MatrixMiniR4 Library User Manual (V1.1.0)

模組清單

模組名稱	功能	內建 / 外接	備註
M1 M2 M3 M4	提供直流馬達驅動功能	內建	
RC1 RC2 RC3 RC4	提供伺服馬達驅動功能	內建	
BTN_UP BTN_DOWN	提供自訂按鈕功能	內建	
LED	提供控制 RGB LED 功能	內建	
Motion	提供加速度、陀螺儀、歐拉角資訊	內建	
Buzzer	提供控制蜂鳴器功能	內建	
OLED	提供控制 OLED 功能	內建	
WiFi	提供 WiFi 相關功能	內建	
PWR	提供電量相關資訊	內建	
I2C0 I2C1 I2C2 I2C3 I2C4	可連搭配下 I2C 介面模組使用 Matrix Motion Matrix Laser Matrix Color	外接	I2C0 與 A3 介面共用
D1 D2 D3 D4	提供數位輸入輸出、PWM 輸出	外接	D1 左側支援 PWM 輸出 D2 左側支援 PWM 輸出 D3 右側支援 PWM 輸

模組名稱	功能	內建 / 外接	備註
			出 D4 右側支援 PWM 輸出
A1 A2 A3	提供數位類比、類比、數位輸入輸出	外接	A1 支援 DAC 功能
Uart	提供 UART 功能	外接	硬體 UART
PS2	提供 PS2 控制手把功能	外接	占用 D2、D3
Vernier	提供 VernierLib 相關功能	外接	
Vision	可搭配 Matrix Vision 模組使用	外接	

API 說明

MatrixMiniR4

> MiniR4

```
/// @brief 初始化
/// @return true: 成功, false: 失敗
bool begin()
```

MiniR4DC

> M1、M2、M3、M4

```
/// @brief 初始化
/// @return true:成功, false:失敗
bool begin(void)

/// @brief 設定反轉方向
/// @param dir true:反轉, false:正轉
```

```
/// @return true:成功, false:失敗
bool setReverse(bool dir)

/// @brief 設定速度
/// @param power -100~100
/// @return true:成功, false:失敗
bool setPower(int16_t power)

/// @brief 設定速度（PID）
/// @param speed -100~100
/// @return true:成功, false:失敗
bool setSpeed(int16_t speed)

/// @brief 旋轉角度
/// @param power 速度 -100~100
/// @param degree 角度 0~32767
/// @return true:成功, false:失敗
bool rotateFor(int16_t power, uint16_t degree)

/// @brief 設定定速PID參數
/// @return true:成功, false:失敗
bool setFixSpeedPID(float kp, float ki, float kd)

/// @brief 設定旋轉PID參
/// @return true:成功, false:失敗
bool setRotatePID(float kp, float ki, float kd)

/// @brief 取得編碼器計數值
/// @return 計數值
int32_t getCounter(void)

/// @brief 取得馬達轉動角度
/// @return 馬達轉動角度
int32_t getDegrees(void)

/// @brief 重設編碼器計數值
```

```
/// @return true:成功, false:失敗
bool resetCounter(void)

/// @brief 馬達煞車
/// @param brake true:brake, false:coast
/// @return true:成功, false:失敗
bool setBrake(bool brake)
```

MiniR4RC

> RC1、RC2、RC3、RC4

```
/// @brief 初始化
/// @return true:成功, false:失敗
bool begin(void)

/// @brief 設定硬體方向
/// @param dir true:正轉, false:反轉
/// @return true:成功, false:失敗
bool setHWDir(bool dir)

/// @brief 設定角度
/// @param angle 0~180
/// @return true:成功, false:失敗
bool setAngle(uint16_t angle)
```

MiniR4BTN

> BTN_UP、BTN_DOWN

```
/// @brief 取得按鈕狀態
/// @return true:按下, false:放開
bool getState(void)
```

MiniR4LED

LED

```
/// @brief 初始化
/// @param pin 使用腳位
void begin(uint8_t pin)

/// @brief 設定顏色（Hex）
/// @param idx LED 編號 1~2
/// @param rgb Hex 顏色
/// @return true:成功, false:失敗
bool setColor(uint8_t idx, uint32_t rgb)

/// @brief 設定顏色（r ,g ,b）
/// @param idx LED 編號 1~2
/// @param r Red 顏色 0~255
/// @param g Green 顏色 0~255
/// @param b Blue 顏色 0~255
/// @return true:成功, false:失敗
bool setColor(uint8_t idx, uint8_t r, uint8_t g, uint8_t b)

/// @brief 設定亮度
/// @param idx LED 編號 1~2
/// @param brightness 亮度 0~255
void setBrightness(uint8_t idx, uint8_t brightness)
```

MiniR4Motion

Motion

```
/// @brief 取得陀螺儀數值
/// @param axis 軸向 (X, Y, Z)
/// @return 指定軸向的數值
double getGyro(AxisType axis)

/// @brief 取得加速度計數值
/// @param axis 軸向 (X, Y, Z)
/// @return 指定軸向的數值
double getAccel(AxisType axis)

/// @brief 取得歐拉角數值
/// @param axis 軸向 (Roll, Pitch, Yaw)
/// @return 指定軸向的數值
int16_t getEuler(AxisType axis)

/// @brief 重置 IMU 數值
/// @return true:成功, false:失敗
bool resetIMUValues(void)
```

