物聯網氣體壓力偵測系統初步測試研製結案報告書

目錄

[前言 1](#_Toc99529476)

[第一章 系統架構介紹 1](#_Toc99529477)

[1-1節 硬體流程 1](#_Toc99529478)

[1-2節 雲端系統流程 2](#_Toc99529479)

[第二章 各部位程式碼介紹 3](#_Toc99529480)

[2-1節 Arduino(WeMos-D1\_WiFiClient)程式碼 3](#_Toc99529481)

[2-2節 雲端系統部分 8](#_Toc99529482)

[2-2-1節 資料庫規劃SQL 8](#_Toc99529483)

[2-2-2節 PHP WebSocket Server 12](#_Toc99529484)

[2-2-3節 val2DB.php (PHP GET RESTful API) 25](#_Toc99529485)

[2-2-4節 chart.php (即時氣體槽訊號顯示UI) 27](#_Toc99529486)

[第三章 最終雲端系統測試 32](#_Toc99529487)

[第四章 結論與備註 33](#_Toc99529488)

[4-1節 結論 33](#_Toc99529489)

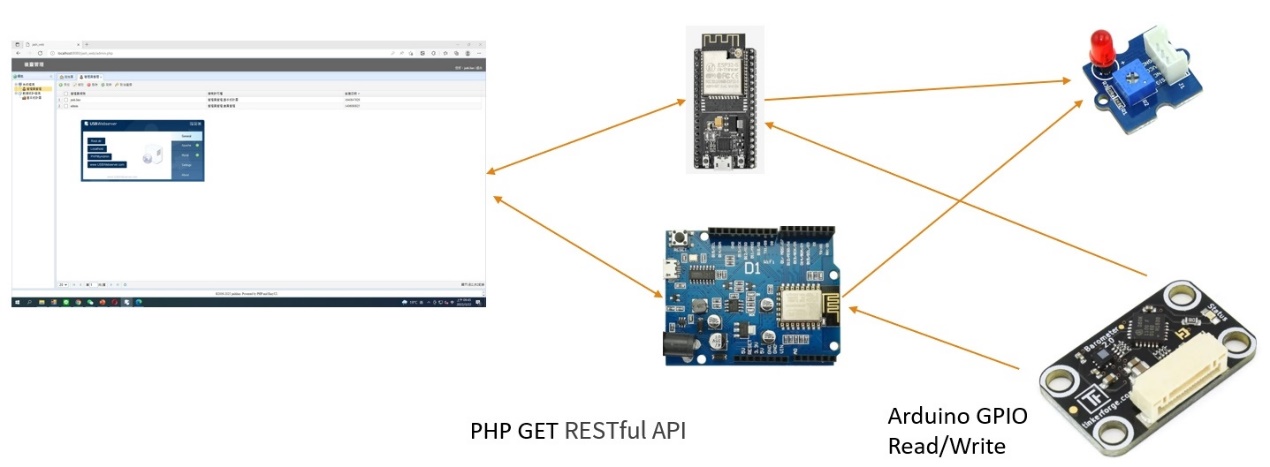
[4-2節 備註 33](#_Toc99529490)

# 前言

由於朋友拜託，想要建議一套可以透過Web Server與單晶片整合的氣體槽的即時監控系統，因此才有此專案的建立。

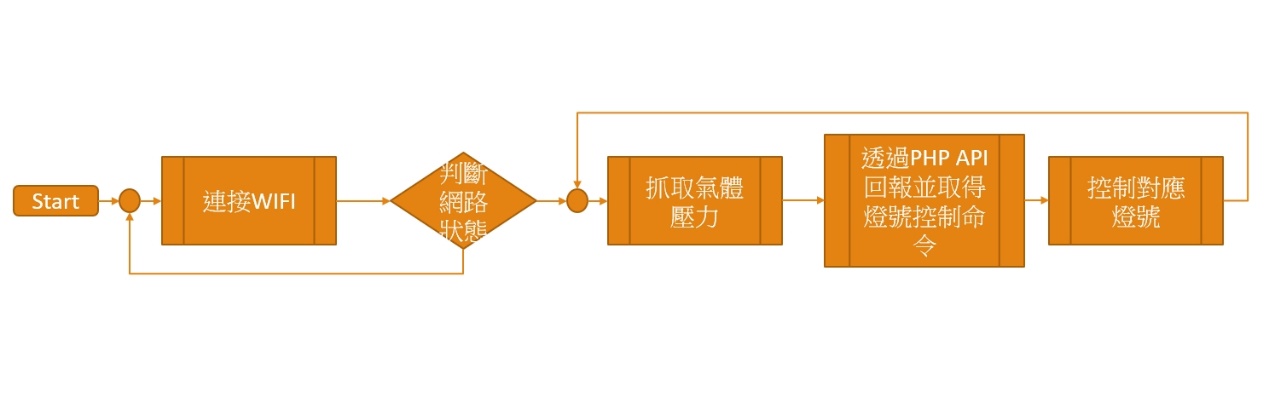
# 第一章 系統架構介紹

由於想要快速建立Web Server與單晶片的通訊機制，因此選用PHP+MySQL和 Arduino(WeMos-D1\_WiFiClient)作為開發環境與設備，其基本架構圖如下：



