

项目源代码文档

main.py

```
1  #!/usr/bin/python3
2  from flask import Flask, render_template_string, request
3  import threading
4  import time
5  from display_manager import DisplayManager
6  import signal
7  import sys
8  import json
9  import os
10
11 app = Flask(__name__)
12 display_manager = DisplayManager()
13 THRESHOLD_FILE = "/tmp/energy_threshold.json" # 共享存储文件
14
15 # 网页控制界面 HTML 模板
16 HTML_TEMPLATE = """
17 <!DOCTYPE html>
18 <html>
19 <head>
20     <title>树莓派 OLED 控制</title>
21     <style>
22         body { font-family: Arial, sans-serif; margin: 20px; }
23         .container { max-width: 600px; margin: 0 auto; }
24         button { padding: 10px 15px; margin: 5px; }
25     </style>
26 </head>
27 <body>
28     <div class="container">
29         <h1>OLED 显示屏控制</h1>
30         <p>当前状态: {{ status }}</p>
31         <p>最后更新: {{ last_update }}</p>
32
33         <h2>控制命令</h2>
34         <button onclick="sendCommand('start')">启动显示</button>
35         <button onclick="sendCommand('stop')">停止显示</button>
36         <button onclick="sendCommand('refresh')">刷新电量</button>
37
38         <h2>电量警告阈值</h2>
```

```

39         <input type="number" id="threshold_input" value="100" placeholder="阈值
(kWh)">
40         <button onclick="setThreshold()">设置</button>
41
42     </div>
43
44     <script>
45         function sendCommand(cmd) {
46             fetch('/command?cmd=' + cmd)
47                 .then(response => response.text())
48                 .then(data => alert(data))
49                 .catch(error => alert('Error: ' + error));
50         }
51
52         function setThreshold() {
53             const threshold = document.getElementById('threshold_input').value;
54             fetch('/set_threshold', {
55                 method: 'POST',
56                 headers: {'Content-Type': 'application/x-www-form-urlencoded'},
57                 body: 'threshold=' + threshold
58             }).then(response => response.text())
59                 .then(data => alert(data));
60         }
61     </script>
62 </body>
63 </html>
64 """
65
66 @app.route('/')
67 def control_panel():
68     status = "运行中" if display_manager.process else "已停止"
69     return render_template_string(HTML_TEMPLATE,
70                                   status=status,
71                                   last_update=time.strftime("%Y-%m-%d %H:%M:%S"))
72
73 @app.route('/command')
74 def handle_command():
75     cmd = request.args.get('cmd', "")
76     if cmd == 'start':
77         display_manager.start()
78         return "显示已启动"
79     elif cmd == 'stop':
80         display_manager.cleanup()
81         return "显示已停止"
82     elif cmd == 'refresh':

```

```

83         # 这里需要实现电量刷新逻辑
84         return "电量已刷新"
85     return "无效命令"
86
87 if not os.path.exists(THRESHOLD_FILE):
88     with open(THRESHOLD_FILE, 'w') as f:
89         json.dump({"threshold": 100}, f) # 默认阈值 100
90
91 @app.route('/set_threshold', methods=['POST'])
92 def set_threshold():
93     try:
94         threshold = float(request.form.get('threshold'))
95         # 保存到共享文件
96         with open(THRESHOLD_FILE, 'w') as f:
97             json.dump({"threshold": threshold}, f)
98         return "阈值已更新为 {}kWh".format(threshold)
99     except Exception as e:
100         return "错误: " + str(e)
101
102 def run_flask():
103     app.run(host='0.0.0.0', port=5000, threaded=True)
104
105 def signal_handler(sig, frame):
106     print("\n 正在关闭服务器...")
107     display_manager.cleanup()
108     sys.exit(0)
109
110 if __name__ == "__main__":
111     # 注册信号处理
112     signal.signal(signal.SIGINT, signal_handler)
113
114     # 启动显示
115     display_manager.start()
116
117     # 在单独线程中启动 Flask
118     flask_thread = threading.Thread(target=run_flask)
119     flask_thread.daemon = True
120     flask_thread.start()
121
122     print("服务器已启动, 请访问 http://树莓派 IP:5000")
123     print("按 CTRL+C 退出")
124
125     try:
126         while True:
127             time.sleep(1)

