

基于 Arduino 板的智能门锁开发

2022 年电力学院开放实习 by ZAer

目录

队伍信息	3
1 研究背景及意义	4
2 各模块介绍	4
2.1 Arduino 板	4
2.2 HC-06 蓝牙模块	4
2.3 RFID 射频模块	5
3 3D 打印技术	7
4 技术路线	11
5 硬件部分	12
5.1 焊接说明	12
5.2 Fritzing 接线图	12
5.3 接线说明	14
5.3.1 接线说明	14
5.3.2 伺服电机	14
5.3.3 HC-06 蓝牙模块	14
5.3.4 RFID 模块	15
5.3.5 按钮	15
5.4 信号流	15
5.5 实物展示	16
6 软件部分	16
6.1 Arduino IDE	16
6.2 Code (Language == C++)	16
6.2.1 代码解释	21

7 测试结果	22
7.1 刷卡开门	22
7.2 蓝牙开门	23
7.3 按钮开门	25
8 创新点	26
9 改进方案	26
10 总结	27

队伍信息



学院: 电力学院

专业: 电气工程及其自动化

成员: 赵知易 祁宇轩 卢嘉铭

学号:

起始日期: 2022 年 4 月 5 日 —— 2022 年 4 月 14 日

1 研究背景及意义

在华工五山校区，大多数宿舍都采用传统的钥匙锁开门方式，而钥匙较小易丢失，加之门锁为金属易生锈、变形，有时需要较大的力气才能开门。本组将基于 Arduino 板，并运用 RFID(Radio Frequency Identification, 射频识别) 通信技术实现隔空刷卡开门功能，并通过 3D 打印制作门锁装置置于门后，并用简易外壳封装电路板置于门前，连接二者后直接运用到实际中。从而有效解决钥匙易丢、开门不易的问题。本组在实现按钮开门、刷卡开门后，进一步开发了手机连接蓝牙 HC-06 模块，输入密码开锁功能。

2 各模块介绍

2.1 Arduino 板

Arduino 是一个开源嵌入式硬件平台，用来供用户制作可交互式的嵌入式项目。此外 Arduino 作为一个开源硬件和开源软件的公司，同时兼有项目和用户社区。该公司负责设计和制造 Arduino 电路板及相关附件。这些产品按照 GNU 宽通用公共许可证 (LGPL) 或 GNU 通用公共许可证 (GPL) 许可的开源硬件和软件分发的，Arduino 允许任何人制造 Arduino 板和软件分发。

Arduino 系列电路板的设计大多使用 Atmel AVR 单片机。这些电路板配有一组数字和模拟 I/O 引脚，可以连接各种扩展板或面包板 (Shields 扩展版) 和其他电路。这些电路板具有串行通信接口，包括某些型号上的通用串行总线 (USB)，也用于从个人电脑加载程序。

软件编程方面，通常使用 C/C++ 编程语言，官方提供了一个 Arduino IDE 用开发。除了使用传统的编译工具链之外，Arduino 项目还提供了一个基于 Processing 语言项目的集成开发环境。另外，一些少儿编程教育软件提供了对 Arduino 的可视化编程。



Figure 1: Arduino Uno

2.2 HC-06 蓝牙模块

HC-06 蓝牙串口通信模块，是基于 Bluetooth Specification V2.0 带 EDR 蓝牙协议的数传模块。无线工作频段为 2.4GHz ISM，调制方式是 GFSK。模块最大发射功率为 4dBm，接收灵敏度-85dBm，板

载 PCB 天线，可以实现 10 米距离通信。

模块采用邮票孔封装方式，模块大小 $27\text{mm} \times 13\text{mm} \times 2\text{mm}$ ，方便客户嵌入应用系统之内，自带 LED 灯，可直观判断蓝牙的连接状态。模块采用 CSR 的 BC417 芯片，支持 AT 指令，用户可根据需要更改角色（主、从模式）以及串口波特率、设备名称等参数，使用灵活。

HC 05/06 适用于串行通信。在这里，安卓应用程序被设计成在按下某个按钮时向蓝牙模块发送串行数据。另一端的蓝牙模块接收数据，并通过蓝牙模块的 TX 引脚（Arduino 的 RX 引脚）将其发送给 Arduino。送入 Arduino 的代码检查接收到的数据并进行比较。如果接收到的数据为 1，则当接收到的数据为 0 时，LED 将点亮或熄灭 HC 05/06 适用于串行通信。在这里，安卓应用程序被设计成在按下某个按钮时向蓝牙模块发送串行数据。另一端的蓝牙模块接收数据，并通过蓝牙模块的 TX 引脚（Arduino 的 RX 引脚）将其发送给 Arduino。送入 Arduino 的代码检查接收到的数据并进行比较。如果接收到的数据为 1，则当接收到的数据为 0 时，LED 将点亮或熄灭。

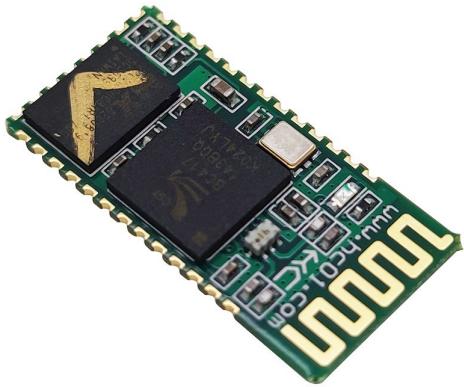


Figure 2: HC-06 蓝牙模块

2.3 RFID 射频模块

RFID 代表射频识别，它是一种非接触式技术，在许多行业中被广泛用于人员跟踪，访问控制，供应链管理，图书馆书籍跟踪，收费站系统等任务。

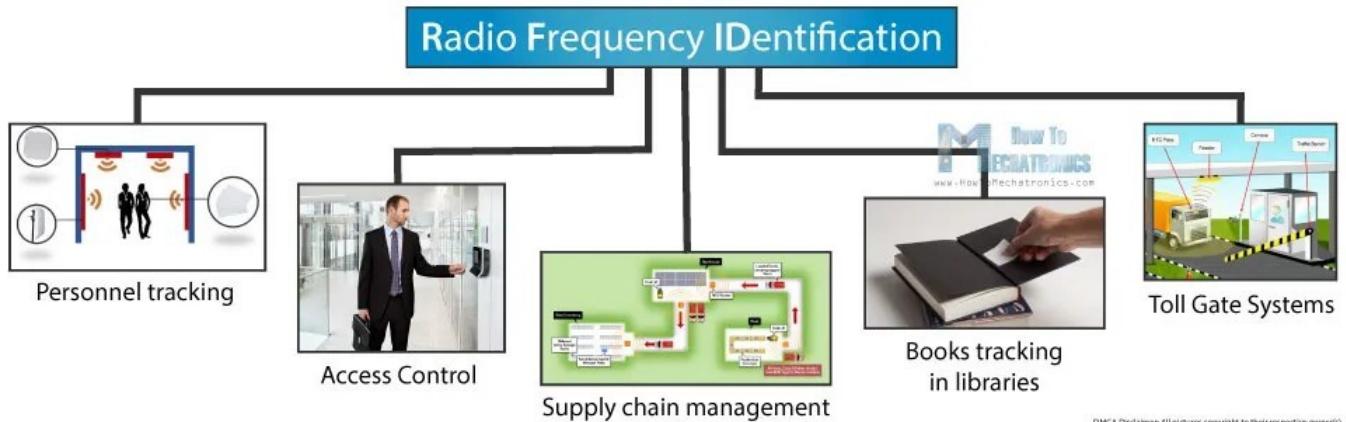


Figure 3: RFID 应用场景

RFID 阅读器由射频模块，控制单元和产生高频电磁场的天线线圈组成。另一方面，电子标签通常是无源元件，仅由天线和电子微芯片组成，因此当它接近收发器的电磁场时，由于感应，其天线线圈中会产生电压，该电压用作微芯片的电源。



Figure 4: RFID 元件构成

数据传输方式 1：当电子标签通电时，它可以从读取器中提取传输的消息，并且它使用一种称为负载操作的技术来实现将消息发送回读取器。打开和关闭电子标签天线上的负载将影响阅读器天线的功耗，这可以通过电压降来测量。电压的这种变化将被捕获为 1 和 0，这就是数据从电子标签传输到读取器的方式。

