ANEXO A

En este anexo se presentan los documentos C++ utilizados:

QMLproyect.pro

```
QT += declarative
QT += sql

include(keyboard/keyboard.pri)
include(table/table.pri)
include(dial/dial.pri)
include(/home/luisenrique/qextserialport-1.2beta2-
standard/src/qextserialport.pri)


SOURCES += \
    main.cpp \
    icon4.cpp \
    icon1.cpp \
    mainclass.cpp \
    timerweigth.cpp \
    timerspeed.cpp \
    icon6.cpp \
    date.cpp \
    timerclock.cpp \
    sdcard.cpp \
    icon5.cpp

OTHER_FILES += \
    ../QMLproyect_build/qml/welcome.qml \
    ../QMLproyect_build/qml/keyboard/KeyboardBinding.qml \
    ../QMLproyect_build/qml/keyboard/Keyboard.qml \
    ../QMLproyect_build/qml/keyboard/Display.qml \
    ../QMLproyect_build/qml/keyboard/ButtonKeyboard.qml \
    ../QMLproyect_build/qml/keyboard/app.qml \
    ../QMLproyect_build/qml/table/TriangleButton.qml \
    ../QMLproyect_build/qml/table/TableDelegate.qml \
    ../QMLproyect_build/qml/table/TableComplete.qml \
    ../QMLproyect_build/qml/table/TableBody.qml \
    ../QMLproyect_build/qml/table/app.qml \
    ../QMLproyect_build/qml/icons/icon4.qml \
    ../QMLproyect_build/qml/ToolBar.qml \
    ../QMLproyect_build/qml/Button.qml \
    ../QMLproyect_build/qml/icons/IconBinding.qml \
    ../QMLproyect_build/qml/icons/icon1.qml \
    ../QMLproyect_build/qml/GridIcons.qml \
    ../QMLproyect_build/qml/icons/icon2.qml \
    ../QMLproyect_build/qml/icons/icon3.qml \
    ../QMLproyect_build/qml/icons/icon5.qml \
    ../QMLproyect_build/qml/icons/icon6.qml \
    ../QMLproyect_build/qml/tools/Display.qml \
    ../QMLproyect_build/qml/dial/Dial.qml \
    ../QMLproyect_build/qml/dial/Display.qml \
    ../QMLproyect_build/qml/dial/Dial2.qml \
```

```
    ../QMLproyect_build/qml/dial/Dial2Complete.qml \
    ../QMLproyect_build/qml/tools/Button.qml \
    ../QMLproyect_build/qml/tools/TriangleButton.qml \
    ../QMLproyect_build/qml/speed/Speed.qml

HEADERS += \
    icon4.h \
    icon1.h \
    mainclass.h \
    timerweigth.h \
    timerspeed.h \
    icon6.h \
    date.h \
    timerclock.h \
    sdcard.h \
    icon5.h
```

Dial.pri

```
SOURCES += \
    $$PWD/dial.cpp

HEADERS += \
    $$PWD/dial.h

INCLUDEPATH += \
    $$PWD/
```

Dial.cpp

```
#include "dial.h"
#include "qmath.h"
#include <QDebug>

dial::dial( QDeclarativeView *view,const QString pathQmlFile  )
{
    setColor("#FF5050");
    view->rootContext()->setContextProperty("DialBinding",this);
    view->setSource(QUrl::fromLocalFile(pathQmlFile));
    connect(this,SIGNAL(ChangeStateColor()),this, SLOT(StateColor()) );
    setCurrentAngle(0);

    SetNormString("1000");
    SetUpperString("25");
    SetLowerString("25");
}


void dial::StateColor()
```

```
{
    if( currentAngle() <= 90+downAngle()  )
    {setColor("#FF5050");}
    else if(currentAngle() < 90+upAngle()  )
    {setColor("#009900");}
    else    {setColor("#FF5050");}
}
int dial::QStringToInt(const QString m, int size)
{
    int num;
    if(m.contains("."))
    {
    QStringList list=m.split(".");

    num=list.at(0).toInt()*qPow(10,size)+
list.at(1).toInt()*qPow(10,size-list.at(1).size());
    }
    else num= m.toInt()*qPow(10,size);

    return num;
}

void dial::getAngles()
{
    if(m_upperInt>m_lowerInt)
    {m_upAngle=25;
          m_downAngle=-((m_lowerInt)*25 /m_upperInt);
    }
    else {m_downAngle=-25;

        m_upAngle=((m_upperInt)*25 /m_lowerInt);
    }

    qDebug()<<m_upAngle;
    qDebug()<<m_downAngle;

}

void dial::setCurrentValue(const QString m)
{
    SetValueString(m);
    int value=QStringToInt(m,3);
    int angle;
    angle=((value-( m_normInt-m_lowerInt ))*-m_downAngle )/(m_lowerInt )
+ 90+ m_downAngle ;

    if(angle<0)
      {angle=0;}
    else if(angle>180)
    {angle=180;}
    setCurrentAngle(angle);
    StateColor();

}
```

Dial.h

```
#ifndef DIAL_H
#define DIAL_H

#include <QObject>
#include <QDeclarativeItem>
#include <QDeclarativeView>
#include <QDeclarativeContext>

class dial : public QObject
{
    Q_OBJECT
    Q_PROPERTY(QColor color READ color WRITE setColor NOTIFY
colorChanged)
    Q_PROPERTY(int currentAngle READ currentAngle WRITE setCurrentAngle
NOTIFY curentAngleChanged)
    Q_PROPERTY(int upAngle READ upAngle WRITE setUpAngle NOTIFY
upAngleChanged)
    Q_PROPERTY(int downAngle READ downAngle WRITE setDownAngle NOTIFY
downAngleChanged)
    Q_PROPERTY(QString normString READ normString WRITE SetNormString
NOTIFY normStringChanged)
    Q_PROPERTY(QString upperString READ upperString WRITE SetUpperString
NOTIFY upperStringChanged)
    Q_PROPERTY(QString normString READ normString WRITE SetNormString
NOTIFY normStringChanged)
    Q_PROPERTY(QString lowerString READ lowerString WRITE SetLowerString
NOTIFY lowerStringChanged)
    Q_PROPERTY(QString valueString READ valueString WRITE SetValueString
NOTIFY valueStringChanged)


public:
    dial(QDeclarativeView *view =0 , const  QString pathQmlFile=0 );

    QColor color() const  {return m_color;}
    void setColor(const QColor color) { m_color= color; }
    int currentAngle() const  {return m_currentAngle;}
    void setCurrentAngle(const int currentAngle) { m_currentAngle=
currentAngle;  }
    int upAngle() const  {return m_upAngle;}
    void setUpAngle(const int upAngle) { m_upAngle= upAngle; }
    int downAngle() const  {return m_downAngle;}
    void setDownAngle(const int downAngle) { m_downAngle= downAngle; }

    QString normString() {return m_normString;}
    void SetNormString(const QString normString) {m_normString=
normString; m_normInt= QStringToInt(normString,3);}
    QString upperString() {return
QString::number(((double)(m_normInt+m_upperInt)/1000) );}
    void SetUpperString(const QString upperString) {m_upperString=
upperString;m_upperInt=QStringToInt(upperString,3);getAngles(); }
    QString lowerString() {return QString::number(((double)(m_normInt-
```

```
m_lowerInt)/1000) );}
    void SetLowerString(const QString lowerString) {m_lowerString=
lowerString;m_lowerInt=QStringToInt(lowerString,3);getAngles();}
    QString valueString() {return m_valueString;}
    void SetValueString(const QString valueString) {m_valueString=
valueString;}


    int QStringToInt(const QString st, int size);
    void setCurrentValue(const QString);
signals:
    void colorChanged();
    void curentAngleChanged();
    void upAngleChanged();
    void downAngleChanged();
    void ChangeStateColor();
    void normStringChanged();
    void upperStringChanged();
    void lowerStringChanged();
    void valueStringChanged();

public slots:
    void StateColor();
    void getAngles();

private:
    QColor m_color;
    int m_currentAngle;
    int m_upAngle;
    int m_downAngle;
    QString m_normString;
    QString m_lowerString;
    QString m_upperString;
    QString m_valueString;
    int m_normInt;
    int m_upperInt;
    int m_lowerInt;


};

#endif // DIAL_H
```

Keyboard.pri

```
HEADERS += \
    $$PWD/KeyboardClass.h
SOURCES += \
    $$PWD/KeyboardClass.cpp

INCLUDEPATH += \
    $$PWD/
```

KeyboardClass.cpp

```
#include "KeyboardClass.h"


KeyboardClass::KeyboardClass( QDeclarativeView *view,const QString
pathQmlFile  )
{
    view->rootContext()->setContextProperty("keyboardBinding",this);
    view->setSource(QUrl::fromLocalFile(pathQmlFile));
    m_edit=false;
    m_view =view;
    setNum(false);
}


void KeyboardClass::changeText(const QString t)
{
    if(t.contains("/u1"))
    {
        setText(text().left(text().size()-1) );
    }
    else if(t.contains("/u2"))
    {
        emit enterPressed();
        emit enterPressed(text());
        setText("");
    }
    else if(t.contains("/u3"))
    {
        emit cancelPressed();
        setText("");
    }

    else
    {setText(text()+t);  }

    emit textChanged(text());
}

void KeyboardClass::setNum(bool c)
{

    if(c)
    {m_view->rootContext()->setContextProperty("numberKeyboard","true");
        m_view->rootContext()-
>setContextProperty("buttonchangenum","false");}
    else
    {m_view->rootContext()->setContextProperty("numberKeyboard","false");
        m_view->rootContext()-
>setContextProperty("buttonchangenum","true");}

}
```

KeyboardClass.h

```
#ifndef KEYBOARDCLASS_H
#define KEYBOARDCLASS_H

#include <QDeclarativeItem>
#include <QTextEdit>
#include <QDeclarativeView>
#include <QDeclarativeContext>

class KeyboardClass : public QTextEdit
{
    Q_OBJECT
    Q_PROPERTY(QString text READ text WRITE setText NOTIFY textChanged)
    Q_PROPERTY(int index READ index WRITE setIndex )
    Q_PROPERTY(bool isEdited READ isEdited WRITE setEdit )

public:

    KeyboardClass(QDeclarativeView *view =0 , const  QString
pathQmlFile=0 );

    QString text() const  {return m_text;}
    void setText(const QString text) { m_text= text; }
    Q_INVOKABLE void changeText( const QString text);

    int index() const {return m_index;}
    Q_INVOKABLE  void setIndex(const int index) {m_index= index;}

    bool isEdited(){return m_edit;}
    Q_INVOKABLE  void setEdit(const bool isEdited) {m_edit= isEdited;}

    Q_INVOKABLE void setNum(bool );

signals:
    void textChanged(const QString );
    void enterPressed();
    void enterPressed(QString text);
    void cancelPressed();

private:
    QString m_text;
    QDeclarativeView *m_view;
    int m_index;
    bool m_edit;
};

#endif
```

Sqlquerymodel.pri

```
SOURCES += \
    $$PWD/sqlquerymodel.cpp \

HEADERS += \
    $$PWD/sqlquerymodel.h \

INCLUDEPATH += \
    $$PWD/
```

Sqlquerymodel.cpp

```
#include "sqlquerymodel.h"

SqlQueryModel::SqlQueryModel(QObject *parent) :
    QSqlQueryModel(parent)
{
}


void SqlQueryModel::setQuery(const QString &query, const QSqlDatabase
&db)
{
    QSqlQueryModel::setQuery(query,db);
    generateRoleNames();
}

void SqlQueryModel::setQuery(const QSqlQuery & query)
{
    QSqlQueryModel::setQuery(query);
    generateRoleNames();
}

void SqlQueryModel::generateRoleNames()
{
    QHash<int, QByteArray> roleNames;
    for( int i = 0; i < record().count(); i++) {
        roleNames[Qt::UserRole + i + 1] =
record().fieldName(i).toAscii();
    }
    setRoleNames(roleNames);
}

QVariant SqlQueryModel::data(const QModelIndex &index, int role) const
{
    QVariant value = QSqlQueryModel::data(index, role);
    if(role < Qt::UserRole)
    {
        value = QSqlQueryModel::data(index, role);
```

```
    }
    else
    {
        int columnIdx = role - Qt::UserRole - 1;
        QModelIndex modelIndex = this->index(index.row(), columnIdx);
        value = QSqlQueryModel::data(modelIndex, Qt::DisplayRole);
    }
    return value;
}
```

Sqlquerymodel.h

```
#ifndef SQLQUERYMODEL_H
#define SQLQUERYMODEL_H

#include <QObject>
#include <QSqlQueryModel>
#include <QSqlRecord>

class SqlQueryModel : public QSqlQueryModel
{
    Q_OBJECT

    void generateRoleNames();
public:
    explicit SqlQueryModel(QObject *parent = 0);
    void setQuery(const QString &query, const QSqlDatabase &db =
QSqlDatabase());
     void setQuery(const QSqlQuery &query);
     QVariant data(const QModelIndex &index, int role) const;


signals:

public slots:

};

#endif // SQLQUERYMODEL_H
```

Sqlite.pri

```
SOURCES += \
    $$PWD/customsqlmodel.cpp \
    $$PWD/sqliteport.cpp

HEADERS += \
    $$PWD/customsqlmodel.h \
    $$PWD/sqliteport.h
```

```
INCLUDEPATH += \
    $$PWD/
```

Customsqlmodel.cpp

```
#include <customsqlmodel.h>
#include <QColor>

customsqlmodel::customsqlmodel(QObject *parent)
: QSqlQueryModel(parent)
{


}


QVariant customsqlmodel::data(const QModelIndex &index, int role) const
{
    QVariant value = QSqlQueryModel::data(index, role);
//    if (value.isValid() && role == Qt::DisplayRole) {
//        if (index.column() == 0)
//            return value.toString().prepend("#");
//        else if (index.column() == 2)
//            return value.toString().toUpper();

//    }
//    if (role == Qt::TextColorRole && index.column() == 1)
//        return qVariantFromValue(QColor(Qt::blue));
    return value;
}
```