```

```
128     except KeyboardInterrupt:
129         signal_handler(None, None)
```

```
1
```

screen.py

```
1  #!/usr/bin/python3
2  from luma.core.interface.serial import i2c
3  from luma.oled.device import sh1106
4  from luma.core.render import canvas
5  from PIL import ImageFont
6  import time
7  import getElectricityBalance
8  import signal
9  import sys
10 import json
11 import os
12
13 class DisplayController:
14     def __init__(self):
15         # 初始化设备
16         self.threshold_file = "/tmp/energy_threshold.json"
17         self.current_threshold = 100 # 默认值
18         self.serial = i2c(port=1, address=0x3C)
19         self.device = sh1106(self.serial)
20         self.running = True
21         self.last_balance = None
22         self.first_load = True # 首次加载标志
23
24         # 加载字体
25         try:
26             self.font = ImageFont.truetype('wqy-microhei.ttc', 14)
27         except:
28             self.font = None
29
30         # 注册信号处理
31         signal.signal(signal.SIGINT, self.signal_handler)
32         signal.signal(signal.SIGTERM, self.signal_handler)
33
34     def signal_handler(self, signum, frame):
35         self.running = False
36         self.device.clear()
37         sys.exit(0)
```

```

38
39     def read_threshold(self):
40         """从共享文件读取最新阈值"""
41         try:
42             with open(self.threshold_file, 'r') as f:
43                 data = json.load(f)
44                 self.current_threshold = float(data["threshold"])
45         except:
46             pass # 保持当前阈值不变
47
48     def show_loading(self):
49         """只在首次显示加载状态"""
50         if self.first_load:
51             with canvas(self.device) as draw:
52                 draw.rectangle(self.device.bounding_box, outline="white", fill="black")
53                 draw.text((10, 30), "Loading...", fill="white", font=self.font)
54                 time.sleep(0.5) # 确保用户能看到加载状态
55
56     def show_data(self, balance):
57         self.read_threshold() # 每次显示前读取最新阈值
58         """显示时间和电量数据"""
59         with canvas(self.device) as draw:
60             draw.rectangle(self.device.bounding_box, outline="white", fill="black")
61             draw.text((5, 10), time.strftime("%H:%M:%S"), fill="white", font=self.font)
62             draw.text((10, 25), "Current Power:", fill="white", font=self.font)
63             draw.text((40, 40), f"{balance:.2f} kWh", fill="white", font=self.font)
64             # 动态阈值判断
65             if balance < self.current_threshold:
66                 warning_text = "!!!!!"
67                 draw.text((97, 10), warning_text, fill="white", font=self.font)
68
69     def run(self):
70         while self.running:
71             try:
72                 # 首次加载显示状态
73                 self.show_loading()
74
75                 # 只在首次或需要刷新时获取电量
76                 if self.first_load or self.should_refresh_balance():
77                     balance = getElectricityBalance.get_electricity_balance()
78                     self.last_balance = balance
79                     self.first_load = False
80
81                 # 显示数据 (时间会自动更新)
82                 self.show_data(self.last_balance)

```

```

83
84         time.sleep(1) # 更新时间间隔
85
86     except Exception as e:
87         print(f"显示错误: {e}")
88         self.first_load = True # 出错后下次重新加载
89         time.sleep(5)
90
91     def should_refresh_balance(self):
92         """自定义电量刷新条件 (例如每 10 分钟) """
93         # 示例: 每 600 秒刷新一次 (10 分钟)
94         return hasattr(self, 'last_refresh_time') and \
95             (time.time() - self.last_refresh_time) > 600
96
97 if __name__ == "__main__":
98     controller = DisplayController()
99     controller.run()
100

