1. **树莓派智能花盆系统代码(Python)**

**project.py**

1. **import** time
2. **import** RPi.GPIO as GPIO

5. GPIO.setmode(GPIO.BCM)
7. ##--------------------超声波设置-----------------------##
8. makerobo\_TRIG = 23  # 超声波模块Tring控制管脚
9. makerobo\_ECHO = 24  # 超声波模块Echo控制管脚
11. # 超声波模块初始化工作
12. **def** makerobo\_setup():
13. GPIO.setwarnings(False)       # 忽略GPIO操作注意警告
14. GPIO.setup(makerobo\_TRIG, GPIO.OUT) # Tring设置为输出模式
15. GPIO.setup(makerobo\_ECHO, GPIO.IN)  # Echo设置为输入模式
17. # 超声波计算距离函数
18. **def** ur\_disMeasure():
20. GPIO.output(makerobo\_TRIG, 0)  # 开始起始
21. time.sleep(0.000002)           # 延时2us
23. GPIO.output(makerobo\_TRIG, 1)  # 超声波启动信号，延时10us
24. time.sleep(0.00001)            # 发出超声波脉冲
25. GPIO.output(makerobo\_TRIG, 0)           # 设置为低电平

28. **while** GPIO.input(makerobo\_ECHO) == 0: # 等待回传信号
29. us\_a = 0
30. us\_time1 = time.time()                # 获取当前时间
31. **while** GPIO.input(makerobo\_ECHO) == 1: # 回传信号截止信息
32. us\_a = 1
33. us\_time2 = time.time()                # 获取当前时间
35. us\_during = us\_time2 - us\_time1          # 转换微秒级的时间
37. # 声速在空气中的传播速度为340m/s, 超声波要经历一个发送信号和一个回波信息，
38. # 计算公式如下所示：
39. **return** us\_during \* 340 / 2 \* 100        # 求出距离
41. ##--------------------蜂鸣器----------------------##
42. makerobo\_Buzzer = 17    # 有源蜂鸣器管脚定义
44. # GPIO设置函数
45. **def** makerobo\_setup2():
46. GPIO.setwarnings(False)                       # 关闭GPIO警告提示
47. GPIO.setup(makerobo\_Buzzer, GPIO.OUT)     # 设置有源蜂鸣器管脚为输出模式
48. GPIO.output(makerobo\_Buzzer, GPIO.HIGH)   # 蜂鸣器设置为高电平，关闭蜂鸟器
50. #  打开蜂鸣器
51. **def** makerobo\_buzzer\_on():
52. GPIO.output(makerobo\_Buzzer, GPIO.LOW)  # 蜂鸣器为低电平触发，所以使能蜂鸣器让其发声
53. # 关闭蜂鸣器
54. **def** makerobo\_buzzer\_off():
55. GPIO.output(makerobo\_Buzzer, GPIO.HIGH) # 蜂鸣器设置为高电平，关闭蜂鸣器
57. # 控制蜂鸣器鸣叫
58. **def** makerobo\_beep(x):
59. makerobo\_buzzer\_on()     # 打开蜂鸣器控制
60. time.sleep(x)            # 延时时间
61. makerobo\_buzzer\_off()    # 关闭蜂鸣器控制
62. time.sleep(x)            # 延时时间
64. **def** destroy():
65. GPIO.output(makerobo\_Buzzer, GPIO.HIGH) # 关闭蜂鸣器鸣叫
66. GPIO.cleanup()                     # 释放资源
68. ##-----------------雨滴传感器-------------------##
69. # 初始状态，假设开始时不下雨
70. is\_raining = 0
72. # 用于跟踪
73. printed\_rain = True
74. printed\_no\_rain = False
76. # 使用树莓派引脚编号
77. GPIO.setmode(GPIO.BCM)
79. # 定义你的引脚编号
80. rain\_pin = 12
82. # 设置引脚为输入模式
83. GPIO.setup(rain\_pin, GPIO.IN)
85. ## -----------------PWM控制RGB灯----------------##
86. # 定义你的引脚编号
87. R\_pin = 25
88. G\_pin = 7
89. B\_pin = 8
91. # 设置引脚为输出模式
92. GPIO.setup(R\_pin, GPIO.OUT)
93. GPIO.setup(G\_pin, GPIO.OUT)
94. GPIO.setup(B\_pin, GPIO.OUT)
96. # 设置频率为100Hz,并初始化 PWM 对象
97. R\_PWM = GPIO.PWM(R\_pin, 100)
98. G\_PWM = GPIO.PWM(G\_pin, 100)
99. B\_PWM = GPIO.PWM(B\_pin, 100)
101. # 启动PWM，初始占空比为 0（灯关闭）
102. R\_PWM.start(0)
103. G\_PWM.start(0)
104. B\_PWM.start(0)
106. # 定义你想要的颜色（0到100之间的值:占空比的百分比）
107. # 要显示纯红色，设为 (100, 0, 0)；纯绿色为 (0, 100, 0)；纯蓝色为 (0, 0, 100)
108. R\_value = 0
109. G\_value = 0
110. B\_value = 100
112. ##-------------------舵机控制------------------##
113. servo\_pin1 = 20  # 控制平移旋转的舵机
114. servo\_pin2 = 21  # 控制俯仰旋转的舵机
116. GPIO.setup(servo\_pin1, GPIO.OUT)
117. GPIO.setup(servo\_pin2, GPIO.OUT)
119. # 设置舵机的频率为50Hz
120. pwm1 = GPIO.PWM(servo\_pin1, 50)
121. pwm2 = GPIO.PWM(servo\_pin2, 50)
123. # 启动PWM，初始占空比为0（舵机位置为0度）
124. pwm1.start(0)
125. pwm2.start(0)
127. **def** set2\_angle(angle):
128. duty = angle / 18 + 2
