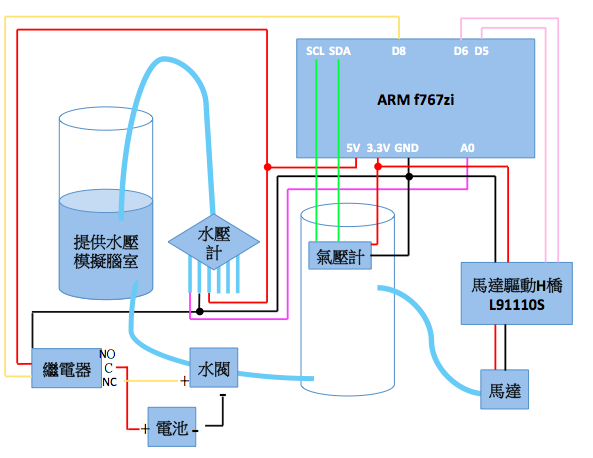
**嵌入式系統期末專題**

10311001 王思敏 10311006 石家瑜

專題簡介

當腦壓因為腦脊髓液過多而上升時，必須適當引流來降低腦壓。本專題以此為概念，模擬當腦壓上升時，利用負壓引流的原理，慢速的抽出腦脊髓液，以免病患因流速過快而感到暈眩。而控制氣室壓力高低是利用即時調整馬達轉速，來達到動態平衡之效果。

硬體接線



接腳

|  |  |
| --- | --- |
| 水壓計 | Nucleo-F767ZI |
| PIN 1 | A0 |
| PIN 2 | Gnd |
| PIN 3 | 5V |

|  |  |
| --- | --- |
| 繼電器 | Nucleo-F767ZI |
| S | D8 |
| + | 5V |
| - | Gnd |
| NO | 水閥正極 |
| C | 電池正極 |
| NC | Null |

|  |  |
| --- | --- |
| 水閥 |  |
| + | 繼電器NO |
| - | 電池負極 |

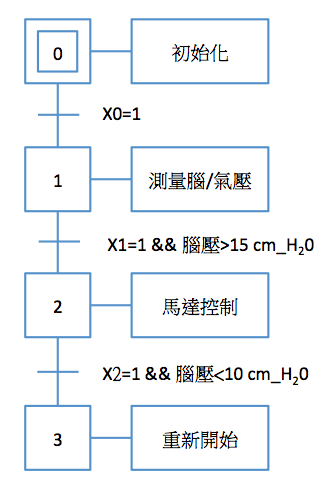
|  |  |
| --- | --- |
| 氣壓計 | Nucleo-F767ZI |
| Vin | 3.3V |
| Gud | Gnd |
| SCL | PB\_8 |
| SDA | PB\_9 |

|  |  |
| --- | --- |
| L9110S | Nucleo-F767ZI |
| Vcc | 3.3V |
| Gud | Gnd |
| A-IA | D5 |
| A-IB | D6 |

API定義

|  |  |  |
| --- | --- | --- |
| 功能 | API | 參數 |
| 初始化氣壓計 | initialize | Null |
| 量氣壓 | air\_pre | Null |
| 量水(腦)壓 | water\_pressure | Null |
| 平均水(腦)壓 | getbrain\_pressure | Null |
| 初始化水閥 | initialize2 | Null |
| 水閥控制 | water\_valve\_flow | Null |
| 控制馬達轉速 | motor.drive | int power |
| 判斷馬達狀態 | PumpControl | float meas, float set, float band, int th, int state |

Grafcet



code

#include "mbed.h"

#include "BMP180.h"

#include "L9110S.h"

//L9110S模組

/\*\*\*\*\* Definitions \*\*\*\*\*/

#define I2C\_ADDR (0xEE) // 1110111x

#define REG\_ADDR\_RESET (0xE0)

#define REG\_ADDR\_ID (0xD0)

#define REG\_ADDR\_CTRL (0xF4)

#define REG\_ADDR\_DATA (0xF6)

#define REG\_ADDR\_AC1 (0xAA)

#define CTRL\_REG\_TEMP (0x2E)

#define CTRL\_REG\_PRESS\_0 (0x34)

#define CTRL\_REG\_PRESS\_1 (0x74)

#define CTRL\_REG\_PRESS\_2 (0xB4)

#define CTRL\_REG\_PRESS\_3 (0xF4)

