

微算機系統實習

第 13 組專案報告

LAB 06

遠端週邊控制實習



組別： 13

班級、姓名與學號：

醫工三 B812110004 葉芸茜

醫工三 B812110011 湯青秀

日期：2024.05.13

一、實驗內容：

1. 學習透過 C++ 和 Python 控制 LED 燈和光敏電阻
2. 學習如何使用回呼函式 (callback function) 回傳給前端光敏電阻的數值
3. 學習如何使用 jQuery 的 \$.ajax() 來定期抓取後端數值
4. 學習如何使用 html 和 javascript 語法
5. 結合感測器與 LED 進行實驗

(1)項目一：網頁顯示光敏電阻的值

- 在麵包板上安裝類比數位訊號轉換器及光敏電阻
- 撰寫 html 及後端程式控制光敏電阻
- 勾選 DETECT，點擊 Submit 切換到偵測畫面，網頁會顯示光敏電阻的值
- 依照設定光敏電阻的門檻值控制 TX2 上 2 顆 LED 燈開關狀態

(2)項目二：切換模式來控制指定 LED 開關事件

- 勾選 NO_DETECT，點擊 Submit 會停止顯示光敏電阻的值，並切換畫面
- 2 顆 LED 燈狀態選項（可選用 Check Box）
- 燈亮、熄滅選項及 Submit 按鈕
- 勾選指定 LED 燈號，點擊 Submit 即可控制 TX2 上的 LED 燈
- 點擊 Submit 後，需在原畫面顯示送出的狀態(在同個畫面)

二、實驗過程及結果：

1. 實驗過程

(1) 撰寫程式碼

(a) 前端

(i) 頁面設計

(ii) 回傳光敏電阻值顯示於網頁



```

1 <script
2   src="https://code.jquery.com/jquery-3.7.1.js"
3   integrity="sha256-eKhay18LEQwp4NKxN+CfCh+3q0VUtJn3QNZ0TciWLP4="
4   crossorigin="anonymous"></script>
5 <script>
6   $("#response").css("fontSize", 40);
7
8   setInterval(GetResult, 1000);
9   function GetResult(){
10     $.ajax({
11       type:"get",
12       url:"/detecting",
13       success:function (result){
14         console.log("Item", result)
15         $('#response').text(result);
16       },
17       error:function(){
18         alert("error");
19       }
20     })
21   }
22 </script>

```

(b) 後端

(i) 頁面跳轉

◦ index to detect(點擊 detect)/index(點擊 no detect)

```

1   if(req.query.botton3 === "Submit"){
2     if(req.query.detecting === "detect"){
3       res.redirect('/detect')
4     }
5     else{
6       res.sendFile('/public/index.html', {root: __dirname })
7     }
8   }
9   else{
10    res.sendFile('/public/index.html', {root: __dirname })
11  }
12
13 })

```

◦ detect to index(點擊 no detect)

```

1 app.get("/detect", (req, res)=> {
2   res.sendFile('/public/detect.html', {root: __dirname })
3   console.log("good")
4   detect()
5   if(req.query.botton3 === "Submit" && req.query.detecting === "noDected"){
6     res.sendFile('/public/index.html', {root: __dirname })
7   }
8
9 })

```

(ii) 判斷按鍵狀態，呼叫 detect(項目一)

```
1 function detect(){
2   var p = new Promise(function(resolve, reject){
3
4     let child_process = require("child_process");
5
6     let process = child_process.spawn('python',[
7       "./gpio_test.py"
8     ]);
9
10    process.stdout.on('data', (data) => {
11      console.log(`stdout: ${data}`);
12      resolve(data)
13      //return data;
14    });
15
16    process.stderr.on('data', (data) => {
17      console.error(`stderr: ${data}`);
18      reject(data)
19    });
20  })
21  return p
22 }
```

(iii) /detecting 持續呼叫 detect()並將光敏電阻的值回傳給.ajax

```
1 app.get("/detecting", (req, res)=> {
2   console.log("good2")
3   detect()
4   .then(function(data){
5     res.send(data)
6   })
7   .catch(error => {
8     console.log(err)
9   })
10 })
11
```