129. GPIO.output(servo\_pin2, True)
130. pwm2.ChangeDutyCycle(duty)
131. time.sleep(1)
132. GPIO.output(servo\_pin2, False)
133. pwm2.ChangeDutyCycle(0)
135. **try**:
136. GPIO.setwarnings(False)
137. # 调用蜂鸣器、超声波初始化函数
138. makerobo\_setup()
139. makerobo\_setup2()
140. **while** True:
142. # 超声波检测水位
143. us\_dis = ur\_disMeasure()   # 获取超声波计算距离
144. **print** (us\_dis, 'cm')       # 打印超声波距离值
145. **print** ('')
146. **if** us\_dis >=11:
147. makerobo\_beep(0.2)  # 控制蜂鸣器鸣叫，延时时间为500ms
148. makerobo\_beep(0.2)  # 控制蜂鸣器鸣叫，延时时间为500ms
149. makerobo\_beep(0.2)  # 控制蜂鸣器鸣叫，延时时间为500ms
150. makerobo\_beep(0.2)  # 控制蜂鸣器鸣叫，延时时间为500ms
151. makerobo\_beep(0.2)  # 控制蜂鸣器鸣叫，延时时间为500ms
153. # 读取雨滴传感器状态
154. **if** GPIO.input(rain\_pin) == 0:
155. is\_raining = GPIO.input(rain\_pin)
156. **print**("It's raining!")
157. # 设置 PWM 占空比，点亮RGB小灯
158. R\_PWM.ChangeDutyCycle(R\_value)
159. G\_PWM.ChangeDutyCycle(G\_value)
160. B\_PWM.ChangeDutyCycle(B\_value)
162. **else**:
163. is\_raining = GPIO.input(rain\_pin)
164. **print**("It's sunny!")
165. # 熄灭小灯
166. R\_PWM.ChangeDutyCycle(0)
167. G\_PWM.ChangeDutyCycle(0)
168. B\_PWM.ChangeDutyCycle(0)
170. **if** is\_raining == 1:
171. **if** **not** printed\_rain:
172. set2\_angle(180)
173. printed\_rain = True
174. printed\_no\_rain = False
175. **else**:
176. **if** **not** printed\_no\_rain:
177. set2\_angle(0)
178. printed\_no\_rain = True
179. printed\_rain = False
181. time.sleep(2)
183. **except** KeyboardInterrupt:  # 按下Ctrl + C时结束程序
184. # 清理GPIO设置
185. GPIO.cleanup()
186. # 关闭蜂鸣器鸣叫
187. GPIO.output(makerobo\_Buzzer, GPIO.HIGH)
188. R\_PWM.stop()
189. G\_PWM.stop()
190. B\_PWM.stop()
191. pwm1.stop()
192. pwm2.stop()

**light.py**

1. **import** time
2. **import** RPi.GPIO as GPIO
3. **from** rpi\_ws281x **import** PixelStrip, Color
4. **import** socket
6. ##-------------------------LED strip配置-----------------##
7. LED\_COUNT = 60        # 灯带上LED灯的数量
8. LED\_PIN = 18        # GPIO引脚号
9. LED\_FREQ\_HZ = 800000  # LED灯的频率
10. LED\_DMA = 10          # DMA通道号
11. LED\_BRIGHTNESS = 255  # 亮度值，取值范围0-255
12. LED\_INVERT = False    # 反转信号
14. # 按钮引脚配置
15. BUTTON\_PIN = 4
17. # 创建PixelStrip对象
18. strip = PixelStrip(LED\_COUNT, LED\_PIN, LED\_FREQ\_HZ, LED\_DMA, LED\_INVERT, LED\_BRIGHTNESS)
19. # 初始化灯带
20. strip.begin()
22. GPIO.setmode(GPIO.BCM)
24. # 设置按钮引脚为输入模式，上拉电阻使之默认为高电平
25. GPIO.setup(BUTTON\_PIN, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)
27. **def** turn\_on(strip, value1,value2,value3):
28. """将LED设置为指定的颜色."""
29. color=Color(value1, value2, value3)
30. **for** i **in** range(strip.numPixels()):
31. strip.setPixelColor(i, color)
32. strip.show()
34. **def** turn\_off(strip):
35. """关闭LED."""
36. **for** i **in** range(strip.numPixels()):
37. strip.setPixelColor(i, Color(0, 0, 0))
38. strip.show()
40. ##-----------------------TCP/IP----------------------------##
41. # 创建socket对象
42. s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)
44. # 绑定到本地的8000端口
45. s.bind(('0.0.0.0', 8000))
47. # 开始监听连接
48. s.listen(1)
50. **def** handle\_client(client\_socket):
51. **while** True:
52. # 接收客户端的数据
53. data = client\_socket.recv(1024)
55. **if** **not** data:
56. **break**
58. # 解析数据为数字
59. brightness = int(data.decode('utf-8'))
60. **print**(brightness)
62. # 设置LED的亮度
63. strip.setBrightness(brightness)
65. turn\_on(strip,255,0,255)
67. **try**:
68. **while** True:
69. # 等待一个客户端的连接
70. **print**("waiting...")
