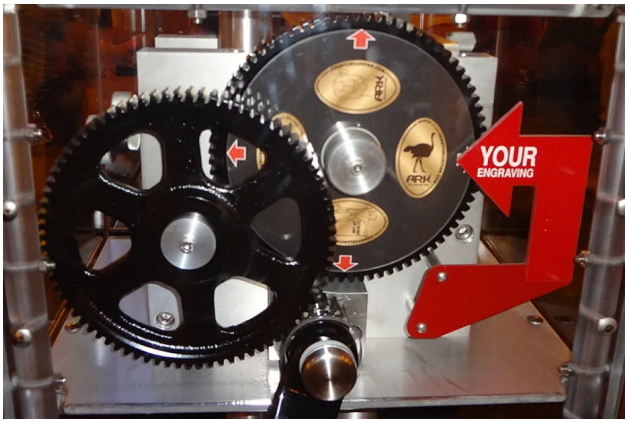


压币机程序设计

基于树莓派的步进电机图形化程序控制

2021 年 7 月 22 日

单位: 天津英赛迪科技有限公司

姓名: 谢远峰

时间: 2021 年 7 月 22 日——2021 年 8 月 6 日

摘要

压币机利用机械挤压的方式将硬币进行二次加工，形成一个细长硬币。新硬币相对于传统硬币被压平或拉伸，并拥有新的设计压花。此类硬币常被用作纪念品。在旅游园区中常见到压币机。

程序设计需求描述

1. 基于树莓派平台 + 可触摸屏幕（基于 Python-tkinter 库的图形化可执行界面程序）
2. 进入程序主界面，显示印花、初始化、开始、使用说明选项
3. 印花图案的选择（程序主页：显示四个图案，代表四种印花格式）
4. 四个印花图案默认情况下为未选择模式，图案底部默认设置按钮颜色为红色 (不选择)
5. 点选印花图案后，按钮颜色由红色转换为绿色
6. 用户进行纪念币打印之前，可点击使用说明进行程序使用流程的查看
7. 选择印花图案后，点击初始化按钮，完成电机初始化
8. 初始化完成后，出现初始化完成窗口，用户进行确认
9. 用户确认初始化完毕后，用户点击开始按钮，程序驱动电机开始移动
10. 打印完成后，出现打印完成，提醒用户拿取纪念币的窗口，等待用户确认
11. 用户点击确认后，自动返回到程序主界面

程序流程介绍

程序常规流程图说明

1. 用户点击目标可执行程序，进入程序主页面
2. 用户点击使用说明，查看程序执行方式 (选)
3. 用户点击任一印花图案，印花图案下方按钮由红转绿
4. 用户点击初始化按键，程序驱动电机进行初始化
5. 程序返回初始化成功窗口，用户点击确认
6. 用户点击开始按键，程序驱动电机进行硬币的印花压制
7. 程序返回硬币制作成功的窗口，用户点击确认
8. 用户拿取已制作好的纪念币
9. 程序返回至默认初始化状态

异常分支说明

1. 用户进入程序主页面，关闭
2. 用户未查看程序使用说明，直接进行程序初始化
3. 用户选择印花图案后，未进行进一步操作
4. 程序初始化失败
5. 用户进行初始化后，未进行进一步操作
6. 程序运行失败
7. 程序初始化过程中关闭程序
8. 程序运行过程中关闭程序