L9110S::L9110S(PinName cw\_r, PinName ccw\_r):

cw\_min(14), cw\_max(70), ccw\_min(14), ccw\_max(70), \_cw(cw\_r), \_ccw(ccw\_r), periode(1000), power\_act(0)

{

\_cw.period\_us( periode);

\_ccw.period\_us(periode);

\_cw.pulsewidth\_us( 0);

\_ccw.pulsewidth\_us(0);

}

/\*\* sets Pulswidth cw(+) ccw(-)

\*

\* @param power positiv in clockwise Dir

\* @param power negativ in counter clockwise Dir

\*

\* in % of range from min\_(cw/ccw) to max\_(cw/ccw)

\*

\*/

void L9110S::drive(int power)

{

// Limit PWM -100 to 100

if (power > 100) power = 100;

if (power < -100) power = -100;

power\_act = power;

// Calc PWM in us

if (power > 0) {power = ((power \* ( cw\_max - cw\_min)) + ( cw\_min \* 100)) \* periode / 10000;}

else { if (power < 0) {power = ((power \* (ccw\_max - ccw\_min)) - (ccw\_min \* 100)) \* periode / 10000;}

else power = 0;}

//cw or ccw Rotate

if (power >= 0){\_ccw.pulsewidth\_us(0); \_cw.pulsewidth\_us(power);}

else {\_cw.pulsewidth\_us(0); \_ccw.pulsewidth\_us(-power);}

}

/\*\* sets Pulswidth and Dir

\*

\* @param dir positiv dir is clockwise direction

\* @param dir negativ dir is counter clockwise direction

\* @param dir zero is stop

\*

\* @param power in % of range from min\_(cw/ccw) to max\_(cw/ccw) allways positiv

\*

\*/

void L9110S::frequency(int hz)

{

periode = 1000000/hz;

}

float L9110S::deg\_diff(float Soll, float Ist)

{

float diff = Soll - Ist;

if(diff > 180) diff = -360 + diff;

if(diff < -180) diff = 360 + diff;

return diff;

}

void L9110S::drive\_diff(float Soll, float Ist, int kp ,int i\_lim, int trash)

{

int p = 0; //Leistungsvariable

float diff = deg\_diff(Soll, Ist ); //Winkeldiffernz

diff = diff - (power\_act / 20.0); //Winkeldiffernz - D Anteil aus Motorleistung 10? bei 100%

if ( fabs(diff) < (trash / 10.0)) {p = 0; drive\_i \*= 0.9;} //Leistung 0, wenn innerhalb Threshold

else

{

p = diff \* kp; //Leistungsbedarf aus Winkeldifferenz mal Kp

drive\_i = drive\_i + ( p / 20.0); //Berechnung I Anteil

if (drive\_i > i\_lim) drive\_i = i\_lim;

if (drive\_i < -i\_lim) drive\_i = -i\_lim;

}

if (p > 100) (p = 100);

if (p < -100) (p = -100);

power\_act = (6 \* power\_act + p) / 7;

drive(drive\_i/10 + p);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//氣壓模組

BMP180::BMP180(PinName sda, PinName scl)

{

i2c\_ = new I2C(sda, scl);

i2c\_owner = true;

i2c\_->frequency(400000);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

BMP180::BMP180(I2C \*i2c) :

i2c\_(i2c)

{

i2c\_owner = false;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

BMP180::~BMP180()

{

if(i2c\_owner) {

delete i2c\_;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int BMP180::init(void)

{

char addr;

char data[22];

int i;

if (checkId() != 0) {

return -1;

}

addr = REG\_ADDR\_AC1;

if (i2c\_->write(I2C\_ADDR, &addr, 1) != 0) {

return -1;

}

if (i2c\_->read(I2C\_ADDR, data, 22) != 0) {

return -1;

}

for (i = 0; i < 11; i++) {

calib.value[i] = (data[2\*i] << 8) | data[(2\*i)+1];

}

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int BMP180::reset(void)

{

char data;

data = REG\_ADDR\_RESET;

if (i2c\_->write(I2C\_ADDR, &data, 1) != 0) {

return -1;

}

data = 0xB6;

if (i2c\_->write(I2C\_ADDR, &data, 1) != 0) {

return -1;

}

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int BMP180::checkId(void)

{

char addr;

char data;

addr = REG\_ADDR\_ID;

if (i2c\_->write(I2C\_ADDR, &addr, 1) != 0) {

return -1;