Customsqlmodel.h

```
#ifndef customsqlmodel_H
#define customsqlmodel_H

#include <QSqlQueryModel>

class customsqlmodel : public QSqlQueryModel
{
    Q_OBJECT

public:
    customsqlmodel(QObject *parent = 0);
    QVariant data(const QModelIndex &item, int role) const;


};

#endif // CUSTOMSQLMODEL_H
```

Sqliteport.cpp

```
#include "sqliteport.h"

sqlite::sqlite(QObject *parent) :
    QObject(parent)
{
}

sqlite::sqlite(const QString& pathDatabase)
{
    connection(pathDatabase);
}

void sqlite::initializeModel(QSqlQueryModel *model, const QString& table)
{
    QString st = QString("SELECT * FROM %1").arg(table);
    model->setQuery(st);
}


QTableView* sqlite::createView(QSqlQueryModel *model, const QString
&title = "")
{
    view = new QTableView;
    view->setModel(model);

    static int offset = 0;
    view->setWindowTitle(title);
    view->resize(model->columnCount()*100,400);
    //view->setMaximumHeight(400);
    //view->setMaximumWidth(400);
    view->move(100 + offset, 100 + offset);
    offset += 20;
    return view;
}

void sqlite::showCurrentTableView()
{
    view->show();
}

void sqlite::setTable(const QString& table)
{

    customModel = new customsqlmodel();
    initializeModel(customModel,table);
    createView(customModel, table);

}


QStringList sqlite::showTables()
{

    return db.tables();
}
```

```
bool sqlite::connection(const QString& pathDatabase)
{
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(pathDatabase);
    if(db.open())
    {
       return true;
    }
    else return false;

}

bool sqlite::   isOpen()
{
    return db.isOpen();
}
```

Sqliteport.h

```
#ifndef SQLITE_H
#define SQLITE_H

#include <QObject>
#include <QSqlDatabase>
#include <QSqlQueryModel>
#include <QTableView>
#include <customsqlmodel.h>
#include <QSqlQuery>
#include "QDebug"


class sqlite : public QObject
{
    Q_OBJECT
public:
    explicit sqlite(QObject *parent = 0);
    explicit sqlite(const QString& pathDatabase);


    QSqlDatabase dbsqliteDatabase() {return db;}
    void initializeModel(QSqlQueryModel *model, const QString& table);
    QTableView* createView(QSqlQueryModel *, const QString &);
    void showCurrentTableView();
    void setTable(const QString& table);
    QStringList showTables();
    bool connection(const QString& pathDatabase);
    bool isOpen();


signals:

public slots:
```

```
private:
    QSqlDatabase db;
    QTableView *view;
    QSqlQueryModel *customModel;
    QSqlQuery *query;


};

#endif // SQLITE_H
```

Table.pri

```
include(../sqlite/sqlite.pri)
include(../queryModelList/sqlquerymodel.pri)


SOURCES += \
    $$PWD/tableclass.cpp

HEADERS += \
    $$PWD/tableclass.h

INCLUDEPATH += \
    $$PWD/
```

Tableclass.cpp

```
#include "tableclass.h"

TableClass::TableClass(QDeclarativeView *view  ,const QString
pathQmlFile, const QSqlDatabase db, const QString tablename)
{
    m_tablename= tablename;
    setQuery( QString("SELECT * FROM %1").arg(tablename), db);
    view->rootContext()->setContextProperty("TableBinding",this);
    view->setSource(QUrl::fromLocalFile(pathQmlFile));
    setMaxRows(rowCount());
    m_pathQML=pathQmlFile;
    m_db=db;
    m_view =view;
    setReadytoAdd(false);
}
```

```
void TableClass::Update()
{
    setQuery( QString("SELECT * FROM %1").arg(m_tablename), m_db);
    m_view->rootContext()->setContextProperty("TableBinding",this);
    setMaxRows(rowCount());
}

void TableClass::deleteRowTable()
{
    QString st=QString("delete from %1  where %2= '%3';").arg(tableName()
, roleNames()[Qt::UserRole+1],key() );
    setQuery(st);
    Update();
}

void TableClass::setReadytoAdd(bool c)
{
    if(c)
    {m_view->rootContext()-
>setContextProperty("buttonAddLabel","Agregar");}
    else
    {m_view->rootContext()->setContextProperty("buttonAddLabel","Agrega
Nuevo");}
    m_readyForAdd=c;
}


void TableClass::addRowTable()
{
    if(m_readyForAdd)
    {emit readyAddRow(); }

}
```

Tableclass.h

```
#ifndef TABLECLASS_H
#define TABLECLASS_H

#include <QObject>
#include <sqlquerymodel.h>
#include <QDeclarativeItem>
#include <QDeclarativeView>
#include <QDeclarativeContext>

#include <QDebug>

class TableClass : public SqlQueryModel
{
    Q_OBJECT
    Q_PROPERTY(QString key READ key  WRITE setKey NOTIFY keySelected)
    Q_PROPERTY(QString name READ name  WRITE setName NOTIFY nameSelected)
    Q_PROPERTY(QString norm READ norm  WRITE setNorm NOTIFY normSelected
)
```

```
    Q_PROPERTY(QString upper READ upper  WRITE setUpper NOTIFY
upperSelected)
    Q_PROPERTY(QString lower READ lower  WRITE setLower NOTIFY
lowerSelected)
    Q_PROPERTY(int rowIndex READ rowIndex WRITE setRowIndex NOTIFY
rowChanged)
    Q_PROPERTY(int maxRows READ maxRows NOTIFY maxRowChanged)


public:
    explicit TableClass(QDeclarativeView *view =0 ,const QString
pathQmlFile=0, const QSqlDatabase db = QSqlDatabase()  , const QString
tablename=0);

    QString key() const {return m_key;}
    Q_INVOKABLE  void setKey(const QString key) {m_key= key;emit
nameSelected(m_key); }
    QString name() const {return m_name;}
    Q_INVOKABLE  void setName(const QString name) {m_name= name;emit
nameSelected(m_name); }
    QString norm() const {return m_norm;}
    Q_INVOKABLE  void setNorm(const QString norm) {m_norm= norm;emit
normSelected(m_norm); }
    QString upper() const {return m_upper;}
    Q_INVOKABLE  void setUpper(const QString upper) {m_upper= upper;emit
upperSelected(m_upper);}
    QString lower() const {return m_lower;}
    Q_INVOKABLE  void setLower(const QString lower) {m_lower= lower;emit
lowerSelected(m_lower);}
    int rowIndex() const {return m_rowIndex;}
    Q_INVOKABLE  void setRowIndex(const int rowIndex) {m_rowIndex=
rowIndex; emit rowChanged(m_rowIndex);}
    int  maxRows() {return m_maxRows;}
    void setMaxRows(const int rows) {m_maxRows=rows ; emit
maxRowChanged(m_maxRows);}
    QString  tableName() {return m_tablename;}
    void setTableName(const QString table) {m_tablename=table; }
    Q_INVOKABLE void deleteRowTable();
    Q_INVOKABLE void addRowTable();


    void Update();


signals:

      void rowChanged(const int );
      void maxRowChanged(const int);
      void keySelected(const QString);
      void nameSelected(const QString);
      void normSelected(const QString);
      void upperSelected(const QString);
      void lowerSelected(const QString);
      void readyAddRow();


public slots:
```

```
     Q_INVOKABLE void setReadytoAdd(bool );

private:

    QString m_key;
    QString m_name;
    QString m_norm;
    QString m_upper;
    QString m_lower;
    QString m_tablename;
    QString m_pathQML;
    QSqlDatabase m_db;
    QDeclarativeView *m_view;
    int m_rowIndex;
    int m_maxRows;
    bool m_readyForAdd;

};

#endif // TABLECLASS_H
```

Date.cpp

```
#include "date.h"
#include "qdebug.h"


date::date(QObject *parent) :
    QObject(parent)
{
}

void date::getdate()
{
        QProcess *process = new QProcess(this);        QString st;
        st="date +%D";
        process->start(st);
        process->waitForFinished(100);
        QString dat= process->readAll();
        setMonth(dat.left(dat.indexOf('/')).toInt());
        dat=dat.mid(dat.indexOf('/')+1);
        setDay(dat.left(dat.indexOf('/')).toInt());
        dat=dat.mid(dat.indexOf('/')+1);
        setYear(dat.left(dat.indexOf('/')).toInt());

}

void date::getTime()
{
        QProcess *process = new QProcess(this);        QString st;
        st="date +%T";
        process->start(st);
        process->waitForFinished(100);
        QString dat= process->readAll();
```

```
        setHour(dat.left(dat.indexOf(':')).toInt());
        dat=dat.mid(dat.indexOf(':')+1);
        setMin(dat.left(dat.indexOf(':')).toInt());
        dat=dat.mid(dat.indexOf(':')+1);
        setSec(dat.left(dat.indexOf(':')).toInt());
}


void date::setDate()
{
        QProcess *process = new QProcess(this);         QString st;
        st="date ";
        st=st+QString::number(month()).rightJustified(2,'0')+
                QString::number(day()).rightJustified(2,'0')+
                QString::number(hour()).rightJustified(2,'0')+
                QString::number(min()).rightJustified(2,'0')+
                QString::number(year()+2000)+
                "."+
                QString::number(sec()).rightJustified(2,'0')
                ;
        process->start(st);
        process->waitForFinished(100);
        qDebug()<<"new set"<<st;
}
```

Date.h

```
#ifndef DATE_H
#define DATE_H

#include <QObject>
#include <QProcess>
class date : public QObject
{
    Q_OBJECT
public:
    explicit date(QObject *parent = 0);
    void setDay( int day) { m_day=day;}
    void setMonth(int month) {m_month= month;}
    void setYear(int year){m_year= year;}
    void setHour(int hour) { m_hour = hour;}
    void setMin(int min) { m_min = min;}
    void setSec(int sec) { m_sec = sec;}

    int day(){return m_day;}
    int month() {return m_month;}
    int year() {return m_year;}
    int hour(){ return m_hour;}
    int min(){return m_min;}
    int sec() {return m_sec;}

    void getdate();
    void getTime();
```

```
    void setDate();
signals:

public slots:


private:
    int m_day;
    int m_month;
    int m_year;
    int m_hour;
    int m_min;
    int m_sec;

};

#endif // DATE_H
```

Icon1.cpp

```
#include "icon1.h"

icon1::icon1(QDeclarativeView *view  ,const QString &pathQmlFile)
{
   // m_dial = new dial(view,pathQmlFile );
    m_view=view;
    m_view->setSource(QUrl::fromLocalFile(pathQmlFile));
    m_view->rootContext()->setContextProperty("Icon1Binding",this);
    CountReject=0;
    CountPass=0;
    timer = new QTimer(this);
    timer->setInterval(100);
    //itemQML = m_view->rootObject();

    QObject *item = view->rootObject();

 QDeclarativeContext *context = m_view->rootContext();
 context->setContextProperty("rejectdisplayicon1",CountReject);
 context->setContextProperty("passdiplayicon1",CountPass);
    context->setContextProperty("NumberDisplay","0");
    context->setContextProperty("valueWeigthQML","name" );
    context->setContextProperty("normQML","norm" );
    context->setContextProperty("upperQML","upper" );
    context->setContextProperty("lowerQML","lower" );

  //  connect(item,SIGNAL(startTimerIcon1()),this, SLOT(timerShot())  );
   // connect(timer,SIGNAL(timeout()),this,SLOT(timerShot()));
}

void icon1::timerShot(const QString m)
{


}
```

Icon1.h

```
#ifndef ICON1_H
#define ICON1_H

#include <QObject>
#include <dial.h>
#include <QTimer>
#include <QDeclarativeItem>
#include <QDeclarativeView>
#include <QDeclarativeContext>

class icon1 : public QObject
{
    Q_OBJECT
public:
    explicit icon1(QDeclarativeView *view =0 ,const QString
&pathQmlFile=0);
signals:

   // void stateChanged(const QString &newState);
//public slots:
//    void SetNormString(const QString norm) { m_dial-
>SetNormString(norm); }
//    void SetLowerString(const QString lower) { m_dial-
>SetLowerString(lower); }
//    void SetUpperString(const QString upper) { m_dial-
>SetUpperString(upper); ;qDebug()<<"change angles";}
//    void getAngles() {    m_dial->getAngles();}
//    void setCurrentValue(const QString value) {    m_dial-
>setCurrentValue(value);}

//private:
//    dial *m_dial;



public slots:
    void SetNormString(const QString norm)
    {    QDeclarativeContext *context = m_view->rootContext();
         context->setContextProperty("normQML",norm);
    }
    void SetUpperString(const QString upper)
    {    QDeclarativeContext *context = m_view->rootContext();
         context->setContextProperty("upperQML",upper);
    }
    void SetLowerString(const QString lower)
    {    QDeclarativeContext *context = m_view->rootContext();
         context->setContextProperty("lowerQML",lower);
    }
   Q_INVOKABLE void SetIncreaseCountPass()
    {    QDeclarativeContext *context = m_view->rootContext();
```

```
          CountPass=CountPass+1  ;context-
>setContextProperty("passdiplayicon1",CountPass);
    }
   Q_INVOKABLE void     SetIncreaseCountReject()
    {    QDeclarativeContext *context = m_view->rootContext();
          CountReject=CountReject+1;context-
>setContextProperty("rejectdisplayicon1",CountReject); ;
    }
    void SetValue(const QString value)
    {    QDeclarativeContext *context = m_view->rootContext();
         context->setContextProperty("valueWeigthQML",value);
    }


    void timerShot(const QString m);

private:
    QDeclarativeView *m_view;
    int CountReject;
    int CountPass;
    QTimer *timer;
    QObject *itemQML;
};

#endif // ICON_H
```

