



UNIVERSITY OF MISSOURI – COLUMBIA
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
"A PROUD TRADITION, ENGINEERING THE FUTURE!"

Automation Distance Sensitive Security Camera

ECE 4220
FINAL PROJECT

May 15, 2015

— Author —

Junkai Cai, Computer Engineering

— Course Instructor —

Mr. Luis A. Rivera Estrada

Abstract

ECE 4220 course site on Blackboard.

Index Terms — *Database systems, Digital circuits, Distance measurement, Embedded software, Interrupters, Microprogramming, Sensor systems, Software engineering, Webcams*

Table of Contents

Abstract	i
Introduction	6
Background	7
Security system with method for locatable portable electronic camera image transmission to a remote receiver (US 6181373 B1)	7
Implementation	8
System description	8
Hardware Discription.....	8
Software Description	9
Constraints	13
Implement Method.....	14
FairCom Database Server	14
HC-SR04 Ultrasonic Sensor	15
Creative Live! Cam Chat HD	16
TS-7250 Single Board Computers	17
User Space Receiver	19
Experiments and Results.....	20
Experiments	20
Results	23
Circuit Overview.....	23
FairCom Program with Distance Receiver	24
FairCom Program for Displaying Records	25
Toggling GPIOAIntType2 in Interrupt Handler	26
Configure as Rising Edge Sensitive.....	27
Configure as Falling Edge Sensitive	28
Reading HC-SR04 sensor by Arduino	29
Camera Image on Laptop.....	30
Discussion and Conclusions	31
FairCom Program with Distance Receiver	31
FairCom Program for Displaying Records	31

Toggling GPIOIntType2 in Interrupt Handler	31
Configure as Rising Edge Sensitive.....	32
Configure as Falling Edge Sensitive	33
Reading HC-SR04 sensor by Arduino	33
Camera Image on Laptop	33
Conclusion	34
Appendices (Code).....	36
Reading Sensor Kernel	36
FairCom Server with Distance Receiver and Comparer	40
FairCom Server Displaying Program.....	53
Reading Sensor by Arduino	61
Reading Image from Webcam by OpenCV	63
Bibliography	65

Table of Figures

Figure 1: User Control Hardware Functional Block Diagram.....	8
Figure 2 General Flow Chart	9
Figure 3 Kernel Module Flow Chart.....	10
Figure 4 User Space Receiver Flow Chart.....	11
Figure 5 FairCom Server Flow Chart	12
Figure 6 HC-SR04 Ultrasonic Sensor.....	15
Figure 7 Circuit Overview	23
Figure 8 Result of FinalProjectFairComRead.....	24
Figure 9 Result of FinalProjectFairComDisplay	25
Figure 10 Kernel Message Buffer when Toggling GPIOIntType2	26
Figure 11 Kernel Message Buffer with Rising Edge Sensitive Interrupt	27
Figure 12 Kernel Message Buffer with Falling Edge Sensitive Interrupt	28
Figure 13 Result of SensorReading by Arduino	29
Figure 14 Camera Image on Laptop before Moving	30
Figure 15 Camera Image on Laptop after Moving	30

Table of Tables

Table 1 Software Block Diagram	9
Table 2 c-treeDB Equivalent Date Type.....	14
Table 3 TS-7250 LCD Header Signals	17
Table 4 GPIOA Register Map (partial).....	18

Introduction

Closed-circuit television (CCTV) security system has a widely use all over the world. It is an effective way to reduce the burglary rate. But crowded human traffic is not always true in some public area but we sometimes still need security camera for these places. If we use traditional CCTV, it will take up a huge server disk space every day for a high-quality 24/7 video recording requirement. Since there will never have a huge human traffic in some area, we can take photo when illegal action is happening instead of recording video. The illegal actions mentioned here include opening door, protected stuff being stealing, or people entering prohibited area.

To detect those illegal action, a distance sensor HC-SR04 will be connected to microprocessor, TS-7250 board. When the distance value return by the sensor exceed a certain limit difference than previous data, the camera connected to TS-7250 board will be triggered and take pictures. The pictures will be save on FairCom Database server, which runs also runs on the TS-7250 board, for future use. Also the time stamp of photo should also be recorded. Administrator can access the file system and pull up pictures took by the camera for security purpose. And the recorded photo can also be used for face recognition to identify suspect. The recorded timestamps will help administrator to locate the CCTV footage period for the whole area (or entrance of building) he/she want to inspect if the picture information is not enough for identify an illegal behavior.

This system can be applied in laboratory or power distribution room to monitor the entrance. And it can also work in museum for anti-theft purpose by putting the sensor directly towards protected items. Once the items is moving, the camera will be triggered.

Background

Security system with method for locatable portable electronic camera image transmission to a remote receiver (US 6181373 B1)

“A portable security system consisting of an electronic camera module with a lens, a Global Positioning System module and a combined memory and transmitter module which includes a microphone. When sound-activated via microphone or activated by manual operation of the button, the camera module photographs the scene and feeds the image to the transmitter module, also the Global Positioning System module receives positional information for satellites and converts the received information into display information, and the transmitter module transmits the image and the display information to a remote receiver for conversion into a printed image on a facsimile receiver and into a display of the transmitter's position on a map.”
("Security system with method for locatable portable electronic camera image transmission to a remote receiver." Abstract).