71. client\_socket, addr = s.accept()
73. # 处理客户端的连接
74. handle\_client(client\_socket)
76. **except** KeyboardInterrupt:  # 按下Ctrl + C时结束程序
77. # 清理GPIO设置
78. GPIO.cleanup()
79. # 关闭灯带
80. turn\_off(strip)
81. # 关闭socket
82. s.close()
83. **GUI软件系统代码(Python)**

**gui.py**

1. **import** RPi.GPIO as GPIO
2. **import** time
3. **from** tkinter **import** \*
4. **from** tkinter **import** messagebox
5. **from** PIL **import** Image, ImageTk
7. #####-------------------------------树莓派部分--------------------------------#####
9. ###----------超声波设置
10. makerobo\_TRIG = 23  # 超声波模块Tring控制管脚
11. makerobo\_ECHO = 24  # 超声波模块Echo控制管脚
13. # 超声波模块初始化工作
14. **def** makerobo\_setup():
15. GPIO.setmode(GPIO.BCM)
16. GPIO.setwarnings(False)       # 忽略GPIO操作注意警告
17. GPIO.setup(makerobo\_TRIG, GPIO.OUT) # Tring设置为输出模式
18. GPIO.setup(makerobo\_ECHO, GPIO.IN)  # Echo设置为输入模式
20. # 超声波计算距离函数
21. **def** ur\_disMeasure():
23. GPIO.output(makerobo\_TRIG, 0)  # 开始起始
24. time.sleep(0.000002)           # 延时2us
26. GPIO.output(makerobo\_TRIG, 1)  # 超声波启动信号，延时10us
27. time.sleep(0.00001)            # 发出超声波脉冲
28. GPIO.output(makerobo\_TRIG, 0)           # 设置为低电平

31. **while** GPIO.input(makerobo\_ECHO) == 0: # 等待回传信号
32. us\_a = 0
33. us\_time1 = time.time()                # 获取当前时间
34. **while** GPIO.input(makerobo\_ECHO) == 1: # 回传信号截止信息
35. us\_a = 1
36. us\_time2 = time.time()                # 获取当前时间
38. us\_during = us\_time2 - us\_time1          # 转换微秒级的时间
40. # 声速在空气中的传播速度为340m/s, 超声波要经历一个发送信号和一个回波信息，
41. # 计算公式如下所示：
42. **return** us\_during \* 340 / 2 \* 100        # 求出距离
44. ###----------蜂鸣器
45. makerobo\_Buzzer = 16    # 有源蜂鸣器管脚定义
47. # GPIO设置函数
48. **def** makerobo\_setup2():
49. GPIO.setwarnings(False)                       # 关闭GPIO警告提示
50. GPIO.setup(makerobo\_Buzzer, GPIO.OUT)     # 设置有源蜂鸣器管脚为输出模式
51. GPIO.output(makerobo\_Buzzer, GPIO.HIGH)   # 蜂鸣器设置为高电平，关闭蜂鸟器
53. #  打开蜂鸣器
54. **def** makerobo\_buzzer\_on():
55. GPIO.output(makerobo\_Buzzer, GPIO.LOW)  # 蜂鸣器为低电平触发，所以使能蜂鸣器让其发声
56. # 关闭蜂鸣器
57. **def** makerobo\_buzzer\_off():
58. GPIO.output(makerobo\_Buzzer, GPIO.HIGH) # 蜂鸣器设置为高电平，关闭蜂鸣器
60. # 控制蜂鸣器鸣叫
61. **def** makerobo\_beep(x):
62. makerobo\_buzzer\_on()     # 打开蜂鸣器控制
63. time.sleep(x)            # 延时时间
64. makerobo\_buzzer\_off()    # 关闭蜂鸣器控制
66. **def** destroy():
67. GPIO.output(makerobo\_Buzzer, GPIO.HIGH) # 关闭蜂鸣器鸣叫
68. GPIO.cleanup()                     # 释放资源

71. ###---------根据距离判断是否发出警报
72. # 循环函数
73. **def** makerobo\_loop():
74. **try**:
75. **global** raw\_data
76. **while** True:
77. us\_dis = ur\_disMeasure()   # 获取超声波计算距离
78. us\_dis = 13.7 - us\_dis
79. **if** us\_dis <=2:
80. makerobo\_beep(0.2)  # 控制蜂鸣器鸣叫，延时时间为200ms
81. makerobo\_beep(0.2)
82. with lock:
83. raw\_data = us\_dis
84. time.sleep(1)            # 延时1000ms
85. **except** KeyboardInterrupt:  # 按下Ctrl + C时结束程序
86. destroy()
88. **def** init1():
89. ##-----------------雨滴传感器-------------------##
90. # 初始状态，假设开始时不下雨
91. is\_raining = 0
93. # 用于跟踪
94. printed\_rain = True
95. printed\_no\_rain = False
97. # 使用树莓派引脚编号
98. GPIO.setmode(GPIO.BCM)
100. # 定义你的引脚编号
101. rain\_pin = 12
103. # 设置引脚为输入模式
104. GPIO.setup(rain\_pin, GPIO.IN)