数据传输方式 2：在读取器和电子标签之间还有另一种数据传输方式，称为反向散射耦合。在这种情况下，电子标签使用部分接收功率来产生另一个电磁场，该电磁场将被阅读器的天线捕获。

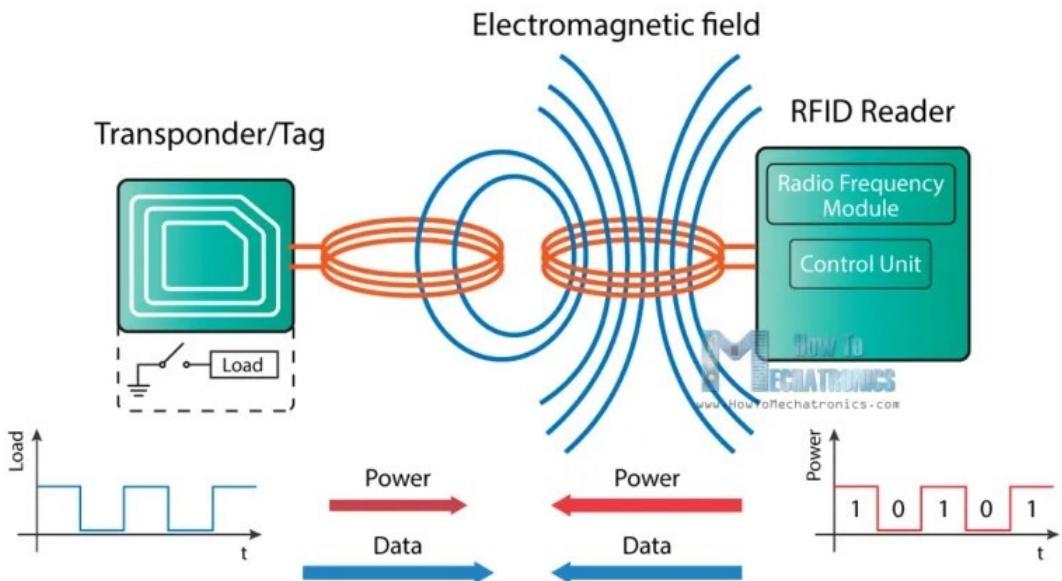


Figure 5: RFID 传输原理

3 3D 打印技术

三维打印（英语：3D printing），又称立体打印、增材制造（英语：Additive Manufacturing, AM）、积层制造，可指任何打印三维物体的过程。] 三维打印主要是一个不断添加的过程，在计算机控制下层叠原材料。三维打印的内容可以来源于三维模型或其他电子数据，其打印出的三维物体可以拥有任何形状和几何特征。三维打印机属于工业机器人的一种。

为了使得制作方法更简单，小组采取 3D 打印来制作组件和伺服电机支架伺服电机。锁定机制基于 Sagittario 的滑动锁设计，将其缩小到原始尺寸的 65%。使用白色树脂填充下 3D 打印锁定机构和伺服电机支架。