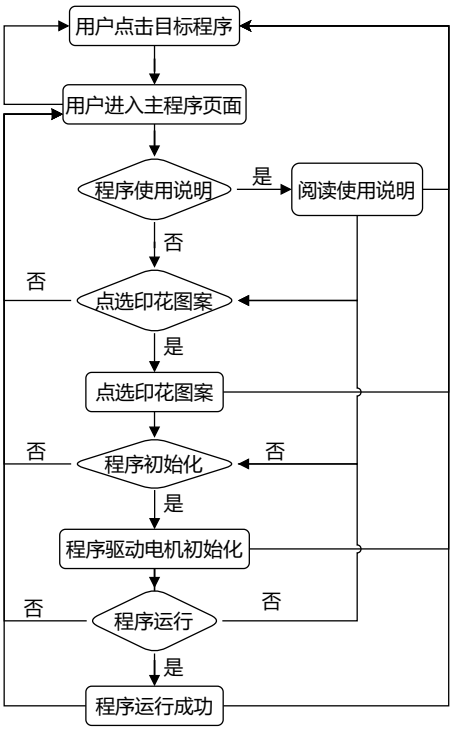
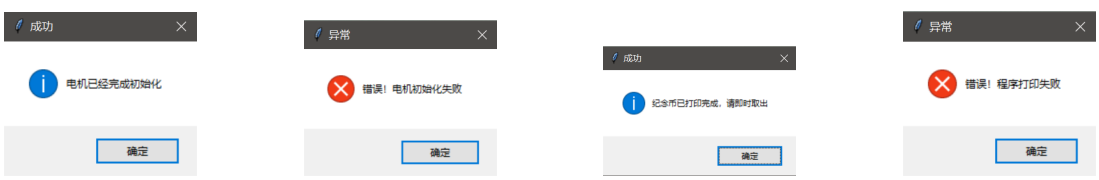