(iv) 判斷按鍵狀態，及對應 LED 及 POWER 狀態(項目二)

```
1 app.get("/index", (req, res)=>{
2     if(req.query.botton1 === "Submit"){
3         if(req.query.led1 === "on"){
4             controLLED("LED1", req.query.POWER)
5         }
6         if(req.query.led2 === "on"){
7             controLLED("LED2", req.query.POWER)
8         }
9     }
10    if(req.query.botton2 === "Mode_shine"){
11        controLLED("Mode_shine", req.query.time)
12    }
13 }
```

```
1 function controLLED(LED, POWER){
2     return new Promise(function(resolve, reject){
3
4         let child_process = require("child_process");
5
6         let process = child_process.spawn('python',[
7             "./gpio_led.py", LED, POWER
8         ]);
9
10        process.stdout.on('data', (data) =>{
11            console.log(`stdout: ${data}`);
12        });
13
14        process.stderr.on('data', (data) =>{
15            console.error(`stderr: ${data}`);
16        });
17    })
18 }
19 }
```

(c) 獲取光敏值

(i) 控制 LED 對應 TX2 上的腳位及初始設定

```
1 def init():
2     GPIO.setwarnings(False)
3
4     GPIO.cleanup()
5     GPIO.setmode(GPIO.BCM)
6     GPIO.setup(output_pin, GPIO.OUT)
7     GPIO.setup(output_pin2, GPIO.OUT)
8     GPIO.setup(SPIMOSI, GPIO.OUT)
9     GPIO.setup(SPIMISO, GPIO.IN)
10    GPIO.setup(SPICKL, GPIO.OUT)
11    GPIO.setup(SPICS, GPIO.OUT)
```

(ii) 根據函式傳入值回傳光敏數值

```
1 def main(argv):
2     init()
3
4     if(len(argv) == 1):
5         adc_value = readadc(photo_ch, SPICLK, SPIMOSI, SPIMISO, SPICS, output_pin, output_pin2)
6         print(adc_value)
7         return adc_value
8     elif(argv[1][1] == 'E'):
9         if(argv[2] == "on"):
10             ledbright(int(argv[1][3]))
11         elif(argv[2] == "off"):
12             leddark(int(argv[1][3]))
13     else:
14         ledshine(int(argv[2]))
```

(iii) 設計光敏數值閾值以控制 LED 狀態

下表為我們希望可以達到的效果

	LED1	LED2
亮	暗	暗
普通	亮	暗
暗	亮	亮


經過觀察發現，當光敏電阻處在越亮的環境中，讀數就會越小，反之則會變大。而在正常光線下，數值大約為 300 左右。因此我們將範圍切割為>450(暗)，450-150(正常)，以及 150 以下(亮)。

```
1 if(adcout > 450):
2     #led2 bright
3     GPIO.output(output2, GPIO.HIGH)
4     #led1 bright
5     GPIO.output(output1, GPIO.HIGH)
6 elif(adcout > 150):
7     #led1 bright
8     GPIO.output(output1, GPIO.HIGH)
9     #led2 dark
10    GPIO.output(output2, GPIO.LOW)
11 else:
12     #led1 dark
13     GPIO.output(output1, GPIO.LOW)
14     #led2 dark
15     GPIO.output(output2, GPIO.LOW)
16 #print(adcout)
17 return adcout
```

(d) LED 控制開關

(i) 控制 LED 對應 TX2 上的腳位及初始設定

(一次只針對易科 led 做初始化，避免每次呼叫函式兩科都被初始化導致原始狀態消失)



```
1 def init(pin):
2     print("initing = ")
3     print(pin)
4     GPIO.setwarnings(False)
5
6     GPIO.cleanup()
7     GPIO.setmode(GPIO.BCM)
8     GPIO.setup(pin, GPIO.OUT)
9     GPIO.setup(SPIMOSI, GPIO.OUT)
10    GPIO.setup(SPIMISO, GPIO.IN)
11    GPIO.setup(SPICLK, GPIO.OUT)
12    GPIO.setup(SPICS, GPIO.OUT)
```