## 1-1節 硬體流程

由於該系統的主要目的是要透過單機片即時監控抓取對應氣體槽的相關參數並回報至雲端系統，因此其程式運作流程圖如下所示:

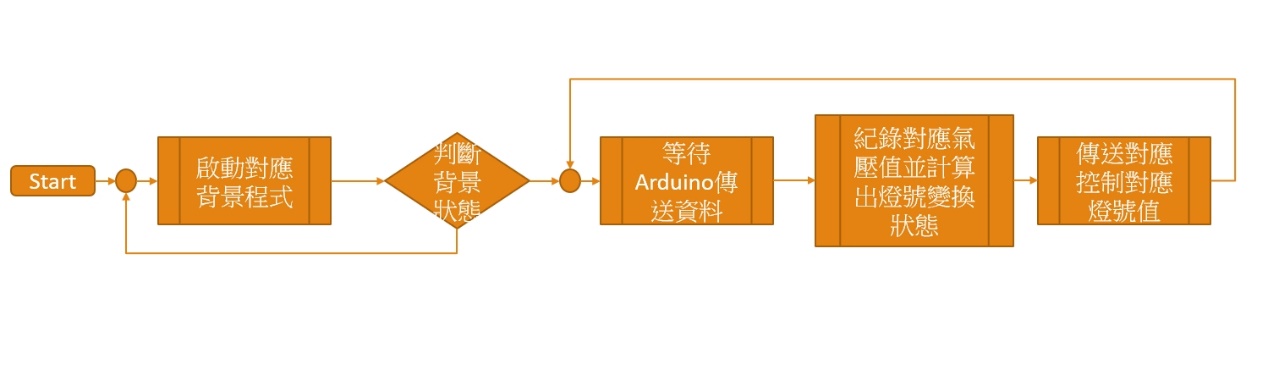


其詳細步驟解說如下:

1. 設備通電連接固定WiFi並取得對應網路IP。
2. 在取得網路存取權後，抓取對應氣體槽的相關參數值。
3. 透過雲端系統已經建立好的PHP GET RESTful API將相關參數傳送至雲端對應DB儲存，並在該連線的回饋中也取得對應的控制命令。
4. 取得並解析雲端系統的控制命令最後讓硬體執行對應的動作。

## 1-2節 雲端系統流程

在介紹完硬體流程之後，接下來就是雲端系統的部分，其程式運作流程圖如下所示:



其詳細步驟解說如下:

1. 啟動WebSocket Server背景服務程式藉此等待單定片呼叫。
2. 將單晶片回傳的氣體參數儲存至MySQL對應的資料庫之中。
3. 將上述的氣體參數值透過已知規則運算轉變成對應的控制命令回給單晶片。

# 第二章 各部位程式碼介紹

此系統的程式就如同系統架構圖所示可以大柱分為兩大區塊，一塊為單晶片的韌體部分，而另一部分為網站的雲端部分，其內容如下所示:

## 2-1節 Arduino(WeMos-D1\_WiFiClient)程式碼

|  |
| --- |
| /\*  \* This sketch sends data via HTTP GET requests to data.sparkfun.com service.  \*  \* You need to get streamId and privateKey at data.sparkfun.com and paste them  \* below. Or just customize this script to talk to other HTTP servers.  \*  \*/  #include <ESP8266WiFi.h>  const char\* ssid = "jash\_TStar";  const char\* password = "asd700502";  const char\* host = "192.168.0.191";  const char\* streamId = "....................";  const char\* privateKey = "....................";  int Blue=D4;//D4 有連接WIFI指示燈  int Green=D5;//D5 命令對應燈  int Red=D6;//D6 命令對應燈  void setup() {  //---  //設定燈號腳位工作模式  pinMode(Blue, OUTPUT);  pinMode(Green, OUTPUT);  pinMode(Red, OUTPUT);  //---設定燈號腳位工作模式    //---  //等待取得網路IP時，設定三燈全亮  digitalWrite(Blue, HIGH);  digitalWrite(Green, HIGH);  digitalWrite(Red, HIGH);  //---等待取得網路IP時，設定三燈全亮  Serial.begin(9600);  delay(50);  // We start by connecting to a WiFi network  Serial.println();  Serial.println();  Serial.print("Connecting to ");  Serial.println(ssid);    WiFi.begin(ssid, password);    while (WiFi.status() != WL\_CONNECTED) {  delay(50);  Serial.print(".");  }    //---  //取得網路IP時，所有燈熄滅  digitalWrite(Blue, LOW);//digitalWrite(Blue, HIGH);  digitalWrite(Green, LOW);  digitalWrite(Red, LOW);  //---取得網路IP時，所有燈熄滅    Serial.println("");  Serial.println("WiFi connected");  Serial.println("IP address: ");  Serial.println(WiFi.localIP());  delay(1000);  }  int value = 0;  void loop() {  value=random(10,50);  Serial.print("connecting to ");  Serial.println(host);    // Use WiFiClient class to create TCP connections  WiFiClient client;  const int httpPort = 8080;  if (!client.connect(host, httpPort)) {  Serial.println("connection failed");  client.stop();  digitalWrite(Blue, HIGH);  return;  }  else  {  digitalWrite(Blue, LOW);  }    // We now create a URI for the request  //String url = "/php\_code/val.php";  String url = "/jash\_web/val2DB.php";  url += "?string=";  url += value;    Serial.print("Requesting URL: ");  Serial.println(url);    // This will send the request to the server  client.print(String("GET ") + url + " HTTP/1.1\r\n" +  "Host: " + host + "\r\n" +  "Connection: close\r\n\r\n");    unsigned long timeout = millis();  while (client.available() == 0)  {  if (millis() - timeout > 5000)  {    Serial.println(">>> Client Timeout(5sec)!");  Serial.println("");  client.stop();  digitalWrite(Blue, HIGH);  return;  }  else  {  digitalWrite(Blue, LOW);  }  }    // Read all the lines of the reply from server and print them to Serial  while(client.available())  {  String line = client.readStringUntil('\r');  int index=line.indexOf("get data=");//搜尋資料列  if(index!=-1)  {  digitalWrite(Blue, LOW);  //----  //過濾不需要資料  int value=line.length();  char Buf[100];  line.toCharArray(Buf, 100);  line="";  //Serial.println(value);  for(int i=0;i<value;i++)  {  if((Buf[i]>=48)&&(Buf[i]<=57))//只留數字範圍  {  line+=Buf[i];  }  }  //----過濾不需要資料  switch(line.toInt())  {  case 0:  Serial.println("Red on & Green off");  digitalWrite(Red, HIGH);  digitalWrite(Green,LOW);  break;  case 1:  Serial.println("Red off & Green on");  digitalWrite(Red, LOW);  digitalWrite(Green,HIGH);  break;  }  break;  }  }    client.stop();  Serial.println("closing connection");  Serial.println("");  delay(1000);  } |

## 2-2節 雲端系統部分

### 2-2-1節 資料庫規劃SQL

|  |
| --- |
| -- phpMyAdmin SQL Dump  -- version 4.0.4.2  -- http://www.phpmyadmin.net  --  -- 主機: localhost  -- 產生日期: 2022 年 01 月 07 日 06:04  -- 伺服器版本: 5.6.13  -- PHP 版本: 5.4.17  SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";  SET time\_zone = "+00:00";  /\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;  /\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;  /\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;  /\*!40101 SET NAMES utf8 \*/;  --  -- 資料庫: `easyui`  --  CREATE DATABASE IF NOT EXISTS `easyui` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci;  USE `easyui`;  -- --------------------------------------------------------  --  -- 表的結構 `easyui\_admin`  --  CREATE TABLE IF NOT EXISTS `easyui\_admin` (  `id` int(10) unsigned NOT NULL AUTO\_INCREMENT COMMENT '自动编号',  `manager` char(20) NOT NULL COMMENT '管理员帐号',  `password` char(40) NOT NULL COMMENT '管理密码',  `auth` char(50) NOT NULL DEFAULT '' COMMENT '权限',  `date` int(10) unsigned NOT NULL COMMENT '创建时间',  PRIMARY KEY (`id`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=5 ;  --  -- 轉存資料表中的資料 `easyui\_admin`  --  INSERT INTO `easyui\_admin` (`id`, `manager`, `password`, `auth`, `date`) VALUES  (1, 'admin', '7c4a8d09ca3762af61e59520943dc26494f8941b', '管理員管理,會員管理', 1406080825),  (4, 'jash.liao', 'fecdda7743d6828422556e3b3ec271c1ecd711a5', '管理員管理,基本統計圖', 1640847929);  -- --------------------------------------------------------  --  -- 表的結構 `easyui\_nav`  --  CREATE TABLE IF NOT EXISTS `easyui\_nav` (  `id` int(10) unsigned NOT NULL AUTO\_INCREMENT COMMENT '自动编号',  `text` char(20) NOT NULL COMMENT '导航名称',  `state` char(10) NOT NULL DEFAULT '' COMMENT '导航状态',  `iconCls` char(20) NOT NULL DEFAULT '' COMMENT '导航图标',  `url` char(50) NOT NULL DEFAULT '' COMMENT '导航链接',  `nid` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '节点',  PRIMARY KEY (`id`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=5 ;  --  -- 轉存資料表中的資料 `easyui\_nav`  --  INSERT INTO `easyui\_nav` (`id`, `text`, `state`, `iconCls`, `url`, `nid`) VALUES  (1, '系統模塊', 'closed', 'icon-system', '', 0),  (2, '管理員管理', 'open', 'icon-manager', 'manager', 1),  (3, '數據統計模塊', 'closed', 'icon-shapes', '', 0),  (4, '基本統計圖', 'open', 'icon-chart', 'chart', 3);  -- --------------------------------------------------------  --  -- 表的結構 `val2db`  --  CREATE TABLE IF NOT EXISTS `val2db` (  `id` int(10) unsigned NOT NULL AUTO\_INCREMENT COMMENT '自动编号',  `date` char(50) NOT NULL COMMENT '寫入時間',  `value` int(10) unsigned NOT NULL COMMENT '數值',  PRIMARY KEY (`id`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=2 ;  --  -- 轉存資料表中的資料 `val2db`  --  INSERT INTO `val2db` (`id`, `date`, `value`) VALUES  (1, '2022-01-07 14:04:10', 32);  /\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;  /\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;  /\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/; |

