The Research of Qt_Embedded and Embedded Linux Application in the Intelligent Monitoring System Control

[1]N. SaiJithendra, [2] D.Aruna kumari, [3]Prof K V Murali Mohan
[1]M. Tech Student, Holy Mary Institute of Technology & Science, Bogaram (V), Keesara (M), R. R Dt.- 501301.
[2]Assistant Professor, ECE, Holy Mary Institute of Technology & Science, Bogaram (V), Keesara (M), R. R Dt.- 501301
[3]Professor,HOD of ECE Dept, Holy Mary Institute of Technology & Science, Bogaram (V), Keesara (M), R. R Dt.- 501301

[1]saijithendra.nalagatla@gmail.com, [2]arunadasari12@gmail.com, [3]kvmmece@gmail.com

I ABSTRACT

Qt is a cross-platform application framework that is widely used for developing application software with a graphical user interface (GUI). It is currently produced by Nokia's Qt Development Frameworks division, which came into being after Nokia's acquisition of the Norwegian company Trolltech, the original producer of Qt. Qt uses standard C++ but makes extensive use of a special code generator (called the Meta Object Compiler, or moc) together with several macros to enrich the language. Non-GUI features include SQL database access, XML parsing, thread management, network support, and a unified cross-platform API for file handling.

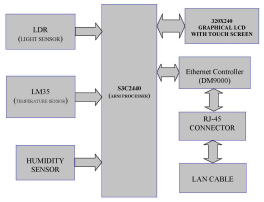
In this project we are implementing high-performance 32-bit microprocessors such as S3C2440, embedded Linux system and Qt / embedded GUI application for Laboratory Intelligent Monitoring System.

Embedded front-end machine use Samsung's S3C2440 ARM processor as the main controller, to which the sensors are connected. We will be using Temperature Sensor and Humidity Sensor for this project. The microprocessor collects the data for the environment of the lab and uses the TCP / IP to send it to the monitoring center for processing.

The ARM processor will also have some relays connected to it for switching off or on any electrical equipment in the laboratory when the laboratory sensors data is abnormal, such as the temperature is too high. ARM Processor is equipped with a touch LCD monitor, which enables us to develop friendly GUI using QT under embedded Linux system to provide functions such as querying and setting the laboratory environment parameters.

Various types of sensors have their own specific device drivers due to the different working principles. So, we will have to make different device drivers for both the sensors.

Black Diagram



Keywords-embedded linux; Qt / Embedded; S3C2440 ARM microprocessor; Intelligent Monitoring System

II INTRODUCTION

The embedded systems which use micro-controller such as 8-bit microcontroller as the main controller has been widely used in various fields, but most of these applications are still in the low-level stag1e of stand-alone use of the embedded system. It is feasible and forward-looking to apply the high-performance 32-bit microprocessors such as S3C2440, embedded linux system and Qt / embedded GUI application to practical industrial control in certain occasion.

Nowadays the management of the domestic laboratories in the research institute and universities has issues of poor real time, high cost and low precision .It is difficult to determine the quality of the environment of the laboratory. So the Laboratory Intelligent Monitoring System should be developed to implement early warning, remote control, real-time monitoring and other functions. This paper focuses on the process and difficult points in the application of embedded GUI based on Qt / Embedded and Linux device driver in the laboratory environment intelligent monitoring system.

III. THE SYSTEM TOPOLOGY

The general framework of the Laboratory Intelligent Monitoring System is divided into three levels from low to high which are ARM front-end machine and its peripheral equipment, PC intelligent monitoring center and remote client terminal.

A. Embedded front-end machine

Embedded front-end machine use Samsung's S3C2440 ARM processor as the main controller, the performance and frequency of which are suitable for real-time video image capture and processing applications. The system hardware architecture is shown in Figure 1.

QT BASICS

Qt [pronounced 'cute'] is a cross-platform application framework that is widely used for developing application software with a graphical user interface (GUI) (in which cases Qt is classified as a widget toolkit), and also used for developing non-GUI programs such as command-line tools and consoles for servers. Qt is most notably

Qt uses standard C++ but makes extensive use of a special code generator (called the *Meta Object Compiler*, or *moc*) together with several macros to enrich the language. Qt can also be used in several other programming languages via language bindings. It runs on the major desktop platforms and some of the mobile platforms. It has extensive internationalization support. Non-GUI features include SQL database access, XML parsing, thread management, network support, and a unified cross-platform application programming interface (API) for file handling.

Distributed under the terms of the GNU Lesser General Public License (among others), Qt is free and open source software. All editions support many compilers, including the GCC C++ compiler and the Visual Studio suite.