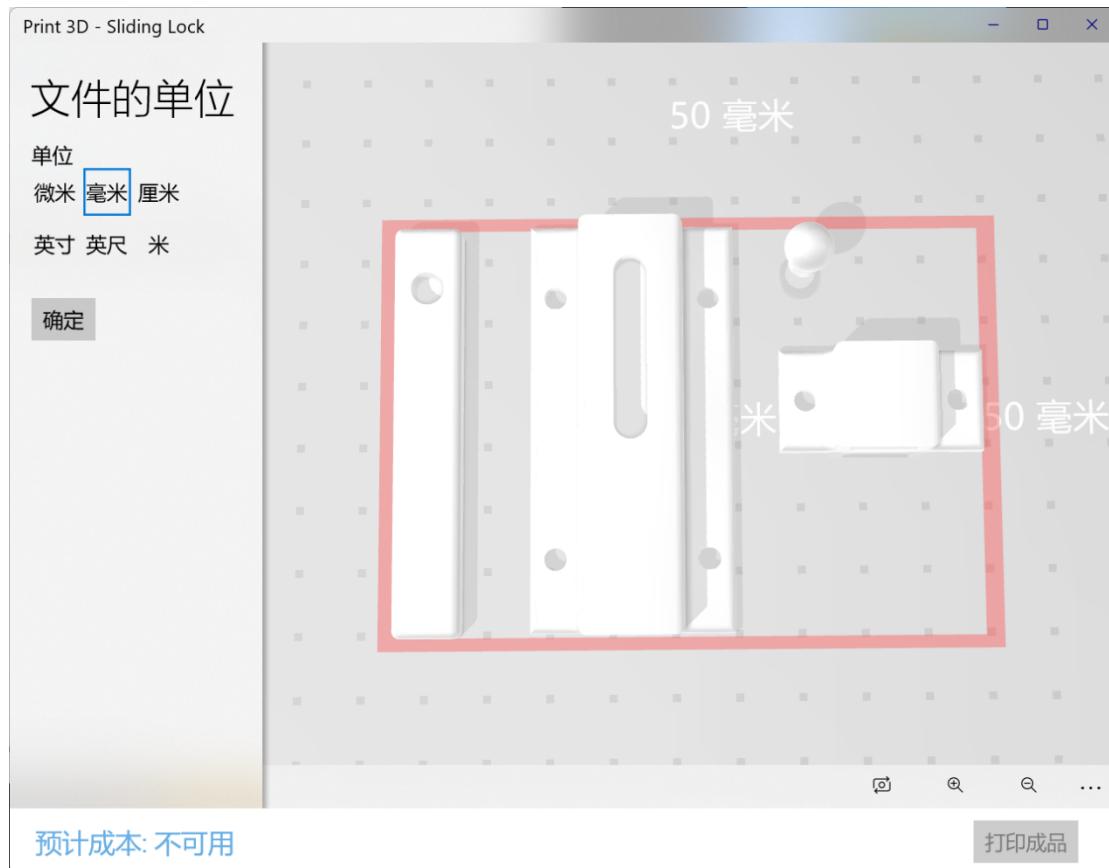


Figure 6: 门锁装置的建模

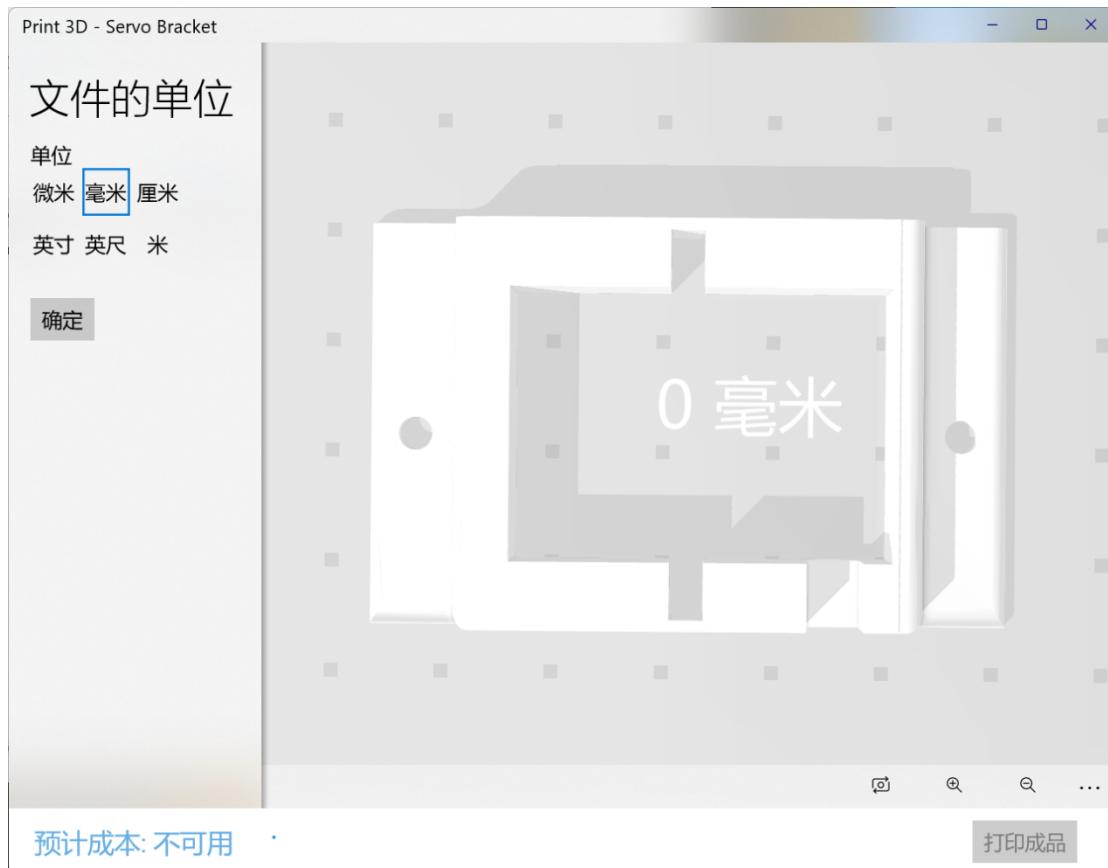


Figure 7: Micro Servo 伺服装置封装外壳的建模

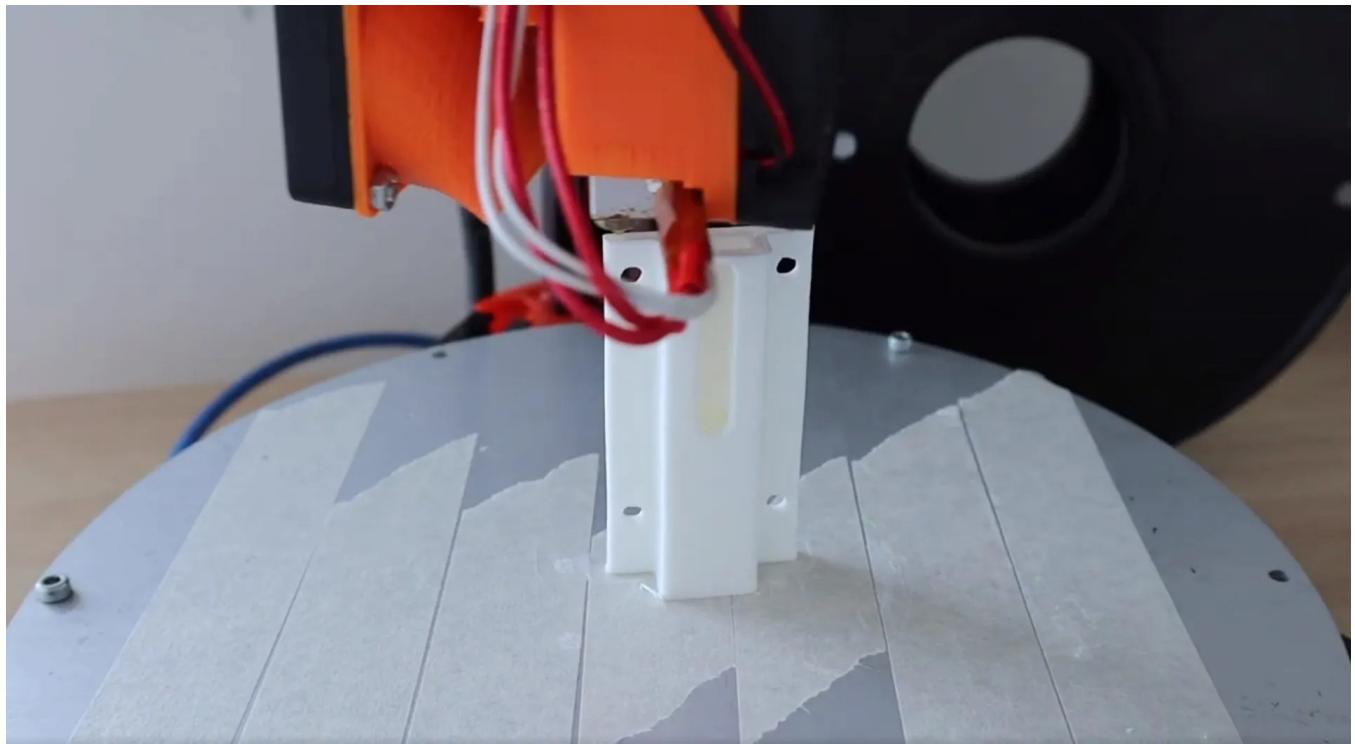


Figure 8: 3D 打印实拍

锁定机制基于 Sagittario 的滑动锁设计，将其缩小到原始尺寸的 65%。使用白色树脂填充下 3D 打印锁定机构和伺服电机支架。

这些部件至少用 50% 的填充物制成。组装较为简单。首先确保主体中的插槽将允许滑动螺栓轴自由滑动，并检查轴是否适合滑动杆上的盲孔。然后调试程序中的 LockPos() 函数调整舵机的转动幅度。

4 技术路线

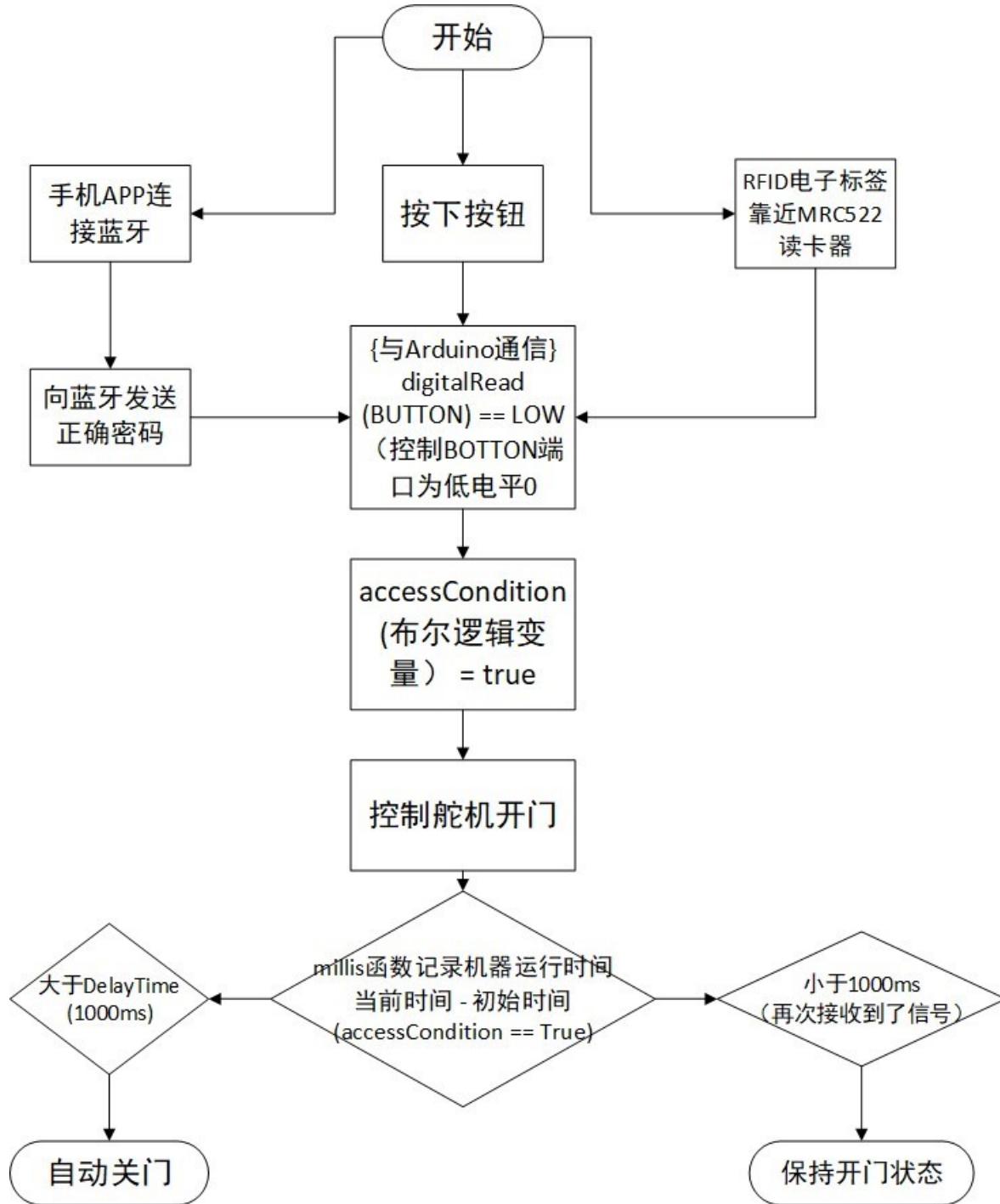


Figure 9: 流程图

我们可以通过三个方式打开门锁。第一个最简单的方式为按按钮开门，通过按按钮给 Arduino 发出一个信号，Arduino 接收信号以后改变 BUTTOM 端节点为低电平。第二个方式是通过蓝牙。用手机