106. ## -----------------PWM控制RGB灯----------------##
107. # 定义你的引脚编号
108. R\_pin = 25
109. G\_pin = 7
110. B\_pin = 8
112. # 设置引脚为输出模式
113. GPIO.setup(R\_pin, GPIO.OUT)
114. GPIO.setup(G\_pin, GPIO.OUT)
115. GPIO.setup(B\_pin, GPIO.OUT)
117. # 设置频率为100Hz,并初始化 PWM 对象
118. R\_PWM = GPIO.PWM(R\_pin, 100)
119. G\_PWM = GPIO.PWM(G\_pin, 100)
120. B\_PWM = GPIO.PWM(B\_pin, 100)
122. # 启动PWM，初始占空比为 0（灯关闭）
123. R\_PWM.start(0)
124. G\_PWM.start(0)
125. B\_PWM.start(0)
127. # 定义你想要的颜色（0到100之间的值:占空比的百分比）
128. # 要显示纯红色，设为 (100, 0, 0)；纯绿色为 (0, 100, 0)；纯蓝色为 (0, 0, 100)
129. R\_value = 0
130. G\_value = 0
131. B\_value = 100
133. ##-------------------舵机控制------------------##
134. servo\_pin1 = 20  # 控制平移旋转的舵机
135. servo\_pin2 = 21  # 控制俯仰旋转的舵机
137. GPIO.setup(servo\_pin1, GPIO.OUT)
138. GPIO.setup(servo\_pin2, GPIO.OUT)
140. # 设置舵机的频率为50Hz
141. pwm1 = GPIO.PWM(servo\_pin1, 50)
142. pwm2 = GPIO.PWM(servo\_pin2, 50)
144. # 启动PWM，初始占空比为0（舵机位置为0度）
145. pwm1.start(0)
146. pwm2.start(0)
148. **def** set2\_angle(angle):
149. duty = angle / 18 + 2
150. GPIO.output(servo\_pin2, True)
151. pwm2.ChangeDutyCycle(duty)
152. time.sleep(1)
153. GPIO.output(servo\_pin2, False)
154. pwm2.ChangeDutyCycle(0)
156. **def** read\_data():
157. **global** state
158. **try**:
159. # 读取雨滴传感器状态
160. **if** GPIO.input(rain\_pin) == 0:
161. is\_raining = GPIO.input(rain\_pin)
162. stste="正在下雨!"
163. # 设置 PWM 占空比，点亮RGB小灯
164. R\_PWM.ChangeDutyCycle(R\_value)
165. G\_PWM.ChangeDutyCycle(G\_value)
166. B\_PWM.ChangeDutyCycle(B\_value)
168. **else**:
169. is\_raining = GPIO.input(rain\_pin)
170. state="晴天!"
171. # 熄灭小灯
172. R\_PWM.ChangeDutyCycle(0)
173. G\_PWM.ChangeDutyCycle(0)
174. B\_PWM.ChangeDutyCycle(0)
176. **if** is\_raining == 1:
177. **if** **not** printed\_rain:
178. set2\_angle(180)
179. printed\_rain = True
180. printed\_no\_rain = False
181. **else**:
182. **if** **not** printed\_no\_rain:
183. set2\_angle(0)
184. printed\_no\_rain = True
185. printed\_rain = False
186. time.sleep(2)
187. **except** KeyboardInterrupt:  # 按下Ctrl + C时结束程序
188. destory()
189. R\_PWM.stop()
190. G\_PWM.stop()
191. B\_PWM.stop()
192. pwm1.stop()
193. pwm2.stop()
195. **def** init2():
196. # LED strip配置
197. LED\_COUNT = 60        # 灯带上LED灯的数量
198. LED\_PIN = 18        # GPIO引脚号
199. LED\_FREQ\_HZ = 800000  # LED灯的频率
200. LED\_DMA = 10          # DMA通道号
201. LED\_BRIGHTNESS = 255  # 亮度值，取值范围0-255
202. LED\_INVERT = False    # 反转信号
204. # 按钮引脚配置
205. BUTTON\_PIN = 4
207. # 创建PixelStrip对象
208. strip = PixelStrip(LED\_COUNT, LED\_PIN, LED\_FREQ\_HZ, LED\_DMA, LED\_INVERT, LED\_BRIGHTNESS)
209. # 初始化灯带
210. strip.begin()
212. GPIO.setmode(GPIO.BCM)
214. # 设置按钮引脚为输入模式，上拉电阻使之默认为高电平
215. GPIO.setup(BUTTON\_PIN, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP)
217. #Flag
218. buttonState = 0
219. beforeState = 0
220. #isLighting = False
221. brightness = 1
223. **def** turn\_on(strip, value1,value2,value3):
224. """将LED设置为指定的颜色."""
225. color=Color(value1, value2, value3)
226. **for** i **in** range(strip.numPixels()):
227. strip.setPixelColor(i, color)
228. strip.show()
230. **def** turn\_off(strip):
231. """关闭LED."""