Qt is developed by an open source project, the Qt Project, involving developers as individuals and from firms working to advance Qt, such as Nokia, Digia, and others. Before the launch of the Qt Project, it was produced by Nokia's Qt Development Frameworks division, which came into being after Nokia's acquisition of the Norwegian company Trolltech, the original producer of Qt. In February 2011 Nokia announced its decision to drop Symbian technologies and base their future smart phones on Microsoft platform instead. One month later Nokia announced the sale of Qt's commercial licensing and professional services to Digia, although Nokia was to remain the main development force behind the framework at that time. On 9 May, it was announced on the Qt Labs website that the groundwork was being laid for the next major version of Qt, with the expectation that Qt 5 would be released in August 2012.

On August 9, 2012, Digia acquired Qt software technologies from Nokia. About 125 Qt developers will be transferred to Digia, with the immediate goal of bringing Qt support to android, iOS and Windows 8 platforms.

Qt Creator IDE:

Qt Creator IDE is a tool for creating Qt applications that eliminates the need for operating system or device emulators. The tool has also been implemented efficiently to minimise download size as well as processor and disk space requirements. The Qt Creator IDE is easy to install and the IDE enables developers to create Qt applications quickly and easily.

Qt tools:

Qt is supplied with several command line and graphical tools to simplify and speed up the development process. Each tool is listed here with a link to its documentation.

- Qt Designer: Create forms visually.
- Qt Assistant: Quickly find the help you need.
- Qt Linguist, lupdate, lrelease: Translate applications to reach international markets.
- qmake: Create makefiles from simple platform-independent project (.pro) files.
- Meta-Object Compiler (moc): Generate meta-object information for QObject subclasses.
- User Interface Compiler (uic): Generate C++ code from user interface files.
- Resource Compiler (rcc): Embed resources into Qt applications during the build process.
- Configuring Qt (qtconfig): X11-based Qt configuration tool with online help.
- Examples and Demos Launcher: A launcher for Qt's Examples and Demonstration programs for Platforms.
- qt3to4 - The Qt 3 to 4 Porting Tool: A tool to assist in porting applications from Qt 3 to Qt 4. (Please note: Code ported from Qt 3 to Qt 4 will not be supported on Qt for Symbian).
- QtDBus XML compiler (qdbusxml2cpp): A tool to convert D-Bus interface descriptions to C++ source code.
- D-Bus Viewer: A tool for examining D-Bus objects and messages.

Qt terms:

1. Widgets: UI components (buttons, message boxes, application windows)

2. Layout Manager: Automatic positioning and resizing of child widgets.

3. Signals and Slots: Inter-object communication.

4. Events: System events (mouse clicks, keyboard ...)

5. Actions: e.g., save action used in tool bar and menu.

Qt Architecture:

- Qt uses native styles to draw UI
- Widgets emulate exact look & feel can be adapted by the developer
- Built on low level APIs of platform
- MFC, Motif, Layered toolkit with thin wrappers. Less performance, less flexibility
- Cross-platform
- Single source for multiple platforms Only requires recompilation

Advantage of Qt:

- **Target multiple platforms from a single source**
- Shorter development time – faster time to market
- Reduced maintenance expense
- Avoid OS-subgroups in development organization

- Enjoy true platform independence
- Target a new platform in weeks, not months
- Rapidly respond to evolving market requirements
- Remain insulated from platform changes
- Qt is actively maintained and developed to support all new mainstream OS variants
- Focus development efforts instead on value-adding innovation
- Qt delivers real, lasting competitive advantage
- Qt increases the productivity of developers by making C++ programming faster, easier and more intuitive
- Qt development tools eliminate common bottlenecks in the development process
- GUI Design & Layout - Qt Designer
- Translation/Localization - Qt Linguist
- Documentation - Qt Assistant
- Cross-platform build system – qmake
- Qt delivers true platform freedom – targeting a new platform is measured in days or weeks, not months or years
- One source code base means less maintenance time and expense – multiplying results of development efforts
- Full access to complete source code on all platforms enables development teams to adapt and extend Qt to meet their unique needs, expediting the development process.

III. THE DESIGN OF GUI AND DEVICE DRIVER OF THE LAB ENVIRONMENT INTELLIGENT MONITORING SYSTEM

The design of GUI for embedded systems is different from that of traditional data computing class software, which often handles mouse or keyboard events to complete a specific calculation, while the former mostly handle events caused by touch screen and other kinds of external devices. Because the embedded systems is resource-constrained, the design mode of the GUI of the traditional PC ,the memory consumption of which is relatively large and take up more CPU time, is not suitable for embedded systems. The lab intelligent monitoring system studied in this paper uses Qt / Embedded under embedded Linux as its GUI development platform, which can fully satisfy the restriction of embedded system resources. The application development framework of Qt / Embedded is shown in Figure 2.