连接蓝牙输入正确的密码，发出信号，改变电平。第三个方式为刷卡开门，我们预先在程序里录入卡的 UID 代码哦，当刷卡时，RFID 模块与 Arduino 进行通信。正确的卡会判断为 True，错误的卡会判断为 False，并闪烁一下红灯。

当 BUTTOM 度端为低电平时，accessCondition == True，从而控制舵机开门。

此外，为了实现按住按钮门不关闭的效果，我们创新性地使用了 millis() 函数来记录机器运行的时间，通过 Arduino 接收两次信号间隔来判断用户是否一直按着按钮，从而弥补了 Delay() 函数的不足，可实现自动关门或手动控制门长开状态。

5 硬件部分

5.1 焊接说明

本方案的焊接技术主要体现在减少接线和移除面包板的功能上。

(1) 采用焊接自制 RG LED 解决了零件包无 RGB LED 灯元件的问题。采用三引脚接头的三个引脚分别与红灯、绿灯和地线焊接在一起，红灯和绿灯串联 220 欧姆的电阻的方法，自制了 RG LED 灯，实现了直接连接 Arduinon 板。

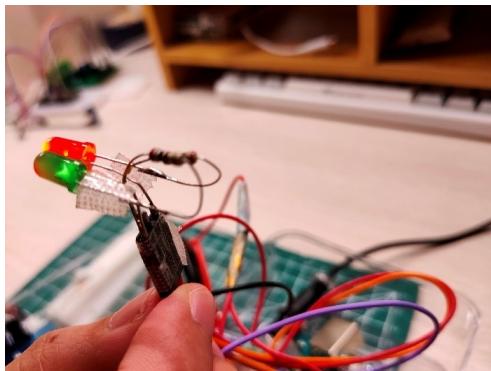


Figure 10: 自制 RG-LED 灯

(2) 采用焊接技术实现原来面包板的串并联功能，移除面包板，使得封装元件更加美观。通过剪线钳剪断漆包线和去掉漆包线上的漆，通过焊接将地线与需要连接舵机、蓝牙模块、RFID 模块的三根地线连接在一起，灵活解决了移除面板板后地线接口不够的尴尬。

5.2 Fritzing 接线图

Fritzing 是一个开源项目，旨在开发用于电子硬件设计的业余或业余 CAD 软件，为设计师和艺术家提供支持，帮助他们从试验原型转向构建更持久的电路。它是在波茨坦应用科学大学开发的。Fritzing 是 GPL 3.0 或更高版本许可下的自由软件，源代码在 GitHub 上免费提供，二进制文件的价格是 GPL 允许的。

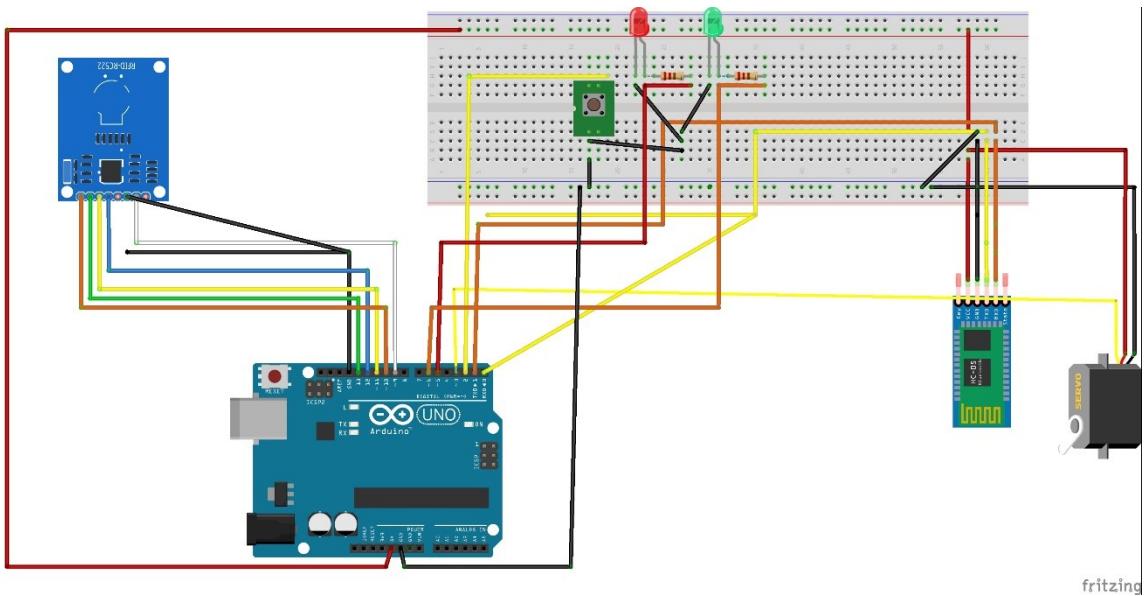


Figure 11: Fritzing 接线图

RC522 传感器将使用 SPI 接口连接到我们的 Arduino。然后，我们将连接一个绿色 LED，它将闪烁以显示标签已被读取且访问已被授予，而红色 LED 将闪烁以指示标签已被读取且访问未被授予。最后，将使用 micro servo 打开和关闭锁定机构。LED 和 micro servo 使用典型的 Arduino 电路连接。