(ii) 根據函式傳入值進行對應 led 編號及狀態動作



```
1 def main(argv):
2     if(int(argv[1][3]) == 1):
3         init(17)
4         led = 17
5     elif(int(argv[1][3]) == 2):
6         init(22)
7         led = 22
8
9     if(len(argv) == 1):
10        adc_value = readadc(photo_ch, SPICLK, SPIMOSI, SPIMISO, SPICS)
11        print(adc_value)
12        return adc_value
13    elif(argv[1][1] == 'E'):
14        if(argv[2] == "on"):
15            print(led)
16            print("on")
17            ledbright(led)
18        elif(argv[2] == "off"):
19            print(led)
20            print("off")
21            leddark(led)
```

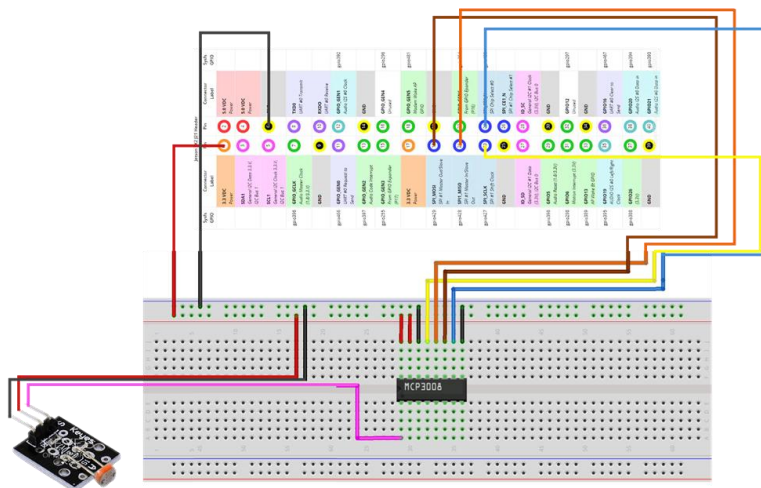
(iii) 傳入 led 編號，控制其亮燈

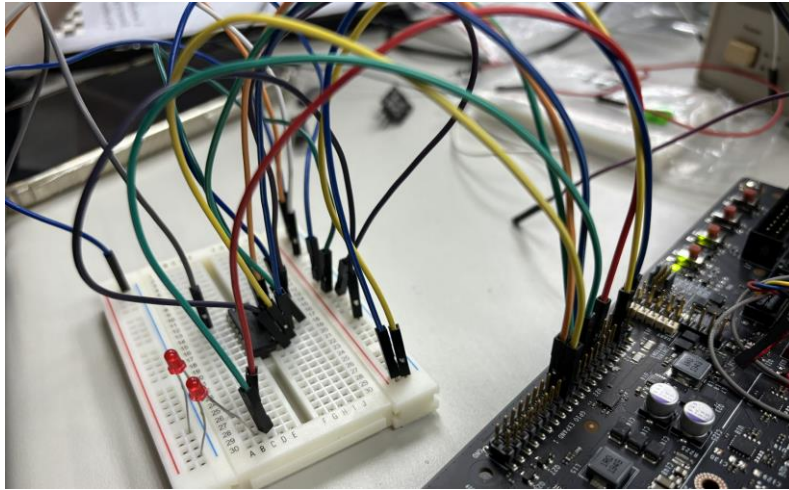


(iv) 傳入 led 編號，控制其暗燈



(2) 運用麵包板連接各組件





LED 1 選用腳位 17(BCM)、11(BOARD)

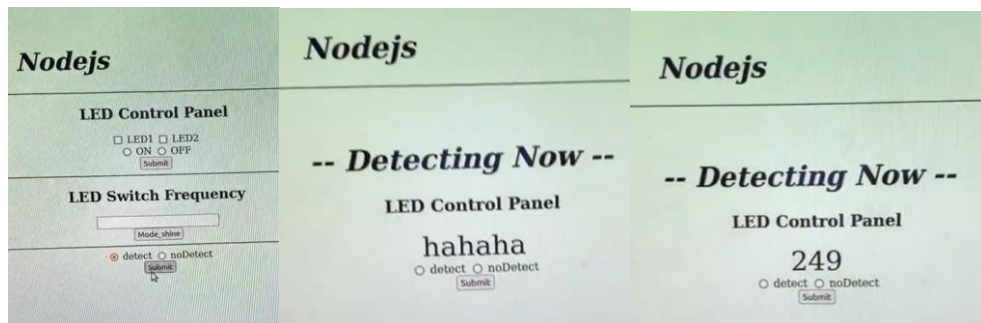
LED 2 選用腳位 27(BCM)、13(BOARD)