}

if (i2c\_->read(I2C\_ADDR, &data, 1) != 0) {

return -1;

}

if (data != 0x55) {

return -1;

}

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int BMP180::startPressure(BMP180::oversampling\_t oss)

{

char data[2];

data[0] = REG\_ADDR\_CTRL;

data[1] = CTRL\_REG\_PRESS\_0 | ((oss & 0x3) << 6);

oss\_ = oss;

if (i2c\_->write(I2C\_ADDR, data, 2) != 0) {

return -1;

}

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int BMP180::getPressure(int \*pressure)

{

char addr, byte[3];

uint32\_t up;

int32\_t b6, x1, x2, x3, b3, p;

uint32\_t b4, b7;

addr = REG\_ADDR\_DATA;

if (i2c\_->write(I2C\_ADDR, &addr, 1) != 0) {

return -1;

}

if (i2c\_->read(I2C\_ADDR, byte, 3) != 0) {

return -1;

}

up = ((byte[0] << 16) | (byte[1] << 8) | byte[2]) >> (8 - oss\_);

b6 = b5 - 4000;

x1 = (b6 \* b6) >> 12;

x1 \*= calib.b2;

x1 >>= 11;

x2 = calib.ac2 \* b6;

x2 >>= 11;

x3 = x1 + x2;

b3 = (((((int32\_t)calib.ac1) \* 4 + x3) << oss\_) + 2);

b3 >>= 2;

x1 = (calib.ac3 \* b6) >> 13;

x2 = (calib.b1 \* ((b6 \* b6) >> 12)) >> 16;

x3 = (x1 + x2 + 2) >> 2;

b4 = (calib.ac4 \* (uint32\_t)(x3 + 32768)) >> 15;

b7 = ((uint32\_t)up - b3) \* (50000 >> oss\_);

p = ((b7 < 0x80000000) ? ((b7 << 1) / b4) : ((b7 / b4) \* 2));

x1 = p >> 8;

x1 \*= x1;

x1 = (x1 \* 3038) >> 16;

x2 = (-7357 \* p) >> 16;

p += (x1 + x2 + 3791) >> 4;

\*pressure = p;

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int BMP180::startTemperature(void)

{

char data[2] = { REG\_ADDR\_CTRL, CTRL\_REG\_TEMP };

if (i2c\_->write(I2C\_ADDR, data, 2) != 0) {

return -1;

}

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int BMP180::getTemperature(float \*tempC)

{

char addr, byte[2];

uint16\_t ut;

int32\_t x1, x2;

addr = REG\_ADDR\_DATA;

if (i2c\_->write(I2C\_ADDR, &addr, 1) != 0) {

return -1;

}

if (i2c\_->read(I2C\_ADDR, byte, 2) != 0) {

return -1;

}

ut = (byte[0] << 8) | byte[1];

x1 = ((ut - calib.ac6) \* calib.ac5) >> 15;

x2 = (calib.mc << 11) / (x1 + calib.md);

b5 = x1 + x2;

\*tempC = (float)(b5 + 8) / 160;

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int BMP180::getTemperature(int16\_t \*tempCx10)

{

char addr, byte[2];

uint16\_t ut;

int32\_t x1, x2;

addr = REG\_ADDR\_DATA;

if (i2c\_->write(I2C\_ADDR, &addr, 1) != 0) {

return -1;

}

if (i2c\_->read(I2C\_ADDR, byte, 2) != 0) {

return -1;

}

ut = (byte[0] << 8) | byte[1];

x1 = ((ut - calib.ac6) \* calib.ac5) >> 15;

x2 = (calib.mc << 11) / (x1 + calib.md);

b5 = x1 + x2;

\*tempCx10 = (b5 + 8) >> 4;

return 0;

}

//腳位定義

I2C i2c(I2C\_SDA, I2C\_SCL);

BMP180 bmp180(&i2c);

Serial pc(USBTX, USBRX);

AnalogIn ain(A0);

DigitalOut waterval(D8);

DigitalOut led1(PB\_0);

L9110S motor\_x(D5, D6);

//參數定義

float brain\_pressure;

float init\_air;

float air;

float deadband = 1.5;

int threshPressure = 1080;

float setPressure;

float speed1 = 0;

int pumpState = 0;

