

通过以上逻辑，程序实现了一个具有多个状态的 micro:bit 锁。用户可以输入密码进行解锁，并在解锁状态下通过大声音触发密码重设功能。

四、 后续工作展望

(GPT-4)对于 Micro: Lock 项目的后续工作展望，以下是几点可能的拓展方向：
1. 增加密码复杂度：当前的密码只包括"A"、"B"和"0"-"2"这几个字符，可以考虑引入更多的字符组合，例如使用加速度计的数据，或者基于光感应器的输入，以增加密码的复杂性。
2. 加入错误输入限制：可以考虑在设备上加入错误输入限制，例如在连续三次或五次密码

输入错误后，设备将暂时锁定一段时间，这样能增加设备的安全性。
3. 增加用户反馈：可以使用更多的音乐或 LED 显示模式，来增强用户的交互体验，例如，当密码输入错误时，可以播放不同的音乐和显示不同的 LED 图案。
4. 多用户支持：考虑增加多用户支持，每个用户可以有各自的密码和信息，这样就可以将一个 Micro: Lock 设备用于多人使用。
5. 加入无线通信功能：使用 micro:bit 的无线通信功能，可以实现远程控制和监控。例如，当密码输入错误过多时，可以向管理员发送通知；或者允许远程修改密码和信息。
6. 密码加密存储：当前的设计中，密码是明文存储的，这在一定程度上存在安全风险。可以考虑在存储和验证密码时引入加密算法，以提高安全性。
7. 优化电池使用：可以考虑在无操作时进入低功耗模式，以延长电池寿命。
8. 设计更优雅的外壳：可以设计和打印一些外壳，让 Micro: Lock 设备看起来更像一个真正的锁。
以上是一些可能的拓展和改进方向，这将使 Micro: Lock 设备更实用，更安全，也更有趣。

五、 小组分工合作

1. 刘泊岩:
- 项目管理：负责项目的整体管理和协调，确保项目按照计划进行。
- 硬件操作：负责实际操作 micro:bit 设备，测试各个功能是否按照预期工作。
- 代码编写：负责编写 Micro: Lock 的主要代码，包括密码输入、密码检查以及状态切换等功能。
- 文档编写：负责编写项目的说明文档，以及项目的总结和展望。

2. 杨一帆:
- 需求分析：负责理解和分析项目的需求，明确项目的目标。
- 测试：在代码编写完毕后，负责进行功能测试，确保所有功能都能正确工作。
- 代码调试：在编写的代码出现问题时，负责找出问题所在并进行修复。
- 用户交互设计：负责设计用户与 Micro: Lock 交互的方式，包括密码输入方式、状态显示方式以及音乐反馈等。
- Poster 制作：在项目的其它内容完成之后，进行 Micro: Lock 的 Poster 制作