The patent mentioned about is a portable system with a GPS module. For a portable design, it also has a remote transmitting part. The camera module will action based on sound collected by the microphone module, which means it detect sound event as an illegal behavior.

The design in this final project is more specific to small in-door area. A certain degree of distance changing is regarded as illegal behavior. It is non-portable so it don't need a GPS module to record location information and a transmit module to transfer image to control unit. The designed system in this project can be used in a laboratory or power distribution room, which may be a noisy place even without human, while the patent mentioned about is not capable in a noisy location as it is a sound based detection security system.

Implementation

System description

Hardware Discription

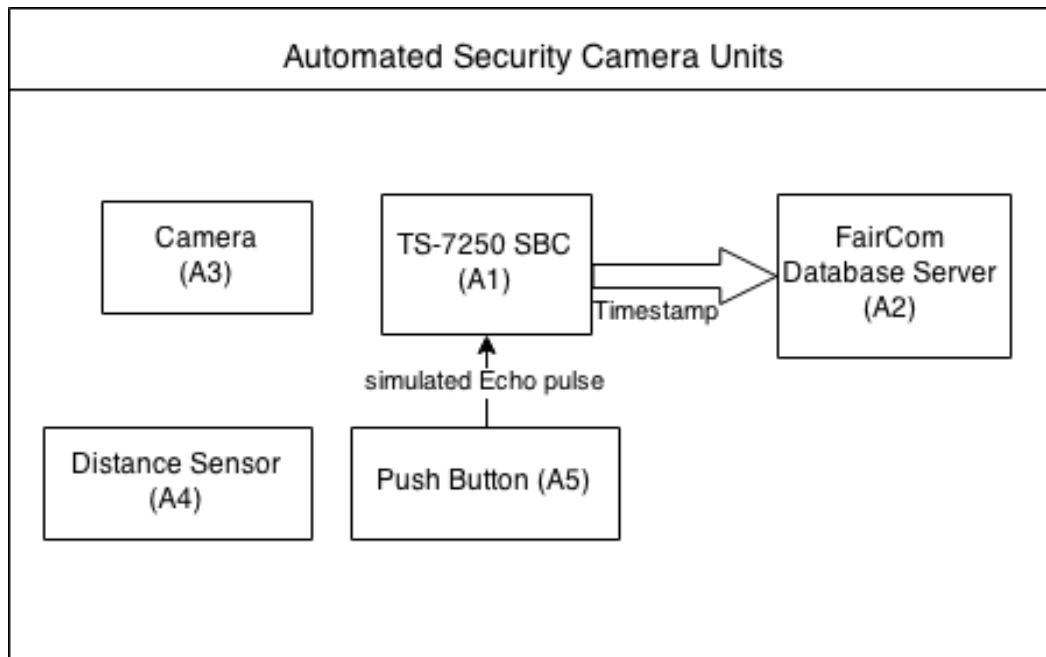


Figure 1: User Control Hardware Functional Block Diagram

Based on what mention in Constraints section, a push button is used to simulate Echo pulse on Distance Sensor so it is connected to TS-7250 board. The camera part is working separately on laptop and the distance sensor part is working separately with Arduino, so they are not connected with TS-7250 board. When a simulated pulse width come and the program find the distance change exceed the threshold a timestamp will be stored into the database.

Software Description

Software Block Diagram

Table 1 Software Block Diagram

Microprocessor			FairCom Database
Camera Controller	Distance Reader	Distance Comparer	Recording timestamp

Four main part of software are designed. Camera Controller is in charge of controlling camera and reading image. Distance Reader is in charge of reading distance data from kernel model. Distance Comparer is in charge of comparing new distance with previous distance to see if the distance change exceed the threshold. If yes, a timestamp will be storing in FairCom database, which is take over by recording timestamp part. Implementation details will be discussed in Implement Method section.