### 2-2-2節 PHP WebSocket Server

啟動BAT

|  |
| --- |
| ..\..\php\php.exe server.php |

WebScoket Server的PHP

|  |
| --- |
| <?php  require 'config.php';//連接DB  error\_reporting(E\_ALL);  set\_time\_limit(0);// 设置超时时间为无限,防止超时  date\_default\_timezone\_set('Asia/shanghai');  if (!function\_exists('array\_column')) {  /\*  方法1：  此函数是PHP5.5后的函数，因此升级PHP到5.5以上即可解决  方法2：  PHP5.5之前版本可自定义一个array\_column()函数  \*/  function array\_column($arr2, $column\_key) {  $data = [];  foreach ($arr2 as $key => $value)  {  $data[] = $value[$column\_key];  }  return $data;  }  }  class WebSocket  {  const LOG\_PATH = './';  const LISTEN\_SOCKET\_NUM = 9;  /\*\*  \* @var array $sockets  \* [  \* (int)$socket => [  \* info  \* ]  \* ]  \* todo 解释socket与file号对应  \*/  private $sockets = [];  private $master;  public function \_\_construct($host, $port) {  try {  $this->master = socket\_create(AF\_INET, SOCK\_STREAM, SOL\_TCP);  // 设置IP和端口重用,在重启服务器后能重新使用此端口;  socket\_set\_option($this->master, SOL\_SOCKET, SO\_REUSEADDR, 1);  // 将IP和端口绑定在服务器socket上;  socket\_bind($this->master, $host, $port);  // listen函数使用主动连接套接口变为被连接套接口，使得一个进程可以接受其它进程的请求，从而成为一个服务器进程。在TCP服务器编程中listen函数把进程变为一个服务器，并指定相应的套接字变为被动连接,其中的能存储的请求不明的socket数目。  socket\_listen($this->master, self::LISTEN\_SOCKET\_NUM);  } catch (\Exception $e) {  $err\_code = socket\_last\_error();  $err\_msg = socket\_strerror($err\_code);  $this->error([  'error\_init\_server',  $err\_code,  $err\_msg  ]);  }  $this->sockets[0] = ['resource' => $this->master];  $pid = get\_current\_user();//LINUX指令 ~ posix\_getpid();  $this->debug(["server: {$this->master} started,pid: {$pid}"]);  while (true) {  try {  $this->doServer();  } catch (\Exception $e) {  $this->error([  'error\_do\_server',  $e->getCode(),  $e->getMessage()  ]);  }  }  }  private function doServer()  {  $write = $except = NULL;  $sockets = array\_column($this->sockets, 'resource');//從輸入數組中的單個列返回值,搜尋resource  $read\_num = socket\_select($sockets, $write, $except, NULL);  // select作为监视函数,参数分别是(监视可读,可写,异常,超时时间),返回可操作数目,出错时返回false;  if (false === $read\_num) {  $this->error([  'error\_select',  $err\_code = socket\_last\_error(),  socket\_strerror($err\_code)  ]);  return;  }  foreach ($sockets as $socket) {  // 如果可读的是服务器socket,则处理连接逻辑  if ($socket == $this->master)  {  $client = socket\_accept($this->master);//獲取read數組中活動的socket，並且把不活躍的從read數組中刪除  // 创建,绑定,监听后accept函数将会接受socket要来的连接,一旦有一个连接成功,将会返回一个新的socket资源用以交互,如果是一个多个连接的队列,只会处理第一个,如果没有连接的话,进程将会被阻塞,直到连接上.如果用set\_socket\_blocking或socket\_set\_noblock()设置了阻塞,会返回false;返回资源后,将会持续等待连接。  if (false === $client)  {  $this->error([  'err\_accept',  $err\_code = socket\_last\_error(),  socket\_strerror($err\_code)  ]);  continue;  }  else  {  self::connect($client);  continue;  }  }  else  {  // 如果可读的是其他已连接socket,则读取其数据,并处理应答逻辑  $bytes = @socket\_recv($socket, $buffer, 2048, 0);  if ($bytes < 9)  {  $recv\_msg = $this->disconnect($socket);  }  else  {  if (!$this->sockets[(int)$socket]['handshake'])  {  self::handShake($socket, $buffer);  continue;  }  else  {  $recv\_msg = self::parse($buffer);  }  }    //PHP array\_unshift 用來在陣列開頭插入一個或多個元素或陣列，用法與 array\_shift 相反。  //array\_unshift ( 原本的陣列 , 要插入的元素或陣列 );  array\_unshift($recv\_msg, 'receive\_msg');  $msg = self::dealMsg($socket, $recv\_msg);  //--OK echo 'jashliao:'.$msg;  $this->broadcast($msg);  }  }  }  /\*\*  \* 将socket添加到已连接列表,但握手状态留空;  \*  \* @param $socket  \*/  public function connect($socket)  {  socket\_getpeername($socket, $ip, $port);  $socket\_info = [  'resource' => $socket,  'uname' => '',  'handshake' => false,  'ip' => $ip,  'port' => $port,  ];  $this->sockets[(int)$socket] = $socket\_info;  $this->debug(array\_merge(['socket\_connect'], $socket\_info));  }  /\*\*  \* 客户端关闭连接  \*  \* @param $socket  \*  \* @return array  \*/  private function disconnect($socket)  {  $recv\_msg = [  'type' => 'logout',  'content' => $this->sockets[(int)$socket]['uname'],  ];  unset($this->sockets[(int)$socket]);  return $recv\_msg;  }  /\*\*  \* 用公共握手算法握手  \*  \* @param $socket  \* @param $buffer  \*  \* @return bool  \*/  public function handShake($socket, $buffer)  {  // 获取到客户端的升级密匙  $line\_with\_key = substr($buffer, strpos($buffer, 'Sec-WebSocket-Key:') + 18);  $key = trim(substr($line\_with\_key, 0, strpos($line\_with\_key, "\r\n")));  // 生成升级密匙,并拼接websocket升级头  $upgrade\_key = base64\_encode(sha1($key . "258EAFA5-E914-47DA-95CA-C5AB0DC85B11", true));// 升级key的算法  $upgrade\_message = "HTTP/1.1 101 Switching Protocols\r\n";  $upgrade\_message .