As QT uses C + + as its programming language, it can implement hybrid programming with linux-C. The header files include both QT-API library and linux system calls libraries. Write the linux system calls as parts of the slots functions which can respond to specific signals in order to achieve the combination of Qt / Embedded and linux-C.

Of course, to achieve reading and writing of a specific device file, there must be device drivers which provide reading and writing operation interface functions. Therefore, we need to complete the preparing, configuring and modifying of the drivers of sensors, cameras and other external expansion device of S3C2440 microprocessor.

The Laboratory Intelligent Monitoring System uses QT to complete GUI on the ARM head-end machine to achieve the graphical display of data collected by a variety of sensors. This article focuses on elaborating the design of the linux drivers of various types of sensors and qtopia application in the system.

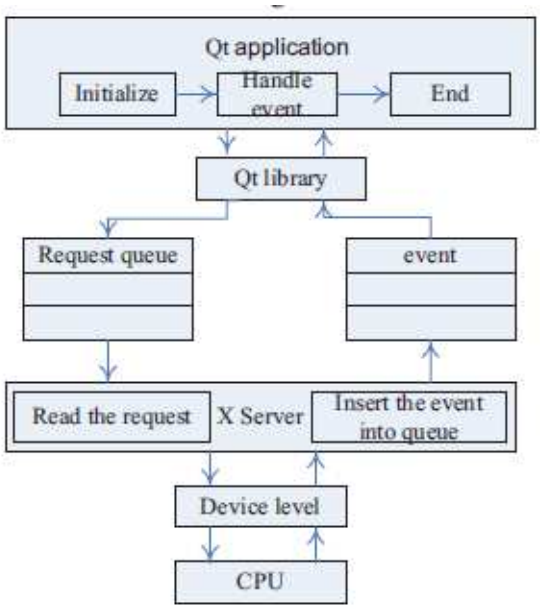


Figure 2. The application development framework of Qt / Embedded

A. Initialization

In the Qt application, firstly a QApplication object is created in main.cpp which is in charge of the main settings and flow controlling of the graphical user interface. Similarly, in the Qtopia a QPEApplication object is created to handle and schedule the events from the system and other source files, including the initialization and the end of the application.

B. Create components

With the help of QT designer ,the programmer can quickly develop relevant GUI components and adjust the size and position, including functions such as displaying the current temperature, humidity, concentrations of carbon dioxide and harmful gases concentration in the laboratory environment and showing whether the infrared sensors open or not. And then define the signals and slots functions, save it as Ui file. The final GUI of lab environment intelligent monitoring system is shown in Figure 3.

C. Event handling

First of all, device driver modules for sensors, camera and so on need to be dynamically loaded into embedded linux operating system kernel which runs on the front-end machines ,providing interface functions of initializing , reading and writing for linux device file so that event handling based on the system time and key can be accomplished in embedded GUI .

As to the display of the data collected by a variety of sensors through the embedded GUI, QTimer function can be called to automatically update the environmental monitoring value in constant time interval. The process of event handling of various types of sensors is shown in Figure 3.

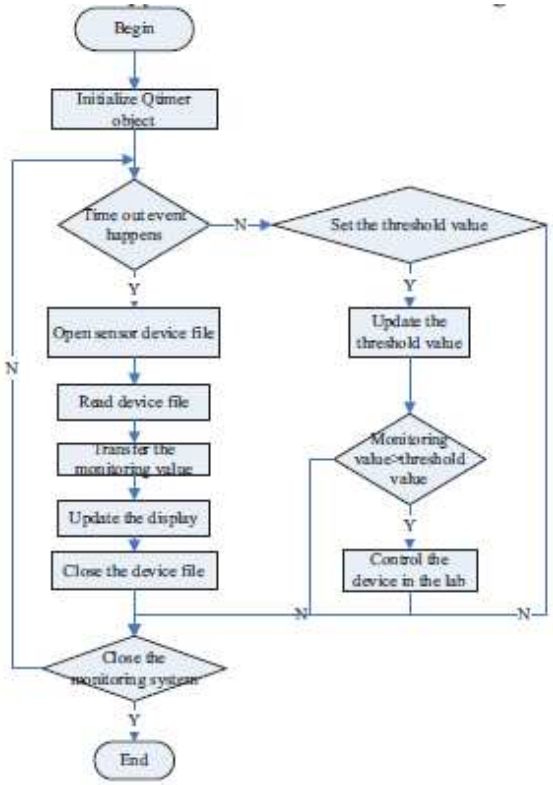


Figure 3 . The process of event handling of various types of sensors

The statement QTimer timer creates an object and then call start () method in the constructor to specify the time-out period. When the timeout event occurs, all kinds of data updating functions handleSensor () will be automatically called.

```
void ILMS::ILMS()
{
connect(&timer,SIGNAL(timeout()),this,
SLOT(handleSensor () ) );
timer.start(180*1000);
/ / the other parts of the constructor definition are not
listed
}
```