232. **for** i **in** range(strip.numPixels()):
233. strip.setPixelColor(i, Color(0, 0, 0))
234. strip.show()
236. **def** set\_brightness(value):
237. **if** value == 0:
238. strip.setBrightness(0)
239. **elif** value == 1:
240. strip.setBrightness(35)
241. **elif** value == 2:
242. strip.setBrightness(100)
243. **else**:
244. strip.setBrightness(255)
246. **def** data\_read():
247. **try**:
248. **while** True:
249. #print(brightness)
250. buttonState = GPIO.input(BUTTON\_PIN)
251. **if** buttonState == 1 **and** beforeState == 0:  #pressed down the moment
252. set\_brightness(brightness)
253. turn\_on(strip,255,0,255)
254. brightness = brightness + 1
255. **if** brightness == 4:
256. brightness = 0
257. **if** buttonState != beforeState:
258. time.sleep(0.05)
259. beforeState = buttonState
261. **except** KeyboardInterrupt:  # 按下Ctrl + C时结束程序
262. # 清理GPIO设置
263. GPIO.cleanup()
264. # 关闭灯带
265. turn\_off(strip)
267. ##-----空气温湿度------------##
268. # 传感器类型和引脚编号
269. sensor = Adafruit\_DHT.DHT11
270. pin = 13
272. # 读取传感器数据
273. humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin)
275. # 获得数据
276. **def** get\_data():
277. **global** tdata
278. **global** hdata
279. tdata = temperature
280. hdata = humidity
282. #####-------------------------------GUI部分-----------------------------------#####
283. # 全局变量
284. username = ""
285. password = ""
287. # 登陆界面
288. **def** login():
289. **global** username, password
290. login\_window = Tk()
291. login\_window.title("登陆")
293. # 添加背景图
295. bg\_image = Image.open("1.png")
296. bg\_photo = ImageTk.PhotoImage(bg\_image)
297. canvas = Canvas(login\_window, width=bg\_image.width, height=bg\_image.height)
298. canvas.pack(fill=BOTH, expand=YES)
299. canvas.create\_image(0, 0, image=bg\_photo, anchor=NW)
301. # 用户名标签和输入框
302. lbl\_username = Label(canvas, text="用户名", bg="#F0E68C")
303. lbl\_username.place(relx=0.3, rely=0.4, anchor=CENTER)
304. txt\_username = Entry(canvas, width=20)
305. txt\_username.place(relx=0.5, rely=0.4, anchor=CENTER)
307. # 密码标签和输入框
308. lbl\_password = Label(canvas, text="密码", bg="#F0E68C")
309. lbl\_password.place(relx=0.3, rely=0.5, anchor=CENTER)
310. txt\_password = Entry(canvas, show="\*", width=20)
311. txt\_password.place(relx=0.5, rely=0.5, anchor=CENTER)
313. # 登陆按钮
314. **def** login\_action():
315. **global** username, password
316. username = txt\_username.get()
317. password = txt\_password.get()
318. **if** username == "" or password == "":
319. messagebox.showerror("错误", "用户名和密码不能为空")
320. **else**:
321. messagebox.showinfo("提示", f"欢迎回来，{username}！")
322. login\_window.destroy()
324. btn\_login = Button(canvas, text="登陆", command=login\_action)
325. btn\_login.place(relx=0.35, rely=0.6, anchor=CENTER)
327. # 注册按钮
328. **def** register\_action():
329. **global** username, password
330. username = txt\_username.get()
331. password = txt\_password.get()
332. **if** username == "" or password == "":
333. messagebox.showerror("错误", "用户名和密码不能为空")
334. **else**:
335. messagebox.showinfo("提示", f"注册成功，欢迎加入我们，{username}！")
337. btn\_register = Button(canvas, text="注册", command=register\_action)
338. btn\_register.place(relx=0.55, rely=0.6, anchor=CENTER)
340. login\_window.mainloop()
342. # 自动浇花系统交互界面
343. **def** auto\_watering():
344. **global** username, password
345. **if** username == "" or password == "":
346. messagebox.showerror("错误", "请先登陆或注册")
347. login()
349. # 创建窗口
350. window = Tk()
351. window.title("自动浇花系统")
353. # 添加背景图
354. bg\_image = Image.open("2.png")
355. bg\_photo = ImageTk.PhotoImage(bg\_image)
356. canvas = Canvas(window, width=bg\_image.width, height=bg\_image.height)
357. canvas.pack(fill=BOTH, expand=YES)
358. canvas.create\_image(0, 0, image=bg\_photo, anchor=NW)
360. # 添加标签
361. lbl = Label(canvas, text=f"欢迎使用自动浇花系统，{username}！", font=("Arial Bold", 24))
362. lbl.place(relx=0.5, rely=0.1, anchor=CENTER)
364. # 添加按钮
365. **def** full\_auto():
366. # 初始化
367. init1()
368. init2()
369. makerobo\_setup()
370. makerobo\_setup2()
371. full\_autor\_window = Toplevel()
372. full\_auto\_window.title("全自动浇花界面")
373. # 背景
374. bg\_image2 = Image.open("11.jpg")