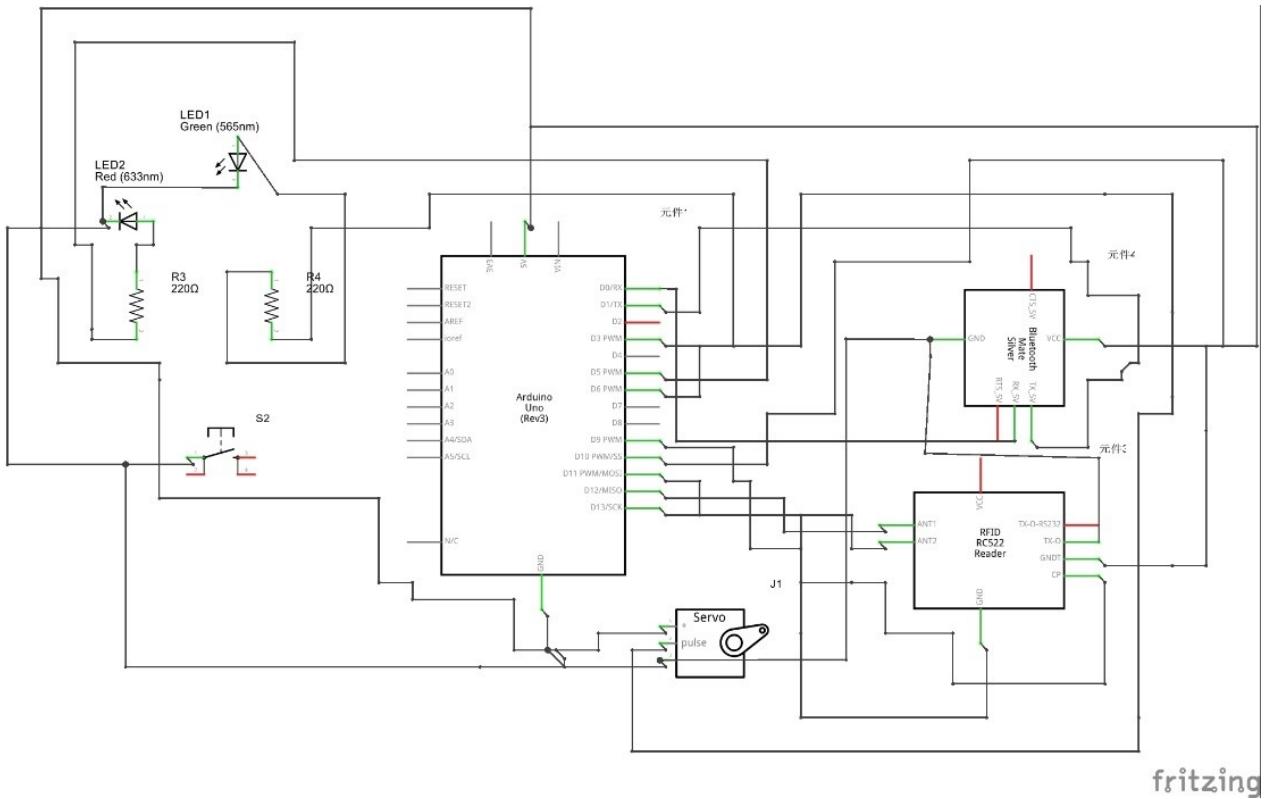


Figure 12: 电路原理图

5.3 接线说明

5.3.1 接线说明

RST_PIN 9——重置 SS_PIN 10——片选

redLEDPin ——5

greenLEDPin ——6

5.3.2 伺服电机

J1 信号——D3 DWM

VCC——5V

GND——GND

5.3.3 HC-06 蓝牙模块

VCC——5V

GND——GND TXD——RXD

RXD——TXD

5.3.4 RFID 模块

RST——9
GND——GND MISI——11
MISO——12
SCK——13
SDA——10

5.3.5 按钮

BUTTON——2

5.4 信号流

在这个系统中，arduino 板共有三个输入信号，分别是按钮、蓝牙、RFID，每种信号的传递方法和协议又有所不同。

其中按钮部分最为简单，只用到了数字引脚 BUTTON 和 GND 两 pin。其中数字引脚 pin 既能提供电压也接收信号。因为这个引脚设置为了 pinMode(BUTTON, INPUT_PULLUP)（上拉电压模式）在不接地的情况下，始终保持着高电平，digitalRead(BUTTON) 输出保持为 1。当按下按钮后，BUTTON 与 GND 接通，digitalRead(BUTTON) 输出 0。在每次循环中，只需要判断这个端口的输出，就能判断是否按下的按钮。

蓝牙部分，arduino 通过串口与 HC06 蓝牙模块通讯的，一共用到了四条线，分别是 RX、TX、Vcc、GND。其中 Vcc、GND 负责为蓝牙模块供电，RX、TX 负责通讯。在 Arduino 控制器上，串口都是位于 Rx 和 Tx 两个引脚。通过设定 Serial.begin(speed) 来开启串口，其中参数 speed 指串口通信波特率，用于设定串口通信速率的参数。串口通信双方必须使用相同的波特率才可以正确通信，这里我们使用了 9600。在使用串口时，Arduino 会在 SRAM 中开辟一段大小为 64B 的空间，串口接收到的数据都会被暂时存放在该空间中，称这个存储空间为缓冲区。当调用 read() 函数时，就会从缓冲区中取出 1B 的数据。这个就是通过蓝牙向 arduino 传输的数据。

RFID 部分，arduino 通过 SPI(Serial Peripheral Interface) 与 RFID 模块通讯。这是一种用于芯片通信的同步串行通信接口规范，主要应用于单片机系统中，类似于 I²C。SPI 设备之间使用全双工模式通信，是一个主机和一个或多个从机的主从模式。主机负责初始化帧，这个数据传输帧可以用于读与写两种操作，片选线路可以从多个从机选择一个来响应主机的请求。这里一共用到了六条线 MISO、MISO、SDA、RST、SCK、Vcc、GUN 其中后面两个负责供电，MISO、MISO 负责在 arduino 和 RFID 之间传输信号，SDA、RST、SCK 负责提供时钟信号、片选信号等辅助信号。最终通过 SPI 库，可读取 RFID 模块中特定位置寄存器的值，并使用 RFID 库解析为 Uid 等信息。

5.5 实物展示

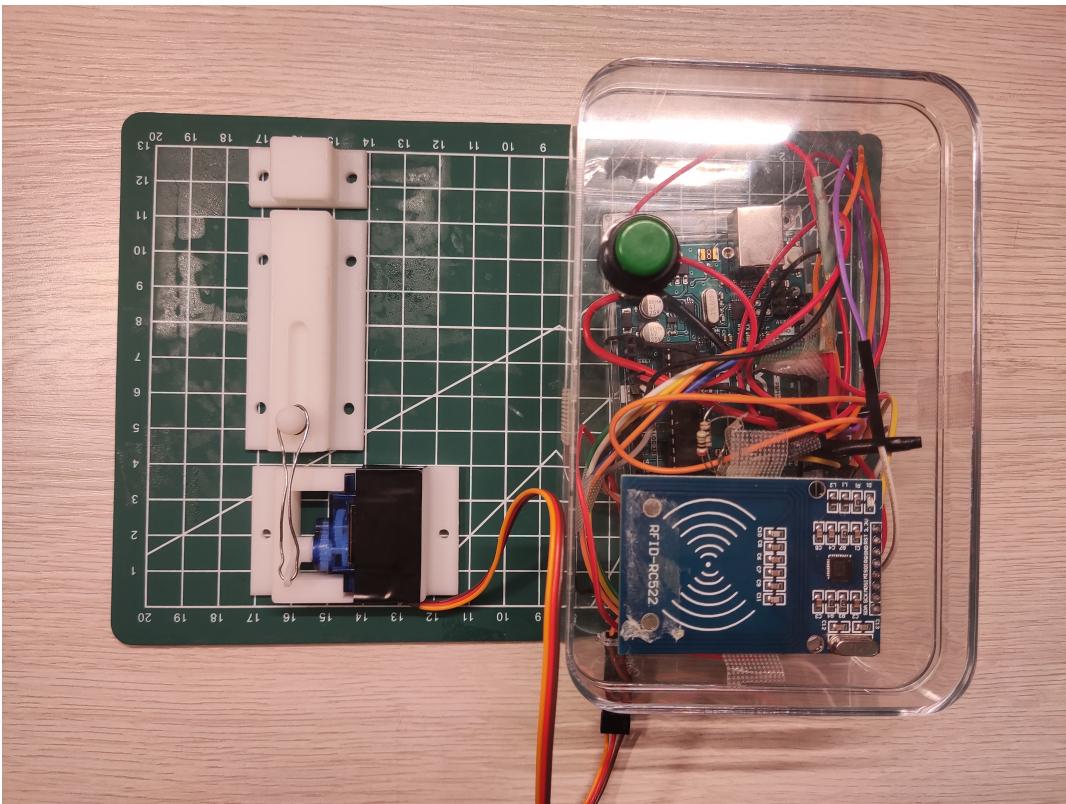


Figure 13: 实物封装效果

6 软件部分

6.1 Arduino IDE

Arduino 集成开发环境包含一个用于编写代码的文本编辑器、消息区、文本控制台、带有常用功能按钮的工具栏和一系列菜单。它连接到 Arduino 硬件以上传程序并与之通信。

6.2 Code (Language == C++)

```
1 #include <RFID.h>
2 #include <Servo.h>
3 #include <SPI.h>
4
5 #define RST_PIN 9 // 重置
6 #define SS_PIN 10 // 片选
7 // 默认的 MOSI = 11, MISO = 12
8 // SCK = 13
```

```

9 #define BUTTON 2
10
11 #define DelayTimeToClose 1000
12
13 RFID rfid(10, 9); // 创建一个实例 RFID class
14 unsigned char str[MAX_LEN]; // MAX_LEN is 16: size of the array
15
16 constexpr int accessGrantedSize = 3;
17 // id卡号列表长度
18 String accessGranted[accessGrantedSize] = {"81579111109", "1231386999", "
19 1415215015015"}; // 可接受的id卡号列表
20
21 Servo lockServo; // 创建Servo实例
22 int lockPos = 15; // 锁定位置
23 int unlockPos = 75; // 开锁位置
24 boolean locked = true; // 上锁状态
25
26
27 int redLEDPin = 5;
28 int greenLEDPin = 6;
29
30
31 void setup()
32 {
33     Serial.begin(9600); // 开启串口，用于与蓝牙模块通讯，或者DeBug
34     SPI.begin(); // 初始化SPI，用于控制RFID模组
35     rfid.init(); // rfid实例初始化
36     pinMode(redLEDPin, OUTPUT); // LED 闪灯
37     pinMode(greenLEDPin, OUTPUT);
38     digitalWrite(redLEDPin, HIGH);
39     delay(200);
40     digitalWrite(greenLEDPin, HIGH);
41     delay(200);
42     digitalWrite(redLEDPin, LOW);
43     delay(200);
44     digitalWrite(greenLEDPin, LOW);
45     lockServo.attach(3);
46     lockServo.write(lockPos); // 上锁