MiniR4BUZZER

Buzzer

```
/// @brief 初始化
/// @param pin 使用的腳位
void begin(uint8_t pin)

/// @brief 發出指定頻率聲音
/// @param frequency 頻率
```

```
/// @param duration 持續時間
void Tone(uint16_t frequency, uint32_t duration)

/// @brief 停止發聲
void NoTone(void)
```

MiniR4I2C

> I2C0、I2C1、I2C2、I2C3、I2C4

MXMotion

```
/// @brief 初始化
/// @return true: 成功, false: 失敗
bool begin();

/// @brief 取得歐拉角 Roll 數值
/// @return Roll 數值
int getRoll();

/// @brief 取得歐拉角 Pitch 數值
/// @return Pitch 數值
int getPitch();

/// @brief 取得歐拉角 Yaw 數值
/// @return Yaw 數值
int getYaw();

/// @brief 取得陀螺儀數值
/// @param axis 軸向 (X, Y, Z)
/// @return 指定軸向的數值
int getGyro(AxisType axis);

/// @brief 取得加速度計數值
```

```
/// @param axis 軸向 (X, Y, Z)
/// @return 指定軸向的數值
int getAccel(AxisType axis);
```

MXLaser

```
/// @brief 初始化
/// @return true: 成功, false: 失敗
bool     begin();

/// @brief 取得距離
/// @return 距離
uint16_t getDistance();
```

MatrixColor

```
/// @brief 初始化
/// @return true: 成功, false: 失敗
bool begin();

void setGamma(bool state);
void setLight(bool state, bool mode, uint8_t pwm);

/// @brief 取得顏色值
/// @param color 顏色種類
/// @return 顏色值
uint8_t getColor(ColorType color);

/// @brief 取得灰階值
/// @return 灰階值
uint8_t getGrayscale();

/// @brief 取得顏色編號
/// @return 顏色編號
uint8_t getColorNumber();
```

MiniR4PWM

D1、D2、D3、D4

```
/// @brief 設定左側 PWM 輸出
/// @param level 0~255
void setPWML(uint8_t level)
{
    pinMode(_pin1, OUTPUT);
    analogWrite(_pin1, level);
}

/// @brief 設定右側 PWM 輸出
/// @param level 0~255
void setPWMR(uint8_t level)
{
    pinMode(_pin2, OUTPUT);
    analogWrite(_pin2, level);
}

    /// @brief 取得左側數位輸入
/// @param pullup 是否啟用上拉電阻
/// @return true:高電位, false:低電位
bool getL(bool pullup = false)
{
    if (pullup) {
        pinMode(_pin1, INPUT_PULLUP);
    } else {
        pinMode(_pin1, INPUT);
    }
    return digitalRead(_pin1);
}
```

```
/// @brief 取得右側數位輸入
/// @param pullup 是否啟用上拉電阻
/// @return true:高電位, false:低電位
bool getR(bool pullup = false)
{
    if (pullup) {
        pinMode(_pin2, INPUT_PULLUP);
    } else {
        pinMode(_pin2, INPUT);
    }
    return digitalRead(_pin2);
}

/// @brief 設定左側數位輸出
/// @param level true:高電位, false:低電位
void setL(bool level = HIGH)
{
    pinMode(_pin1, OUTPUT);
    digitalWrite(_pin1, level);
}

/// @brief 設定右側數位輸出
/// @param level true:高電位, false:低電位
void setR(bool level = HIGH)
{
    pinMode(_pin2, OUTPUT);
    digitalWrite(_pin2, level);
}

/// @brief 切換左側數位輸出
void toggleL()
{
    pinMode(_pin1, OUTPUT);
    digitalWrite(_pin1, !digitalRead(_pin1));
}
```

```
/// @brief 切換右側數位輸出
void toggleR()
{
    pinMode(_pin2, OUTPUT);
    digitalWrite(_pin2, !digitalRead(_pin2));
}
```

MiniR4HC04（US）

```
/// @brief 取得 HC-SR04 距離
/// @return 距離 (公分), -1: timeout
float getDistance(void)
```

MiniR4DAC、MiniR4Analog（包含 MiniR4PWM 方法）

A1

```
/// @brief 設定 DAC 左側輸出
void setDACR(int level)
{
    pinMode(_pin1, OUTPUT);
    analogWrite(_pin1, level);
}
```

A1、A2、A3

```
/// @brief 取得左側類比輸入
/// @return 0~1023
int getAIL(void)
{
    pinMode(_pin1, INPUT);
    return analogRead(_pin1);
```

```
}

/// @brief 取得右側類比輸入
/// @return 0~1023
int getAIR(void)
{
    pinMode(_pin2, INPUT);
    return analogRead(_pin2);
}
```

MiniR4Power

PWR

```
/// @brief 設定電池節數
/// @param cell 電池節數
/// @return true:成功, false:失敗
bool setBattCell(uint8_t cell)

/// @brief 取得電池電壓
/// @return 電壓 (V)
float getBattVoltage(void)

/// @brief 取得電池電量百分比
/// @return 電量百分比 (%)
float getBattPercentage(void)
```