Icon4.cpp

```
#include "icon4.h"
#include "QDebug"

icon4::icon4(QDeclarativeView *view  ,const QString &pathQmlFile, const
QSqlDatabase &db , const QString &tablename)
{

   m_table=  new TableClass (view, pathQmlFile, db, tablename );
   m_keyboard = new KeyboardClass(view,pathQmlFile );
   m_table->setRowIndex(0);
   connect(m_keyboard,SIGNAL(enterPressed(QString)), this,SLOT(
setTable(QString) ));

connect(this,SIGNAL(readytoAddNewRow(bool)),m_table,SLOT(setReadytoAdd(bo
ol)) );
   connect(m_table,SIGNAL(readyAddRow()),this,SLOT(addRow()) );


   connect(m_table, SIGNAL(nameSelected(const QString)),this ,
SIGNAL(nameSelected(const QString)) );
   connect(m_table, SIGNAL(normSelected(const QString)),this ,
SIGNAL(normSelected(const QString)) );
   connect(m_table, SIGNAL(upperSelected(const QString)),this ,
SIGNAL(upperSelected(const QString)) );
   connect(m_table, SIGNAL(lowerSelected(const QString)),this ,
SIGNAL(lowerSelected(const QString)) );
```

```
}

void icon4::setTable(QString newValue)
{
    qDebug()<<"hello";
    if(m_keyboard->isEdited())
    {
        if(m_keyboard->index()<=4)//edit existed rows
    {QString roleName;
    roleName= m_table->roleNames()[Qt::UserRole+m_keyboard->index()+1];
    QString keyrole;
    keyrole= m_table->roleNames()[Qt::UserRole+1];
    QString st=QString("UPDATE  %1 SET %2='%3' WHERE %4 =
'%5'").arg(m_table->tableName() ,roleName,newValue, keyrole,m_table-
>key() );
    m_table->setQuery(st);
    m_table->Update();

    if( roleName.contains("name"))
    {m_table->setName(newValue);}
    else if( roleName.contains("norm"))
    {m_table->setNorm(newValue);}
    else if( roleName.contains("upper"))
    {m_table->setUpper(newValue);}
    else if( roleName.contains("lower"))
    {m_table->setLower(newValue);}
        }

        else if(m_keyboard->index()<=8)//add new row
    {QString roleName=m_table->roleNames()[Qt::UserRole+m_keyboard-
>index()+1-4];
    if( roleName.contains("name"))
    {m_table->setName(newValue);}
    else if( roleName.contains("norm"))
    {m_table->setNorm(newValue);}
    else if( roleName.contains("upper"))
    {m_table->setUpper(newValue);}
    else if( roleName.contains("lower"))
    {m_table->setLower(newValue);}
    m_StringForNewRow=QString("INSERT INTO %1 (name, norm, upper, lower)
").arg(m_table->tableName() );

m_StringForNewRow.append(QString("VALUES('%1','%2','%3','%4')").arg(m_tab
le->name(),m_table->norm(),m_table->upper(),m_table->lower()  ));
    if((m_table->name().size()>0)&&(m_table->norm().size()>0)&&(m_table-
>upper().size()>0)&&(m_table->lower().size()>0) )
    {
        emit readytoAddNewRow(true);
        }

    }
        if(m_keyboard->index()>=9)

        {setDateClass.getdate();
        setDateClass.getTime();
```

```
        switch (m_keyboard->index())
        {

        case 9: {setDateClass.setDay(newValue.toInt());
setDateClass.setDate(); };break;
        case 10: {setDateClass.setMonth(newValue.toInt());
setDateClass.setDate(); };break;
        case 11: {setDateClass.setYear(newValue.toInt());
setDateClass.setDate();};break;
        case 12: {setDateClass.setHour(newValue.toInt());
setDateClass.setDate();};break;
        case 13: {setDateClass.setMin(newValue.toInt());
setDateClass.setDate();};break;
        case 14: {setDateClass.setSec(newValue.toInt());
setDateClass.setDate();};break;

        }
        }




    }
}

void icon4::addRow()
{
    m_table->setQuery(m_StringForNewRow);
    m_table->Update();
    m_table->setReadytoAdd(false);
    m_table->setRowIndex(m_table->maxRows()-1);
    m_StringForNewRow.clear();
}
```

Icon4.h

```
#ifndef ICON4_H
#define ICON4_H

#include <QObject>
#include <QDeclarativeItem>
#include <QDeclarativeView>
#include <QDeclarativeContext>
#include <QSqlDatabase>
#include <KeyboardClass.h>
#include <sqliteport.h>
#include <tableclass.h>
#include <date.h>

class icon4 : public QObject
{
    Q_OBJECT
```

```
public:
    explicit icon4(QDeclarativeView *view =0 ,const QString
&pathQmlFile=0, const QSqlDatabase &db = QSqlDatabase() , const QString
&tablename=0);

    QString key() {return m_table->key();}
    QString name() {return m_table->name();}
    QString norm() {return m_table->norm();}
    QString upper() {return m_table->upper();}
    QString lower() {return m_table->lower();}


signals:
    void readytoAddNewRow(bool);
    void nameSelected(const QString);
    void normSelected(const QString);
    void upperSelected(const QString);
    void lowerSelected(const QString);


public slots:
    void setTable(QString);
    void addRow();
private:

   KeyboardClass *m_keyboard;
   TableClass *m_table;
   QString m_StringForNewRow;
   date setDateClass;

};

#endif // ICON4_H
```

Icon5.cpp

```
#include "icon5.h"



icon5::icon5(QDeclarativeView *view  ,const QString &pathQmlFile,const
QSqlDatabase &db, const QString &tablename)

{
    m_view=view;
    m_view->setSource(QUrl::fromLocalFile(pathQmlFile));
    sdcardport=  new sdcard  (db);
    m_view->rootContext()->setContextProperty("SDcardBinding",this);
    sdcardport->createSimpleTable(tablename);
    dataSDcard = new QList< QByteArray>;
```

```
}

void icon5::insertRangeOfValuesToDatabase(QByteArray value, const int
num)
{
    dataSDcard->append(value);
    if(dataSDcard->size()>=num)
    {
        sdcardport->inserttoSimpleTable(*dataSDcard);
        dataSDcard->clear();
    }
}

void icon5::insertRangeOfValuesToDatabase(QByteArray value)
{
    dataSDcard->append(value);
    if(dataSDcard->size()>=5)
    {
        sdcardport->inserttoSimpleTable(*dataSDcard);
        dataSDcard->clear();
    }
}
```

Icon5.h

```
#ifndef ICON5_H
#define ICON5_H
//#define pathDatabaseSDcard "/home/luisenrique/luis"
//#define pathSDcard "/media/SD"

#define pathDatabaseSDcard "/usr/local/scripts"
#define pathSDcard "/sddisk"

#include <QObject>
#include <QDeclarativeItem>
#include <QDeclarativeView>
#include <QDeclarativeContext>
#include <sdcard.h>
#include <QDir>

class icon5 : public QObject
{
    Q_OBJECT
public:
    explicit icon5(QDeclarativeView *view =0 ,const QString
&pathQmlFile=0,const QSqlDatabase &db = QSqlDatabase() , const QString
&tablename=0);

    Q_INVOKABLE  void createSdCardFile() {


                                            if(dataSDcard->size()>0)
                                            {
```

```
                                                    sdcardport-
>inserttoSimpleTable(*dataSDcard);
                                                     dataSDcard->clear();
                                                }

                                                QDir dir;
                                                QString st = pathSDcard;
                                                st.append("/files");
                                                if(!dir.exists(st))
                                                {
                                                if(dir.mkdir(st))
                                                {sdcardport-
>createCsv(pathDatabaseSDcard);}
                                                }
                                                else sdcardport-
>createCsv(pathDatabaseSDcard);
                                                sdcardport-
>disconnectSDcard(pathSDcard);
                                               }



signals:

public slots:
    void insertRangeOfValuesToDatabase(QByteArray value, const int num);
    void insertRangeOfValuesToDatabase(QByteArray value);

private:
 //   date *dateconfig;
    QDeclarativeView *m_view;
    sdcard *sdcardport;
    QList< QByteArray> *dataSDcard;

};

#endif // ICON5_H
```

Icon6.cpp

```
#include "icon6.h"

icon6::icon6(QDeclarativeView *view  ,const QString &pathQmlFile)

{
    m_view=view;
    m_view->setSource(QUrl::fromLocalFile(pathQmlFile));

}
```

Icon6.h

```
#ifndef ICON6_H
#define ICON6_H

#include <QObject>
#include <date.h>
#include <QDeclarativeItem>
#include <QDeclarativeView>
#include <QDeclarativeContext>

class icon6 : public QObject
{
    Q_OBJECT
public:
    explicit icon6(QDeclarativeView *view =0 ,const QString
&pathQmlFile=0);

signals:

public slots:

private:
 //   date *dateconfig;
    QDeclarativeView *m_view;

};

#endif // ICON5_H
```

Main.cpp

```
#include <QtGui/QApplication>
#include <QCoreApplication>
#include <QDeclarativeEngine>

#include <QDeclarativeView>
#include <QGraphicsObject>
#include <mainclass.h>

int main(int argc, char *argv[])
{

        QApplication a(argc, argv);
        QDeclarativeView view;

        MainClass Mai(&view,"qml/welcome.qml");

        QObject::connect(view.engine(),SIGNAL(quit()),
QCoreApplication::instance(), SLOT(quit()));

        view.show();
        return a.exec();
}
```

Mainclass.cpp

```
#include "mainclass.h"



MainClass::MainClass(QDeclarativeView *view , const  QString pathQmlFile
)
{

    sqlite Sqlite("/usr/local/sql/test3.db") ;
    view->setWindowFlags(Qt::FramelessWindowHint);//make window border
desapear
    icon4 *Icon4 = new icon4(view, pathQmlFile,Sqlite.dbsqliteDatabase(),
"t1" );

    icon1 *Icon1 = new icon1(view,pathQmlFile);
    icon5 *Icon5 = new
icon5(view,pathQmlFile,Sqlite.dbsqliteDatabase(),"TableWeigth");

    timerWeigth *TimerW = new timerWeigth(view,pathQmlFile);
    timerSpeed *TimerS = new timerSpeed(view,pathQmlFile);
    timerClock *TimerC = new timerClock(view,pathQmlFile);



//    Icon1->SetLowerString(Icon4->lower());
//    Icon1->SetUpperString(Icon4->upper());
//    Icon1->SetNormString(Icon4->norm());


    connect(Icon4,SIGNAL(normSelected(const QString)), Icon1,
SLOT(SetNormString(QString)));
    connect(Icon4,SIGNAL(upperSelected(const QString)), Icon1,
SLOT(SetUpperString(QString)) );
    connect(Icon4,SIGNAL(lowerSelected(const QString)), Icon1,
SLOT(SetLowerString(QString)) );


connect(TimerW,SIGNAL(WeightDetected(QByteArray)),Icon5,SLOT(insertRangeO
fValuesToDatabase(QByteArray)) );

    view->rootContext()->setContextProperty("Display3icon4",Icon4-
>upper());


}
```

Mainclass.h

```
#ifndef MAINCLASS_H
#define MAINCLASS_H

#include <QObject>
#include <QDeclarativeView>
#include <QGraphicsObject>
#include <QDeclarativeProperty>
#include <QDeclarativeContext>

#include<icon6.h>
#include<icon5.h>
#include<icon4.h>
#include<icon1.h>
#include<timerweigth.h>
#include<timerspeed.h>
#include<timerclock.h>
#include<sdcard.h>

class MainClass : public QObject
{
    Q_OBJECT
public:
    MainClass(QDeclarativeView *view =0 , const  QString pathQmlFile=0 );

signals:

public slots:

};

#endif // MAINCLASS_H
```

Sdcard.cpp

```
#include "sdcard.h"
#include <QDebug>



sdcard::sdcard(const QSqlDatabase &d)
{

    db=d;
    if(db.isOpen())
    {         query = new QSqlQuery(); }

}

void sdcard::createSimpleTable(QString table)
{

    QString st;
```

```
    st.append( QString(" CREATE TABLE %1 (t1key integer primary key,
weigth text, time TIMESTAMP DEFAULT CURRENT_TIMESTAMP);").arg(table));

            query->exec(QString("DROP TABLE %1;").arg(table));
                query->exec(st);
            setCurrentTable(table);
}

void sdcard::inserttoSimpleTable(QByteArray value)
{
    QString values=value;
    QString st;
    st.append( QString("INSERT INTO %1 (weigth) VALUES( '%2'
);").arg(CurrentTable()). arg(values));
    query->exec(st);

}

void sdcard::inserttoSimpleTable(QList<QByteArray> values)
{
    QString st;
    st.append(QString("INSERT INTO %1 (weigth)  ").arg(CurrentTable()) );

    for(int i=0; i<values.size();i++ )
    {
        QString value=values.at(i);

        if(i==0)
        { st.append(QString( "SELECT '%1' AS weigth ").arg(value) );  }
        else st.append(QString( "UNION ALL SELECT '%1' ").arg(value) );
    }
    query->exec(st);
}


void sdcard::createCsv(const QString path)
{
    QProcess process;
    QString st;
    st.append("sh ");
    //process.start("bash /home/luisenrique/luis/script ");
    st.append(path);
    st.append("/script");
    process.start(st);
    process.waitForFinished();
}



void sdcard::disconnectSDcard(const QString path )
{
    QProcess process;
   // process.start("bash /home/luisenrique/luis/script ");
    QString st;
    st.append("umount ");
    st.append(path);
    process.start(st);
```

```
    process.waitForFinished();
}
```