= "Upgrade: websocket\r\n";  $upgrade\_message .= "Sec-WebSocket-Version: 13\r\n";  $upgrade\_message .= "Connection: Upgrade\r\n";  $upgrade\_message .= "Sec-WebSocket-Accept:" . $upgrade\_key . "\r\n\r\n";  socket\_write($socket, $upgrade\_message, strlen($upgrade\_message));// 向socket里写入升级信息  $this->sockets[(int)$socket]['handshake'] = true;  socket\_getpeername($socket, $ip, $port);  $this->debug([  'hand\_shake',  $socket,  $ip,  $port  ]);  // 向客户端发送握手成功消息,以触发客户端发送用户名动作;  $msg = [  'type' => 'handshake',  'content' => 'done',  ];  $msg = $this->build(json\_encode($msg));  socket\_write($socket, $msg, strlen($msg));  return true;  }  /\*\*  \* 解析数据  \*  \* @param $buffer  \*  \* @return bool|string  \*/  private function parse($buffer)  {  $decoded = '';  $len = ord($buffer[1]) & 127;  if ($len === 126) {  $masks = substr($buffer, 4, 4);  $data = substr($buffer, 8);  } else if ($len === 127) {  $masks = substr($buffer, 10, 4);  $data = substr($buffer, 14);  } else {  $masks = substr($buffer, 2, 4);  $data = substr($buffer, 6);  }  for ($index = 0; $index < strlen($data); $index++) {  $decoded .= $data[$index] ^ $masks[$index % 4];  }  return json\_decode($decoded, true);  }  /\*\*  \* 将普通信息组装成websocket数据帧  \*  \* @param $msg  \*  \* @return string  \*/  private function build($msg)  {  $frame = [];  $frame[0] = '81';  $len = strlen($msg);  if ($len < 126) {  $frame[1] = $len < 16 ? '0' . dechex($len) : dechex($len);  } else if ($len < 65025) {  $s = dechex($len);  $frame[1] = '7e' . str\_repeat('0', 4 - strlen($s)) . $s;  } else {  $s = dechex($len);  $frame[1] = '7f' . str\_repeat('0', 16 - strlen($s)) . $s;  }  $data = '';  $l = strlen($msg);  for ($i = 0; $i < $l; $i++) {  $data .= dechex(ord($msg{$i}));  }  $frame[2] = $data;  $data = implode('', $frame);  return pack("H\*", $data);  }  /\*\*  \* 拼装信息  \*  \* @param $socket  \* @param $recv\_msg  \* [  \* 'type'=>user/login  \* 'content'=>content  \* ]  \*  \* @return string  \*/  private function dealMsg($socket, $recv\_msg)  {  /\*  $msg\_type = $recv\_msg['type'];  $msg\_content = $recv\_msg['content'];  $response = [];  switch ($msg\_type) {  case 'login':  $this->sockets[(int)$socket]['uname'] = $msg\_content;  // 取得最新的名字记录  $user\_list = array\_column($this->sockets, 'uname');  $response['type'] = 'login';  $response['content'] = $msg\_content;  $response['user\_list'] = $user\_list;  break;  case 'logout':  $user\_list = array\_column($this->sockets, 'uname');  $response['type'] = 'logout';  $response['content'] = $msg\_content;  $response['user\_list'] = $user\_list;  break;  case 'user':  $uname = $this->sockets[(int)$socket]['uname'];  $response['type'] = 'user';  $response['from'] = $uname;  $response['content'] = $msg\_content;  break;  }  \*/  $k=0;  $data\_array = array();  $response = [];  $query = mysql\_query("SELECT id,value FROM val2db ORDER BY id DESC LIMIT 0,6") or die('SQL 錯誤！');  while (!!$row = mysql\_fetch\_array($query)) {  $data\_array[$k] = $row[1];  // --OK echo $row[1].'\r\n';  $k++;  }  $response['d\_array'] = array\_reverse($data\_array);  return $this->build(json\_encode($response));  }  /\*\*  \* 广播消息  \*  \* @param $data  \*/  private function broadcast($data)  {  foreach ($this->sockets as $socket) {  if ($socket['resource'] == $this->master) {  continue;  }  socket\_write($socket['resource'], $data, strlen($data));  }  }  /\*\*  \* 记录debug信息  \*  \* @param array $info  \*/  private function debug(array $info)  {  $time = date('Y-m-d H:i:s');  array\_unshift($info, $time);  $info = array\_map('json\_encode', $info);  file\_put\_contents(self::LOG\_PATH . 'websocket\_debug.log', implode(' | ', $info) . "\r\n", FILE\_APPEND);  }  /\*\*  \* 记录错误信息  \*  \* @param array $info  \*/  private function error(array $info)  {  $time = date('Y-m-d H:i:s');  array\_unshift($info, $time);  $info = array\_map('json\_encode', $info);  file\_put\_contents(self::LOG\_PATH . 'websocket\_error.log', implode(' | ', $info) . "\r\n", FILE\_APPEND);  }  }  $ws = new WebSocket("127.0.0.1", "8081"); |