375. bg\_photo2 = ImageTk.PhotoImage(bg\_image2)
376. full\_autor\_canvas = Canvas(full\_autor\_window, width=bg\_image2.width, height=bg\_image2.height)
377. full\_autor\_canvas.pack(fill=BOTH, expand=YES)
378. full\_autor\_canvas.create\_image(0, 0, image=bg\_photo2, anchor=NW)
379. **print**('调用全自动浇花函数')
380. tip\_label = Label(full\_autor\_window, text="提示：请按下ctrl+C中断: ",font=("Arial", 20))
381. tip\_label.place(relx=0, rely=0.1, anchor=NW)
382. label = Label(check\_weather\_window, text="当前天气情况: ",font=("Arial", 20))
383. label.place(relx=0, rely=0.3, anchor=NW)
384. **def** update\_lb():
385. **global** raw\_data
386. **global** state  # 声明全局变量
387. label.config(text="当前天气情况:".format(raw\_data))
388. label.config(text="当前水位情况:".format(state))
389. full\_autor\_window.after(1000, update\_lb)  # 每隔1秒钟更新一次文本
390. t1 = threading.Thread(target=read\_data)
391. t2 = threading.Thread(target=update\_lb)
392. t1.start()
393. t2.start()
394. btn\_full\_auto = Button(canvas, text="全自动",font=("Arial", 18), bg="green", fg="white", command=full\_auto)
395. btn\_full\_auto.place(relx=0.35, rely=0.4, anchor=NW)
397. **def** semi\_auto():
398. semi\_auto\_window = Toplevel()
399. semi\_auto\_window.title("半自动")
401. bg\_image2 = Image.open("3.png")
402. bg\_photo2 = ImageTk.PhotoImage(bg\_image2)
403. semi\_auto\_canvas = Canvas(semi\_auto\_window, width=bg\_image2.width, height=bg\_image2.height)
404. semi\_auto\_canvas.pack(fill=BOTH, expand=YES)
405. semi\_auto\_canvas.create\_image(0, 0, image=bg\_photo2, anchor=NW)
407. # 查看晴雨天
408. **def** check\_weather():
409. # 初始化
410. init1()
411. # 创建窗口
412. check\_weather\_window = Toplevel()
413. check\_weather\_window.title("查看晴雨天状况")
414. # 背景
415. bg\_image3 = Image.open("4.jpg")
416. bg\_photo3 = ImageTk.PhotoImage(bg\_image3)
417. check\_water\_level\_canvas = Canvas(check\_weather\_window, width=bg\_image3.width, height=bg\_image3.height)
418. check\_water\_level\_canvas.pack(fill=BOTH, expand=YES)
419. check\_water\_level\_canvas.create\_image(0, 0, image=bg\_photo3, anchor=NW)
420. **print**('调用查看晴雨天函数')
421. tip\_label = Label(check\_weather\_window, text="提示：请按下ctrl+C中断: ",font=("Arial", 20))
422. tip\_label.place(relx=0, rely=0.1, anchor=NW)
423. label = Label(check\_weather\_window, text="当前天气情况: ",font=("Arial", 20))
424. label.place(relx=0, rely=0.3, anchor=NW)
425. **def** update\_lb():
426. **global** state  # 声明全局变量
427. label.config(text="当前天气情况:".format(state))
428. check\_weather\_window.after(1000, update\_lb)  # 每隔1秒钟更新一次文本
429. t1 = threading.Thread(target=read\_data)
430. t2 = threading.Thread(target=update\_lb)
431. t1.start()
432. t2.start()
433. check\_weather\_window.mainloop()
434. btn\_check\_weather = Button(semi\_auto\_window, text="查看晴雨天",font=("Arial", 18), command=check\_weather)
435. btn\_check\_weather.place(relx=0.3, rely=0.3, anchor=NW)
437. # 查看土壤湿度
438. **def** check\_soil\_humidity():
439. # 调用查看土壤湿度函数
440. # 创建窗口
441. check\_soil\_humidity\_window = Toplevel()
442. check\_soil\_humidity\_window.title("查看土壤湿度")
443. # 背景
444. bg\_image3 = Image.open("5.jpg")
445. bg\_photo3 = ImageTk.PhotoImage(bg\_image3)
446. check\_soil\_humidity\_canvas = Canvas(check\_soil\_humidity\_window, width=bg\_image3.width, height=bg\_image3.height)
447. check\_soil\_humidity\_canvas.pack(fill=BOTH, expand=YES)
448. check\_soil\_humidity\_canvas.create\_image(0, 0, image=bg\_photo3, anchor=NW)
449. **print**('调用土壤湿度检测函数')
450. tip\_label = Label(check\_soil\_humidity\_window, text="提示：土壤湿度在（17,44）范围内最适宜植物生长",font=("Arial", 20))
451. tip\_label.place(relx=0, rely=0, anchor=NW)
452. **def** water():
453. messagebox.showinfo("提示", "开始浇水！")
454. btn\_control\_on = Button(check\_soil\_humidity\_window, text="开始抽水",font=("Arial", 18), command=water)
455. btn\_control\_on.place(relx=0, rely=0.1, anchor=NW)
456. # 创建历史数据查询按钮
457. **def** show\_history\_data():
458. # 创建历史数据查询窗口
459. history\_window = Toplevel(check\_soil\_humidity\_window)
460. history\_window.title('历史数据查询')
462. # 创建日历组件
463. cal = Calendar(history\_window)
464. cal.pack(padx=10, pady=10)
466. # 设置一周的第一天为星期日
