ภาคผนวก

ภาคผนวก ก

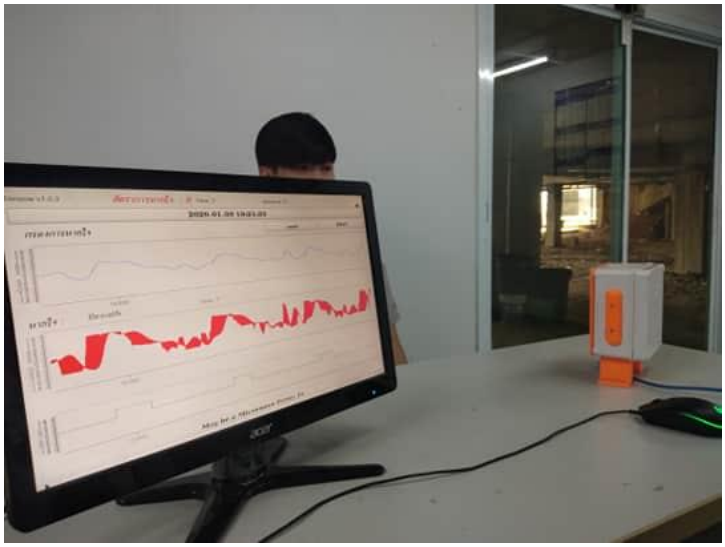
อุปกรณ์ระบบวิเคราะห์อัตราการหายใจด้วยไมโครเวฟเซ็นเซอร์

ภาพที่ ก.1 อุปกรณ์ระบบวัดอัตราการหายใจด้วยไมโครเวฟเซ็นเซอร์

ภาพที่ ก.2 หน้าจอแสดงผลของระบบวัดอัตราการหายใจด้วยไมโครเวฟเซ็นเซอร์

ภาพที่ ก.3 การติดตั้งส่วนของระบบวัดอัตราการหายใจด้วยไมโครเวฟเซ็นเซอร์

จากภาพที่ ก.3 ทำการติดตั้งอุปกรณ์วัดอัตราการหายใจด้วยไมโครเวฟเซ็นเซอร์ในระยะที่ 70 เซนติเมตร บนโต๊ะคัดกรองผู้ป่วย โดยโมดูลเซนเซอร์ไมโครเวฟต้องหันไปในทิศทางเดียวกับผู้ป่วยหรือให้ตรงกับตัวผู้ป่วย



ภาพที่ ก.4 การติดตั้งส่วนหน้าจอมอนิเตอร์ของระบบ

จากภาพที่ ก.4 ติดตั้งหน้าจอมอนิเตอร์ต่อกับอุปกรณ์วัดอัตราการหายใจ เพื่อให้ผู้คัดกรองสามารถดูกราฟการหายใจและอัตราการหายใจได้

ภาคผนวก ข

โปรแกรมระบบวิเคราะห์อัตราการหายใจด้วยไมโครเวฟเซ็นเซอร์

ภาพที่ ข.1 หน้าแสดงผลของระบบวัดอัตราการหายใจด้วยไมโครเวฟเซ็นเซอร์

จากภาพที่ ข.1 คือหน้าแสดงผลของระบบวัดอัตราการหายใจด้วยไมโครเวฟเซ็นเซอร์ อธิบายการทำงานดังนี้