//量氣壓與氣溫

int air\_pre (void) {

bmp180.startTemperature();

wait\_ms(5); // Wait for conversion to complete

float temp;

if(bmp180.getTemperature(&temp) != 0) {

pc.printf("Error getting temperature\n\r");

}

bmp180.startPressure(BMP180::ULTRA\_LOW\_POWER);

wait\_ms(10); // Wait for conversion to complete

int pressure = 0;

if(bmp180.getPressure(&pressure) != 0) {

pc.printf("Error getting pressure\n\r");

}

return pressure;

}

//量測與校正水壓

float water\_pressure (void){

AnalogIn ain(A0);

float voltage = ain.read();

float sensorValue = ((voltage - 0.04)/0.09); //Voltage = Vout/Vs , expressed in percentage // Vs = Vin reference voltage

// print out the value you read:

return sensorValue;

}

//水閥控制

void water\_valve\_flow (void){

waterval = 1;

wait(0.5);

waterval = 0;

wait(0.5);

}

//馬達需求狀態判斷

int PumpControl(float meas, float set, float band, int th, int state){

if(meas>th){

state = 0;

pc.printf("Over the negative pressure threshold:%d hPa\n\r",th);

pc.printf(" .... Pump is stopped \n\r");

motor\_x.drive(1);

return state;

}

if(meas<=(set - band)) state = 1; //state=1為加速

if(meas>set - band && meas<set + band){

state = 2; //state=2為定速

water\_valve\_flow();

}

if(meas>=(set + band)) state = 3; //state=3為減速

return state;

}

//對氣壓計進行初始化

void initialize(void){

while(1){

if(bmp180.init() != 0) {

pc.printf("Error communicating with BMP180\n\r");

wait(1);

}

else {

pc.printf("Initialized BMP180\n\r");

wait(1);

break;

}

}

}

//取得10比腦壓平均，並將單位換為百帕

float getbrain\_pressure(void){

float bra\_pre = 0;

for (int i = 0;i < 10;i++){

bra\_pre = bra\_pre + water\_pressure()\*10;

}

bra\_pre = (bra\_pre/10)-8;

return bra\_pre;

}

//初始化水閥

void initialize2(void){

waterval = 0;

}

//取得10比氣壓平均，並將單位換為百帕

float airpre(void){

int a = 0;

for (int i = 0;i < 10;i++){

a = a + air\_pre();

}

return a/1000;

}

//初始化

void action0(){

initialize();

initialize2();

}

//顯示腦壓與氣壓值

void action1(){

brain\_pressure = getbrain\_pressure();

init\_air = airpre();

pc.printf("brain pressure: %f hPa\n\r Air pressure:%f hPa\n\r",brain\_pressure, init\_air);

wait(1);

}

//馬達控制

void action2(){

speed1 = ((brain\_pressure-10)/5)\*20;

setPressure = init\_air+brain\_pressure-3;

pc.printf("Set pressure:%f\n\r", setPressure);

pumpState = PumpControl(air,setPressure,deadband,threshPressure,pumpState);

if(pumpState!=0){

motor\_x.drive(speed1); //馬達開始運作

pc.printf("Pump is activated \n\r");

}

if(pumpState!=0){

pumpState = PumpControl(air,setPressure,deadband,threshPressure,pumpState);

//調整馬達速度

if(pumpState==3){

speed1 = speed1-1;

motor\_x.drive (speed1);

pc.printf("Slow down the pump... \n\r");

}

if(pumpState==1 & speed1!=0){

speed1 = speed1+1;

motor\_x.drive (speed1);

pc.printf("Speed up the pump... \n\r");

}

}

brain\_pressure = getbrain\_pressure();

air = airpre();

pc.printf("brain pressure: %f hPa\n\r Air pressure:%f hPa\n\r",brain\_pressure + init\_air, air);

}

int x0=1;

int x1=0;

int x2=0;

int x3=0;

void grafcet0(){

if(x0==1){

action0();

x0=0;

x1=1;

}

if(x1==1){

action1();

if(brain\_pressure > 15){

x1=0;

x2=1;

}

}

if(x2==1 && brain\_pressure > 15 ){

action2();

if (brain\_pressure < 10){

led1 = 1;

wait(5);

led1 = 0;

x3=1;

x2=0;

}

}

if(x3==1 && brain\_pressure < 10){

x0=1;

x3=0;

}

}

int main(void) {

while(1){

grafcet0();

}

}

結果