Sdcard.h

```
#ifndef SDCARD_H
#define SDCARD_H

#include <QObject>
#include <QSqlDatabase>
#include <QSqlQueryModel>
#include <QSqlQuery>
#include <QProcess>

class sdcard : public QObject
{
    Q_OBJECT
public:
    explicit sdcard(const QSqlDatabase &d = QSqlDatabase());
    bool isOpen() {db.isOpen(); }
    void createSimpleTable( QString table);
    void setCurrentTable(QString name) { currenttablename=name;}
    QString CurrentTable() {return currenttablename;}
    void inserttoSimpleTable(QByteArray value);
    void inserttoSimpleTable(QList<QByteArray> values );
    void createCsv(const QString path);

    void disconnectSDcard(const QString path );

signals:

public slots:


private:
    QSqlDatabase db;
    QString currenttablename;
    QSqlQuery *query;

};
```

Timerclock.cpp

```
#include "timerclock.h"
#include "qextserialport.h"
#define PORTNAME "/dev/ttyUSB0"  //HMI


timerClock::timerClock( QDeclarativeView *view,const QString pathQmlFile
)

{
    timer= new QTimer(this);
    timer->setInterval(1000);
```

```
    getdate();
    getTime();
    m_view=view;
    view->rootContext()->setContextProperty("TimerClockBinding",this);
    view->setSource(QUrl::fromLocalFile(pathQmlFile));
    connect(timer,SIGNAL(timeout()),this,SLOT(doTimer()));
  //  m_keyboard = new KeyboardClass(view,"" );
//    connect(m_keyboard,SIGNAL(enterPressed(QString)), this,SLOT(
setClock(QString) ));



    QDeclarativeContext *context = m_view->rootContext();
    context->setContextProperty("dayLabel", "0" );
    context->setContextProperty("monthLabel", "0" );
    context->setContextProperty("yearLabel","0" );

    context->setContextProperty("hourLabel", "0" );
    context->setContextProperty("minLabel", "0" );
    context->setContextProperty("secLabel", "0" );

}

void timerClock::doTimer()
{
    QDeclarativeContext *context = m_view->rootContext();
    getdate();
    getTime();
    context->setContextProperty("dayLabel", day() );
    context->setContextProperty("monthLabel",month() );
    context->setContextProperty("yearLabel", year() );

    context->setContextProperty("hourLabel", hour() );
    context->setContextProperty("minLabel", min() );
    context->setContextProperty("secLabel",sec() );
}

void timerClock::setClock(QString newValue)
{

    if(m_keyboard->isEdited())
{
    switch (m_keyboard->index())
    {

    case 9: {setDay(newValue.toInt()); setDate(); };break;
    case 10: {setMonth(newValue.toInt()); setDate(); };break;
    case 11: {setYear(newValue.toInt()); setDate();};break;
    case 12: {setHour(newValue.toInt()); setDate();};break;
    case 13: {setMin(newValue.toInt()); setDate();};break;
    case 14: {setSec(newValue.toInt()); setDate();};break;

    }
}

}
```

Timerclock.h

```
#ifndef TIMERCLOCK_H
#define TIMERCLOCK_H

#include <QObject>
#include <QObject>
#include <QDeclarativeItem>
#include <QDeclarativeView>
#include <QDeclarativeContext>
#include <QDebug>
#include <QTimer>
#include <date.h>
#include <KeyboardClass.h>

class QextSerialPort;


class timerClock : public date
{
    Q_OBJECT

     Q_PROPERTY(bool Timer READ Timer WRITE setTimer NOTIFY ChangeTimer)

public:
    explicit timerClock(QDeclarativeView *view =0 , const  QString
pathQmlFile=0 );

    bool Timer()  {return timerState;}
    Q_INVOKABLE  void setTimer(bool Timer) {timerState=Timer;
                                            if(Timer)
                                            {timer->start();}
                                            else
                                            {timer->stop();}

                                           }

signals:
    void ChangeTimer();

public slots:
    void doTimer();
    void setClock(QString);

private:
QTimer *timer;
bool timerState;
QDeclarativeView *m_view;
KeyboardClass *m_keyboard;

};
```

```
#endif // TIMERWEIGTH_H
```

Timerspeed.cpp

```
#include "timerspeed.h"

//static int fd;

timerSpeed::timerSpeed( QDeclarativeView *view,const QString pathQmlFile
)

{
    timer= new QTimer(this);
    timer2=new QTimer(this);
    setT(999);


    system("/etc/rc.d/init.d/leds stop");


    fd = open("/dev/GPIO-Control", O_RDWR);
    if (fd < 0)
    {
    perror("open GPIO control");
    SLOT( close() );}

    for(int i=0; i<4;i++)
    {ioctl(fd, 1, i);}
    view->rootContext()->setContextProperty("TimerSpeedBinding",this);
    view->setSource(QUrl::fromLocalFile(pathQmlFile));
    connect(timer,SIGNAL(timeout()),this,SLOT(doTimer1()));
    connect(timer2,SIGNAL(timeout()),this,SLOT(doTimer2()));
    setF(5);
    setTimer(true);
}

void timerSpeed::doTimer1()
{

    ioctl(fd, 1, 0);
    timer->stop();
    timer2->start();
//    qDebug()<<"end timer 1";
//    qDebug()<<H();qDebug()<<L();
}

void timerSpeed::doTimer2()
{

    timer2->stop();
    timer->start();
    ioctl(fd, 0, 0);
//    qDebug()<<"end timer 2 ";
//    qDebug()<<H();qDebug()<<L();
```

```
}
```

Timerspeed.h

```
#ifndef TIMERSPEED_H
#define TIMERSPEED_H

#include <QObject>
#include <QObject>
#include <QDeclarativeItem>
#include <QDeclarativeView>
#include <QDeclarativeContext>
#include <QDebug>
#include <QTimer>
#include "sys/ioctl.h"
#include "sys/stat.h"
#include <fcntl.h>
#include <linux/kernel.h>

class timerSpeed : public QObject
{
    Q_OBJECT

     Q_PROPERTY(bool Timer READ Timer WRITE setTimer NOTIFY ChangeTimer)
     Q_PROPERTY(int H READ H WRITE setH NOTIFY Hchange)
    Q_PROPERTY(int F READ F WRITE setF NOTIFY Fchange)


public:
    explicit timerSpeed(QDeclarativeView *view =0 , const  QString
pathQmlFile=0 );

    bool Timer()  {return timerState;}
    Q_INVOKABLE  void setTimer(bool Timer) {timerState=Timer;
                                            if(Timer)
                                            {timer->start();}
                                            else
                                            {timer->stop();
                                                timer2->stop();}
                                           }
    int T() {return t;}
    Q_INVOKABLE void setT(const int pe) {t=pe;}
    Q_INVOKABLE int H() {return timer->interval();}
    Q_INVOKABLE void setH(const int H)  { if((H<=t)&&(H>=0)){th=H; tl=t-
th; timer->setInterval(th);timer2->setInterval(tl); emit Hchange();
ioctl(fd, 0, 1); }}
    Q_INVOKABLE  int L() {return timer2->interval();}
    Q_INVOKABLE void setL(const int lo)  { tl=lo; tl=t-tl; timer-
>setInterval(th);timer2->setInterval(tl); ioctl(fd, 0, 1); }



    Q_INVOKABLE  int F() {return f;}
```

```
    Q_INVOKABLE void setF(const int fr)  {  if(  (fr>=0) &&(fr<=50)  )
{f=fr;; emit  Fchange();} }


    Q_INVOKABLE void setOn() {    ioctl(fd, 0, 1);}
    Q_INVOKABLE void setOff() {    ioctl(fd, 1, 1);}

    Q_INVOKABLE void readySettings() { if(F()<=50) { setH(F()*999/50);}

                                qDebug()<<"H="<<H();

qDebug()<<"L="<<L();
                               }


signals:
    void ChangeTimer();
    void Hchange();
    void Fchange();

public slots:
    void doTimer1();
    void doTimer2();

private:
   int fd;
QTimer *timer;
QTimer *timer2;
bool timerState;
int t;
int th;
int tl;
int f;

};

#endif // TIMERWEIGTH_H
```

Timerweigth.cpp

```
#include "timerweigth.h"
#include "qextserialport.h"
#define PORTNAME "/dev/ttyUSB0"  //HMI


timerWeigth::timerWeigth( QDeclarativeView *view,const QString
pathQmlFile  )

{
    timer= new QTimer(this);
    timer->setInterval(1000);
    PortSettings settings = {BAUD9600, DATA_8,PAR_EVEN, STOP_1,
FLOW_XONXOFF, 10};
    port = new QextSerialPort(PORTNAME, settings,
```

```
QextSerialPort::EventDriven);
    m_view=view;
    view->rootContext()->setContextProperty("TimerWeightBinding",this);
    view->setSource(QUrl::fromLocalFile(pathQmlFile));
    connect(timer,SIGNAL(timeout()),this,SLOT(doTimer()));
    connect(port, SIGNAL(readyRead()), SLOT(getWeigth()));

    port->setPortName(PORTNAME);
    port->open(QIODevice::ReadWrite);

    if(port->isOpen())
    {qDebug()<<"port is open";}

}

void timerWeigth::doTimer()
{
    qDebug()<<"send";
    port->write("S31;MAV?;");
}

void timerWeigth::getWeigth()
{
    buffer.append(port->readAll());
    if(buffer.size()==8)
    {
        qDebug()<<"buffer="<<buffer;
        if(!buffer.startsWith("-"))
        {
            QDeclarativeContext *context = m_view->rootContext();
            context->setContextProperty("valueWeigthQML",buffer);
            emit WeightDetected(buffer);
        }


        buffer.clear();}


//    buffer.append(port->readAll());
//    if(buffer.size()==8)
//    {
//        if(buffer.startsWith("-"))
//        {
//            qDebug()<<buffer.mid(1);
//            QDeclarativeContext *context = m_view->rootContext();
//            context-
>setContextProperty("valueWeigthQML",buffer.mid(1));
//            emit WeightDetected(buffer);
//        }


//        buffer.clear();}
}

void timerWeigth::Calibrate()
{
    qDebug()<<"cal";
```

```
    port->write("TAS;");
}
```

Timerweigth.h

```
#ifndef TIMERWEIGTH_H
#define TIMERWEIGTH_H

#include <QObject>
#include <QObject>
#include <QDeclarativeItem>
#include <QDeclarativeView>
#include <QDeclarativeContext>
#include <QDebug>
#include <QTimer>

class QextSerialPort;


class timerWeigth : public QObject
{
    Q_OBJECT

     Q_PROPERTY(bool Timer READ Timer WRITE setTimer NOTIFY ChangeTimer)

public:
    explicit timerWeigth(QDeclarativeView *view =0 , const  QString
pathQmlFile=0 );

    bool Timer()  {return timerState;}
    Q_INVOKABLE  void setTimer(bool Timer) {timerState=Timer;
                                            if(Timer)
                                            {timer->start();}
                                            else
                                            {timer->stop();}

                                           }

    Q_INVOKABLE void Calibrate();

signals:
    void ChangeTimer();
    void WeightDetected(QByteArray);

public slots:
    void doTimer();
    void getWeigth();

private:
QTimer *timer;
bool timerState;
QextSerialPort *port;
QByteArray buffer;
QDeclarativeView *m_view;
```

```
};

#endif // TIMERWEIGTH_H
```

ANEXO B

En este anexo se presentan los documentos QML utilizados:

Dentro de la carpeta Dial:

Dial.qml

```
// import QtQuick 1.0 // to target S60 5th Edition or Maemo 5
import QtQuick 1.1
//import "jsfile.js" as loco

Item {
     id: root
     property real  value : DialBinding.currentAngle
     property string normdial: DialBinding.normString
     property string upperdial: DialBinding.upperString
     property string lowerdial: DialBinding.lowerString
     property int angleup: DialBinding.upAngle
     property int angledown: DialBinding.downAngle
Rectangle{
     width: 550; height: 360
     //color:"#FF5050"
     //color: "#009900"
     //color: DialBinding.color

     Image {id:imagedial; source: "images/test.png"}

     Display {

            anchors.horizontalCenter: imagedial.horizontalCenter
            y:200
            width: 250
            //width: 800
            height: 60
            //text: DialBinding.valueString

       }


     Image {
         id: needle
         x: imagedial.width/2;
         y: 302
         smooth: true
         source: "images/pointer2.png"
         transform: Rotation {
             id: needleRotation
             origin.x: needle.width/2; origin.y: 15
             //angle: Math.min(Math.max(90, root.value*2.6 + 90), 133)
             angle: (root.value+90);

             Behavior on angle {
                 SpringAnimation {
                     spring: 1.4
                     damping: .15
```

```
                    }
                }
            }
        }



    //labels
     Text{anchors.horizontalCenter:
imagedial.horizontalCenter;y:67;text:normdial;
         font.bold: true;font.pixelSize:  parent.height*.1;
horizontalAlignment: Text.AlignRight;
         font.family: "Helvetica_240_75.qpf"}




     //display
     Rectangle{id:lowerclock; x:imagedial.width/2-lowerclock.width/2;y:65

         color:"transparent"; width:100;height:240;
         Text{anchors.horizontalCenter: parent.horizontalCenter;
text:lowerdial; color:"black"
             font.bold: true;
             font.pixelSize:  parent.height*.09; horizontalAlignment:
Text.AlignRight;
             font.family: "Helvetica_240_75.qpf"
         }
         transform: Rotation {
             //origin.x: 9; origin.y: 67
             origin.x: 50; origin.y: lowerclock.height+60
              //  origin.x: lowerclock.width/2; origin.y:
lowerclock.height
             angle: angledown
             //angle: slider.x
         }
            }
     Rectangle{id:upperclock; x:imagedial.width/2-upperclock.width/2;y:65

         color:"transparent"; width:100;height:240;
         Text{anchors.horizontalCenter: parent.horizontalCenter;
text:upperdial;color:"black"
             font.bold: true
             font.pixelSize:  parent.height*.09; horizontalAlignment:
Text.AlignRight;
             font.family: "Helvetica_240_75.qpf"}
         transform: Rotation {
             //origin.x: 9; origin.y: 67
             origin.x: 50; origin.y: upperclock.height+60
             //angle: angleupperqml
             angle: angleup

         }

         states: [

             State {
```

```
                // name: "Icon1_state"; when: (dial.value>90-5+angledown
)&&(dial.value<angleup+90+10)
                 name: "Icon1_state"; when: (dial.value>90+angledown
)&&(dial.value<angleup+90)
             PropertyChanges { target: imagedial; source:"images/Fond-
dial-meter-verde.png" }

             }

         ]

            }

 }}
```