หมายเลขที่ 1 เมื่อหายใจครบ 5 ครั้งจะทำการคำนวนอัตรากายหายใจต่อ 1 นาที

หมายเลขที่ 2 เวลาของการหายใจต่อ 1 นาที

หมายเลขที่ 3 อัตราการหายใจและแจ้งความผิดปกติของการหายใจ

หมายเลขที่ 4 เวลาและวันที่ของวันที่ตรวจวัดอัตราการหายใจ

หมายเลขที่ 5 เมื่อกดปุ่ม Reset จะเริ่มต้นการทำงานของระบบวัดอัตราการหายใจใหม่

หมายเลขที่ 6 เมื่อกดปุ่ม Exit เมื่อต้องการปิดโปรแกรม

ภาคผนวก ค

โปรแกรม

```
#include "GraphSharpWindow.h"

#include "ui_GraphSharpWindow.h"

#include <SkIoSharp.h>

#include <SkIoPlayload.h>

#include <qwt_plot_curve.h>

#include <qwt_series_data.h>

#include <stdlib.h>

#define  MAX_TIME_SERIES       5000

#define  MAX_AMPLITUDE_FILTER 5000

#include <QCoreApplication>

#include <mqttpublish.h>

#include <QThread>


/*-------------------------------------------------------------------------------------------------------------*/

int millisecondx = 0;

int secondx = 0;

bool T_Up, T_Down = false ;

int beeeeeeee  = 0;

int time_count =0;

bool stat_time_count = false;

int A =0;
```

```
class GraphSharpWindowPrivate : public QObject{
public  :
    QScopedPointer<SkIoSharp> sharp;
    QScopedPointer<QwtPlotCurve>            pBody;
    QScopedPointer< QVector<double> >       xBody;
    QScopedPointer< QVector<double> >       yBody;
    QScopedPointer<QwtPlotCurve>            pHeart;
    QScopedPointer< QVector<double> >       xHeart;
    QScopedPointer< QVector<double> >       yHeart;
    QScopedPointer<QwtPlotCurve>            pBreath;
    QScopedPointer< QVector<double> >       xBreath;
    QScopedPointer< QVector<double> >       yBreath;
    QScopedPointer< QVector<double> >       ySpectrum;
    QScopedPointer<QwtPlotCurve>            hCurve;
    QScopedPointer<QwtPlotCurve>            bCurve;
    QScopedPointer< QVector<double> >       hPhase;
    QScopedPointer< QVector<double> >       bPhase;
    QScopedPointer< QVector<double> >       xWave;
    QScopedPointer< QVector<double> >       yWave;
    QScopedPointer<QwtPlotCurve>            pAVg;
```

```
    QScopedPointer< QVector<double> >       xAvg;
    QScopedPointer< QVector<double> >       yAvg;
    double                                  nInput;
    double                                  hPrev;
    double                                  bPrev;
    double                                  minX;
    int                                 nPoints;
    int                                 nWindow;
    int                                 nAvg;};
GraphSharpWindow::GraphSharpWindow(QWidget *parent) : QMainWindow(parent),
ui(new Ui::GraphSharpWindow), d(new GraphSharpWindowPrivate()){
    ui->setupUi(this);
    d->pBody.reset(new QwtPlotCurve("Body"));
    d->xBody.reset(new QVector<double>());
    d->yBody.reset(new QVector<double>());
    d->pHeart.reset(new QwtPlotCurve("Heart"));
    d->xHeart.reset(new QVector<double>());
    d->yHeart.reset(new QVector<double>());
    d->pBreath.reset(new QwtPlotCurve("Breath"));
    d->xBreath.reset(new QVector<double>());
    d->yBreath.reset(new QVector<double>());
```

```
d->hCurve.reset(new QwtPlotCurve("HeartPhase"));

d->bCurve.reset(new QwtPlotCurve("BodyPhase"));

d->hPhase.reset(new QVector<double>());

d->bPhase.reset(new QVector<double>());

d->xWave.reset(new QVector<double>());