图 1: 程序运行流程图



(a) 初始化成功 (b) 初始化异常 (c) 程序打印成功 (d) 程序打印异常

图 2: 程序提示样例

附录

A：针脚说明图

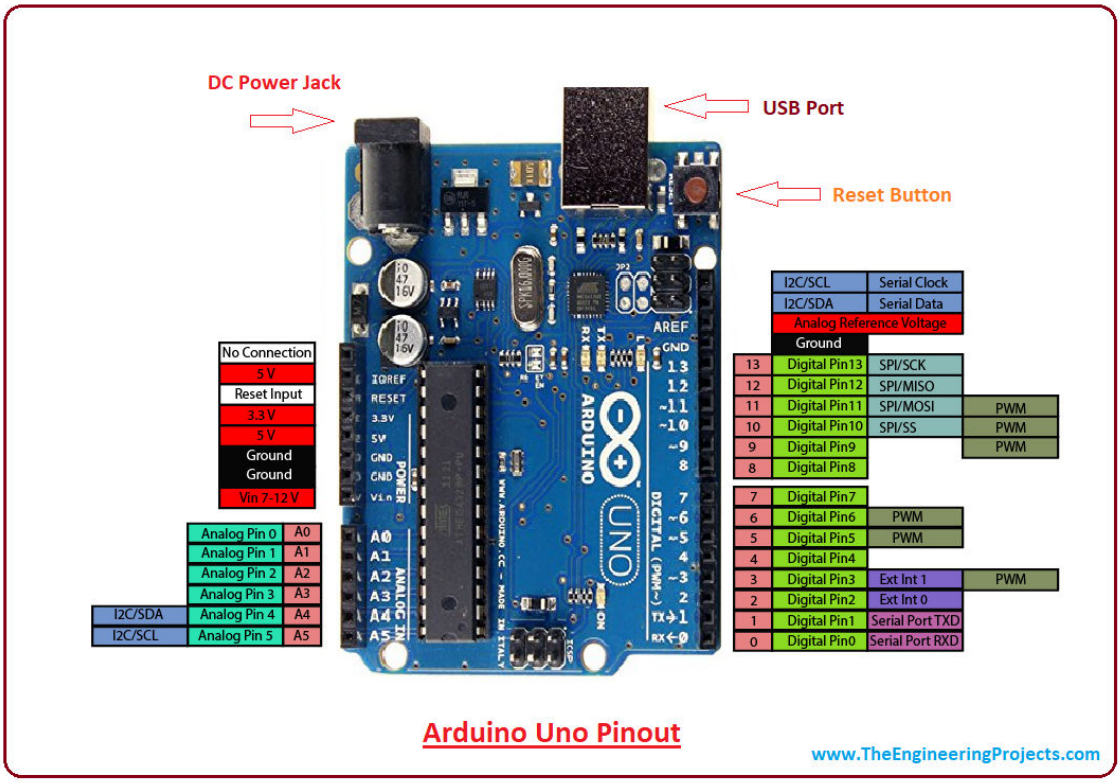


图 3: Arduino 针脚

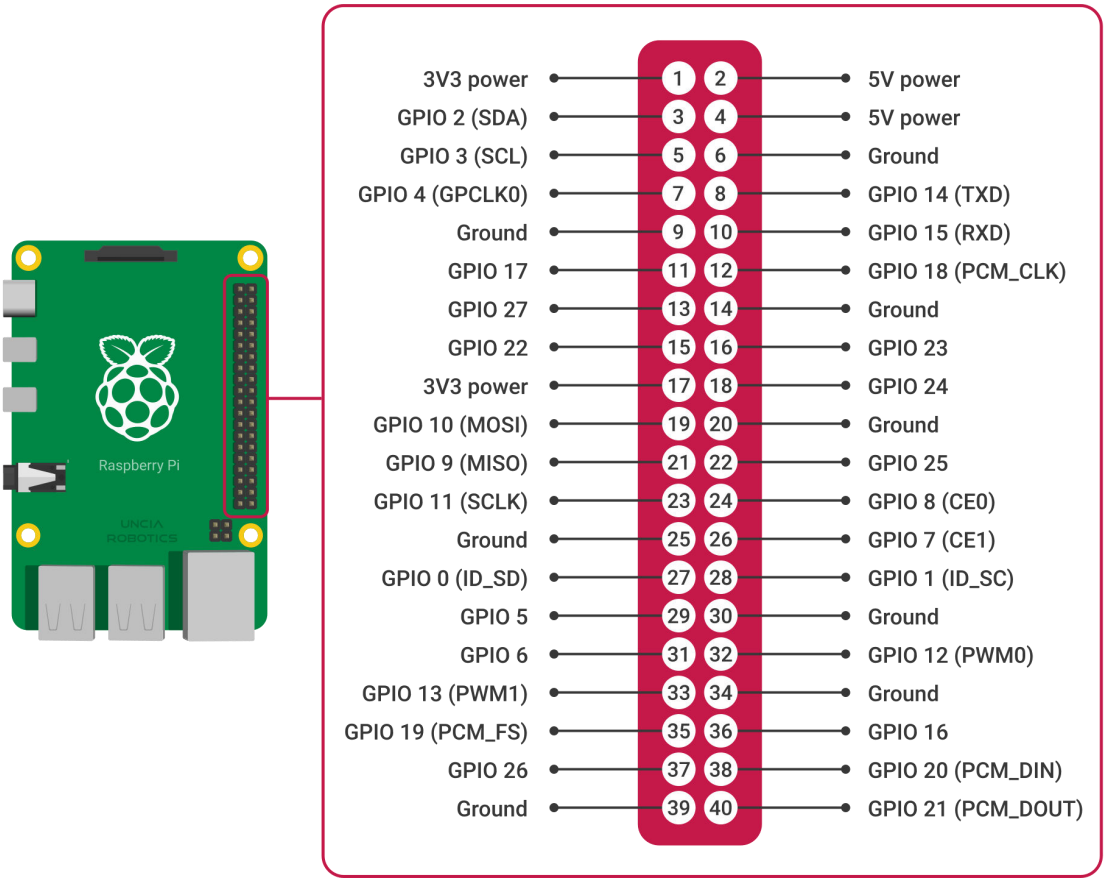


图 4: Arduino 针脚

B：Arduino Fritzing 图样[1]