Display.qml

```
import QtQuick 1.0

BorderImage {
    id: image

    property alias text : displayText.text
    property alias currentOperation : operationText
    property int  sizetext: parent.height*.50


    source: "images/display.png"
    border { left: 10; top: 10; right: 10; bottom: 10 }

    Text {
        id: displayText
        anchors {
            right: parent.right; verticalCenter: parent.verticalCenter;
verticalCenterOffset: -1
            rightMargin: 6; left: operationText.right
        }
        font.bold: true
        font.family: "Helvetica_240_75.qpf"
       font.pixelSize:  parent.height*.55;
        //font.pixelSize:  70;
        horizontalAlignment: Text.AlignRight; elide: Text.ElideRight

        color: "#343434";
        wrapMode: Text.WrapAnywhere
    }
    Text {
        id: operationText
        font.pixelSize:  sizetext; horizontalAlignment: Text.AlignRight;
elide: Text.ElideRight
        color: "#343434"; smooth: true
        anchors { left: parent.left; leftMargin: 6; verticalCenterOffset:
-3; verticalCenter: parent.verticalCenter }
    }
```

```
}
```

Dentro de la carpeta Icons:

Icon1.qml

```
// import QtQuick 1.0 // to target S60 5th Edition or Maemo 5
import QtQuick 1.1

import "../dial" as Dial


Item{
id:icon1

    signal quitClicked
property color colortext: "white"


Image {
    id: quitButton
    y: 0
    x:750
    source: "images/quit.png"
    MouseArea {
        anchors.fill: parent
        onClicked: {icon1.quitClicked()  }
    }}

//Dial.Dial{x:0;y:50}

    Dial.Dial2Complete{id: dialshow;y:60}



Rectangle {
 x:569
 y:50
 id: info
 // 6 rows
 color: "#1C1C1C" //"#282828"
 width: 230
 height: 320
 property string text1row1: "Product:" ;            property string
text2row1: TableBinding.name;
 property string text1row2: "Upper:";               property string
text2row2: "+ "+ TableBinding.upper
 property string text1row3: "Norm:";                property string
text2row3: TableBinding.norm
 property string text1row4: "Lower:";               property string
text2row4: "- "+ TableBinding.lower
 property string text1row5:  "Pass:";               property string
text2row5: passdiplayicon1
 property string text1row6: "Reject:";              property string
```

```
text2row6: rejectdisplayicon1
 property string color1row1: "#3a2828";                    property string
color2row1: "#2E2E2E";
 property string color1row2: "#3a2828";                    property string
color2row2: "#2e2e2e";
 property string color1row3: "#3a2828";                    property string
color2row3: "#2e2e2e";
 property string color1row4: "#3a2828";                    property string
color2row4: "#2e2e2e";
 property string color1row5: "#3a2828";                    property string
color2row5: "#2e2e2e";
 property string color1row6: "#3a2828";                    property string
color2row6: "#2e2e2e";


 Column {
         id:columnInfo
         spacing: 5
         x:columnInfo.spacing
         y:columnInfo.spacing*3/2

         Rectangle { id: row1; width: info.width-columnInfo.spacing*2;
height: (info.height-columnInfo.spacing*(6+1))/6
             Row {
                 Rectangle { color:info.color1row1; width:
(row1.width)/2; height: row1.height
                     Text{
                         text: info.text1row1
                         font.bold: true
                         x:20
                         anchors.verticalCenter: parent.verticalCenter
                         color: colortext
                         font.pixelSize:  parent.height*.50;
horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
                         }
                             }
                 Rectangle { color: info.color2row1; width: row1.width/2;
height: row1.height
                     Text{
                         text: info.text2row1
                         font.bold: true
                         x:20
                         anchors.verticalCenter: parent.verticalCenter
                         color: colortext
                         font.pixelSize:  parent.height*.50;
horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
                         }
                         }

                  }}
         Rectangle { id: row2;  width: info.width-columnInfo.spacing*2;
height: (info.height-columnInfo.spacing*(6+1))/6
             Row {
                 Rectangle { color: info.color1row2; width:
(row1.width)/2; height: row1.height
                     Text{
```

```
                        text: info.text1row2
                        font.bold: true
                        x:20
                        anchors.verticalCenter: parent.verticalCenter
                        color: colortext
                        font.pixelSize:  parent.height*.50;
horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
                         }
                         }
                Rectangle { color: info.color2row2; width: row1.width/2;
height: row1.height
                    Text{

                        text: info.text2row2
                        font.bold: true
                        x:20
                        anchors.verticalCenter: parent.verticalCenter
                        color: colortext
                        font.pixelSize:  parent.height*.50;
horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
                        }
                        }
                }}
         Rectangle { id: row3; width: info.width-columnInfo.spacing*2;
height: (info.height-columnInfo.spacing*(6+1))/6
             Row {
                Rectangle { color: info.color1row3; width:
(row1.width)/2; height: row1.height
                    Text{
                        text: info.text1row3
                        font.bold: true
                        x:20
                        anchors.verticalCenter: parent.verticalCenter
                        color: colortext
                        font.pixelSize:  parent.height*.50;
horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
                        }
                        }
                Rectangle { color: info.color2row3; width: row1.width/2;
height: row1.height
                    Text{
                        text: info.text2row3
                        font.bold: true
                        x:20
                        anchors.verticalCenter: parent.verticalCenter
                        color: colortext
                        font.pixelSize:  parent.height*.50;
horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
                        }
                        }
                }}
         Rectangle { id: row4; width: info.width-columnInfo.spacing*2;
height: (info.height-columnInfo.spacing*(6+1))/6
             Row {
                Rectangle { color: info.color1row4; width:
(row1.width)/2; height: row1.height
                    Text{
```

```
                        text: info.text1row4
                        font.bold: true
                        x:20
                        anchors.verticalCenter: parent.verticalCenter
                        color: colortext
                        font.pixelSize:  parent.height*.50;
horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
                        }

                }
                Rectangle { color: info.color2row4; width: row1.width/2;
height: row1.height
                        Text{
                        text:  info.text2row4
                        font.bold: true
                        x:20
                        anchors.verticalCenter: parent.verticalCenter
                        color: colortext
                        font.pixelSize:  parent.height*.50;
horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
                        }

                }

                 }}
         Rectangle { id: row5; width: info.width-columnInfo.spacing*2;
height: (info.height-columnInfo.spacing*(6+1))/6
             Row {
                Rectangle { color: info.color1row5; width:
(row1.width)/2; height: row1.height
                        Text{
                        text: info.text1row5
                        font.bold: true
                        x:20
                        anchors.verticalCenter: parent.verticalCenter
                        color: colortext
                        font.pixelSize:  parent.height*.50;
horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
                        }

                }
                Rectangle { color: info.color2row5; width: row1.width/2;
height: row1.height
                        Text{
                        text: info.text2row5
                        font.bold: true
                        x:20
                        anchors.verticalCenter: parent.verticalCenter
                        color: colortext
                        font.pixelSize:  parent.height*.50;
horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
                        }
                }

                 }}
         Rectangle { id: row6; width: info.width-columnInfo.spacing*2;
height: (info.height-columnInfo.spacing*(6+1))/6
```

```
            Row {
                Rectangle { color: info.color1row6; width:
(row1.width)/2; height: row1.height
                    Text{
                        text: info.text1row6
                        font.bold: true
                        x:20
                        anchors.verticalCenter: parent.verticalCenter
                        color: colortext
                        font.pixelSize:  parent.height*.50;
horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
                    }
                }
                Rectangle { color: info.color2row6; width: row1.width/2;
height: row1.height
                    Text{
                        text: info.text2row6
                        font.bold: true
                        x:20
                        anchors.verticalCenter: parent.verticalCenter
                        color: colortext
                        font.pixelSize:  parent.height*.50;
horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
                    }
                }

              }}
        }//end column

 }//end info



}
```

Icon2.qml

```
// import QtQuick 1.0 // to target S60 5th Edition or Maemo 5
import QtQuick 1.1

Item{
id:icon2
    signal quitClicked


Image {
    id: quitButton
    y: 0
    x:750
    source: "images/quit.png"
    MouseArea {
        anchors.fill: parent
        onClicked: {icon2.quitClicked()  }
```

```
    }}

Rectangle {
    width: 100
    height: 62
    color: "yellow"
}
}
```

Icon3.qml

```
import QtQuick 1.1

import "../speed" as Speed



Item {



id: icon3
    //id:icon3
        signal quitClicked


Speed.Speed{}


    Image {
        id: quitButton
        y: 0
        x:750
        source: "images/quit.png"
        MouseArea {
            anchors.fill: parent
            onClicked: {icon3.quitClicked()  }
        }}
    //property int width: 100
    //property int height: 100

    property int heighttrianglebutton:100
    property int widthtrianglebutton:100



}
```

Icon4.qml

```
// import QtQuick 1.0 // to target S60 5th Edition or Maemo 5
import QtQuick 1.1
import "../table" as Table
import "../tools" as Tools
Item{
id:icon4

    signal quitClicked
property int widthofdisplay:  250
property int heightofdisplay: 50

property bool stateAddRow: false
property bool stateKeyboard: false

Table.TableComplete{id: table;x:40}

Image {
    id: deleteicon
    x: 570
    y: 340
    source: "images/DeleteRed.png"
    MouseArea {
        id: mouseareadelete
        anchors.fill: parent
        //color: "black"
        onClicked: {TableBinding.deleteRowTable()}
    }

    states: State {
        name: "mouseareadelete"; when: mouseareadelete.pressed == true
        PropertyChanges { target: deleteicon; opacity: .4 }
    }
    Text{
        y:deleteicon.height
        text:"Eliminar"
        color:"white"
         anchors.horizontalCenter: parent.horizontalCenter
    }

}



Image {
    id: addIcon
    x: deleteicon.x+deleteicon.width+60
    y: 340
    source: "images/AddGreen.png"
    MouseArea {
        id: mouseareadd
        anchors.fill: parent
        //color: "black"
        onClicked: {
            if(stateAddRow==false) stateAddRow=true; else
stateAddRow=false;

            TableBinding.setName("");
```

```
            TableBinding.setNorm("");
            TableBinding.setUpper("");
            TableBinding.setLower("");
            TableBinding.addRowTable();

        }
    }

    states: State {
        name: "mouseareadd"; when: mouseareadd.pressed == true
        PropertyChanges { target: addIcon; opacity: .4 }
         PropertyChanges {target: display1icon4;  text:"Nuevo Nombre"}
         PropertyChanges {target: display2icon4;  text:"Nueva Normal"}
         PropertyChanges {target: display3icon4;  text:"Nuevo Maximo"}
         PropertyChanges {target: display4icon4;  text:"Nuevo Minimo"}

    }
    Text{
        id:addIconLabel
        y:addIcon.height
        text:buttonAddLabel
        color:"white"
         anchors.horizontalCenter: parent.horizontalCenter
    }

}



Image {
    id: quitButton
    y: 0
    x:750
    source: "images/quit.png"
    MouseArea {
        anchors.fill: parent
        onClicked: {icon4.quitClicked()  }
    }}



 Grid{
     id: displaysicon4
     columns: 1
     x: 530
     y:  70
     spacing: 12
 Tools.Display {

        id: display1icon4
        width: widthofdisplay
        height: heightofdisplay
        text: TableBinding.name

        MouseArea {
            id: displayicon4name
```

```
            anchors.fill: parent
            onClicked: { keyB.show();
keyboardBinding.changeText(TableBinding.name);
                stateKeyboard=true;

keyboardBinding.setIndex(1);keyboardBinding.setNum(false)}
        }
    }

 Tools.Display {

        id: display2icon4

        width: widthofdisplay
        height: heightofdisplay
        text: TableBinding.norm
        MouseArea {
            id: displayicon4norm

            anchors.fill: parent
            onClicked: { keyB.show();
keyboardBinding.changeText(TableBinding.norm);
                stateKeyboard=true;
                keyboardBinding.setIndex(2);keyboardBinding.setNum(true)
}
        }

    }
 Tools.Display {

        id: display3icon4
        width: widthofdisplay
        height: heightofdisplay

        text: TableBinding.upper
        MouseArea {
            id: displayicon4upper

            anchors.fill: parent
            onClicked: { keyB.show();
keyboardBinding.changeText(TableBinding.upper);
                stateKeyboard=true;
                keyboardBinding.setIndex(3);keyboardBinding.setNum(true)
}

        }

    }
 Tools.Display {

        id: display4icon4
        width: widthofdisplay
        height: heightofdisplay
        text: TableBinding.lower

        MouseArea {
            id: displayicon4lower
```

```
            anchors.fill: parent
            onClicked: { keyB.show();
keyboardBinding.changeText(TableBinding.lower);
                stateKeyboard=true;
                keyboardBinding.setIndex(4);keyboardBinding.setNum(true)}
        }
    }



 }

 states: [

     State {
         name: "keyboardtrue"; when: stateKeyboard == true
         PropertyChanges {target: icon4; x:-800  }
         PropertyChanges {target: keyB; onHide:{icon4.x=0 ; keyB.x=-
800;stateKeyboard=false } }

     },
 State {
     name: "addnewrawtotable"; when: stateAddRow == true

     PropertyChanges {target: displayicon4name;  onClicked:{
keyB.show();keyboardBinding.changeText(TableBinding.name);

keyboardBinding.setIndex(5);keyboardBinding.setNum(false);stateKeyboard=t
rue;}}
     PropertyChanges {target: displayicon4norm;  onClicked:{
keyB.show();keyboardBinding.changeText("");

keyboardBinding.setIndex(6);keyboardBinding.setNum(true);stateKeyboard=tr
ue;}}
     PropertyChanges {target: displayicon4upper;  onClicked:{
keyB.show();keyboardBinding.changeText("");

keyboardBinding.setIndex(7);keyboardBinding.setNum(true);stateKeyboard=tr
ue;}}
     PropertyChanges {target: displayicon4lower;  onClicked:{
keyB.show();keyboardBinding.changeText("");

keyboardBinding.setIndex(8);keyboardBinding.setNum(true);stateKeyboard=tr
ue;}}

 }




]


}
```