```

```

46  pinMode(BUTTON, INPUT_PULLUP);
47  Serial.println("Already init"); // 向串口输出信息
48  Serial.println("Place card/tag near reader...");
49 }
50
51 // START LOOP
52 // 将开锁和亮灯放在循环的最后执行，便于插入蓝牙和按钮控制
53 void loop()
{
54
55 // 判断卡的UID是否在列表里面
56 if (rfid.findCard(PICC_REQIDL, str) == MI_OK) // 如果有ID卡靠近时才执行
57 {
58     Serial.println("Card found");
59
60     String temp = ""; // 临时变量，用于储存RFID的序列号
61     if (rfid.anticoll(str) == MI_OK) // 放碰撞检测，可以识别靠近的多张卡
62     {
63         Serial.print("The card's ID number is : ");
64
65         for (int i = 0; i < 4; i++) // 记录标签序列号
66         {
67             temp = temp + (0x0F & (str[i] >> 4));
68             temp = temp + (0x0F & str[i]);
69         }
70
71         Serial.println(temp); // 向串口输出序列号
72         accessCondition = checkAccess(temp); // 调用checkAccess函数，检查是否
73        否
74     }
75     rfid.selectTag(str); // 锁卡防止重复读卡
76 }
77 rfid.halt();
78
79 // 判断是否有人按下按钮
80 if (digitalRead(BUTTON) == LOW)
81 {
82     accessCondition = true;
83 }

```

```

84 // 判断蓝牙接受到的信号是否为密码
85 while (Serial.available() > 0)
86 {
87     char inputByte = Serial.read();
88     if (inputByte == 'S')
89     {
90         accessCondition = true;
91         break;
92     }
93     // 如果不接受，返回accessCondition=false，而且闪一下红灯
94     digitalWrite(redLEDPin, LOW);
95     delay(50);
96     digitalWrite(redLEDPin, HIGH); // 循环最后还会控制一次灯，不会出现红灯
97     和绿灯常亮的情况}
98
99 // 循环的最后汇总‘accessCondition，locked，lastTime’这三个信息，并控制
100 servo
101 tell_the_condition_to_control_the_servo(accessCondition, locked,
102 lastTime);
103 lock_condition_to_led(locked);
104
105 boolean checkAccess(String & temp)
106 {
107     boolean granted = false;
108     for (int i = 0; i <= (accessGrantedSize - 1); i++) // 遍历接受的id卡列表
109     {
110         if (accessGranted[i] == temp)
111         {
112             return true;
113         }
114     }
115
116     // 如果不接受，返回accessCondition=false，而且闪一下红灯
117     digitalWrite(redLEDPin, LOW);
118     delay(50);
119     digitalWrite(redLEDPin, HIGH); // 循环最后还会控制一次灯，不会出现红灯和

```

```

    绿灯常亮的情况
120   return false;
121 }
122
123 // DO THIS EVERY LOOP
124 // cheak if we should switch the servo
125 // 每次循环运行，判断三种情况，控制开锁和自动上锁
126 void tell_the_condition_to_control_the_servo(boolean & accessCondition ,
127   boolean & locked , unsigned long & lastTime)
128 {
129   unsigned long previousTime = millis();
130   if (locked)
131   {
132     if (accessCondition)
133     {
134       lockServo.write(unlockPos);
135       locked = false;
136       lastTime = previousTime;
137     }
138   }
139   else
140   {
141     if (accessCondition)
142     {
143       lastTime = previousTime;
144     }
145   }
146   else
147   {
148     if (previousTime - lastTime > DelayTimeToClose)
149     {
150       lockServo.write(lockPos);
151       locked = true;
152       lastTime = 0;
153     }
154     // else do nothing 这时门锁是开启的，保持不变
155   }
156   // reset the global variable 'accessCondition'
157   accessCondition = false;

```

```

157 }
158
159 // 根据是否上锁，控制led输出
160 void lock_condition_to_led(boolean locked)
161 {
162     if (locked)
163     {
164         digitalWrite(redLEDPin, HIGH);
165         digitalWrite(greenLEDPin, LOW);
166     }
167     else
168     {
169         digitalWrite(greenLEDPin, HIGH);
170         digitalWrite(redLEDPin, LOW);
171     }
172 }

```

6.2.1 代码解释

此代码用到了<RFID>、<Servo>、<SPI>这三个库，其中 Servo 负责控制舵机，RFID 和 SPI 负责 Arduino 板和 MFRC522 模块通讯。

在预处理中设置好了 MFRC522、Servo、LED 的控制端口，延时关门的时间，舵机开关对应位置，便于修改。分别创建了 RFID 和 Servo 实例，便于后面程序直接调用控制。设定了拥有访问权限的 RFID 字符串列表。

setup() 函数，在 Arduino 通电后执行一次，对 Serial、SPI、rfid 三个对象初始化，红绿 LED 灯交替闪烁，设定好 lockServo 控制的端口，控制舵机上锁，并设定好按钮控制端端口。最后通过串口，向蓝牙模块和电脑的串口监视器输出初始化完成的信息。

loop() 函数，在 Arduino 执行完 setup() 函数后将一直循环执行，在这个函数中，对三种输入都做了判断，若需要开门，则将 accessCondition 设置为 true，在每次循环后都会检查。整个程序耦合度十分低，若想要添加新的开锁方式，只需控制 accessCondition 的值就能完成控制。

三种输入包括 RFID 的读取，蓝牙读取，按钮状态检测，三个输入条件只需达成一个就可开门。

其中 RFID 的读取部分，首先识别靠近的 RFID 卡，将其序列号转化为一个字符串 temp，检查 temp 是否在 accessGranted 序列中，若存在，则将 accessCondition 赋值为 true。若不存在则表明是未经授权的 ID 卡，此时不开门，将 accessCondition 赋值为 false，并使红灯闪烁。

蓝牙读取部分，检查输入的字符串是否为正确的密码，若是，则将 accessCondition 赋值为 true。

按钮检测部分，读取 PIN2 的电位，因为这个引脚的模式设置为了 INPUT_PULLUP，在不接地的情况下，一直保持高电位。当按下按钮，此引脚与 ground 接通，变成低电位，将 accessCondition 赋值为 true。

telltheconditiontocontroltheservo() 函数，在每次循环后执行，判断四种情况，控制开锁和自动上锁。

这个函数使用了三个全局变量，来判断当前状态，locked 是门锁状态、accessCondition 是是否需要开门、previousTime 是上次开锁时间。还有 DelayTimeToClose 是延迟关锁时间。通过判断 previousTime - lastTime 和 DelayTimeToClose 来实现延迟关锁，避免使用 delay 函数导致系统中断。

1. locked = true, accessCondition = true 开门、更新时间
 2. locked = false, accessCondition = true 此时门还未关，只需更新时间来延迟关门时间
 3. locked = false, accessCondition = false, (previousTime - lastTime > DelayTimeToClose) 此时门未关，accessCondition 为 false，而且超时。需要关门和重新设定时间。
 4. 第四种情况是什么都不干，这占了绝大多数时间。
- lockconditionto_led() 函数，判断门锁状态以控制 LED 灯，若上锁亮红灯，否则亮绿灯。