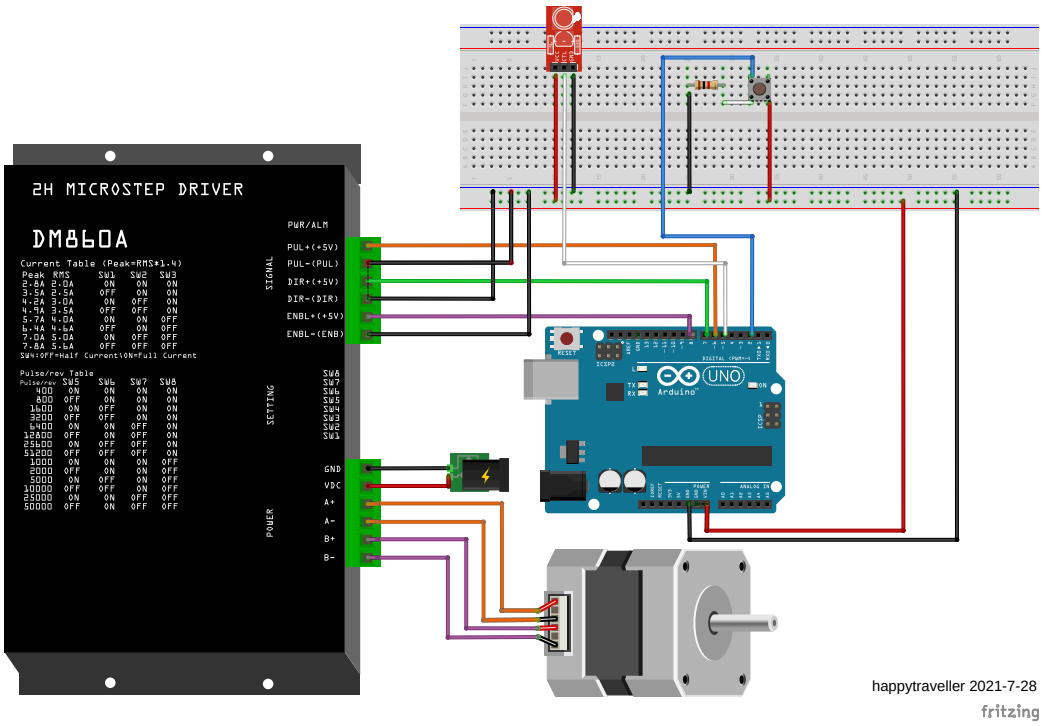


图 5: 实物接线图

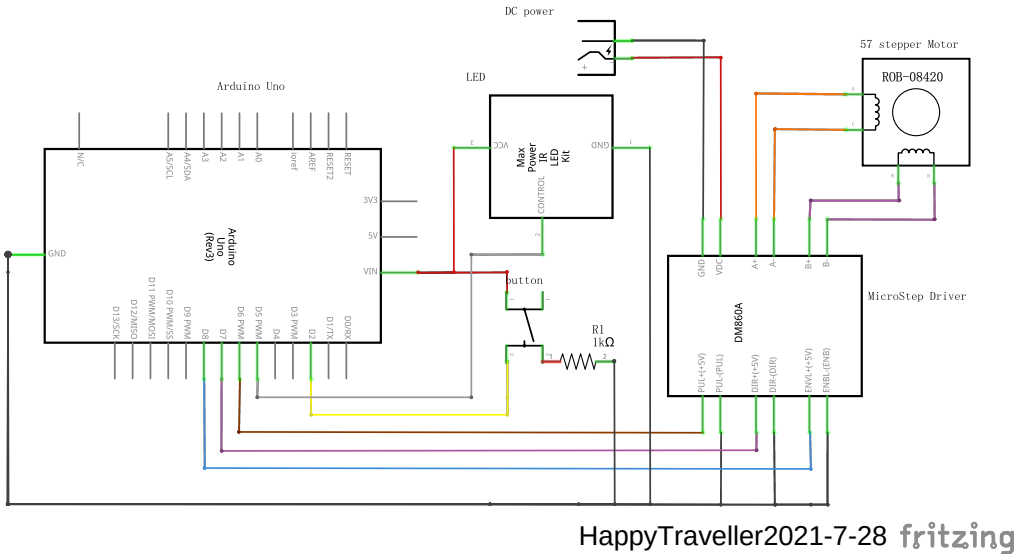


图 6: 针脚连接示意图

运用 Arduino 对步进电机驱动的传动装置进行测试，元件包括：传动装置、Microstep Driver（电机驱动板）、面包板、Arduino Uno R3 开发板、限位开关、数控 LED 灯、若干导线、直流电源。利用 Arduino 官方 IDE 进行程序烧写。实现逻辑为传动块从靠近电机端移动到远离电机端，移动过程中不设置停留限位开关中断触发后，传动块从远离电机端移动到靠近电机端，中途不停留，完成后，停等一秒，而后重新执行远离电机程序。Arduino Uno 仅设置 2、3 两个外部中断接口，参考附录 A。程序内容参考附录 C。