467. calendar.setfirstweekday(calendar.SUNDAY)
468. # 创建查询按钮
469. **def** query\_data():
470. # 获取选择的日期
471. selected\_date = cal.selection\_get()
473. # 查询数据
474. data = []
475. with open('humidity\_data.txt', 'r') as f:
476. **for** line **in** f:
477. date\_str, humidity\_str = line.strip().split(',')
478. date = datetime.strptime(date\_str, '%Y-%m-%d %H:%M:%S')
479. **if** date.date() == selected\_date:
480. data.append((date, float(humidity\_str)))
482. # 显示查询结果
483. result\_text.delete('1.0', END)
484. **if** len(data) == 0:
485. result\_text.insert(END, '没有查询到数据！')
486. **else**:
487. **for** item **in** data:
488. result\_text.insert(END, '{}: {:.4f}\n'.format(item[0].strftime('%Y-%m-%d %H:%M:%S'), item[1]))
490. query\_button = Button(history\_window, text='查询', command=query\_data)
491. query\_button.pack(padx=10, pady=10)
493. # 创建显示查询结果的文本框
494. result\_text = Text(history\_window, width=30, height=10)
495. result\_text.pack(padx=10, pady=10)
496. history\_window.mainloop()
497. btn\_history = Button(check\_soil\_humidity\_window, text="历史数据查询",font=("Arial", 18), command=show\_history\_data)
498. btn\_history.place(relx=0.25, rely=0.1, anchor=NW)
499. label = Label(check\_soil\_humidity\_window, text="当前土壤湿度为：",font=("Arial", 20))
500. label.place(relx=0, rely=0.2, anchor=NW)
502. check\_soil\_humidity\_window.mainloop()
504. btn\_check\_soil\_humidity = Button(semi\_auto\_window, text="查看土壤湿度",font=("Arial", 18), command=check\_soil\_humidity)
505. btn\_check\_soil\_humidity.place(relx=0.55, rely=0.3, anchor=NW)
507. # 手动开关照明设备
508. **def** control\_light():
509. # 初始化
510. init2()
511. # 创建窗口
512. control\_light\_window = Toplevel()
513. control\_light\_window.title("手动开关照明")
514. # 背景
515. bg\_image3 = Image.open("4.jpg")
516. bg\_photo3 = ImageTk.PhotoImage(bg\_image3)
517. control\_light\_canvas = Canvas(control\_light\_window, width=bg\_image3.width, height=bg\_image3.height)
518. control\_light\_canvas.pack(fill=BOTH, expand=YES)
519. control\_light\_canvas.create\_image(0, 0, image=bg\_photo3, anchor=NW)
520. **print**('调用手动开关照明函数')
521. tip\_label = Label(control\_light\_window, text="提示：点击按钮控制开关",font=("Arial", 20))
522. tip\_label.place(relx=0, rely=0.1, anchor=NW)
523. **def** control\_on():
524. messagebox.showinfo("提示", "已打开")
525. btn\_control\_on = Button(control\_light\_window, text="打开",font=("Arial", 20), command=control\_on)
526. btn\_control\_off.place(relx=0.35, rely=0.5, anchor=NW)
527. **def** control\_off():
528. messagebox.showinfo("提示", "已关闭")
529. btn\_control\_off = Button(control\_light\_window, text="关闭",font=("Arial", 20), command=control\_off)
530. btn\_control\_on.place(relx=0.55, rely=0.5, anchor=NW)
531. control\_light\_window.mainloop()
532. btn\_control\_light = Button(semi\_auto\_window, text="手动开关照明设备",font=("Arial", 18), command=control\_light)
533. btn\_control\_light.place(relx=0.3, rely=0.5, anchor=NW)
534. **def** check\_water\_level():
535. # 初始化
536. makerobo\_setup()
537. makerobo\_setup2()
538. # 创建窗口
539. check\_water\_level\_window = Toplevel()
540. check\_water\_level\_window.title("查看蓄水池水位状况")
541. # 背景
542. bg\_image3 = Image.open("4.jpg")
543. bg\_photo3 = ImageTk.PhotoImage(bg\_image3)
544. check\_water\_level\_canvas = Canvas(check\_water\_level\_window, width=bg\_image3.width, height=bg\_image3.height)
545. check\_water\_level\_canvas.pack(fill=BOTH, expand=YES)
546. check\_water\_level\_canvas.create\_image(0, 0, image=bg\_photo3, anchor=NW)
547. **print**('调用超声波测距函数')
548. tip\_label = Label(check\_water\_level\_window, text="提示：请按下ctrl+C中断: ")
549. tip\_label.place(relx=0, rely=0.1, anchor=NW)
550. label = Label(check\_water\_level\_window, text="当前水位为: ",font=("Arial", 20))
551. label.place(relx=0, rely=0.3, anchor=NW)
552. **def** update\_label():
553. **global** raw\_data  # 声明全局变量
554. label.config(text="当前水位为:".format(raw\_data))
555. check\_water\_level\_window.after(1000, update\_label)  # 每隔1秒钟更新一次文本
556. t1 = threading.Thread(target=makerobo\_loop)
557. t2 = threading.Thread(target=update\_label)
558. t1.start()
559. t2.start()
560. check\_water\_level\_window.mainloop()
