****

A级达标线上测试报告



**学院** 计算机科学与技术学院 **专业**  计算机科学与技术

**学号**

**姓名**

**手机** **完成日期** 2024-9-30

**成绩**

# 题目名称: 湿度监测仿真系统

# 一、题目要求

本题目为设计一个基于Arduino UNO的湿度监测仿真系统，使用Proteus软件进行仿真。系Arduino UNO（Atmega328P）通过串行接口组件与上位机 PC 进行双向

通信，PC 上位机软件向 Arduino UNO 发送学生自己的学号，Arduino UNO 收到

后在 LCD 上显示学生的学号，并且向 PC 机发送当前的湿度值。PC 上位机软件

显示收到的湿度值

1. 设计思路

设计思路总结如下：

1. 软件工具和环境 ：

使用了括Proteus 8.9 SP2、Arduino IDE和Virtual Serial Port Driver (VSPD)，以进行硬件仿真和串口通信仿真。

系统通过PC与Arduino进行串口通信，PC作为上位机，模拟实际的环境数据与设备交互。

利用rust的eframe与egui库编写GUI。

2. 串口通信流程 ：

通过VSPD创建一对虚拟串口（COM1和COM2）,模拟Arduino与PC之间的实际通信场景。

使用 `serialport` 库来打开、读取和写入串口，PC与Arduino之间的学号和湿度值的传输都通过串口完成。

3. Arduino与PC交互 ：

Arduino通过串口接收PC发送的学号数据，并将当前环境的湿度信息回传给PC。

在LCD屏幕上显示学号和实时湿度值，控制逻辑是湿度数据通过DHT11传感器获取，并通过串口发送给PC。

4. 系统结构与组件 ：

PC端 ：GUI程序允许用户输入学号并选择串口，接收来自Arduino的湿度数据并显示。此部分使用`eframe`库实现了GUI界面。

Arduino端 ：Arduino负责读取湿度传感器的数据，将数据通过串口返回给PC，并根据湿度值控制电机。

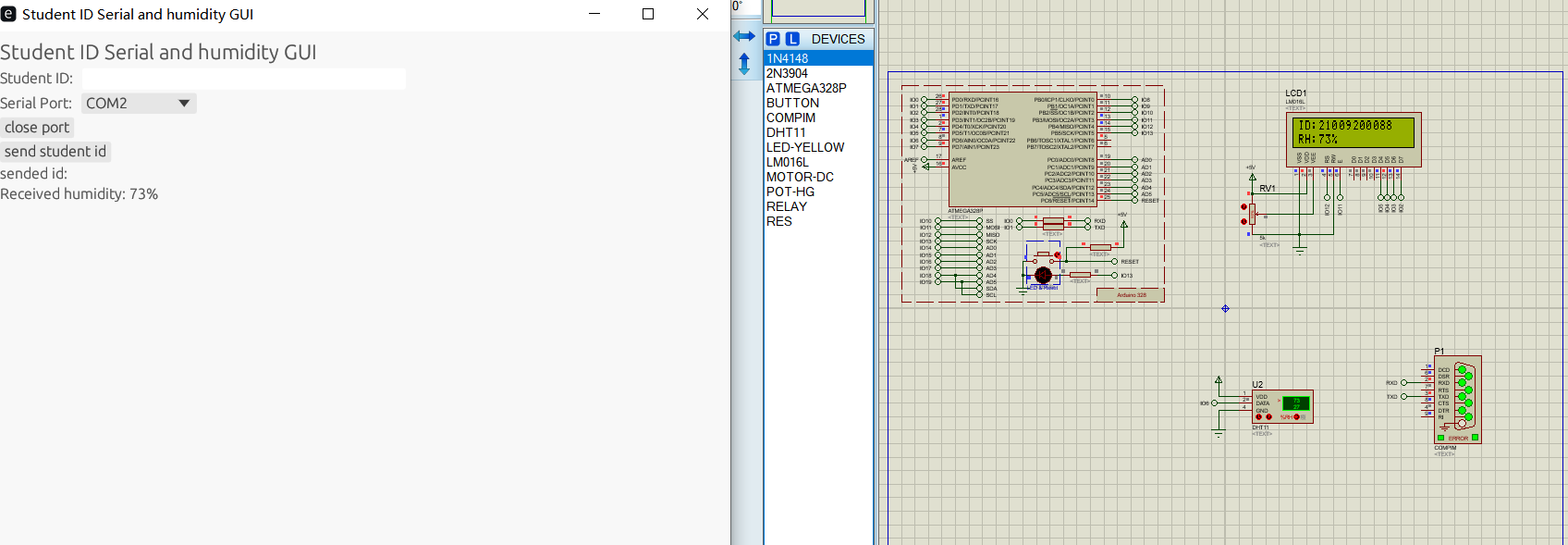
5. 多线程与异步设计 ：

在PC端，为了避免阻塞主线程，串口数据的接收部分是异步的，通过后台线程来不断读取串口数据。

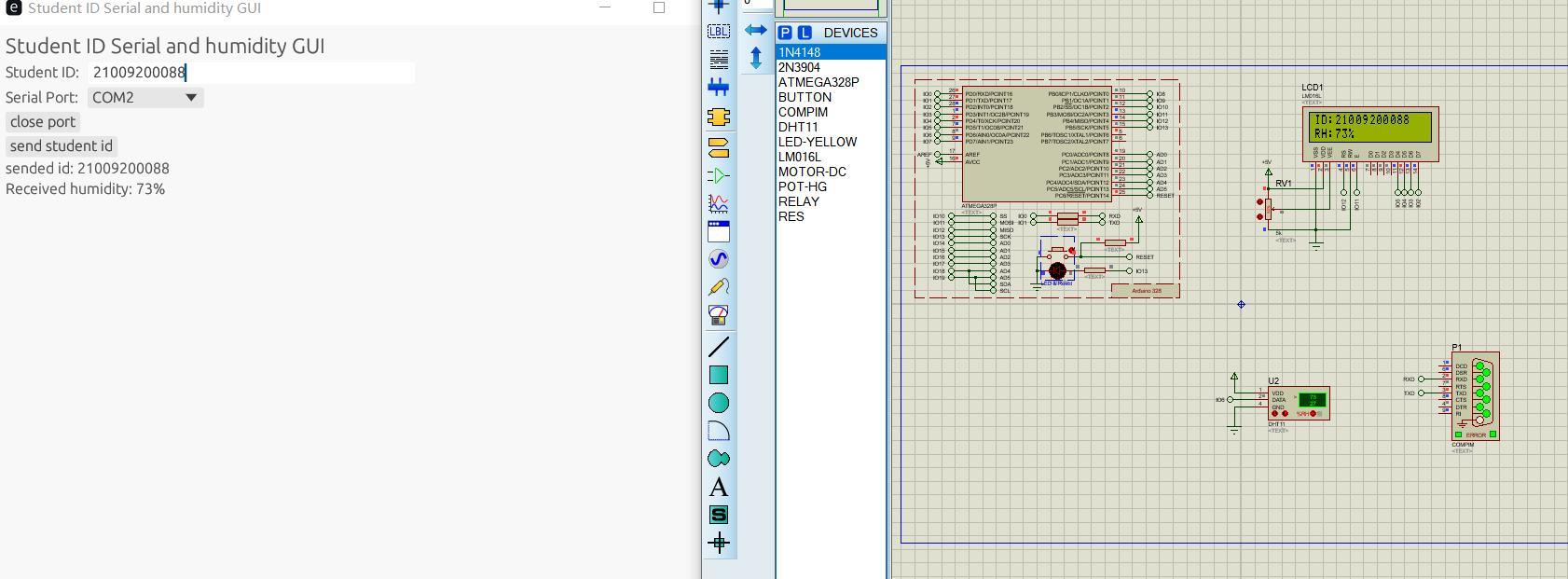
共享数据（如湿度值）通过 `Arc<Mutex>` 进行保护，以保证多个线程可以安全地访问和更新湿度值。