Icon5.qml

```
import QtQuick 1.1

Item{
id:icon5
    signal quitClicked


Image {
    id: quitButton
    y: 0
    x:750
    source: "images/quit.png"
    MouseArea {
        anchors.fill: parent
        onClicked: {icon5.quitClicked()  }
    }}


Rectangle {


    Rectangle{
        x:430
        y:170
        width: 200
        height: 80
        color: "transparent"


    Text {
        y:500
        id: message
        anchors {
            right: parent.right; verticalCenter: parent.verticalCenter;
verticalCenterOffset: -1
            rightMargin: 6;
        }
        font.bold: true
        font.family: "Helvetica_240_75.qpf"
       font.pixelSize:  parent.height*.55;
        //font.pixelSize:  70;
        horizontalAlignment: Text.AlignRight; elide: Text.ElideRight

        color: "white"
        wrapMode: Text.WrapAnywhere
        text: "Aprete el\nicono para\ngrabar"
    }
    }

        Rectangle{
            x:430
            y:message.y+350
```

```
            width: 200
            height: 80
            color: "transparent"


        Text {
            id: message2
            anchors {
                right: parent.right; verticalCenter:
parent.verticalCenter; verticalCenterOffset: -1
                rightMargin: 6;
            }
            font.bold: true
            font.family: "Helvetica_240_75.qpf"
           font.pixelSize:  parent.height*.30;
            //font.pixelSize:  70;
            horizontalAlignment: Text.AlignRight; elide: Text.ElideRight

            color: "#8A0808"
            wrapMode: Text.WrapAnywhere
            text: "Solo Puede grabar\n una vez"
        }
        }

        Image {
            id: sdconnectOn
            x: 78
            y: 80
            source: "images/diskusb.png"

            MouseArea {
                id: mouseareaicon1
                anchors.fill: parent
                onClicked: SDcardBinding.createSdCardFile()
            }
            states: State {
                name: "pressedicon1"; when: mouseareaicon1.pressed ==
true
                PropertyChanges { target: sdconnectOn; opacity: .4 }
            }
        }


 }
}
```

Icon6.qml

```
import QtQuick 1.1

import "../keyboard" as Keyboard
import "../tools" as Tools
```

```
Item{
    id: icon6
    signal quitClicked

    Image {
        id: quitButton
        y: 0
        x:750
        source: "images/quit.png"
        MouseArea {
            anchors.fill: parent
            onClicked: {icon6.quitClicked()  }
        }}
 Grid{
     y:100
     columns: 2; rows: 3
     spacing: 12
     Rectangle{
         width:150; height:64
         color: "transparent"
     Text {
         anchors {
             right: parent.right; verticalCenter: parent.verticalCenter;
verticalCenterOffset: -1
             rightMargin: 6;
         }
         font.bold: true
         font.family: "Helvetica_240_75.qpf"
        font.pixelSize:  parent.height*.55;
         //font.pixelSize:  70;
         horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
         color: "white";
         wrapMode: Text.WrapAnywhere
         text:"Dia :"
     }}

    Tools.Display {
         width: 150
         height: 64
           text:  dayLabel
           MouseArea {
               anchors.fill: parent
               onClicked: {
keyB.show();keyboardBinding.setIndex(9);keyboardBinding.changeText(dayLab
el);
               keyboardBinding.setNum(true)}
           }
       }


    Rectangle{
        width:150; height:64
        color: "transparent"
    Text {
        anchors {
            right: parent.right; verticalCenter: parent.verticalCenter;
```

```
verticalCenterOffset: -1
            rightMargin: 6;
        }
        font.bold: true
        font.family: "Helvetica_240_75.qpf"
       font.pixelSize:  parent.height*.55;
        //font.pixelSize:  70;
        horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
        color: "white";
        wrapMode: Text.WrapAnywhere
        text:"Mes :"
    }}

    Tools.Display {

         width: 150
         height: 64
           text:  monthLabel
           MouseArea {
               anchors.fill: parent
               onClicked: {
keyB.show();keyboardBinding.setIndex(10);keyboardBinding.changeText(month
Label) ;keyboardBinding.setNum(true) }
           }
       }

    Rectangle{
        width:150; height:64
        color: "transparent"
    Text {
        anchors {
            right: parent.right; verticalCenter: parent.verticalCenter;
verticalCenterOffset: -1
            rightMargin: 6;
        }
        font.bold: true
        font.family: "Helvetica_240_75.qpf"
       font.pixelSize:  parent.height*.55;
        //font.pixelSize:  70;
        horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
        color: "white";
        wrapMode: Text.WrapAnywhere
        text:"Año :"
    }}

    Tools.Display {

         width: 150
         height: 64
           text:  yearLabel
           MouseArea {
               anchors.fill: parent
               onClicked: {
keyB.show();keyboardBinding.setIndex(11);keyboardBinding.changeText(yearL
abel);keyboardBinding.setNum(true)}
           }
       }
```

```
}

 Grid{
     x:400
     y:100
     columns: 2; rows: 3
     spacing: 12
     Rectangle{
         width:150; height:64
         color: "transparent"
     Text {
         anchors {
             right: parent.right; verticalCenter: parent.verticalCenter;
verticalCenterOffset: -1
             rightMargin: 6;
         }
         font.bold: true
         font.family: "Helvetica_240_75.qpf"
        font.pixelSize:  parent.height*.55;
         //font.pixelSize:  70;
         horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
         color: "white";
         wrapMode: Text.WrapAnywhere
         text:"Hora :"
     }}

    Tools.Display {
         width: 150
         height: 64
           text:  hourLabel
           MouseArea {
               anchors.fill: parent
               onClicked: {
keyB.show();keyboardBinding.setIndex(12);keyboardBinding.changeText(hourL
abel);keyboardBinding.setNum(true)}
           }
       }


    Rectangle{
        width:150; height:64
        color: "transparent"
    Text {
        anchors {
            right: parent.right; verticalCenter: parent.verticalCenter;
verticalCenterOffset: -1
            rightMargin: 6;
        }
        font.bold: true
        font.family: "Helvetica_240_75.qpf"
       font.pixelSize:  parent.height*.55;
        //font.pixelSize:  70;
        horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
        color: "white";
        wrapMode: Text.WrapAnywhere
        text:"Min :"
    }}
```

```
    Tools.Display {

          width: 150
          height: 64
            text:  minLabel
            MouseArea {
                anchors.fill: parent
                onClicked: {
keyB.show();keyboardBinding.setIndex(13);keyboardBinding.changeText(minLa
bel);keyboardBinding.setNum(true)}
            }
       }

    Rectangle{
        width:150; height:64
        color: "transparent"
    Text {
        anchors {
            right: parent.right; verticalCenter: parent.verticalCenter;
verticalCenterOffset: -1
            rightMargin: 6;
        }
        font.bold: true
        font.family: "Helvetica_240_75.qpf"
       font.pixelSize:  parent.height*.55;
        //font.pixelSize:  70;
        horizontalAlignment: Text.AlignRight; elide: Text.ElideRight
        color: "white";
        wrapMode: Text.WrapAnywhere
        text:"Seg :"
    }}

    Tools.Display {

          width: 150
          height: 64
            text:  secLabel
            MouseArea {
                anchors.fill: parent
                onClicked: {
keyB.show();keyboardBinding.setIndex(14);keyboardBinding.changeText(secLa
bel);keyboardBinding.setNum(true)}

            }
       }




    Tools.Button {id:reaybutton;x:0 ; width:120;height: 70; operation:
"calibrar"
        y:250
        color: 'red' ; onClicked: TimerWeightBinding.Calibrate() }
}
```

```
    //Keyboard.Keyboard {numberkeyboard: true;
buttonchangenumkeyboard:false;  x:-background.width; id: keyB;
widthItem:800; heightItem: 480;onEnterClicked: keyB.x= -background.width
; onQuitClicked: keyB.x= -background.width   }


}
```

IconBinding.qml

```
import QtQuick 1.1

Item {
    id: binding
    signal show
    signal hide
    property string returnqml: "../welcome.qml"
    property string targetqml: "icon1.qml"

    Loader { id: pageLoader ; focus:true ;  anchors.fill: parent}

    onShow: {pageLoader.source=targetqml ; }

    Connections{
     ignoreUnknownSignals: true
     target:pageLoader.item
     onQuitClicked: { pageLoader.source = returnqml;
binding.hide();TimerWeightBinding.setTimer(false);
         TimerClockBinding.setTimer(false) }
     }

}
```

Dentro de la carpeta Keyboard:

App.qml

```
// import QtQuick 1.0 // to target S60 5th Edition or Maemo 5
import QtQuick 1.1

Item {
    id: rect;
    width: 800
    height: 480

    KeyboardBinding{id: keyB; returnqml: "app.qml" }
   ButtonKeyboard { onClicked: keyB.show() }

}
```

ButtonKeyboard.qml

```
// import QtQuick 1.0 // to target S60 5th Edition or Maemo 5
import QtQuick 1.1


BorderImage {
    id: button

    property alias operation: buttonText.text
    property string color: ""

    signal clicked
    signal pressedAndHold

    source: "images/button-" + color + ".png"; clip: true
    border { left: 10; top: 10; right: 10; bottom: 10 }

    Rectangle {
        id: shade
        anchors.fill: button; radius: 10; color: "black"; opacity: 0
    }
    Text {
        id: buttonText
        anchors.centerIn: parent; anchors.verticalCenterOffset: -1
        font.pixelSize: parent.width > parent.height ? parent.height * .5
: parent.width * .5
        style: Text.Sunken; color: "white"; styleColor: "black"; smooth:
true
    }

    MouseArea {
        id: mouseArea
        anchors.fill: parent
        onClicked: {
           button.clicked()
        }
        onPressAndHold: {
            button.pressedAndHold()
        }
    }

    states: State {
        name: "pressed"; when: mouseArea.pressed == true
        PropertyChanges { target: shade; opacity: .4 }
    }
}
```

Display.qml

```
// import QtQuick 1.0 // to target S60 5th Edition or Maemo 5
import QtQuick 1.1
```

```
import QtQuick 1.0

BorderImage {
    id: image

    property alias text : displayText.text
    property alias currentOperation : operationText
    property int  sizetext: parent.height*.50


    source: "images/display.png"
    border { left: 10; top: 10; right: 10; bottom: 10 }

    Text {
        id: displayText
        anchors {
            right: parent.right; verticalCenter: parent.verticalCenter;
verticalCenterOffset: -1
            rightMargin: 6; left: operationText.right
        }
        font.bold: true
        font.family: "Helvetica_240_75.qpf"
       font.pixelSize:  parent.height*.55;
        //font.pixelSize:  70;
        horizontalAlignment: Text.AlignRight; elide: Text.ElideRight

        color: "#343434";
        wrapMode: Text.WrapAnywhere
    }
    Text {
        id: operationText
        font.pixelSize:  sizetext; horizontalAlignment: Text.AlignRight;
elide: Text.ElideRight
        color: "#343434"; smooth: true
        anchors { left: parent.left; leftMargin: 6; verticalCenterOffset:
-3; verticalCenter: parent.verticalCenter }
    }
}
```