d->yWave.reset(new QVector<double>());

d->xAvg.reset(new QVector<double>());

d->yAvg.reset(new QVector<double>());

d->ySpectrum.reset(new QVector<double>());

d->nInput  = 1;

d->nWindow = 8;

d->nPoints = 1;

d->hPrev   = 0.0;

d->bPrev   = 0.0;

d->minX    = 0;

d->nAvg    = 0;

SkIoPlayload *playload;

tm_dd_int_ = playload->NowString("dd").toInt();

tm_hh_int_ = playload->NowString("hh").toInt();
```

```
    d->pBody->attach(ui->sharpBody);
    d->pBody->setSamples(*d->xBreath.data(), *d->yBreath.data());
    d->pBody->setPen(QPen(QColor("green"), 1));
    d->pBody->setStyle(QwtPlotCurve::Lines);
    d->pHeart->attach(ui->sharpHeart);
    d->pHeart->setSamples(*d->xBreath.data(), *d->yBreath.data());
    d->pHeart->setPen(QPen(QColor("blue"), 1));
    d->pHeart->setStyle(QwtPlotCurve::Lines);
    d->pBreath->attach(ui->sharpBreath);
    d->pBreath->setSamples(*d->xBreath.data(), *d->yBreath.data());
    d->pBreath->setPen(QPen(QColor("red"), 1));
    d->pBreath->setStyle(QwtPlotCurve::Lines);
    d->xWave->clear();
    d->yWave->clear();
---------- ส่วนกำหนดค่าแสดงกราฟ ----------------------------------------
    ui->sharpBody->setAxisScale(QwtPlot::yLeft     ,  -2,      2, 0.3);
    ui->sharpHeart->setAxisScale(QwtPlot::yLeft    ,  -2,      2, 0.3);
    ui->sharpBreath->setAxisScale(QwtPlot::yLeft   ,  -2,      2, 0.3);
    ui->sharpHeart->setAxisScale(QwtPlot::xBottom ,    0, 2500   , 2500);
    ui->sharpBreath->setAxisScale(QwtPlot::xBottom,    0, 2500   , 2500);
    ui->sharpBody->setAxisScale(QwtPlot::xBottom   ,    0, 2500   , 2500);
```

```
    d->sharp.reset(new SkIoSharp());

    d->sharp->connect(d->sharp.data(), SIGNAL(sharpChange(SkIoPlayload*)), this,
SLOT(sharpChange(SkIoPlayload*)));

    analyze.reset(new AnalyzeSignal());

    connect(analyze.data(), SIGNAL(OnSignalObject(IdentifyAction, signalObject)), this,
SLOT(OnSignalObject(IdentifyAction, signalObject)));}

GraphSharpWindow::~GraphSharpWindow()

{    analyze.reset();

    d->pBody.reset();

    d->pHeart.reset();

    d->pBreath.reset();

    d->sharp.reset();

    d->ySpectrum.reset();

    d.reset();

    delete ui;}

----------  ส่วนของการวัดการเจอตัวบุคคล ----------

void GraphSharpWindow::sharpChange(SkIoPlayload *playload)

{ if (playload->Type() == SkIoPlayload::kWaveform) {

        if(show_ui == true){

            this->sharpSmooth(playload->Body());

            int value = playload->Body();
```

```
analyze->addSignal(value);
millisecondx++;
if(millisecondx == 100 ){
    secondx++;
    millisecondx = 0;              }
d->xBody->append((d->nInput)/d->nPoints);
d->yBody->append((int)A);
if (d->yBody->size() > MAX_TIME_SERIES) {
    d->xBody->remove(0, d->yBody->size() - MAX_TIME_SERIES);
    d->yBody->remove(0, d->yBody->size() - MAX_TIME_SERIES);              }
d->pBody->setSamples(*d->xBody.data(), *d->yBody.data());
d->xHeart->append((d->nInput)/d->nPoints);
d->yHeart->append((double)playload->Breath() / (32768.0/1.0));
if (d->yHeart->size() > MAX_TIME_SERIES) {
 d->xHeart->remove(0, d->yHeart->size() - MAX_TIME_SERIES);
d->yHeart->remove(0, d->yHeart->size() - MAX_TIME_SERIES);              }
d->pHeart->setSamples(*d->xHeart.data(), *d->yHeart.data());