1. 仿真结果展示

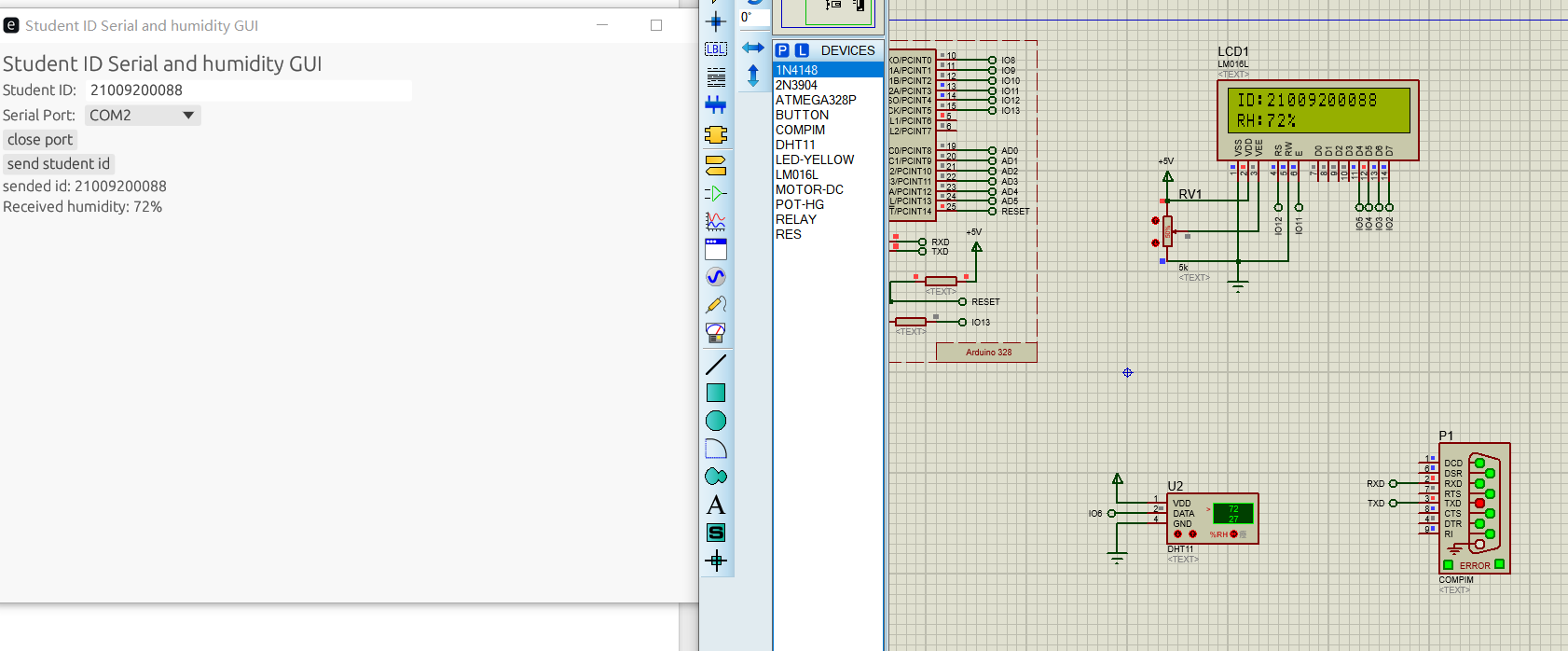
学号发送前：



学号发送后：



调整湿度后：



四、参考文献

1. 程序设计

上位机源码

// 结构体定义

//

struct SerialGuiApp {

student\_id: String, // 存储学号的字符串

humidity\_value: Arc<Mutex<String>>, // 线程安全的共享字符串，用于存储湿度值

port\_list: Vec<String>, // 存储检测到的串口列表

selected\_port: Option<String>, // 当前选中的串口名

port: Option<Arc<Mutex<Box<dyn SerialPort>>>>,// 可选的串口对象，用于线程安全地访问串口

is\_port\_open: bool, // 标志位，表示串口是否已打开

}

//

// 为 SerialGuiApp 实现默认构造函数

//

impl Default for SerialGuiApp {

fn default() > Self {

Self {

student\_id: String::new(), // 初始化空的学号

humidity\_value: Arc::new(Mutex::new(String::new())), // 初始化空的湿度值，并使用 Arc 和 Mutex 来进行线程安全的共享

port\_list: serialport::available\_ports()