直流电源供电 12V、电流为 0.6A，细分程度设置为 2，衰减程度设置为 20%，电流设置上限为 1.5A，SW1-SW8 的设置参数为 (1,1,1,1,1,1,0,1)。参数的具体讲解参考附录 C

[1]图样与实际程序存在接线出入，可根据实际情况进行相应调整

C：程序参考及硬件参数介绍

Listing 1: **arduino.c**

```
// 针脚定义
const int stepPin = 6;          // PUL -Pulse
const int dirPin = 7;           // DIR -Direction
const int enPin = 8;            // ENA -Enable
const byte ledPin = 4;          // LED -Interrupt
const byte interruptPin = 2;    // KEY - Interrupt
volatile byte state = LOW;      // 定义默认输入状态
int reset_num = 0;              // 电机转数

void setup() {
    // 针脚设置
    pinMode(stepPin, OUTPUT);
    pinMode(dirPin, OUTPUT);
    pinMode(enPin, OUTPUT);
    pinMode(ledPin, OUTPUT);
    pinMode(interruptPin, INPUT_PULLUP);
    // 启动使能端
    digitalWrite(enPin, LOW);
    // 串口通信初始化
    Serial.begin(9600);
    // 监视中断输入引脚的变化
    attachInterrupt(digitalPinToInterrupt(
        interruptPin), stateChange, HIGH);
    // 串口打印转数数据
    Serial.println(reset_num);
    // 延迟确保通电后数据为0
    delay(3000);
}

void loop() {
    // 循环结构内部，设置正向函数
    // 从靠近电机端移动到远离电机端
    // 默认LED提示灯为熄灭状态
    state = LOW;
    digitalWrite(ledPin, state);
    digitalWrite(dirPin, LOW);
    while (reset_num <= 20) {
        for (int x = 0; x < 200; x++) {
            digitalWrite(stepPin, HIGH);
            delayMicroseconds(320);
            digitalWrite(stepPin, LOW);
            delayMicroseconds(320);
        }
        reset_num++;
        Serial.println(reset_num);
    }
}

void stateChange() {
    // 设置限位开关模拟
    // 触碰开关后，进入中断程序
    // 从远离电机端移动到靠近电机端
    // 移动过程中，LED显示灯长亮
    state = HIGH;
    digitalWrite(ledPin, state);
    digitalWrite(dirPin, HIGH);
    while (reset_num >= 2) {
        for (int x = 0; x < 200; x++) {
            digitalWrite(stepPin, HIGH);
            delayMicroseconds(320);
            digitalWrite(stepPin, LOW);
            delayMicroseconds(320);
        }
        reset_num--;
        Serial.println(reset_num);
    }
    state = HIGH;
    // 普通的延迟函数实质上是中断函数
    // 由于函数中断优先级高于时间中断，故无法执行
    for (int i = 0; i < 60; i++) {
        delayMicroseconds(10000);
    }
}
```

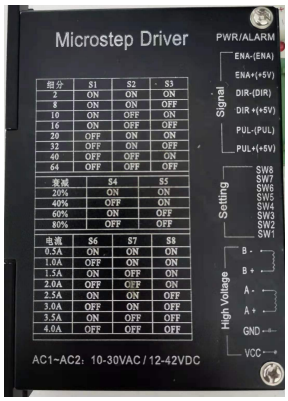


图 7: 硬件参数说明

如右图所示，通过 8 个接口进行硬件参实的调节，其中 SW1-SW3 实现细分度的调整细分程度越高，单圈所需的脉冲次数也越多，在细分度为 2 的前提下，单圈脉冲次数为 200。细分度共分为 8 级，分别为 (2, 8, 10, 16, 20, 32, 40,64)，在细分度为 2 的前提下，S1, S2, S3 均置为 ON(开启状态)。

给电机线圈充电过程是线性的，当线圈的电流大于一定值的时候，单片机输入衰减时钟让电流达到稳定。快慢衰减是指在 H 桥中的选择场效应管的导通截至来控制电流放电速度的快慢，放电快则为快衰减，慢则为慢衰减。衰减程度共分为 4 级，分别为 (20%, 40%, 60%, 80%) 本次测试采用 20% 慢速档。