handleSensor ()
will update the current environmental monitoring data of the sensors, which requires the driven interface functions of sensors.

D. The design of Linux device driver for sensors

Various types of sensors have their own specific device drivers due to the different working principles. Take Digital temperature sensor as an example to elaborate the design method of Linux device driver for sensors. Function handleTemperature () mainly uses the digital temperature sensor driver function read (), which is renamed form the function BYTE DS18b20_read_byte (void) . static struct file_operations s3c2440_18b20_fops =

```
{
. owner = THIS_MODULE,
. read = s3c2440_18b20_read,
};
```

The prototype of this function is DS18b20_read_byte, providing a reading method for DS18b20 device file.

```
BYTE DS18b20_read_byte (void)
{
BYTE i = 0;
BYTE byte = 0;
for (i = 0; i < 8; i++)
{
s3c2440_gpio_cfgpin(DS18B20_PIN,
DS18B20_PIN_OUTP);
s3c2440_gpio_setpin(DS18B20_PIN, LOW);
udelay(1);
byte >>= 1;
s3c2440_gpio_setpin(DS18B20_PIN, HIGH);
s3c2440_gpio_cfgpin(DS18B20_PIN,
DS18B20_PIN_INP);
if(s3c2440_gpio_getpin(DS18B20_PIN)) byte |=
0x80;
udelay(60);
}
return byte;
}
```

The read () function returns the byte stream buf which contains the LS byte and MS byte corresponding to buf [0] and buf [1]. The value of former four bits of buf [0] is 2-4 ~ 2-1.In the laboratory environment, the general accuracy of temperature measurement can be negligible. The former our bits of buf [1] are sign flags, while we just use bit11 as the sign flag. Move buf [0] to the right four bits and then add it with the value of buf [1] ,we will get the current temperature of the laboratory environment.

In addition, system drivers also provide reset(), write(), proc() functions for DS18B20 which is available for the calls of other applications

IV SENSORS

A **sensor** is a device that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument. They are used for various purposes including measurement or information transfer.

An electronic sensor is any device that uses electricity to sense a change in physical quantity, and then through a voltage change, send a signal to a device that captures this information. Some sensors measure properties directly, other sensors measure properties indirectly, using conversions or calculations to determine results. Sensors are generally categorized by the type of phenomenon that they measure, rather than the functionality of the sensor itself.

There are many different things to measure -- heat, light, humidity, sound, level, weight etc. each of these requires a different sensors. There are so many kinds of sensors.

MECHANICAL SENSORS:

Mechanical sensors measure a property through mechanical means, although the measurement itself may be collected electronically. An example of a mechanical sensor is a strain gauge. The strain gauge measures the physical deformation of a component by experiencing the same strain as the component, yet the change in resistance of the strain gauge is measured electrically. Other types of mechanical sensors include:

- Pressure sensors
- Accelerometers
- Potentiometers
- Gas and fluid flow meters
- Humidity sensors

ELECTRICAL:

Electrical sensors measure electric and magnetic properties. An example of an electrical sensor is an ohmmeter, which is used to measure electrical resistance between two points in a circuit. An ohmmeter sends a fixed voltage through one probe, and measures the returning voltage through a second probe. The drop in voltage is proportional to the resistance, as dictated by Ohm's Law. Other electrical sensors include:

- Voltmeter/Ammeter
- Metal detector
- RADAR
- Magnetometer

THERMAL:

Although all thermal sensors measure changes in temperature, there are a variety of types of thermal sensors, each with specific uses, temperature ranges, and accuracies. Some types of thermal sensors include:

- Thermometers
- Thermocouples
- Thermistors
- Bi-metal thermometers

OPTICAL:

Optical sensors detect the presence of light waves. This could include light in the visible spectrum, or outside the visible spectrum, in the case of infrared sensors. Some types of optical sensors include:

- Photo detectors
- Infrared sensors
- Fiber optic sensors
- Interferometers

OTHER TYPES OF SENSORS:

There are many other types of sensors:

- Radiation sensors, including Geiger counters and dosimeters
- Motion sensors, including radar guns ,Infrared detectors and speedometers
- Acoustic, including sonar and seismometers
- Gyroscopes
- Microphones
- Video cameras
- Hall Effect probes (magnetic field)
- Remote control devices
- Photocells

Sensors may be simple physical measurement systems, or complex electronic devices requiring sophisticated data acquisition systems. No matter the type of sensor, input type, or output type, every sensor has inherent characteristics that allow the user to select the right sensor for the task at hand.