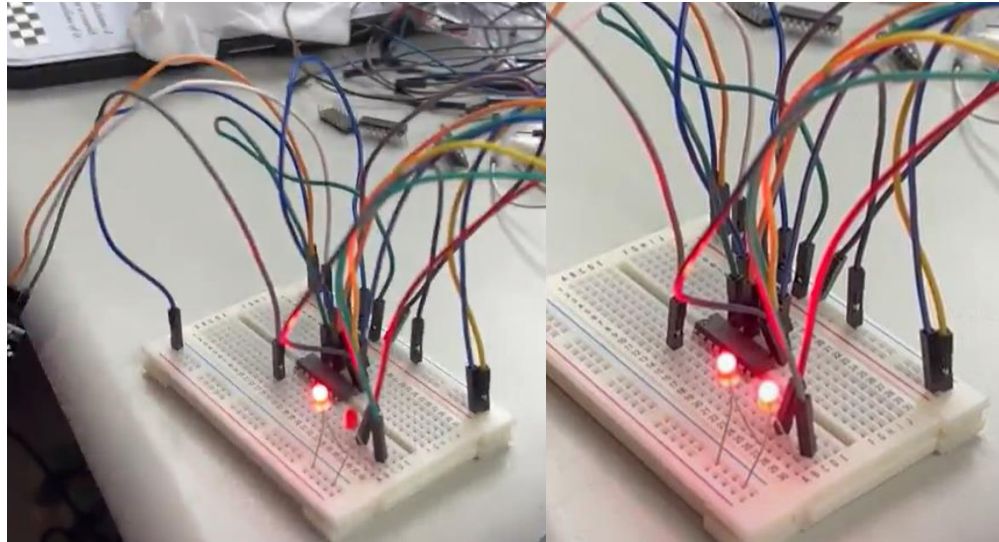
2. 預期實驗結果

- (1) 項目一：勾選 detect，點擊 submit，開啟偵測畫面，網頁會顯示實時光敏電阻的數值。隨著光敏數值的變動，依據閾值控制兩顆 LED 的實時開關狀態
- (2) 項目二：勾選 no detect，點擊 submit，切換下一個頁面。畫面可勾選兩顆 LED(LED 1、LED 2)，及要選擇的對應狀態(on、off)，點擊 submit 後可控制 TX2 上的 LED 開關狀態

3. 實際上的結果

https://drive.google.com/file/d/1BvFquShfJCulsXSILxTG_8VGs1M4PXfT/view?usp=sharing





4. 遇到的問題&問題怎麼解決

(1) 切換頁面

- (a)我們不會切換頁面，不知道要怎麼讓她在 redirect 之後直接可以顯示另一個 html 檔，用了 Post 他就不會動了，於是
- (b)我們在 /detect 路由的一開始 sendfile detect.html 給他，他就會動了！
- (c)可惜，好景不長，我們發現 ajax 如果 get detect 的話，他就會，一直，抓到我們為了顯示 send 給他的 html 檔案，於是，他就會顯示一大堆 html 文字
- (d)如何解決的呢？我們另外給他開了一個路由叫做 /detecting，他就只負責呼叫 detect() 這個 function，就不會干擾到其他東西了

(2) 出大事了！怎麼辦，沒辦法把 detect 的 output 傳出去！

- (a)喔！他一直在輸出 "good"，原來是因為我們第一個 print 了一個 good 出來，被他抓到了
- (b)把 print 的值變成電阻值了！stdout 的地方明明可以抓到的，但是離開函數之後他就沒辦法知道怎麼抓到他了
- (c)嘗試 console.log(resolve) 希望可以得到它的 output，結果它確實可以顯示了，html 也能抓到了，但是，他輸出的是 [Object object]
- (d)嘗試用 .toString() 把它轉換，結果 console 可以成功顯示數字(string)，但是傳到 html 的輸出，就變成 "[Object object]"
- (e)最後，我們發現，detect() 後面應該要使用 .then

和.catch 才能根據 promise 裡面的使用情況得到 output，
所以我們在.then 裡面 send(data)，就大功告成了
(3) 終於弄好了數值顯示了！終於回到了熟悉的網頁開關 LED 的部分，但是.....