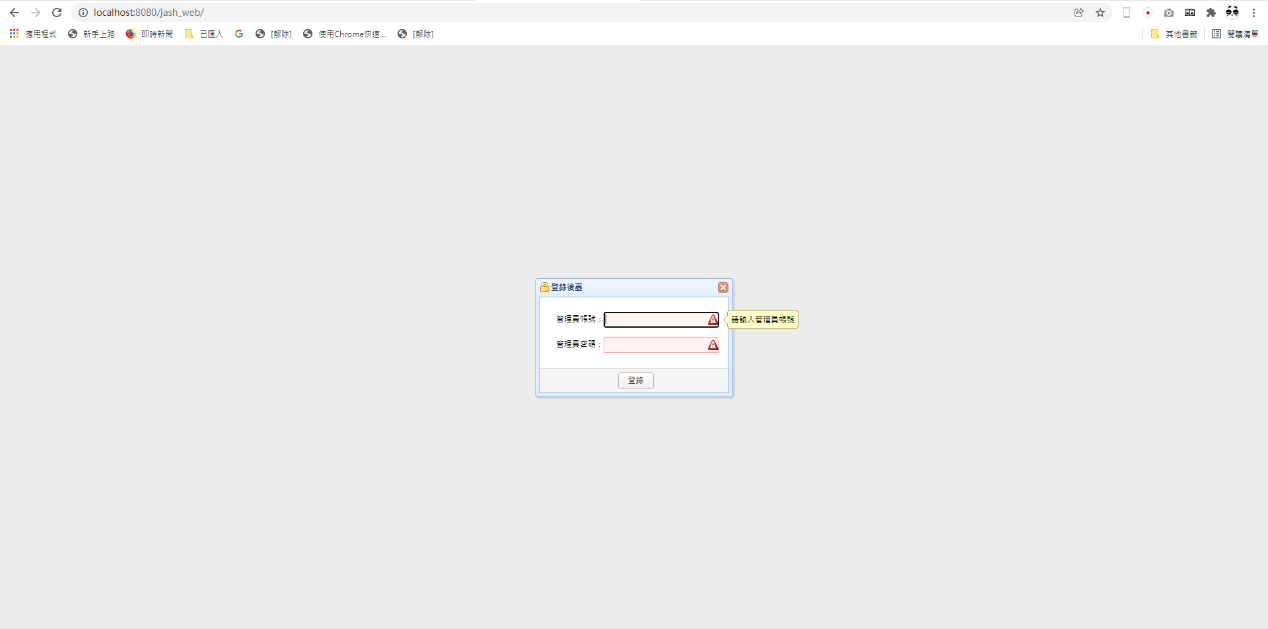
### 2-2-3節 val2DB.php (PHP GET RESTful API)