d->xBreath->append((d->nInput)/d->nPoints);
d->yBreath->append((double)playload->Breath() / (32768.0/2.0));
if (d->yBreath->size() > MAX_TIME_SERIES) {
 d->xBreath->remove(0, d->yBreath->size() - MAX_TIME_SERIES);
```

```
        d->yBreath->remove(0, d->yBreath->size() - MAX_TIME_SERIES);
        }
        d->pBreath->setSamples(*d->xBreath.data(), *d->yBreath.data());

        if ((d->nInput / d->nPoints) > 2500) {
            ui->sharpHeart->setAxisScale(QwtPlot::xBottom , (d->nInput / d->nPoints) -
2500, (d->nInput / d->nPoints) + 1, 2500);
            ui->sharpBreath->setAxisScale(QwtPlot::xBottom, (d->nInput / d->nPoints) -
2500, (d->nInput / d->nPoints) + 1, 2500);
            ui->sharpBody->setAxisScale(QwtPlot::xBottom, (d->nInput / d->nPoints) -
2500, (d->nInput / d->nPoints) + 1, 2500); }
        ui->sharpHeart->replot();
        ui->sharpBreath->replot();
        ui->sharpBody->replot();
        d->nInput++;
        if(d->nInput >= 100000){    // Reset UI Graph
            test_UI();
            QThread::msleep(100);
            d->sharp.reset(new SkIoSharp());
            d->sharp->connect(d->sharp.data(), SIGNAL(sharpChange(SkIoPlayload*)), this,
SLOT(sharpChange(SkIoPlayload*))); } }
```

```
------------------------------ ส่วนของการ นับอัตราการหายใจ ----------------------

        QString Ber = QString("%1").arg((double)playload->Breath());

        int Be = Ber.toInt();

        if (Be > 13495){

            T_Down = true;

            A =1; }

        if (Be < 13495){

            T_Up = true;

            T_Down = false;

            A =0; }

        if(T_Down != false && T_Up !=false){

            beeeeeeee = beeeeeeee+1;

            stat_time_count = true;

            T_Down = false ;

            T_Up   = false; }

        ui->label_Time->setText(" Time : "+QString::number(secondx));

        ui->label_A->setText(" Time : "+QString::number(A));

        if(secondx == 60){

             if(beeeeeeee >= 15 || beeeeeeee <= 20){

                ui->label_Breath_time->setText(" nomal "+QString::number(beeeeeeee));

                secondx = 0; }
```

```
                if(beeeeeeee > 15 && beeeeeeee < 20) {

                    ui->label_Breath_time->setText(" abnomal
"+QString::number(beeeeeeee));

                    secondx = 0; }

          beeeeeeee =0; }

        ui->label_br->setStyleSheet("QLabel { color : red;}");

        ui->label_br->setText(" อัตราการหายใจ  : "+QString::number(playload->turnBreate()));

        microwave(playload->turnHeart(),playload->turnBreate());

        ui->pushButton_time->setStyleSheet("QLabel { color : blue;}");

        QString tm = playload->NowString("yyyy-MM-dd hh:mm:ss");

        ui->pushButton_time->setText(tm);

        int tm_dd_int = playload->NowString("dd").toInt();

        int tm_hh_int = playload->NowString("hh").toInt(); }

    delete playload;}

void GraphSharpWindow::OnSignalObject(IdentifyAction action, signalObject so){

void GraphSharpWindow::on_pushButton_clicked()

{    d->sharp.reset(new SkIoSharp());

    d->sharp->connect(d->sharp.data(), SIGNAL(sharpChange(SkIoPlayload*)), this,
SLOT(sharpChange(SkIoPlayload*)));}

void GraphSharpWindow::sharpSmooth(int body)

{    int    avg    = 0;
```

```