- (a)從上個禮拜開始，我們的光敏電阻就讀不到數字了，明明上次的 lab 都可以正常讀取，為甚麼呢？（我們甚至直接拿了 lab 5 的 code 去執行，他卻都給我一直顯示 0，有時候會突然蹦出幾個 2、3 之類的，看起來真的很像在接觸不良吧）
- (b)於是，我們換了 ic、換了線、換了 ADC、換了麵包板，可是你為什麼還是不會好呢???
- (c)經過了快兩個小時，終於發現，因為電阻的輸出腳和接地腳的線接反了，換過來就讀到了
- (d)這樣下來終於搞定第一項目了，到了最後的第二項，我們發現 LED2 怎麼關不掉？
- (e)我們把 python 檔案中所有步驟都 print 出來，一步步篩查發現明明都是正常的，而且 LED1 呼叫的函數都是一樣的，為甚麼只有 LED2 有問題呢？
- (f)我們突然發現，如果把 LED2 連接的腳位線拔掉再接回去，他就不會亮了！如果是控制導致的問題，那接回去時應該還是亮著的啊！於是，我們合理懷疑是腳位本身的問題，因此決定，換一隻腳，結果就好了:(!

三、程式碼

index.js
<pre>const express = require("express"); const app = express(); app.use(express.static('./public')); app.get("/index", (req, res)=>{ if(req.query.botton1 === "Submit"){ if(req.query.led1 === "on"){ controlLED("LED1", req.query.POWER) } if(req.query.led2 === "on"){ controlLED("LED2", req.query.POWER) } } })</pre>

```

    }
    if(req.query.botton2 === "Mode_shine"){
        controLLED("Mode_shine", req.query.time)
    }

    if(req.query.botton3 === "Submit"){
        if(req.query.detecting === "detect"){
            res.redirect('/detect')
        }
        else{
            res.sendFile('/public/index.html', {root:
__dirname })
        }
    }
    else{
        res.sendFile('/public/index.html', {root: __dirname })
    }
}

}))

app.get("/detect", (req, res)=> {
    res.sendFile('/public/detect.html', {root: __dirname })
    console.log("good")
    detect()
    if(req.query.botton3 === "Submit" && req.query.detecting ===
"noDected"){
        res.sendFile('/public/index.html', {root: __dirname })
    }
})

app.get("/detecting", (req, res)=> {
    console.log("good2")
    detect()
    .then(function(data){
        res.send(data)
    })
    .catch(error => {
        console.log(err)
    })
})

function controLLED(LED, POWER){
    return new Promise(function(resolve, reject){

        let child_process = require("child_process");

        let process = child_process.spawn('python',[
            "./gpio_led.py", LED, POWER

```

```

    });

    process.stdout.on('data', (data) =>{
        console.log(`stdout: ${data}`);

    });

    process.stderr.on('data', (data) =>{
        console.error(`stderr: ${data}`);
    });
    })
}

function detect(){
    var p = new Promise(function(resolve, reject){

        let child_process = require("child_process");

        let process = child_process.spawn('python',[
            "./gpio_test.py"
        ]);

        process.stdout.on('data', (data) => {
            console.log(`stdout: ${data}`);
            resolve(data)
            //return data;
        });

        process.stderr.on('data', (data) => {
            console.error(`stderr: ${data}`);
            reject(data)
        });
    })
    return p
}

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
    console.log(`server is running on port ${PORT}.`);
});