7 测试结果

用户可以选择在门外刷卡解锁和蓝牙解锁，开门状态绿灯亮、关门状态红灯亮。
在门内用按钮开关开门。

7.1 刷卡开门

使用正确的密码卡，门锁打开，绿灯亮，1 秒后关门，红灯亮

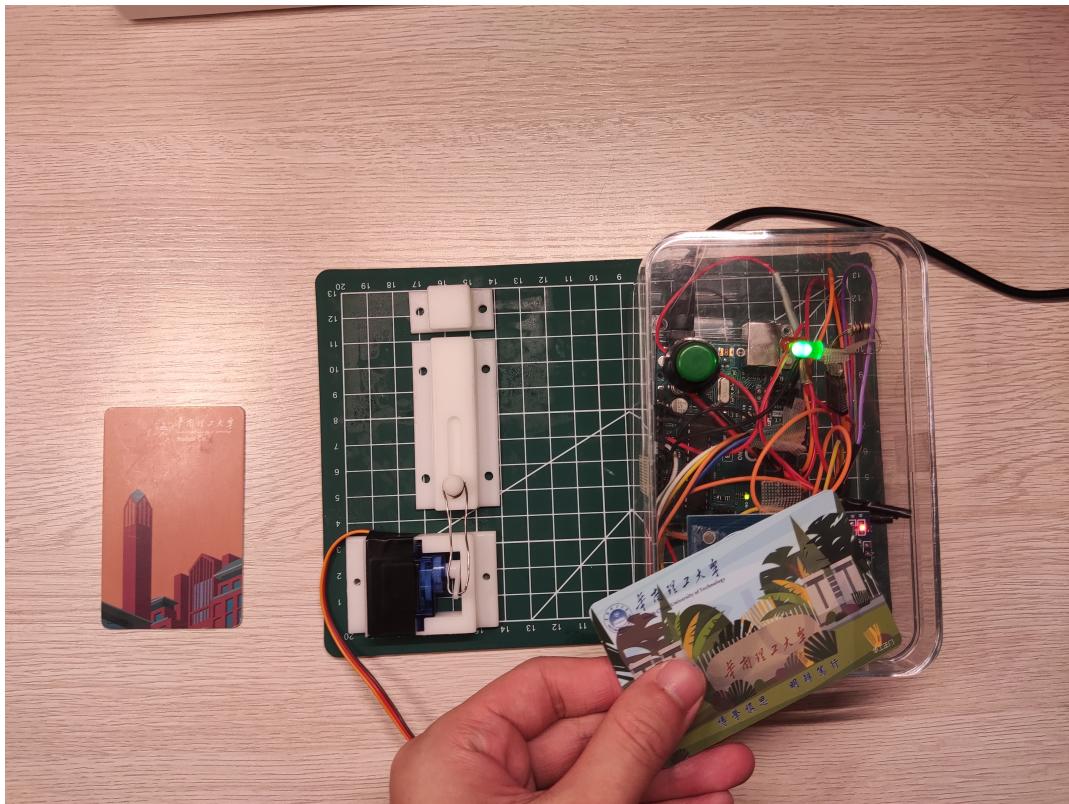


Figure 14: 录入 UID 的卡 IN

使用错误的卡，门锁不开，红灯闪烁

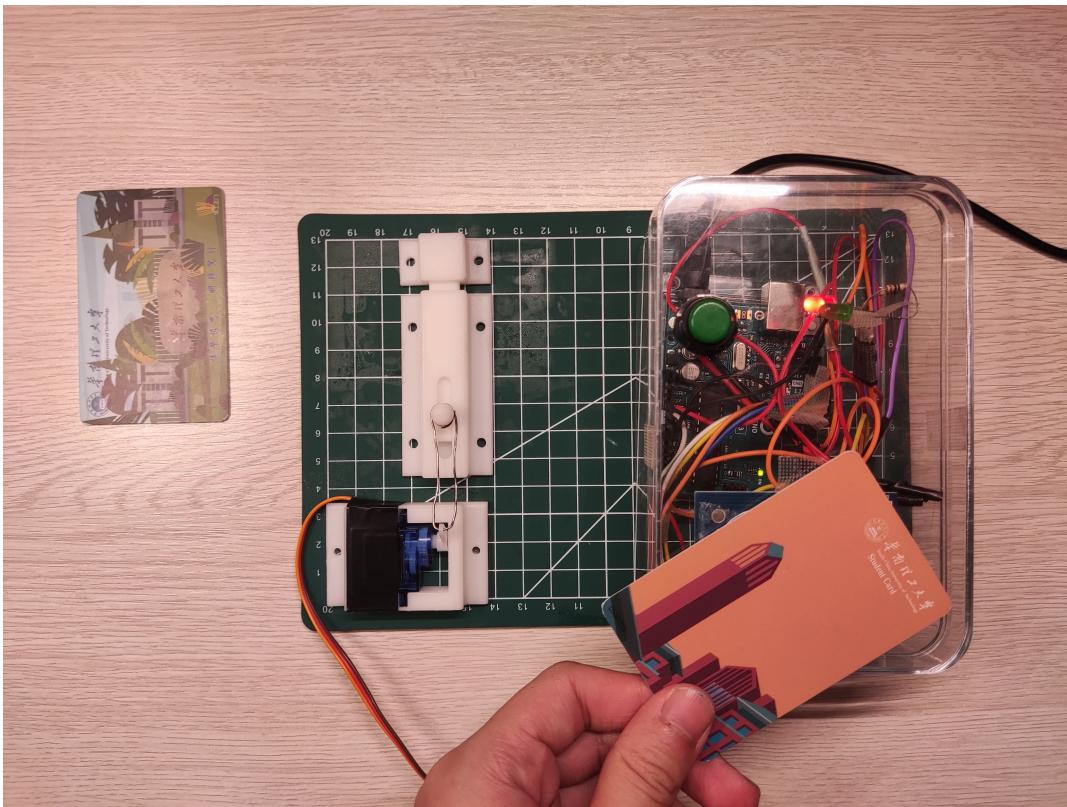


Figure 15: 未录入 UID 的卡 OUT

7.2 蓝牙开门

蓝牙解锁，输入正确的门锁密码，门锁打开，绿灯亮，1s 后关门红灯亮；

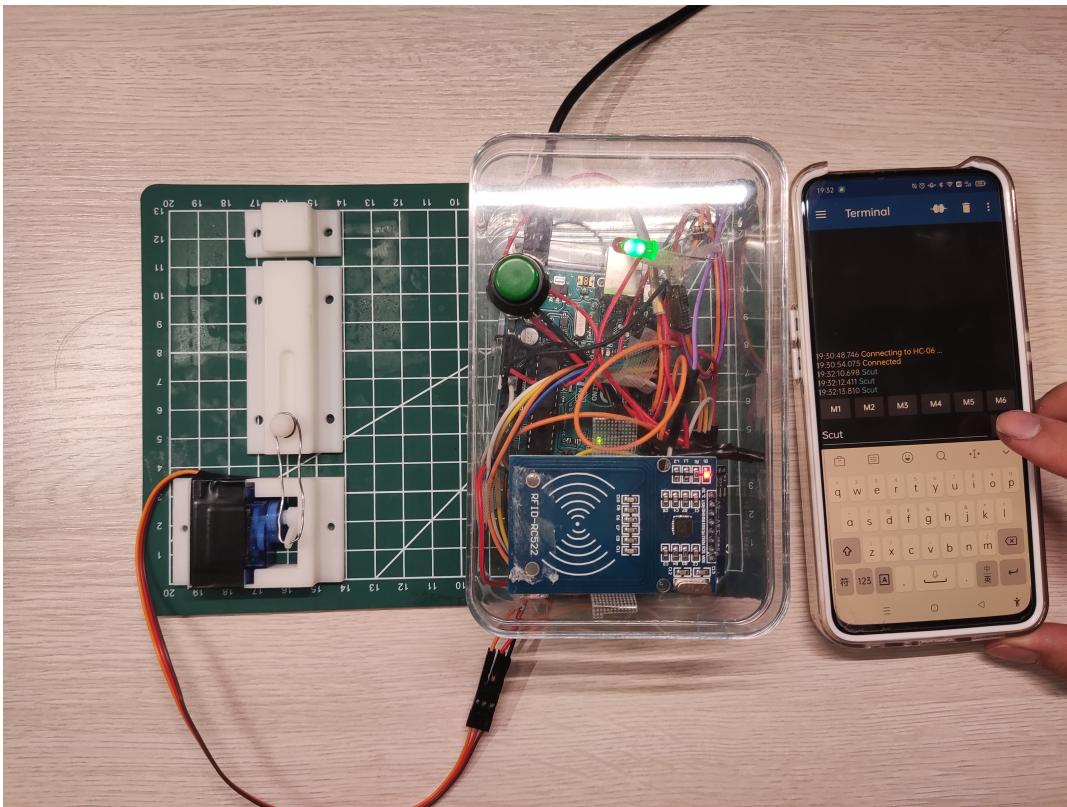


Figure 16: SCUT IN

输入错误的门锁密码，门锁不开，红灯闪烁

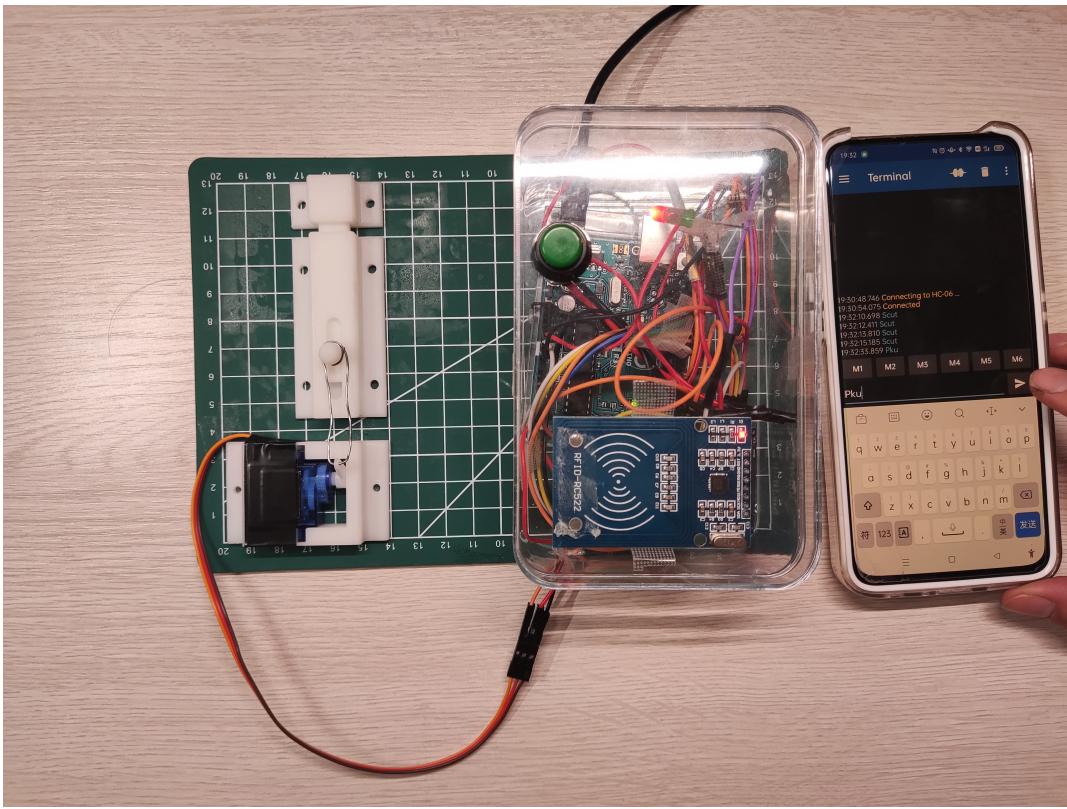


Figure 17: PKU OUT

7.3 按钮开门

按钮解锁，按下按钮，门锁打开，绿灯亮；长按按钮，门锁保存打开状态；松开按钮 1s 后关门，红灯亮

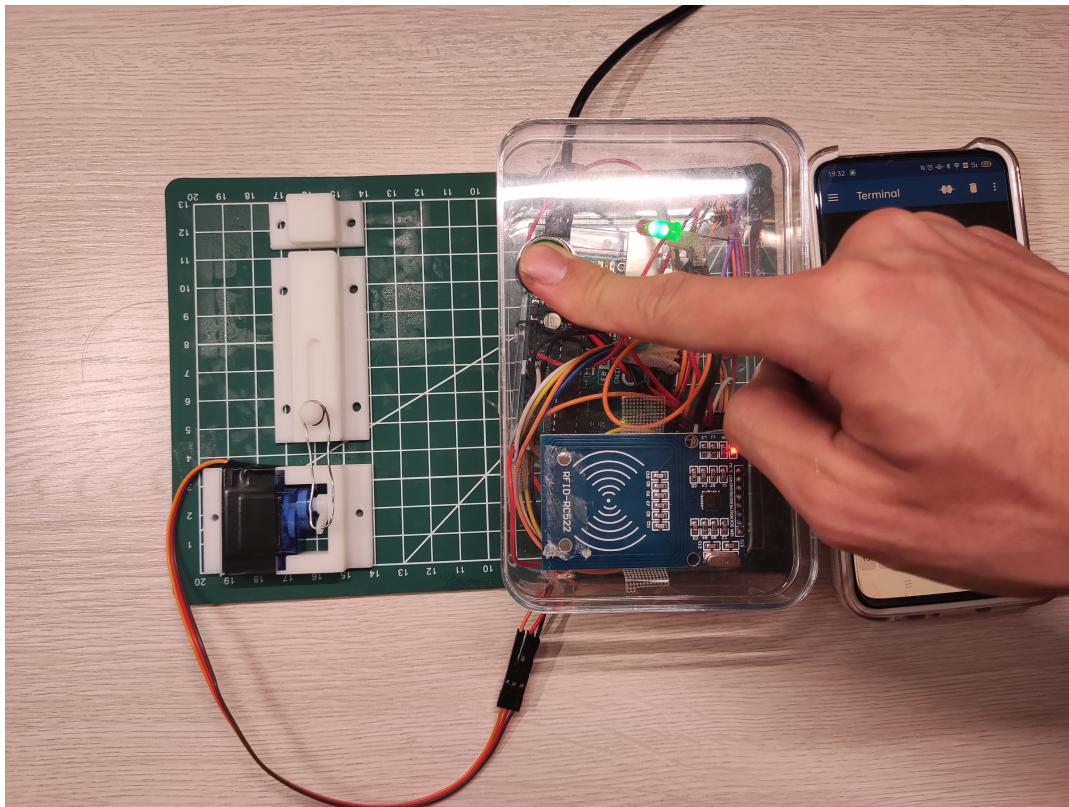


Figure 18: 按钮开门

8 创新点

1. 我们创新地合并了三条路径，并将逻辑简单化：刷卡/通过蓝牙输入正确密码/按按钮—与 Arduino 通信—改变 BUTTOM 电平高低—控制代码里布尔逻辑变量 boolean accesscondition = True/-False —舵机开/关。