    double weight = 0.0;


    d->yAvg->append(body);
    if (d->yAvg->size() > d->nWindow) {
        for (int index = 0; index < d->nWindow; index++) {
            if ((d->yAvg->at(index) > -8000) && (d->yAvg->at(index) < 8000)) {
                avg += (d->yAvg->at(index)/ 4);
            } else {
                avg += (d->yAvg->at(index)); } }
        d->xBreath->append((d->nWindow) + d->nAvg++);
        d->yBreath->append((double)((avg/ d->nWindow) / (32768.0/2.0)));
        if (d->yBreath->size() > MAX_TIME_SERIES) {
            d->xBreath->remove(0, d->yBreath->size() - MAX_TIME_SERIES);
            d->yBreath->remove(0, d->yBreath->size() - MAX_TIME_SERIES);}
        d->pBreath->setSamples(*d->xBreath.data(), *d->yBreath.data());
        d->yAvg->remove(0, d->yAvg->size() - d->nWindow);
        ui->sharpBreath->replot();}}
void GraphSharpWindow::getBreate(){
}QString GraphSharpWindow::turnBreate(){
    return fBreate;}
QString GraphSharpWindow::turnHeart(){
```

```
    return fHeart;}
void GraphSharpWindow::on_pushBtn_clicked()
{    if(show_ui == false){
        show_ui = true;
    }else{
        ui->pushBtn->setText(" SHOW Graph ");
        show_ui = false; }}
void GraphSharpWindow::on_pushButton_time_clicked()
{  d->sharp.reset(new SkIoSharp());
    d->sharp->connect(d->sharp.data(), SIGNAL(sharpChange(SkIoPlayload*)), this,
SLOT(sharpChange(SkIoPlayload*)));
}void GraphSharpWindow::test_UI(){
    d->pBody.reset(new QwtPlotCurve("Body"));
    d->xBody.reset(new QVector<double>());
    d->yBody.reset(new QVector<double>());
    d->pHeart.reset(new QwtPlotCurve("Heart"));
    d->xHeart.reset(new QVector<double>());
    d->yHeart.reset(new QVector<double>());
    d->pBreath.reset(new QwtPlotCurve("Breath"));
    d->xBreath.reset(new QVector<double>());
    d->yBreath.reset(new QVector<double>());
```

```
d->hCurve.reset(new QwtPlotCurve("HeartPhase"));

d->bCurve.reset(new QwtPlotCurve("BodyPhase"));

d->hPhase.reset(new QVector<double>());

d->bPhase.reset(new QVector<double>());

d->xWave.reset(new QVector<double>());

d->yWave.reset(new QVector<double>());

d->xAvg.reset(new QVector<double>());

d->yAvg.reset(new QVector<double>());

d->ySpectrum.reset(new QVector<double>());

d->nInput  = 1;

d->nWindow = 8;

d->nPoints = 1;

d->hPrev   = 0.0;

d->bPrev   = 0.0;

d->minX    = 0;

d->nAvg    = 0;

d->pBody->attach(ui->sharpBody);

d->pBody->setSamples(*d->xBreath.data(), *d->yBreath.data());

d->pBody->setPen(QPen(QColor("green"), 1));

d->pBody->setStyle(QwtPlotCurve::Lines);

d->pHeart->attach(ui->sharpHeart);
```

```
    d->pBreath->setSamples(*d->xBreath.data(), *d->yBreath.data());
    d->pHeart->setPen(QPen(QColor("blue"), 1));
    d->pHeart->setStyle(QwtPlotCurve::Lines);
    d->pBreath->attach(ui->sharpBreath);
    d->pBreath->setSamples(*d->xBreath.data(), *d->yBreath.data());
    d->pBreath->setPen(QPen(QColor("red"), 1));
    d->pBreath->setStyle(QwtPlotCurve::Lines);
    d->xWave->clear();
    d->yWave->clear();
    ui->sharpBody->setAxisScale(QwtPlot::yLeft     ,  -2,      2, 1);
    ui->sharpBody->setAxisScale(QwtPlot::xBottom   ,    0, 2500   , 2500);
    ui->sharpHeart->setAxisScale(QwtPlot::yLeft    ,  -2,      2, 0.3);