.map(|ports| ports.into\_iter().map(|p| p.port\_name).collect())

.unwrap\_or\_else(|\_| vec![]), // 获取系统中可用的串口列表，如果无法获取则返回空列表

selected\_port: None, // 初始化时没有选中任何串口

port: None, // 初始化时没有打开任何串口

is\_port\_open: false, // 初始化时串口未打开

}

}

}

//

// 实现 GUI 应用程序的更新逻辑

//

impl eframe::App for SerialGuiApp {

fn update(&mut self, ctx: &egui::Context, \_frame: &mut eframe::Frame) {

egui::CentralPanel::default().show(ctx, |ui| { // 创建一个中央面板来显示 GUI 元素

ui.heading("Student ID Serial and humidity GUI"); // 设置窗口的标题

// 创建一个水平布局，显示学号输入框

ui.horizontal(|ui| {

ui.label("Student ID:"); // 显示标签 "Student ID:"

ui.text\_edit\_singleline(&mut self.student\_id); // 输入框，允许用户输入学号并存储到 student\_id

});

// 创建一个水平布局，显示串口选择下拉框

ui.horizontal(|ui| {

ui.label("Serial Port:"); // 显示标签 "Serial Port:"

egui::ComboBox::from\_label("") // 创建一个下拉框组件

.selected\_text(self.selected\_port.clone().unwrap\_or\_else(|| "Serial Port".to\_string())) // 显示当前选中的串口，若无则显示 "Serial Port"

.show\_ui(ui, |ui| { // 在 UI 中显示串口选择列表

for port in &self.port\_list { // 遍历可用的串口列表

ui.selectable\_value(&mut self.selected\_port, Some(port.clone()), port); // 用户选择一个串口时，更新 selected\_port

}

});

});

// 创建一个水平布局，显示打开/关闭串口的按钮

ui.horizontal(|ui| {

let button\_text = if self.is\_port\_open { "close port" } else { "open port" }; // 根据串口状态显示不同的按钮文本

if ui.button(button\_text).clicked() { // 如果用户点击了按钮

if self.is\_port\_open { // 如果串口已经打开

self.close\_port(); // 调用 close\_port 方法关闭串口

} else { // 如果串口未打开

if let Some(port\_name) = &self.selected\_port {

self.open\_port(&port\_name.clone()); // 如果有选中的串口，打开它

} else {

eprintln!("please select a serial port first"); // 如果没有选择串口，显示错误提示

}

}

}

});

// 创建发送学号的按钮

if ui.button("send student id").clicked() { // 如果用户点击了发送学号按钮

if self.is\_port\_open { // 如果串口已经打开

self.send\_student\_id(); // 调用 send\_student\_id 方法将学号发送到串口

} else {

ui.label("please open a serial port first"); // 如果串口未打开，提示用户先打开串口

}

}

// 显示发送的学号

ui.label(format!("sended id: {}", self.student\_id)); // 在 GUI 中显示发送的学号

// 显示接收到的湿度值

let humidity = self.humidity\_value.lock().unwrap().clone(); // 从 Mutex 中锁定湿度值并克隆出来

ui.label(format!("Received humidity: {}", humidity)); // 在 GUI 中显示接收到的湿度值

});

ctx.request\_repaint(); // 请求重新绘制界面，保证显示的数据是最新的

}

}

// 串口操作相关方法

//

impl SerialGuiApp {

// 打开串口的方法

fn open\_port(&mut self, port\_name: &str) {

match serialport::new(port\_name, 9600)