2. 为了实现按住按钮门不关闭的效果，我们创新性地使用了 millis() 函数来记录机器运行的时间，通过 Arduino 接收两次信号间隔来判断用户是否一直按着按钮，从而弥补了 Delay() 函数的不足，可实现自动关门或手动控制门长开状态。
3. 我们还根据舵机大小进行了三维建模，运用 3D 打印技术打印了特制的门框。
4. 我们通过焊接自制了 RG LED 灯，并解决了移除面包板的困难。

9 改进方案

1. 增强安全性。出于展示的方便和美观，我们采用树脂材料 3D 打印门锁的方法。可以换用更加坚固的金属门锁可以加强门锁的安全性。
2. 本实习实物采用 Arduino Uno R3 原装开发板。实际上 Arduino 无论在硬件还是软件都已经高度

实现开源，例如在网购网站上面很容易就能找到兼容 Arduino 的 esp32 开发板，性能是 Arduino 的十倍，然而价格仅为它的五分之一。

3. 增加录卡模式。本方案采取识别卡的信号之后编进代码行的方法来实现录卡，添加卡和删除卡并不是很方便。可以向通过蓝牙模块发送命令来打开录卡模式，实现录卡删卡的功能。
4. 增加双电源供电模式。本方案解决了电池不持久的问题，但是并不能预防停电问题，所以可以采取双电源供电策略，通过定期更换电池来保证门锁能够持续运行。

10 总结

本方案解决了传统门锁出现生锈和钥匙容易丢失的问题，具有更强的安全性，更强的信息化和现代化的特点，充分利用了学生随身物品——校卡和手机储存和发送信息的功能，并且通过添加长按按钮方式长时间打开门锁，体现了方案的人性化。另外，对比主流的蓝牙宿舍门锁采用电池供电的形式，本方案采用通过电源适配器供电，实现了门锁的持续运行。