```

getElectricityBalance.py

```

1  # -*- coding: utf-8 -*-
2  import time
3  import subprocess
4  import pytesseract
5  from PIL import Image
6  import mss
7  import re
8
9  # 配置 Tesseract 路径
10 pytesseract.pytesseract.tesseract_cmd = '/usr/bin/tesseract'
11
12 def click_at(x, y):
13     """使用 xdotool 模拟点击"""
14     subprocess.run(f"xdotool mousemove {x} {y} click 1", shell=True)
15
16 def get_electricity_balance():
17     try:
18         # 点击微信第一个窗口 "南林智慧校园"
19         click_at(661, 306)
20         time.sleep(2)
21         # 点击 "一卡通"
22         click_at(904, 862)
23         time.sleep(2)
24         # 点击 "一卡通自助服务"

```

```
25     click_at(892, 710)
26     time.sleep(2)
27     # 点击 "缴电费"
28     click_at(712, 572)
29     time.sleep(2)
30     # 选择 "本部玄武公寓用电"
31     click_at(354, 472)
32     time.sleep(2)
33     # 楼栋选择 20 栋
34     # 点击 "楼栋" 选项
35     click_at(1801, 741)
36     time.sleep(2)
37     # 键盘输入 "2" , 选择 20 栋
38     subprocess.run(['xdotool', 'key', '2'])
39     time.sleep(1)
40     subprocess.run(['xdotool', 'key', 'Return'])
41     time.sleep(2)
42     click_at(1803, 862)
43     subprocess.run("xdotool type '201202'", shell=True)
44     time.sleep(1)
45     # 点击并输入房间号 "201202"
46     click_at(1493, 798)
47     time.sleep(2)
48     power_bbox = (1621, 892, 1854, 935)
49     # 处理电量数据
50     text = extract_text_from_coords(power_bbox)
51     match = re.search(r'(\d+\.\d+)', text)
52     if match:
53         balance = float(match.group(1))
54     else:
55         raise ValueError("No balance found in text")
56
57     return balance
58
59 except Exception as e:
60     print(f"获取电量失败: {str(e)}")
61     return 0.0 # 返回默认值
62
63 if __name__ == "__main__":
64     try:
65         balance = get_electricity_balance()
66         print(f"当前电量: {balance} 度")
67     except Exception as e:
68         print(f"操作失败: {str(e)}")
69
```

auto_cutoff_if_empty.py

```
1  #!/usr/bin/env python3
2  import RPi.GPIO as GPIO
3  import time
4  from datetime import datetime
5
6  # 硬件配置
7  SENSOR_PIN = 27      # 红外传感器 GPIO(BCM27)
8  SERVO_PIN = 18        # 舵机 GPIO(BCM18)
9  PWM_FREQ = 50         # SG90 频率(Hz)
10 UP_ANGLE = 180        # 有人角度
11 DOWN_ANGLE = 0        # 无人角度
12 DEBOUNCE_TIME = 0.5   # 防抖时间(秒)
13
14
15 class ServoControl:
16     def __init__(self):
17         # 初始化设置
18         GPIO.setmode(GPIO.BCM)
19         GPIO.setup(SENSOR_PIN, GPIO.IN)
20         GPIO.setup(SERVO_PIN, GPIO.OUT)
21
22         # PWM 初始化
23         self.pwm = GPIO.PWM(SERVO_PIN, PWM_FREQ)
24         self.pwm.start(0)
25         self.current_angle = None # 初始角度未设置
26         self.last_state = None    # 初始状态未设置
27
28         # 传感器预热
29         print("传感器预热中(30 秒)...")
30         time.sleep(30)
31         print("系统就绪, 开始检测")
32
33     def set_angle(self, angle):
34         """设置舵机角度并保持"""
35         if angle == self.current_angle:
36             return # 已在目标角度
37
38         # 计算占空比 (0°=2.5%, 180°=12.5%)
39         duty = angle / 18 + 2.5
40         self.pwm.ChangeDutyCycle(duty)
41         time.sleep(5) # 确保转动完成
42         self.pwm.ChangeDutyCycle(0) # 停止 PWM 防止抖动
43         self.current_angle = angle
```



```

44         print(f"角度设置: {angle}°")
45
46     def run(self):
47         try:
48             # 获取初始状态并设置初始角度
49             initial_state = GPIO.input(SENSOR_PIN)
50             if initial_state == 1: # 初始有人
51                 self.set_angle(UP_ANGLE)
52             else: # 初始无人
53                 self.set_angle(DOWN_ANGLE)
54             self.last_state = initial_state
55
56             while True:
57                 current_state = GPIO.input(SENSOR_PIN)
58                 print( current_state)
59                 # 只有状态变化时才动作
60                 if current_state != self.last_state:
61                     if current_state == 1: # 从无人变有人
62                         self.set_angle(UP_ANGLE)
63                     else: # 从有人变无人
64                         self.set_angle(DOWN_ANGLE)
65                     self.last_state = current_state
66
67                 time.sleep(DEBOUNCE_TIME) # 检测间隔
68
69         except KeyboardInterrupt:
70             print("\n 正在停止...")
71         finally:
72             self.pwm.stop()
73             GPIO.cleanup()
74             print("系统关闭")
75
76 if __name__ == "__main__":
77     controller = ServoControl()
78     controller.run()
79