.timeout(Duration::from\_millis(1000)) // 设置串口的超时时间为 1000 毫秒

.open() { // 尝试打开指定的串口

Ok(port) => {

self.port = Some(Arc::new(Mutex::new(port))); // 如果成功打开串口，将其存储到 port 中，并使用 Arc 和 Mutex 进行线程安全的共享

self.is\_port\_open = true; // 标记串口为打开状态

self.start\_receiving\_data(); // 启动接收数据的线程

},

Err(e) => {

eprintln!("fail to open port: {}", e); // 如果打开失败，打印错误信息

}

}

}

// 关闭串口的方法

fn close\_port(&mut self) {

self.port = None; // 将串口对象置为空，表示关闭了串口

self.is\_port\_open = false; // 将串口状态设置为关闭

}

// 发送学号到串口的方法

fn send\_student\_id(&mut self) {

if let Some(port) = &self.port { // 如果串口已打开

let mut port = port.lock().unwrap(); // 获取串口的线程安全锁

let \_ = port.write(self.student\_id.as\_bytes()); // 将学号转换为字节数组，并写入串口

}

}

}

//

// 程序入口函数

//

fn main() > Result<(), eframe::Error> {

let options = eframe::NativeOptions::default(); // 配置 eframe 的默认选项

eframe::run\_native(

"Student ID Serial and humidity GUI", // 设置窗口标题

options, // 应用程序选项

Box::new(|\_cc| Box::new(SerialGuiApp::default())), // 初始化 SerialGuiApp 并启动 GUI 应用程序

)

}

//

Arduino程序代码

#include <Wire.h>

#include <dht11.h> // 包含DHT11传感器库

#include <LiquidCrystal.h> // 包含LCD控制库

// 定义引脚

#define DHT11PIN 6 // 定义DHT11传感器连接的引脚

#define EMPIN 7 // 定义电机连接的引脚

// 初始化LCD显示屏，定义其控制和数据引脚

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

dht11 DHT11; // 定义DHT11对象，用于读取湿度

String str = ""; // 用于接收学号的字符串

int length\_, last = 0; // length\_存储字符串长度，last用于保存学号末位

int setHumidity, realHumidity = 0; // 设定的目标湿度和实际湿度值

void setup() {

Serial.begin(9600); // 初始化串口，设置波特率为9600

lcd.begin(16, 2); // 初始化LCD，设置为16列2行

pinMode(EMPIN, OUTPUT); // 设置电机引脚为输出模式

randomSeed(analogRead(0)); // 使用模拟输入初始化随机数种子

}

void loop() {

// 读取DHT11传感器的湿度数据

DHT11.read(DHT11PIN);

realHumidity = (int)DHT11.humidity; // 将湿度值转换为整数并存储

// 保证每次循环生成随机湿度，不依赖串口数据

setHumidity = random(30, 80); // 生成30到80之间的随机湿度值

lcd.print("ID:"); // 在LCD上显示 "ID:"

if (Serial.available() > 0) { // 检查串口是否有数据输入

while (Serial.available() > 0) {

str += (char)Serial.read(); // 逐字符读取串口输入并拼接成字符串

delay(10); // 稍作延时，确保数据完整接收

}

lcd.println(str); // 在LCD上显示学号

str = ""; // 清空字符串，准备下次读取

}

lcd.setCursor(0, 1); // 将光标定位在LCD的第1行

lcd.print("RH:"); // 显示湿度标签

lcd.print(realHumidity); // 显示当前湿度值

lcd.println("%"); // 显示百分号

lcd.home(); // 将光标返回到LCD的左上角

Serial.print("Humidity:"); // 向串口输出湿度信息

Serial.print(realHumidity);

Serial.println("%");

// 根据当前湿度值和设定值控制电机

if (realHumidity <= setHumidity) {

digitalWrite(EMPIN, HIGH); // 当湿度低于目标值时，启动电机

} else {

digitalWrite(EMPIN, LOW); // 否则，关闭电机

}

delay(1000); // 延迟1秒，控制循环的执行频率

}