    ui->sharpBreath->setAxisScale(QwtPlot::yLeft   ,  -2,      2, 0.3);
    ui->sharpHeart->setAxisScale(QwtPlot::xBottom ,    0, 2500   , 2500);
    ui->sharpBreath->setAxisScale(QwtPlot::xBottom,    0, 2500   , 2500);
    d->sharp.reset(new SkIoSharp());
    d->sharp->connect(d->sharp.data(), SIGNAL(sharpChange(SkIoPlayload*)), this,
SLOT(sharpChange(SkIoPlayload*)));
    analyze.reset(new AnalyzeSignal());
    connect(analyze.data(), SIGNAL(OnSignalObject(IdentifyAction, signalObject)), this,
SLOT(OnSignalObject(IdentifyAction, signalObject)));
```

```
}void GraphSharpWindow::on_pushButton_reset_ui_clicked()
{ test_UI();
    QThread::msleep(100);
}void GraphSharpWindow::on_pushButton_exit_clicked()
{    QCoreApplication::exit();
}QString GraphSharpWindow::NowString(QString format)
{    QDateTime dt    = QDateTime::currentDateTime();
    QString tmstr   = dt.toString(format);
    return tmstr;
}QString tmOld;
QString tmNew;
QString tmNow;
bool isChanging = true;
bool errorStart = true;
int oldHeart = 80;
int timeOld;
int timeNow;
int timerCount = 0;
int globalCount = 0;
void GraphSharpWindow::microwave(int Heart,int Breath){ // HR Detection
    tmNow = NowString("ss");
```

```
timeNow = tmNow.toInt();
if(tmNew != tmNow){ // Increases the two timers used
    timerCount++;
    if(globalCount <= 200){
        globalCount++; } }
if(Heart*1.05 >= oldHeart && Heart*0.95 <= oldHeart){
    isChanging = false; // If new heart rate doesn't change by more than 5%, ignore. }
else {    isChanging = true; }
//-------ERROR DETECTION-------------------------
if(globalCount <= 120){ // Calibrating time (2 mins)
    timerCount = 0;
    ui->pushBtn->setText(" Please Wait: System Calibrating "); }
else { if(Heart > 100){ // High Heart Rate
        if(errorStart){ // Starts 30s timer
            tmOld = NowString("ss");
            timeOld = tmOld.toInt();
            errorStart = false;  }
        if(timerCount < 30){
            if(isChanging){
                qDebug()<< "May be a Microwave Error";}
```

```
            ui->pushBtn->setText(" May be a Microwave Error " +
QString::number(timerCount) + "s"); }
            else{ // Once over 30s, patient has high heart rate
                if(isChanging){
                    qDebug()<< "Alert: High Breath "; }
                ui->pushBtn->setText(" Alert: High Breath  "); }}
        else if(Breath <= 100 && Breath >= 60){ // Normal HR
            errorStart = true;
            timerCount = 0; // Resets error (30s) timer
            ui->pushBtn->setText(" Normal ");}
        else if(Breath < 60){ // Low HR
            if(errorStart){ // Starts 30s timer
                tmOld = NowString("ss");
                timeOld = tmOld.toInt();
                errorStart = false;}
            if(timerCount < 30){ // Under 30s (may be error)
                if(isChanging){
                    qDebug()<< "May be a Microwave Error"; }
          ui->pushBtn->setText(" May be a Microwave Error: " +
QString::number(timerCount) + "s");           } else { // Over 30s; patient has low heart rate
                if(isChanging){
```

```
                qDebug()<< "Alert: Low Breath ";}

            ui->pushBtn->setText(" Alert: Low Breath  ");}}}

  old Breath  = Breath  ;

tmNew = tmNow;}
```