Keyboard.qml

```
import QtQuick 1.0


Item{
    id:keyboard;
    property int widthItem: 800
    property int heightItem: 480

    property bool numberkeyboard: numberKeyboard
    property bool buttonchangenumkeyboard: buttonchangenum
    property bool buttonuppercase: true
    property string textbuttonkeyboardchange: "123"
    signal enterClicked
```

```
    signal quitClicked
 //   signal clicked

    width: widthItem
    height: heightItem



Rectangle {
    id: keyboard1
    y:heightItem*1/5

    color: "#282828"

        property bool uppercase: false



         Rectangle {

             Display {

                 id: displayKeyboard
                 width: widthItem*3/4
                 height: 64
                 x:widthItem/2-displayKeyboard.width/2
                  y:10
                  text: keyboardBinding.text
             }

             Image {
                 id: quitButton
                 x: widthItem-quitButton.width -widthItem/50
                 y: 0

                 source: "images/quit.png"
                 MouseArea {
                     anchors.fill: parent
                     onClicked: {
keyboardBinding.changeText("/u3");keyboard.quitClicked()   }
                 }}



               id: column;//spacing: 6
                width: 800; height: 400
                color: "#282828"
                property real h:55
                property real w:widthItem/10-12
                Row {
                    id: line1keyboard
                    x:10
                    y:displayKeyboard.height+20
                    spacing: 6
                    ButtonKeyboard {id:q; width: column.w; height:
column.h; operation: "q"; color: 'blue';onClicked:
{{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)}
```

```
keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:w; width: column.w; height:
column.h; operation: "w"; color: 'blue';onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)}  }
                    ButtonKeyboard {id:e; width: column.w; height:
column.h; operation: "e"; color: 'blue';onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)}  }
                    ButtonKeyboard {id:r; width: column.w; height:
column.h; operation: "r"; color: 'blue';onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)}  }
                    ButtonKeyboard {id:t; width: column.w; height:
column.h; operation: "t"; color: 'blue';onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)}  }
                    ButtonKeyboard {id:y; width: column.w; height:
column.h; operation: "y"; color: 'blue';onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)}  }
                    ButtonKeyboard {id:u; width: column.w; height:
column.h; operation: "u"; color: 'blue';onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)}  }
                    ButtonKeyboard {id:i; width: column.w; height:
column.h; operation: "i"; color: 'blue';onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)}  }
                    ButtonKeyboard {id:o; width: column.w; height:
column.h; operation: "o"; color: 'blue';onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)}  }
                    ButtonKeyboard {id:p; width: column.w; height:
column.h; operation: "p"; color: 'blue';onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)}  }

                }
                Row {
                    id: line1numberkeyboard
                    x:10 -widthItem
                    y:displayKeyboard.height+20
                    spacing: 6
                    ButtonKeyboard { width: column.w; height: column.h;
operation: "7"; color: 'blue';onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)}  }
                    ButtonKeyboard { width: column.w; height: column.h;
operation: "8"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {width: column.w; height: column.h;
operation: "9"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }


                }


                Row {
                    id:line2keyboard
                    y:line1keyboard.y+column.h+10
                    x:column.w/2+10
                    spacing: 6
                    ButtonKeyboard {id:a; width: column.w; height:
column.h; operation: "a"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
```

```
                    ButtonKeyboard {id:s; width: column.w; height:
column.h; operation: "s"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:d; width: column.w; height:
column.h; operation: "d"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:f; width: column.w; height:
column.h; operation: "f"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:g; width: column.w; height:
column.h; operation: "g"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:h; width: column.w; height:
column.h; operation: "h"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:j; width: column.w; height:
column.h; operation: "j"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:k; width: column.w; height:
column.h; operation: "k"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:l; width: column.w; height:
column.h; operation: "l"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:ene; width: column.w; height:
column.h; operation: "ñ"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }

                }
                Row {
                    id:line2numberkeyboard
                    y:line1keyboard.y+column.h+10
                    x:column.w/2+10-widthItem
                    spacing: 6
                    ButtonKeyboard { width: column.w; height: column.h;
operation: "4"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard { width: column.w; height: column.h;
operation: "5"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard { width: column.w; height: column.h;
operation: "6"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }

                }


                Row {
                    id:line3keyboard
                    y:line2keyboard.y+column.h+10
                    x:column.w+20
                    spacing: 6

                    ButtonKeyboard {id:z; width: column.w; height:
column.h; operation: "z"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:x; width: column.w; height:
```

```
column.h; operation: "x"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:c; width: column.w; height:
column.h; operation: "c"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:v; width: column.w; height:
column.h; operation: "v"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:b; width: column.w; height:
column.h; operation: "b"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:n; width: column.w; height:
column.h; operation: "n"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard {id:m; width: column.w; height:
column.h; operation: "m"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard { width: column.w; height: column.h;
operation: ";"; color: 'purple' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }

                }


                Row {
                    id:line3numberkeyboard
                    y:line2keyboard.y+column.h+10
                    x:column.w+20-widthItem
                    spacing: 6

                    ButtonKeyboard { width: column.w; height: column.h;
operation: "1"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard { width: column.w; height: column.h;
operation: "2"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard { width: column.w; height: column.h;
operation: "3"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }

                }

                Row {
                    id:line4keyboard
                    y:line3keyboard.y+column.h+10
                    x:22+column.w*5/2
                    spacing: 6
                    //ButtonKeyboard { width: column.w*5/2; height:
column.h; }
                    ButtonKeyboard {  width: column.w*5; height:
column.h; operation: " " ; color: 'green' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard { width: column.w; height: column.h;
operation: "?"; color: 'purple' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }

                }
```

```
                Row {
                    id:line4numberkeyboard
                    y:line3keyboard.y+column.h+10
                    x:22+column.w*5/2-widthItem
                    spacing: 6
                    //ButtonKeyboard { width: column.w*5/2; height:
column.h; }
                    ButtonKeyboard {  width: column.w*4-6; height:
column.h; operation: " " ; color: 'green' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard { width: column.w; height: column.h;
operation: "0"; color: 'blue' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }
                    ButtonKeyboard { width: column.w; height: column.h;
operation: "."; color: 'purple' ;onClicked:
{keyboardBinding.changeText(operation); keyboardBinding.setEdit(true)} }

                }



////////////////////////////////////////////////////////////////////////
/////////
  //button to uppercase
                ButtonKeyboard {

                    id: upperbutton;
                    width: column.w; height: column.h;
                    y:line2keyboard.y+column.h+10
                    x:10
                    onClicked:
{if(keyboard1.uppercase==false)keyboard1.uppercase = true; else
keyboard1.uppercase = false; }

                    Image{source: "images/arrowup.png"; anchors.centerIn:
parent;}
                }
////////////////////////////////////////////////////////////////////////
/////////
 //button to button erase
                ButtonKeyboard {

                    id: deletebutton;
                    width:column.w*3/2; height: column.h;
                    y:line2keyboard.y+column.h+10
                    x:column.w+620
                    onClicked:
{keyboardBinding.changeText("/u1");keyboardBinding.setEdit(true)}
                    Image{source: "images/deletearrow.png";
anchors.centerIn: parent;}
                }


////////////////////////////////////////////////////////////////////////
/////////
 //button to numberkeyboard
               ButtonKeyboard {
```

```
Text{id:textbuttonnumber;text:textbuttonkeyboardchange; anchors.centerIn:
parent; style: Text.Sunken; color: "white"; styleColor: "black"; smooth:
true;
                        font.pixelSize: parent.width > parent.height ?
parent.height * .5 : parent.width * .5}
                    id: numberkeyboardd;
                    width: column.w*5/2; height: column.h;
                    y:line3keyboard.y+column.h+10
                    x:15
                    onClicked: if(keyboard.numberkeyboard==false)
keyboard.numberkeyboard = true;else keyboard.numberkeyboard=false

                }



////////////////////////////////////////////////////////////////////////
/////////
//button to enter
                ButtonKeyboard {

                    id: buttonEnter;
                    width: column.w*5/2; height: column.h
                    y:line3keyboard.y+column.h+10
                    x:22+column.w*5/2+420
                    //onClicked:
if(keyboard.numberkeyboard==false)keyboard.numberkeyboard = true; else
keyboard.numberkeyboard= false
                    onClicked: { keyboardBinding.changeText("/u2");
keyboard.enterClicked() }
                    Image{source: "images/enterarrow.png";
anchors.centerIn: parent;}
                }


////////////////////////////////////////////////////////////////////////
///
        }
//////////////////////////////////////////////////////

          states: [

              State{
              name: "changetonumberkeyboard";when: keyboard.numberkeyboard
== true
              PropertyChanges { target: line1keyboard; x: -widthItem}
              PropertyChanges { target: line1numberkeyboard; x:
10+widthItem/3}
              PropertyChanges { target: line2keyboard; x: -widthItem}
              PropertyChanges { target: line2numberkeyboard; x:
column.w/2+10+widthItem/3}
              PropertyChanges { target: line3keyboard; x: -widthItem}
              PropertyChanges { target: line3numberkeyboard;
x:column.w+20+widthItem/3}
              PropertyChanges { target: line4keyboard; x: -widthItem}
              PropertyChanges { target: line4numberkeyboard;
```

```
x:22+column.w*5/2}
              PropertyChanges { target: textbuttonnumber; text:"abc"}
              }
                 ]


}//end keboard1

states:
    [
    State {
    name: "changetouppercase"; when: keyboard1.uppercase == true
    PropertyChanges { target: q; operation: "Q"}
    PropertyChanges { target: w; operation: "W"}
    PropertyChanges { target: e; operation: "E"}
    PropertyChanges { target: r; operation: "R"}
    PropertyChanges { target: t; operation: "T"}
    PropertyChanges { target: y; operation: "Y"}
    PropertyChanges { target: u; operation: "U"}
    PropertyChanges { target: i; operation: "I"}
    PropertyChanges { target: o; operation: "O"}
    PropertyChanges { target: p; operation: "P"}
    PropertyChanges { target: a; operation: "A"}
    PropertyChanges { target: s; operation: "S"}
    PropertyChanges { target: d; operation: "D"}
    PropertyChanges { target: f; operation: "F"}
    PropertyChanges { target: g; operation: "G"}
    PropertyChanges { target: h; operation: "H"}
    PropertyChanges { target: j; operation: "J"}
    PropertyChanges { target: k; operation: "K"}
    PropertyChanges { target: l; operation: "L"}
    PropertyChanges { target: ene; operation: "Ñ"}
    PropertyChanges { target: z; operation: "Z"}
    PropertyChanges { target: x; operation: "X"}
    PropertyChanges { target: c; operation: "C"}
    PropertyChanges { target: v; operation: "V"}
    PropertyChanges { target: b; operation: "B"}
    PropertyChanges { target: n; operation: "N"}
    PropertyChanges { target: m; operation: "M"}


    },

     State{  //hide letters keyboard
     name: "ff";when: keyboard.buttonchangenumkeyboard == false
     PropertyChanges { target: numberkeyboardd; x: -widthItem}
     PropertyChanges { target: upperbutton; x: -widthItem}
     }
]


}
```

Keyboardbinding.qml

```
import QtQuick 1.1

Item {
    id:binding
    signal show
    signal hide
    property string returnqml: "../welcome.qml"
    Loader { id: pageLoader ; focus:true ;  anchors.fill: parent}

    onShow: {pageLoader.source="Keyboard.qml" }

    Connections{
     ignoreUnknownSignals: true
     target:pageLoader.item
     onEnterClicked: { binding.hide()}
     onQuitClicked: { binding.hide() }
     }


}
```

Dentro de la capeta speed:

Speed.qml

```
import QtQuick 1.1
import "../tools" as Tools

Rectangle
{
    id:d

    Rectangle{

        x:30
        y:100
        color:"#282828"
        height: 310
        width:800/1.1

      Tools.Display {

           id: displayicon3
             width: 800/1.4
             height: 180
             y:20
             x:buttonup.width+gridButtons.x+20
             text: TimerSpeedBinding.F
         }

      Tools.Button {id:stopbutton;x: displayicon3.x+displayicon3.width-
stopbutton.width; width:120;height: 70;operation: "Parar"
          y:displayicon3.y+displayicon3.height+20
          color: 'red' ;onClicked: TimerSpeedBinding.setOff();}
```

```
      Tools.Button {id:reaybutton;x: displayicon3.x+displayicon3.width-
reaybutton.width-450; width:120;height: 70;operation: "Listo"
          y:displayicon3.y+displayicon3.height+20
          color: 'red' ;onClicked: TimerSpeedBinding.readySettings();    }


      Tools.Button {id:startbutton;x: displayicon3.x+displayicon3.width-
startbutton.width-150; width:200;height: 70;operation: "Empezar"
          y:displayicon3.y+displayicon3.height+20
          color: 'green' ;onClicked: TimerSpeedBinding.setOn();}


Grid{
 id:gridButtons
 anchors.verticalCenter: parent.verticalCenter
 columns: 1
 spacing: 20
 width:buttonup.width
 x:20
      Tools.TriangleButton
      {
         id: buttonup; width: 100; height: 100; opacity: 0.9;
         onClicked: TimerSpeedBinding.setF(TimerSpeedBinding.F+1)


      }

      Tools.TriangleButton
      {

          transform: Rotation{ origin.x:buttondown.width/2;
origin.y:buttondown.height/2; angle: 180}
          id: buttondown; width: 100; height: 100; opacity: 0.9
          onClicked: TimerSpeedBinding.setF(TimerSpeedBinding.F-1)
      }

     }

 }



}
```

Dentro de la carpeta table:

App.qml

```
import QtQuick 1.1

Item{
    width:800
    height: 480
```

```
    Rectangle{
        width: 800
        height: 480
    TableComplete{x:40}
    }
}
```

TableBody.qml

```
// import QtQuick 1.0 // to target S60 5th Edition or Maemo 5
import QtQuick 1.1

import QtQuick 1.1

Item{

property int widthItem: 390
property int heightItem: TableBinding.maxRows*60+30
property int heightRow: 60
property int tableY: 0

width: widthItem
height: heightItem


Rectangle{
    width:widthItem
    height: heightItem
    color: "#282828"
      y:tableY*heightRow

ListView {
          id: list_view1
          clip: true
          anchors.margins: 10
          anchors.fill: parent
          model: TableBinding
          delegate: TableDelegate {
              width: widthItem
              height: heightRow
          }

          interactive : false
          highlight: highlight
          highlightFollowsCurrentItem: false
          focus: true


          Component {
              id: highlight
              Rectangle {
                  focus: true
                    id: test
```

```
                    width: widthItem-30; height: heightRow
                    color: "lightsteelblue"; radius: 5
                    y:TableBinding.rowIndex*heightRow

                }
            }

        }

}
}
```