```

night_power_saving_control.py

```

1  #!/usr/bin/env python3
2  import RPi.GPIO as GPIO
3  import time
4  from datetime import datetime
5
6  # 硬件配置

```

```

7  SENSOR_PIN = 27      # 红外传感器 GPIO(BCM27)
8  SERVO_PIN = 18       # 舵机 GPIO(BCM18)
9  PWM_FREQ = 50        # SG90 频率(Hz)
10 UP_ANGLE = 180       # 有人角度
11 DOWN_ANGLE = 0       # 无人角度
12 DEBOUNCE_TIME = 0.5  # 防抖时间(秒)
13
14 def is_night_time():
15     now = datetime.now().time()
16     return now.hour >= 23 or now.hour < 6
17
18 class ServoControl:
19     def __init__(self):
20         # 初始化设置
21         GPIO.setmode(GPIO.BCM)
22         GPIO.setup(SENSOR_PIN, GPIO.IN)
23         GPIO.setup(SERVO_PIN, GPIO.OUT)
24
25         # PWM 初始化
26         self.pwm = GPIO.PWM(SERVO_PIN, PWM_FREQ)
27         self.pwm.start(0)
28         self.current_angle = None # 初始角度未设置
29         self.last_state = None    # 初始状态未设置
30
31         # 传感器预热
32         print("传感器预热中(30 秒)...")
33         time.sleep(30)
34         print("系统就绪, 开始检测")
35
36     def set_angle(self, angle):
37         """设置舵机角度并保持"""
38         if angle == self.current_angle:
39             return # 已在目标角度
40
41         # 计算占空比 (0°=2.5%, 180°=12.5%)
42         duty = angle / 18 + 2.5
43         self.pwm.ChangeDutyCycle(duty)
44         time.sleep(5) # 确保转动完成
45         self.pwm.ChangeDutyCycle(0) # 停止 PWM 防止抖动
46         self.current_angle = angle
47         print(f"角度设置: {angle}°")
48
49     def run(self):
50         try:
51             # 获取初始状态并设置初始角度

```

```

52         initial_state = GPIO.input(SENSOR_PIN)
53         if initial_state == 1: # 初始有人
54             self.set_angle(UP_ANGLE)
55         else: # 初始无人
56             self.set_angle(DOWN_ANGLE)
57         self.last_state = initial_state
58
59         while True:
60             current_state = GPIO.input(SENSOR_PIN)
61             print( current_state)
62             # 只有状态变化时才动作
63             if current_state != self.last_state:
64                 if current_state == 1: # 从无人变有人
65                     self.set_angle(UP_ANGLE)
66                 else: # 从有人变无人
67                     self.set_angle(DOWN_ANGLE)
68                 self.last_state = current_state
69
70             time.sleep(DEBOUNCE_TIME) # 检测间隔
71
72         except KeyboardInterrupt:
73             print("\n 正在停止...")
74         finally:
75             self.pwm.stop()
76             GPIO.cleanup()
77             print("系统关闭")
78
79     if __name__ == "__main__":
80         while True:
81             if is_night_time():
82                 controller = ServoControl()
83                 controller.run()
84
85

```