```

gpio_test.py

```

import Jetson.GPIO as GPIO
import time
import sys

```

```

SPICLK = 11 #23
SPIMISO = 9 #21
SPIMOSI = 10 #19
SPICS = 8 #24
output_pin = 17 #11
output_pin3 = 27 #13
output_pin2 = 22 #15
output_pin4 = 5 #29

photo_ch = 0 #27

def init():
    GPIO.setwarnings(False)

    GPIO.cleanup()
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(output_pin, GPIO.OUT)
    GPIO.setup(output_pin2, GPIO.OUT)
    GPIO.setup(SPIMOSI, GPIO.OUT)
    GPIO.setup(SPIMISO, GPIO.IN)
    GPIO.setup(SPICLK, GPIO.OUT)
    GPIO.setup(SPICS, GPIO.OUT)

# get light
def readadc(adcnun, clockpin, mosipin, misopin, cspin, output1,
output2):
    if((adcnun > 7) or (adcnun < 0)):
        return -1
    GPIO.output(cspin, True)

    GPIO.output(clockpin, False)
    GPIO.output(cspin, False)

    commandout = adcnun
    commandout |= 0x18
    commandout <= 3
    for i in range(5):
        if (commandout & 0x80):
            GPIO.output(mosipin, True)
        else:
            GPIO.output(mosipin, False)
        commandout <= 1
        GPIO.output(clockpin, True)
        GPIO.output(clockpin, False)

    adcout = 0
    for i in range(12):
        GPIO.output(clockpin, True)
        GPIO.output(clockpin, False)
        adcout <= 1

```

```

        if(GPIO.input(misopin)):
            adcout |= 0x1

GPIO.output(cspin, True)
adcout >>= 1

if(adcout > 450):
    #led2 bright
    GPIO.output(output2, GPIO.HIGH)
    #led1 bright
    GPIO.output(output1, GPIO.HIGH)
elif(adcout > 150):
    #led1 bright
    GPIO.output(output1, GPIO.HIGH)
    #led2 dark
    GPIO.output(output2, GPIO.LOW)
else:
    #led1 dark
    GPIO.output(output1, GPIO.LOW)
    #led2 dark
    GPIO.output(output2, GPIO.LOW)
#print(adcout)
return adcout

# ledbright
def ledbright(led):
    if(led == 1):
        GPIO.output(output_pin, True)
    if(led == 2):
        GPIO.output(output_pin2, True)

def leddark(led):
    if(led == 1):
        GPIO.output(output_pin, False)
    if(led == 2):
        GPIO.output(output_pin2, False)

def ledshine(time):
    for i in range(2*time):
        adc_value = readadc(photo_ch, SPICLK, SPIMOSI, SPIMISO,
SPICS, output_pin, output_pin2)
        print(adc_value)
        if(i % 2 == 0):
            GPIO.output(output_pin, True)
            GPIO.output(output_pin2, True)
            GPIO.output(output_pin3, False)
            GPIO.output(output_pin4, False)
        else:
            GPIO.output(output_pin3, True)
            GPIO.output(output_pin4, True)

```

```

        GPIO.output(output_pin1, False)
        GPIO.output(output_pin2, False)
        time.sleep(1000000/adc_value)
        GPIO.output(output_pin1, False)
        GPIO.output(output_pin2, False)
        GPIO.output(output_pin3, False)
        GPIO.output(output_pin4, False)

def main(argv):
    init()

    if(len(argv) == 1):
        adc_value = readadc(photo_ch, SPICLK, SPIMOSI, SPIMISO,
        SPICS, output_pin, output_pin2)
        print(adc_value)
        return adc_value
    elif(argv[1][1] == 'E'):
        if(argv[2] == "on"):
            ledbright(int(argv[1][3]))
        elif(argv[2] == "off"):
            leddark(int(argv[1][3]))
    else:
        ledshine(int(argv[2]))

if __name__ == '__main__':
    main(sys.argv)

```

gpio_led.py

```

import Jetson.GPIO as GPIO
import time
import sys

SPICLK = 11 #23
SPIMISO = 9 #21
SPIMOSI = 10 #19
SPICS = 8 #24
output_pin = 17 #11
output_pin2 = 27 #13
output_pin3 = 22 #15
output_pin4 = 5 #29

photo_ch = 0 #27

def init(pin):
    print("initing = ")
    print(pin)
    GPIO.setwarnings(False)