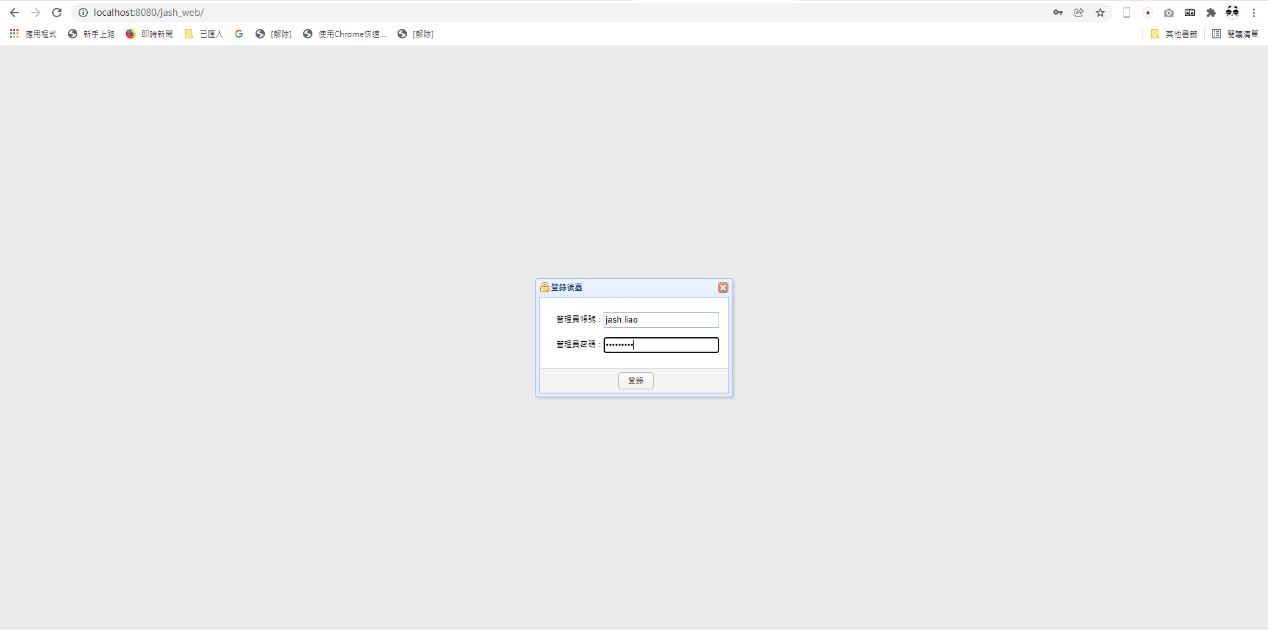
|  |
| --- |
| <?php  require 'config.php';//連接DB  require 'ws.php';//PHP WebSocket Client，因為JS 不能被Arduino 執行 所以只能找PHP  //error\_reporting(0);//停止報錯  date\_default\_timezone\_set("Asia/Taipei");  $time = date("Y-m-d H:i:s");  //echo $time.'<br/>';    //---  //接收Arduino資料  $val = $\_GET['string'];  /\*  $bf = fopen("test.txt","a+");  fwrite ($bf,$val."\r\n");  fclose($bf);  \*/  //---接收Arduino資料  //---  //將資料寫入DB  //echo "INSERT INTO val2db (date, value) VALUES ('$time','$val')";  mysql\_query("INSERT INTO val2db (date, value) VALUES ('$time','$val')") or die('SQL 錯誤！');  mysql\_close();  //---將資料寫入DB  //---  //觸發WebSocket Server 後推資料動力  $ws\_client = new ws(array  (  'host' => '127.0.0.1',  'port' => 8081,  'path' => ''  ));  $result = $ws\_client->send('message');  $ws\_client->close();  //---觸發WebSocket Server 後推資料動力    //---  //回傳Arduino資料  echo "get data=";  if(($val%2)==0)  {  echo 0;  }  else  {  echo 1;  }  //---回傳Arduino資料    ?> |

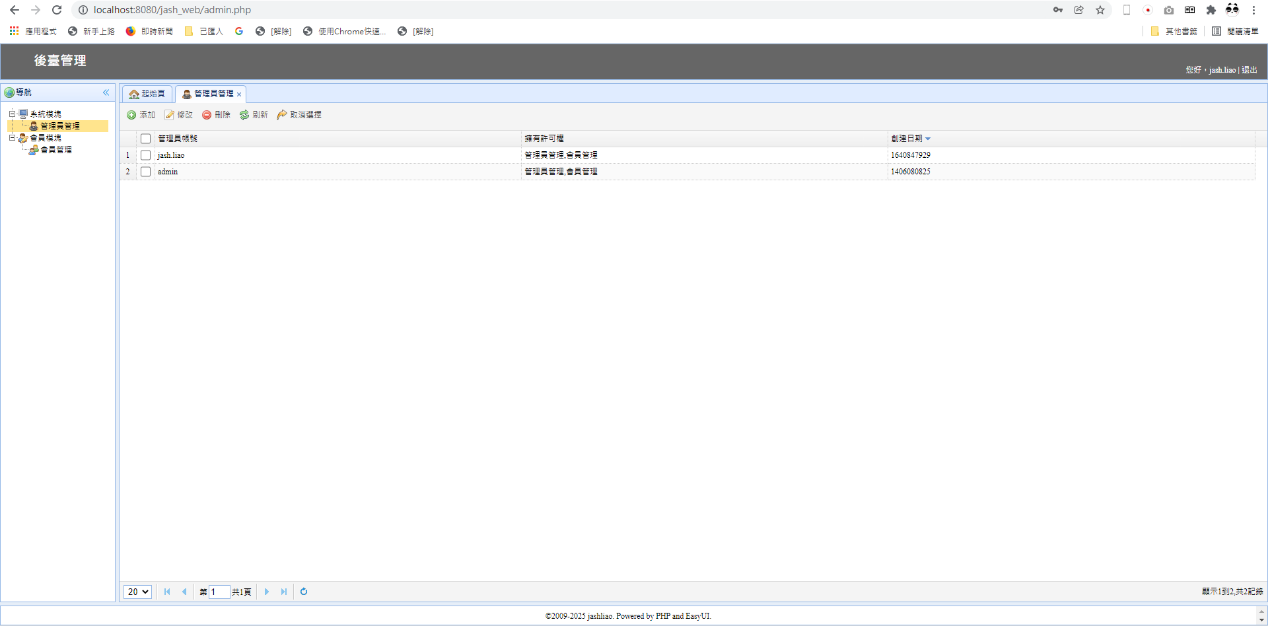
### 2-2-4節 chart.php (即時氣體槽訊號顯示UI)