电流大小同样可分为 8 级，分别为 (0.5A, 1.0A, 1.5A, 2.0A, 2.5A, 3.0A, 3.5A, 4.0A)，本次测试采用 1.5A 档位。

程序主界面

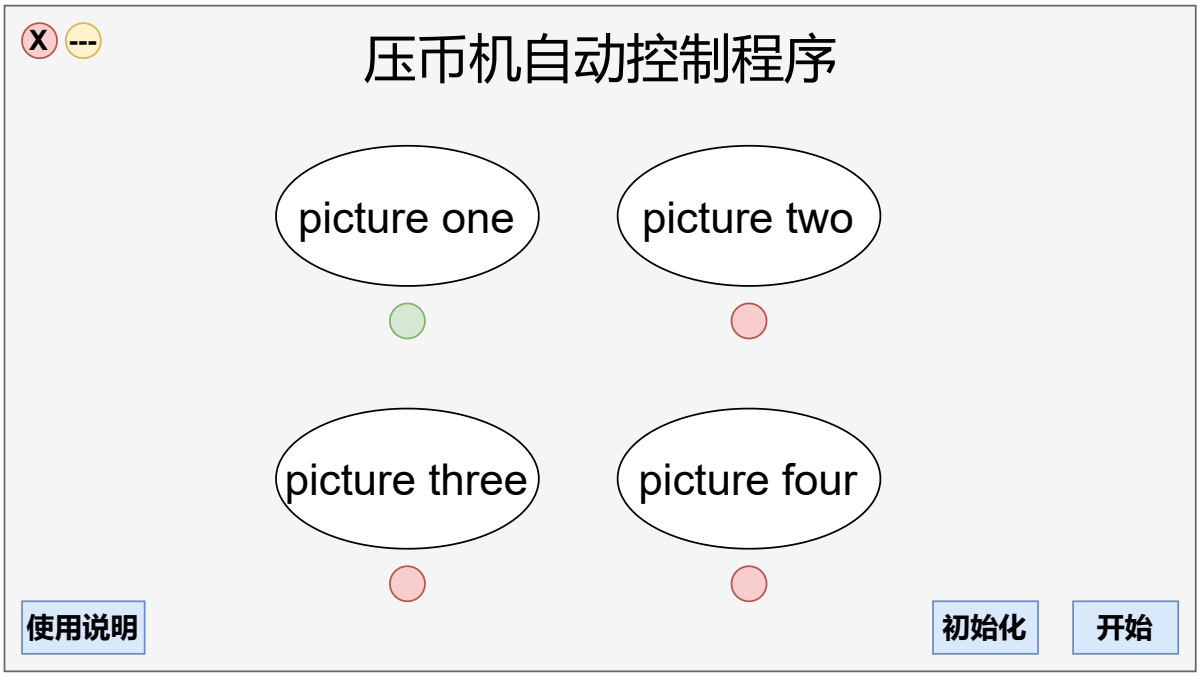


图 8: 程序主界面

接口说明和功能说明

I2C

-

SPI：串行外设接口

- 概述

  Raspberry Pi 系列的设备配备了许多 SPI 总线。SPI 可以用来连接各种各样的外围设备–显示器、网络控制器（以太网、CAN 总线）、UART 等等。

- 硬件支撑

  所有 Raspberry Pi 设备共有的 BCM2835 内核有 3 个 SPI 控制器。

  SPI0，在所有 Pis 的头部都有（尽管有一个替代的映射，只在计算模块上可用）。

  SPI1，可用于 40 针版本的 Pis。

  SPI2，只适用于计算模块

  BCM2711 增加了另外 4 条 SPI 总线–SPI3 至 SPI6，每条都有 2 个硬件芯片选择。所有这些都可以在 40 针头上使用（只要没有其他东西试图使用相同的引脚）。

- 针脚替代说明图

  网站参考

- 信号定义

  - SCK : Serial Clock 串行时钟
  - MOSI : Master Output,Slave Input 主发从收
  - MISO : Master Input,Slave Output 主收从发
  - SS/CS : Slave Select 片选信号