561. # 查看蓄水池水位状况
562. btn\_check\_water\_level = Button(semi\_auto\_window, text="查看蓄水池水位状况",font=("Arial", 18),command=check\_water\_level)
563. btn\_check\_water\_level.place(relx=0.55, rely=0.5, anchor=NW)
564. **def** check\_air\_humidity():
565. # 创建窗口
566. check\_air\_humidity\_window = Toplevel()
567. check\_air\_humidity\_window.title("查看空气温湿度")
568. # 背景
569. bg\_image3 = Image.open("5.jpg")
570. bg\_photo3 = ImageTk.PhotoImage(bg\_image3)
571. check\_air\_humidity\_canvas = Canvas(check\_air\_humidity\_window, width=bg\_image3.width, height=bg\_image3.height)
572. check\_air\_humidity\_canvas.pack(fill=BOTH, expand=YES)
573. check\_air\_humidity\_canvas.create\_image(0, 0, image=bg\_photo3, anchor=NW)
574. **print**('调用空气温湿度检测函数')
575. tip\_label1 = Label(check\_air\_humidity\_window, text="提示：夏季室温25℃时，湿度在（40,50）内较舒适",font=("Arial", 20))
576. tip\_label1.place(relx=0, rely=0, anchor=NW)
577. tip\_label2 = Label(check\_air\_humidity\_window, text="          冬季室温18℃时，湿度在（60,70）内较舒适",font=("Arial", 20))
578. tip\_label2.place(relx=0, rely=0.1, anchor=NW)
579. label = Label(check\_air\_humidity\_window, text="当前温度为:",font=("Arial", 20))
580. label.place(relx=0, rely=0.3, anchor=NW)
581. label = Label(check\_air\_humidity\_window, text="当前空气湿度为：%",font=("Arial", 20))
582. label.place(relx=0, rely=0.3, anchor=NW)
583. **def** update\_label():
584. **global** tdata
585. **global** hdata  # 声明全局变量
586. label.config(text="当前温度为:".format(tdata))
587. label.config(text="当前湿度为:".format(hdata))
588. check\_air\_humidity\_window.after(10000, update\_label)  # 每隔10秒钟更新一次文本
589. t1 = threading.Thread(target=get\_data)
590. t2 = threading.Thread(target=update\_label)
591. t1.start()
592. t2.start()
593. check\_air\_humidity\_window.mainloop()
594. # 查看空气温湿度
595. btn\_check\_air\_humidity = Button(semi\_auto\_window, text="查看空气温湿度",font=("Arial", 18), command=check\_air\_humidity)
596. btn\_check\_air\_humidity.place(relx=0.2, rely=0.7, anchor=NW)
597. # 返回按钮
598. **def** back():
599. semi\_auto\_window.destroy()
601. btn\_back = Button(semi\_auto\_window, text="返回",font=("Arial", 18), command=back)
602. btn\_back.place(relx=0.5, rely=0.7, anchor=CENTER)
604. semi\_auto\_window.mainloop()
606. btn\_semi\_auto = Button(canvas, text="半自动",font=("Arial", 18), bg="blue", fg="white", command=semi\_auto)
607. btn\_semi\_auto.place(relx=0.55, rely=0.4, anchor=NW)
609. # 退出登录按钮
610. **def** logout():
611. **global** username, password
612. username = ""
613. password = ""
614. messagebox.showinfo("提示", "已退出登录")
615. window.destroy()
617. btn\_logout = Button(canvas, text="退出登录", bg="red", fg="white",font=("Arial", 18), command=logout)
618. btn\_logout.place(relx=0.5, rely=0.8, anchor=CENTER)
620. window.mainloop()

623. # 主程序
624. **if** \_\_name\_\_ == '\_\_main\_\_':
625. login()
626. auto\_watering()
627. **多档位紫灯照明APP的Socket通信脚本(C#)**

ButtonController.cs

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** System.Net.Sockets;
4. **using** System.Text;
5. **using** UnityEngine;
7. **public** **class** ButtonController : MonoBehaviour
8. {
9. **private** TcpClient client;
10. **private** NetworkStream stream;
12. **void** Start()
13. {
14. // 创建一个TCP客户端连接到树莓派服务器
15. client = **new** TcpClient("192.168.137.232", 8000);
16. stream = client.GetStream();
17. }
19. **public** **void** OnButton0Click()
20. {
21. SendMessage("0");
22. Debug.Log("0 clicked!");
23. }
25. **public** **void** OnButton1Click()
26. {
27. SendMessage("35");
28. Debug.Log("1 clicked!");
29. }
31. **public** **void** OnButton2Click()
32. {
33. SendMessage("115");
34. Debug.Log("2 clicked!");
35. }
36. **public** **void** OnButton3Click()
37. {
38. SendMessage("255");
39. Debug.Log("3 clicked!");
40. }
42. **public** **void** SendMessage(string message)
43. {
44. // 将消息转换为字节数组并发送
45. byte[] data = Encoding.UTF8.GetBytes(message);
46. stream.Write(data, 0, data.Length);
47. }
49. **public** **void** quit()
50. {
51. Application.Quit();
52. Debug.Log("quit clicked!");
53. }
55. **void** OnApplicationQuit()
56. {
57. // 在应用程序退出时关闭连接
58. stream.Close();
59. client.Close();
60. }
61. }