TableComplete.qml

```
import QtQuick 1.1

Item{

property int widthItem: 800
property int heightItem: 480
property int heightRow: 60
width:widthItem
height: heightItem

    TableBody{id: table; y:heightRow; height:320}


    Rectangle{
        x:table.x
        y:table.y-height
        width:table.width
        height: heightRow
        color: "#3a2828"
    Row {
        anchors.margins: 4
        anchors.fill: parent
        spacing: 4;
        Text {
            id: nametext
            text: "Nombre"
            width: parent.width/5
            color: "white"
            font.pixelSize:  parent.height*.30;
            y:20
        }
        Text {
            id: normtext
            text: "Normal"
            width: parent.width/5;
            color: "white"
            font.pixelSize:  parent.height*.30; horizontalAlignment:
Text.AlignRight; elide: Text.ElideRight
            y:20

        }
```

```
        Text {
            id: uppertext
            text: "Maximo"
            width: parent.width/5;
            color: "white"
            font.pixelSize:  parent.height*.30; horizontalAlignment:
Text.AlignRight; elide: Text.ElideRight
            y:20

        }

        Text{
            id:lowertext
            text: "Minimo"
            width: parent.width/5;
            color: "white"
            font.pixelSize:  parent.height*.30; horizontalAlignment:
Text.AlignRight; elide: Text.ElideRight
            y:20

        }

    }}


    Rectangle{
        id:buttonsupanddown
        color:"#2b2b33"
        x: table.x+table.width
        y: table.y
        height: table.height
        width: 90
    Grid
    {

    anchors.verticalCenter: parent.verticalCenter
    columns: 1
    spacing: 9

    TriangleButton {
        id: toUp
        width: buttonsupanddown.width; height:
table.height/2.5;onClicked: if(table.tableY!=0)
{table.tableY=table.tableY+1} opacity: 0.9}
    TriangleButton {
        id: toDown
          transform: Rotation{ origin.x:toDown.width/2;
origin.y:toDown.height/2; angle: 180}

          width: buttonsupanddown.width; height:
table.height/2.5;onClicked: table.tableY=table.tableY-1 ;opacity: 0.9}
      }
      }

}
```

TableDelegate.qml

```
import QtQuick 1.1

Item {
    id: delegate
    clip: true
    anchors.margins: 4
    Rectangle{
        id: re
        width:parent.width
        height: parent.height
        color:"transparent"

        Text {
            id: key
            text: t1key
        }

    Row {
        anchors.margins: 4
        anchors.fill: parent
        spacing: 4;
        Text {
            id: nametext
            text: name
            width: delegate.width/5
            color: "white"
            font.bold: true
            font.pixelSize:  parent.height*.40;
            y:20
        }
        Text {
            id: normtext
            text: norm
            width: delegate.width/5;
            color: "white"
            font.bold: true
            font.pixelSize:  parent.height*.40; horizontalAlignment:
Text.AlignRight; elide: Text.ElideRight
            y:20

        }

        Text {
            id: uppertext
            text: upper
            width: delegate.width/5;
            color: "white"
            font.bold: true
            font.pixelSize:  parent.height*.40; horizontalAlignment:
Text.AlignRight; elide: Text.ElideRight
            y:20
```

```
        }

        Text{
            id:lowertext
            text: lower
            width: delegate.width/5;
            color: "white"
            font.bold: true
            font.pixelSize:  parent.height*.40; horizontalAlignment:
Text.AlignRight; elide: Text.ElideRight
            y:20

        }

    }
    MouseArea{
        anchors.fill: parent
        onClicked: {
            TableBinding.setKey(key.text);
            TableBinding.setName(nametext.text);
            TableBinding.setNorm(normtext.text);
            TableBinding.setUpper(uppertext.text);
            TableBinding.setLower(lowertext.text);
            TableBinding.setRowIndex(delegate.y/delegate.height);
            TableBinding.setReadytoAdd(false);

                }
    }

    }
}
```

TriangleButton.qml

```
import QtQuick 1.1


Item {
    id: container

    signal clicked

    property string text

    BorderImage {
        id: buttonImage
        source: "images/triangle-icon.png"
        width: container.width; height: container.height
    }
    BorderImage {
        id: pressed
        opacity: 0.1
        source: "images/triangle-icon.png"
        width: container.width; height: container.height
    }
    MouseArea {
```

```
        id: mouseRegion
        anchors.fill: buttonImage
        onClicked: { container.clicked(); }
    }
    Text {
        color: "white"
        anchors.centerIn: buttonImage; font.bold: true; font.pixelSize:
15
        text: container.text; style: Text.Raised; styleColor: "black"
    }
    states: [
        State {
            name: "Pressed"
            when: mouseRegion.pressed == true
            PropertyChanges { target: pressed; opacity: 1.3 }
        }
    ]
}
```

Dentro de la carpeta tools:

Button.qml

```
import QtQuick 1.1


BorderImage {
    id: button

    property alias operation: buttonText.text
    property string color: ""


    signal clicked

    source: "images/button-" + color + ".png"; clip: true
    border { left: 10; top: 10; right: 10; bottom: 10 }

    Rectangle {
        id: shade
        anchors.fill: button; radius: 10; color: "black"; opacity: 0
    }

    Text {
        id: buttonText
        anchors.centerIn: parent; anchors.verticalCenterOffset: -1
        font.pixelSize: parent.width > parent.height ? parent.height * .5
: parent.width * .5
        style: Text.Sunken; color: "white"; styleColor: "black"; smooth:
true
    }

    MouseArea {
        id: mouseArea
```

```
        anchors.fill: parent
        onClicked: {

            button.clicked()
        }
    }

    states: State {
        name: "pressed"; when: mouseArea.pressed == true
        PropertyChanges { target: shade; opacity: .4 }
    }
}
```

Display.qml

```
import QtQuick 1.0

BorderImage {
    id: image

    property alias text : displayText.text
    property alias currentOperation : operationText
    property int  sizetext: parent.height*.50


    source: "images/display.png"
    border { left: 10; top: 10; right: 10; bottom: 10 }

    Text {
        id: displayText
        anchors {
            right: parent.right; verticalCenter: parent.verticalCenter;
verticalCenterOffset: -1
            rightMargin: 6; left: operationText.right
        }
        font.bold: true
        font.family: "Helvetica_240_75.qpf"
       font.pixelSize:  parent.height*.55;
        //font.pixelSize:  70;
        horizontalAlignment: Text.AlignRight; elide: Text.ElideRight

        color: "#343434";
        wrapMode: Text.WrapAnywhere
    }
    Text {
        id: operationText
        font.pixelSize:  sizetext; horizontalAlignment: Text.AlignRight;
elide: Text.ElideRight
        color: "#343434"; smooth: true
        anchors { left: parent.left; leftMargin: 6; verticalCenterOffset:
-3; verticalCenter: parent.verticalCenter }
    }
}
```

TriangleButton.qml

```
import QtQuick 1.1


Item {
    id: container

    signal clicked

    property string text

    BorderImage {
        id: buttonImage
        source: "images/triangle-icon.png"
        width: container.width; height: container.height
    }
    BorderImage {
        id: pressed
        opacity: 0.1
        source: "images/triangle-icon.png"
        width: container.width; height: container.height
    }
    MouseArea {
        id: mouseRegion
        anchors.fill: buttonImage
        onClicked: { container.clicked(); }
    }
    Text {
        color: "white"
        anchors.centerIn: buttonImage; font.bold: true; font.pixelSize:
15
        text: container.text; style: Text.Raised; styleColor: "black"
    }
    states: [
        State {
            name: "Pressed"
            when: mouseRegion.pressed == true
            PropertyChanges { target: pressed; opacity: 1.3 }
        }
    ]


}
```

Archivos QML principales:

Button.qml

```
import QtQuick 1.0

Item {
    id: container

    signal clicked

    property string text

    BorderImage {
        id: buttonImage
        source: "images/toolbutton.sci"
        width: container.width; height: container.height
    }
    BorderImage {
        id: pressed
        opacity: 0
        source: "images/toolbutton.sci"
        width: container.width; height: container.height
    }
    MouseArea {
        id: mouseRegion
        anchors.fill: buttonImage
        onClicked: { container.clicked(); }
    }
    Text {
        color: "white"
        anchors.centerIn: buttonImage; font.bold: true; font.pixelSize:
15
        text: container.text; style: Text.Raised; styleColor: "black"
    }
    states: [
        State {
            name: "Pressed"
            when: mouseRegion.pressed == true
            PropertyChanges { target: pressed; opacity: 1 }
        }
    ]
}
```

GridIcons.qml

```
import QtQuick 1.1

Item {

    property int xGrid: 0
    GridView{
```

```
        id: listIcons
        x:xGrid


        Image {
            id: imageicon1
            x: 78
            y: 80
            source: "images/scheduled_tasks.png"

            MouseArea {
                id: mouseareaicon1
                anchors.fill: parent
                //color: "black"
                onClicked:
{icon1.show();TimerWeightBinding.setTimer(true)}
            }



            states: State {
                name: "pressedicon1"; when: mouseareaicon1.pressed ==
true
                PropertyChanges { target: imageicon1; opacity: .4 }
            }
        }


        Image {
            id: imageicon2
            x: 342
            y: 80
            source: "images/internet.png"
            MouseArea {
                id: mouseareaicon2
                anchors.fill: parent
                onClicked: {icon2.show()}
            }


            states: State {
                name: "pressedicon2"; when: mouseareaicon2.pressed ==
true
                PropertyChanges { target: imageicon2; opacity: .4 }
            }
        }

        Image {
            id: imageicon3
            x: 619
            y: 80
            source: "images/desktop.png"
            MouseArea {
                id: mouseareaicon3
                anchors.fill: parent
                onClicked: {icon3.show()}
            }
```

```
            states: State {
                name: "pressedicon3"; when: mouseareaicon3.pressed ==
true
                PropertyChanges { target: imageicon3; opacity: .4 }
            }

        }

        Image {
            id: imageicon4
            x: 90
            y: 268
            source: "images/my_documents.png"
            MouseArea {
                id: mouseareaicon4
                anchors.fill: parent
                onClicked: {icon4.show()}
            }



            states: State {
                name: "pressedicon4"; when: mouseareaicon4.pressed ==
true
                PropertyChanges { target: imageicon4; opacity: .4 }
            }
        }

        Image {
            id: imageicon5
            x: 348
            y: 268
            source: "images/workgroup.png"
            MouseArea {
                id: mouseareaicon5
                anchors.fill: parent
                onClicked: {icon5.show()}
            }
            states: State {
                name: "pressedicon5"; when: mouseareaicon5.pressed ==
true
                PropertyChanges { target: imageicon5; opacity: .4 }
            }
        }

        Image {
            id: imageicon6
            x: 619
            y: 268
            source: "images/help_and_support.png"
            MouseArea {
                id: mouseareaicon6
                anchors.fill: parent
                onClicked:
{icon6.show();TimerClockBinding.setTimer(true)}
            }
            states: State {
```

```
                name: "pressedicon6"; when: mouseareaicon6.pressed ==
true
                PropertyChanges { target: imageicon6; opacity: .4 }
            }
        }



    }
}
```

ToolBar.qml

```
import QtQuick 1.0

Item {
    id: toolbar

    property alias button1Label: button1.text
    property alias button2Label: button2.text

    property bool button1appear: false
    property bool button2appear: true
    property bool bothbuttonsappear: false


    signal button1Clicked(string msg)
    signal button2Clicked(string msg)


    BorderImage { source: "images/titlebar.sci"; width: parent.width;
height: parent.height + 14; y: -7 }

//    Row {
//        anchors.right: parent.right; anchors.rightMargin: 5; y: 3;
height: 32; spacing: 2


    Button { anchors.rightMargin: 5; y: 3;
        x:background.width

        id: button1; width: 235; height: 32
        //onClicked: {toolbar.button1Clicked(""); if(LedOfConnect==1) {
toolBar. botton1activatedbutton2activated= false;
toolBar.botton1activatedbutton2desactivated=true}
else{toolBar.botton1activatedbutton2activated=true}  }
        onClicked: toolbar.button1Clicked("")

    }

    Button {
            id: button2; width: 235; height: 32; y:3;
            x:background.width
```

```
            onClicked: { toolbar.button2Clicked("") }
        }


    states:[

        State {
        name: "button1sd"; when: toolBar.button1appear == true
        PropertyChanges { target: button1; x: +2}
        //PropertyChanges { target: button2; x:background.width}
            },
        State {
        name: "button2sd"; when: toolBar.button2appear == true
        //PropertyChanges { target: button1; x: background.width}
        PropertyChanges { target: button2; x:+background.width-
button2.width-2}
            },
        State {
        name: "bothbuttons"; when: toolBar.bothbuttonsappear == true
        PropertyChanges { target: button1; x: 2}
        PropertyChanges { target: button2; x:background.width-
button2.width-2}
            }



    ]



}
```

Welcome.qml

```
// import QtQuick 1.0 // to target S60 5th Edition or Maemo 5
import QtQuick 1.1
import "icons" as Icons
import "keyboard" as Keyboard
import "dial" as Dial

Item{
    id:item
    width:800;
    height: 480
    signal startTimerIcon1();

Image {

    id: background


    anchors.rightMargin: 0
    anchors.bottomMargin: 0
    anchors.leftMargin: 0
```

```
    anchors.topMargin: 0

    source: "images/graywallpaper.png"

    anchors.fill: parent



////////////////////////////////////////////////////////////////////////////
////////////

////////////////////////////////////////////////////////////////////////////
////////////
    //list of icons
    Icons.IconBinding{id: icon1; onShow: {gridIcons.x=-800;}}
    Icons.IconBinding{id: icon2; targetqml:"icon2.qml";onShow:
gridIcons.x=-800  }
    Icons.IconBinding{id: icon3; targetqml:"icon3.qml";onShow:
gridIcons.x=-800 }
    Icons.IconBinding{id: icon4; targetqml:"icon4.qml";onShow:
gridIcons.x=-800  }
    Icons.IconBinding{id: icon5; targetqml:"icon5.qml";onShow:
gridIcons.x=-800 }
    Icons.IconBinding{id: icon6; targetqml:"icon6.qml";onShow:
gridIcons.x=-800 }
    Keyboard.KeyboardBinding{id: keyB; onHide: keyB.x=-800;onShow:
keyB.x=0    }


    GridIcons{id:gridIcons;xGrid:0}

//    Button {width:100; height:100;onClicked:
{TimerSpeedBinding.setTimer(true);TimerSpeedBinding.setOn()  }}


            ToolBar {
                    id: toolBar
                    height: 40; anchors.bottom: parent.bottom; width:
background.width; opacity: 0.9
                    //button1Label:  "Connect"
                    //button1Label:
                    button2Label: "Salir"
                    onButton2Clicked: Qt.quit()
                    //onButton2Clicked: item.startTimerIcon1()

            }


    }
}
```