SENSOR CHARACTERISTICS:

Some sensor characteristics include:

- Input Range
- Output Range
- Accuracy
- Repeatability
- Resolution

INPUT RANGE:

Input range is the maximum measurable range that the sensor can accurately measure. For example, a compression load cell may have an input range of 0 - 5000 pounds. The load cell cannot accurately measure "negative", or tensile loads, or compressive loads greater than 5000 pounds. Generally, quantities outside of the input range can be measured, but characteristics such as accuracy and repeatability may be compromised when the input is outside of the specified range.

OUTPUT RANGE:

Output range generally refers to electronic sensors, and is the range of electrical output signal that the sensor returns. However, the output range could be a physical displacement, such as in a spring scale, or rotation, such as in a clock-style analog thermometer. The output range is related to the input range by the conversion algorithm specific to the sensor type, and the algorithm may include factors based on the calibration of the specific sensor.

ACCURACY:

Accuracy actually refers to the amount of error, or inaccuracy that may be present in a sensor. Accuracy can be stated as a unit of measurement, such as +/- 5 pounds, or as a percentage, such as 95%. In most cases, increased accuracy results in an increased cost for a sensor.

REPEATABILITY:

Repeatability, as the name implies, refers to how often a sensor under the same input conditions will return the same value. If a sensor is designed to be used over and over again, it is important that the output value is accurate over every measurement cycle for the life of the sensor. Repeatability is determined by calibration testing of the sensor using known inputs.

RESOLUTION:

Resolution is the smallest unit of measurement that the sensor can accurately measure. Some transducers return output signals in discrete steps, and therefore the resolution is easily defined. Resolution can be stated as a unit of measurement or as a percentage. For electronic sensors, resolution is also dictated by the resolution of the signal conditioning hardware or software.

These qualities are common to all sensors, no matter what characteristic is being measured. All of these traits must be considered when selecting the right sensor for the specific needs of a test.

V. RESULTS

VI. CONCLUSION

In the post-PC era, the embedded system technology develops rapidly and the design of embedded GUI and the linux device drivers are important and indispensable components of it. This paper focuses on solving the issues of poor real time, high cost, low precision and incapability of determining whether the lab environment is in line with the body's health indicators in the laboratory management of domestic institutions of higher learning. It develops a laboratory intelligent monitoring system with S3C2440 microprocessor as its main controller, elaborating the difficult points of the development of the GUI applications based on Qt / Embedded and Linux drivers for various types of sensors in the project. With a perfect support of the embedded system technology, we believe that the intelligent monitoring system will have better performance and broader market prospect.

REFERENCES

[1] Samsung Electronics Co Ltd. Users' Manual S3C2440A V0.12 [M] . March,2004.

[2] Yun Sin-quan, Lu Qiang, Qian Pei-del. One implementation of Linux application based on Qt / Embedded [J]. Computer Application and Software, 2006,23 (2): 105-107.

[3] Trolltech. Online Reference Documentation [EB / OL]. Http://doc. Trolltech. Com /.

[4] Chen Kun, Chen Yun-qiu, Liu Xin. Application design based on Qt / Embedded and embedded Linux [J]. Computer and Digital Engineering, 2009,37 (1): 156-161

Author introduction: Liu Yang, male, born in Jinzhou ,Hubei province. He comes from Software College of Northeastern University, the main research direction is embedded systems.

Acknowledgement: National Ministry of Education College Students Innovative Experiment funded projects (090164)

N.Sai Jithendra did B.Tech in Electrical and Electronics Engineering from JNTU Kakinada and pursuing M.Tech in Embedded systems from Holy Mary institute of Technology and science, JNTU Hyderabad. His interested areas are Embedded systems.

D.Aruna Kumari is Completed M.Tech in ECE, JNTU Kakinada. B. Tech in Electronics and Communication Engineering from Bapatla Engineering college, Nagarjuna University, Guntur, Currently working as Asst.Professor in the ECE Department at Holy Mary Institute of Technology **and** Science (College of Engineering), Hyderabad. Her interested areas are EMTL, communication, Antinas, Microwaves.