|  |
| --- |
| <?php  /\*  //user.php作為來源  error\_reporting(0);//停止報錯    date\_default\_timezone\_set("Asia/Taipei");  echo date("Y-m-d H:i:s")."<br/>";　// 常用的完整表示法，分別為年、月、日、時、分、秒，輸出結果類似 2013-06-05 05:12:50  \*/  ?>  <script src="echarts/echarts.min.js"></script>  <script type="text/javascript" src="easyui/jquery.min.js"></script>  <!-- JS 顯示時間區域 -->  <div align="center">  <font size="6">  <span id='clock'>    </span>  </font>  </div>  <!-- <button onclick="StopFunction()">Stop it</button> -->  <a href="#" class="easyui-linkbutton" data-options="iconCls:'icon-reload'" style="width:80px" onclick="StopFunction()">Stop it</a>  <!-- 为ECharts准备一个具备大小（宽高）的Dom -->  <div id="main" style="width: 960px;height:720px;"></div>  <script type="text/javascript">  // 基于准备好的dom，初始化echarts实例  var myChart = echarts.init(document.getElementById('main'));  var gCount=0;  var t;  var ws = new WebSocket("ws://localhost:8081");  ws.onmessage = function (e) {  /\*  var msg = JSON.parse(e.data);  var sender, user\_name, name\_list, change\_type;  switch (msg.type) {  case 'system':  sender = '系统消息: ';  break;  case 'user':  sender = msg.from + ': ';  break;  case 'handshake':  var user\_info = {'type': 'login', 'content': uname};  sendMsg(user\_info);  return;  case 'login':  case 'logout':  user\_name = msg.content;  name\_list = msg.user\_list;  change\_type = msg.type;  dealUser(user\_name, change\_type, name\_list);  return;  }  var data = sender + msg.content;  listMsg(data);  \*/  if (e.data.indexOf('d\_array') !== -1)  {  var msg = JSON.parse(e.data);  showtime();  showecharts(msg.d\_array);  console.log(msg.d\_array);  }  };    function showtime()  {  var now,hours,minutes,seconds,timeValue;  now = new Date();  hours = now.getHours();  minutes = now.getMinutes();  seconds = now.getSeconds();  timeValue = now.getFullYear()+"年";  timeValue += (((now.getMonth()+1) < 10) ? " 0" : " ")+(now.getMonth()+1)+"月";  timeValue += ((now.getDate()< 10) ? " 0" : " ")+now.getDate()+"日&emsp;";  timeValue += (hours >= 12) ? "下午 " : "上午 ";  timeValue += ((hours > 12) ? hours - 12 : hours) + " 點";  timeValue += ((minutes < 10) ? " 0" : " ") + minutes + " 分";  timeValue += ((seconds < 10) ? " 0" : " ") + seconds + " 秒";  clock.innerHTML = timeValue;  }    function showecharts(value)  {  var option;    var SwitchData = value;  var X\_Data = ["V1","V2","V3","V4","V5","V6"];    // 指定图表的配置项和数据  option = {  title: {  text: 'Arduino Value'  },  tooltip: {},  legend: {  data:['Value']  },  xAxis: {  data: ["V1","V2","V3","V4","V5","V6"]  },  yAxis: {},  series: [{  name: 'Value',  type: 'bar',  data: SwitchData  }]  };  // 使用刚指定的配置项和数据显示图表。  myChart.setOption(option);  }    function showAuto()  {  showtime();  showecharts([]);  }    function StopFunction()  {  clearTimeout(t);  history.go(0);  }    $(function(){  showAuto();  })    </script> |

# 第三章 最終雲端系統測試







# 第四章 結論與備註

## 4-1節 結論

01.因為錢伯目前跑路中，無法最終調校

02.記得先收訂金

## 4-2節 備註

正所謂有夢最美希望相隨，不一定哪一天錢伯會回來要求繼續開發，為了怕忘記先傳到GITHUB備份:『https://github.com/jash-git/IOT-Smart-Application-Design-Practice-Class-Project-2022』，有興趣的也請自取。