```



```

GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
GPIO.setup(pin, GPIO.OUT)
GPIO.setup(SPIMOSI, GPIO.OUT)
GPIO.setup(SPIMISO, GPIO.IN)
GPIO.setup(SPICLK, GPIO.OUT)
GPIO.setup(SPICS, GPIO.OUT)

# get light
def readadc(adcnun, clockpin, mosipin, misopin, cspin):
    print("adcing")
    if((adcnun > 7) or (adcnun < 0)):
        return -1
    GPIO.output(cspin, True)

    GPIO.output(clockpin, False)
    GPIO.output(cspin, False)

    commandout = adcnun
    commandout |= 0x18
    commandout <= 3
    for i in range(5):
        if (commandout & 0x80):
            GPIO.output(mosipin, True)
        else:
            GPIO.output(mosipin, False)
        commandout <= 1
        GPIO.output(clockpin, True)
        GPIO.output(clockpin, False)

    adcout = 0
    for i in range(12):
        GPIO.output(clockpin, True)
        GPIO.output(clockpin, False)
        adcout <= 1
        if(GPIO.input(misopin)):
            adcout |= 0x1

    GPIO.output(cspin, True)

    adcout >= 1
    return adcout

# ledbright
def ledbright(led):
    print("brighting")
    print(led)
    GPIO.output(led, True)

```

```

# leddark
def leddark(led):
    print("darking")
    print(led)
    GPIO.output(led, False)

def ledshine(time):
    for i in range(2*time):
        adc_value = readadc(photo_ch, SPICLK, SPIMOSI, SPIMISO,
SPICS)
        print(adc_value)
        if(i % 2 == 0):
            GPIO.output(output_pin, True)
            GPIO.output(output_pin2, True)
            GPIO.output(output_pin3, False)
            GPIO.output(output_pin4, False)
        else:
            GPIO.output(output_pin3, True)
            GPIO.output(output_pin4, True)
            GPIO.output(output_pin1, False)
            GPIO.output(output_pin2, False)
        time.sleep(100000/adc_value)
    GPIO.output(output_pin1, False)
    GPIO.output(output_pin2, False)
    GPIO.output(output_pin3, False)
    GPIO.output(output_pin4, False)

def main(argv):
    if(int(argv[1][3]) == 1):
        init(17)
        led = 17
    elif(int(argv[1][3]) == 2):
        init(22)
        led = 22

    if(len(argv) == 1):
        adc_value = readadc(photo_ch, SPICLK, SPIMOSI, SPIMISO,
SPICS)
        print(adc_value)
        return adc_value
    elif(argv[1][1] == 'E'):
        if(argv[2] == "on"):
            print(led)
            print("on")
            ledbright(led)
        elif(argv[2] == "off"):
            print(led)
            print("off")
            leddark(led)

```

```
if __name__ == '__main__':  
    main(sys.argv)
```

detect.html

```
<!DOCTYPE html>  
  
<style>  
    .all{  
        background:#e8fcef;  
        width:600px;  
        height:800px;  
        border:10px solid #b3dbb1;  
        margin:0 auto;  
        text-align: center;  
  
    }  
    .all h1{  
        font-size: 40px;  
        font-weight:500px;  
        padding-left:75px;  
        padding-top:50px;  
        font-style:italic;  
        text-align: left;  
    }  
    .all p{  
        font-size:20px;  
        font-weight:200px;  
        opacity:0.5;  
    }  
</style>  
  
<html lang="en">  
    <head>  
        <meta charset="UTF-8">  
        <meta name="viewport" content="width=\, initial-  
scale=1.0">  
        <title>Nodejs</title>  
    </head>  
    <body>  
        <div class="all">  
            <p>  
                <h1>Nodejs</h1>  
            </p>  
            <hr>  
            <h1> -- Detecting Now -- </h1>
```

```

        <form action="/index" method="get">
            <h2>LED Control Panel</h2>

            <div id="response" method="post"> hahaha </div>

            <div>
                <input type="radio" id="detect"
name="detecting" value="detect">
                <label for="ON">detect</label>

                <input type="radio" id="noDetect"
name="detecting" value="noDetect">
                <label for="OFF">noDetect</label>
            </div>
            <input type="submit" name="botton3"
value="Submit">
        </form>
    </div>
    <script
src="https://code.jquery.com/jquery-3.7.1.js"
integrity="sha256-
eKhayi8LEQwp4NKxN+CfCh+3qOVUtJn3QNZ0TciWLP4="
crossorigin="anonymous"></script>
    <script>
        $("#response").css("fontSize", 40);

        setInterval(GetResult, 1000);
        function GetResult(){
            $.ajax({
                type:"get",
                url:"/detecting",
                success:function (result){
                    console.log("Item", result)
                    $('#response').text(result);
                },
                error:function(){
                    alert("error");
                }
            })
        }
    </script>
</body>
</html>

```

index.html

```
<!DOCTYPE html>
```

```
<style>
```

```

    .all{
        background:#e8fcef;
        width:600px;
        height:800px;
        border:10px solid #b3dbb1;
        margin:0 auto;
        text-align: center;

    }
    .all h1{
        font-size: 40px;
        font-weight:500px;
        padding-left:75px;
        padding-top:50px;
        font-style:italic;
        text-align: left;
    }
    .all p{
        font-size:20px;
        font-weight:200px;
        opacity:0.5;
    }
</style>

<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=\, initial-
scale=1.0">
        <title>Nodejs</title>
    </head>
    <body>
        <div class="all">
            <p>
                <h1>Nodejs</h1>
            </p>
            <hr>
            <form action="/index" method="get">
                <h2>LED Control Panel</h2>
                <div>
                    <input type="checkbox" id="led1" name="led1"
>
                    <label for="LED1">LED1</label>

                    <input type="checkbox" id="led2" name="led2"
>
                    <label for="LED2">LED2</label>
                </div>

                <div>

```

```

        <input type="radio" id="ON" name="POWER"
value="on">
        <label for="ON">ON</label>

        <input type="radio" id="OFF" name="POWER"
value="off">
        <label for="OFF">OFF</label>
    </div>
    <input type="submit" name="botton1"
value="Submit">
    <hr>

    <h2>LED Switch Frequency</h2>
    <input type="text" name="time">
    <br>
    <input type="submit" name="botton2"
value="Mode_shine">
    <hr>

    <div>
        <input type="radio" id="detect"
name="detecting" value="detect">
        <label for="ON">detect</label>

        <input type="radio" id="noDetect"
name="detecting" value="noDetect">
        <label for="OFF">noDetect</label>
    </div>
    <input type="submit" name="botton3"
value="Submit">
    </form>
</div>

</body>
</html>

```

四、本次實驗過程說明與解決方法：

1. 實驗過程

撰寫程式碼 → 連接麵包板、感測器、IC、LED → 執行 index.js →
 頁面操作 → (項目一)勾選 detect 並提交 → 開始偵測光敏電阻數值並
 顯示在網頁 → (項目二)勾選 no detect 並提交，跳轉頁面 → 勾選
 led 編號及狀態並提交 → 維持同一頁面，led 根據勾選值改變開關狀態

2. 解決方法

(1)問題一：頁面切換問題

◦ 描述：

無法正確切換頁面，使用 Post 後無法正常跳轉，ajax 抓取

detect 路由時顯示 HTML 代碼。

。解決方案：

在/detect 路由的一開始用 sendfile 發送 detect.html，暫時解決問題。

新增/detecting 路由，只負責呼叫 detect()函數，避免干擾其他內容。

(2)問題二：無法傳出 detect 的輸出

。描述：

函數輸出"good"，抓不到正確的 stdout 數據，傳到 HTML 的輸出變成[Object object]。

。解決方案：

檢查 print 輸出，確保只輸出需要的數據。

使用.then 和.catch 處理 detect()的 Promise，在.then 中 send(data)正確傳遞數據。

(3)問題三：光敏電阻讀取數據問題

。描述：

光敏電阻讀不到數據，即使使用之前 Lab 的代碼也是一樣，懷疑接觸不良。

。解決方案：

更換 IC、線、ADC 和麵包板。

發現電阻輸出腳和接地腳接反，調整後讀到數據。

(4)問題四：LED2 無法關閉

。描述：

LED2 無法關閉，重新接線後才會熄滅。

。解決方案：

檢查 Python 代碼和每個步驟的輸出。

懷疑是腳位問題，更換 LED2 連接的腳位後問題解決。

五、分工：

學號、組員	貢獻比例	工作內容
B812110004 葉芸茜	50%	文書處理、實驗設計與實作、程式規劃、測試與除錯
B812110011 湯青秀	50%	文書處理、實驗設計與實作、程式規劃、測試與除錯