General Flow chart

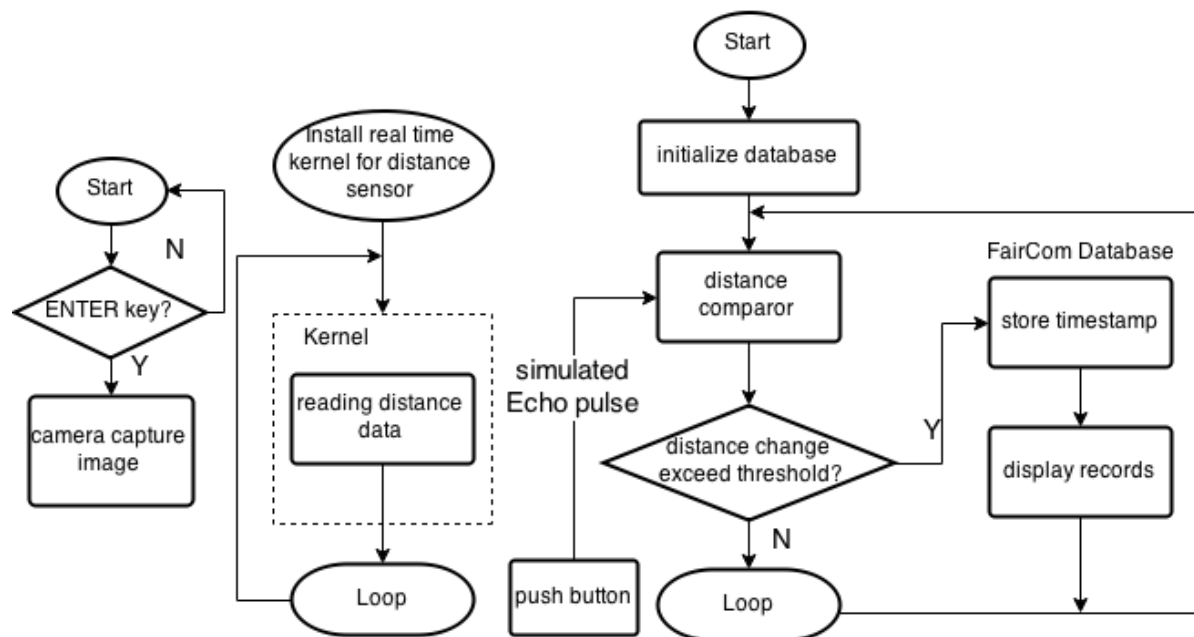


Figure 2 General Flow Chart

This flow chart shows a general software architecture for the whole system. Blocks without arrow are separate and there is no communication between them.

Kernel Module Flow chart

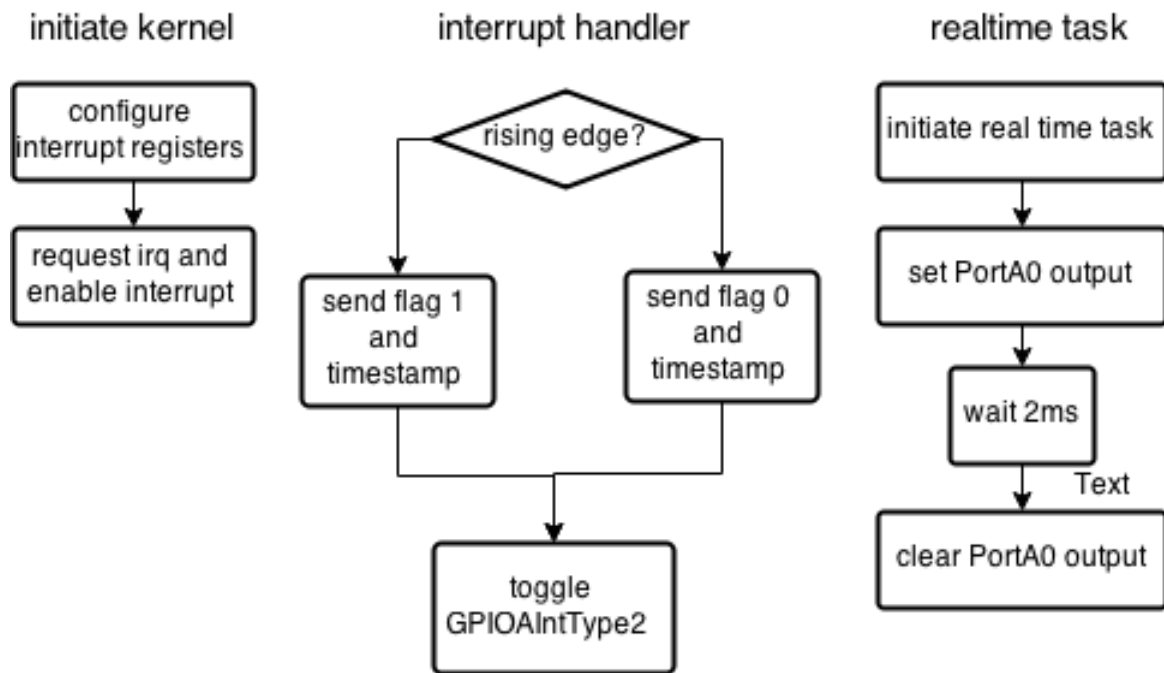


Figure 3 Kernel Module Flow Chart

The real-time task is generating the trigger pulse on PortA0, and PortA1 is configured to enable interrupt to measure the pulse width. To correctly use PortA, some register needs to be configured.

Bit 0 of PADDR is set so it is output trigger pulse to “Trig” pin of sensor, while bit 1 of PADDR is clear so it can read pulse from “Echo” pin of sensor. GPIOIntType1 register is set at bit 1 so the interrupt type is edge sensitive. GPIOADB register is set on bit 1 so it can support debounce requirement. GPIOIntType2 is set at bit 1 when the kernel module is initiated so the PortA1 will generate interrupt at a rising edge. In the interrupt handler, GPIOIntType2 is toggled so the interrupt polarity is switching between rising edge and falling edge so we can measure the pulse width. Remember we must clear the interrupt in the interrupt handler. Also, interrupt should be cleared, released, and disabled when removing the module.

User Space Receiver Flow Chart

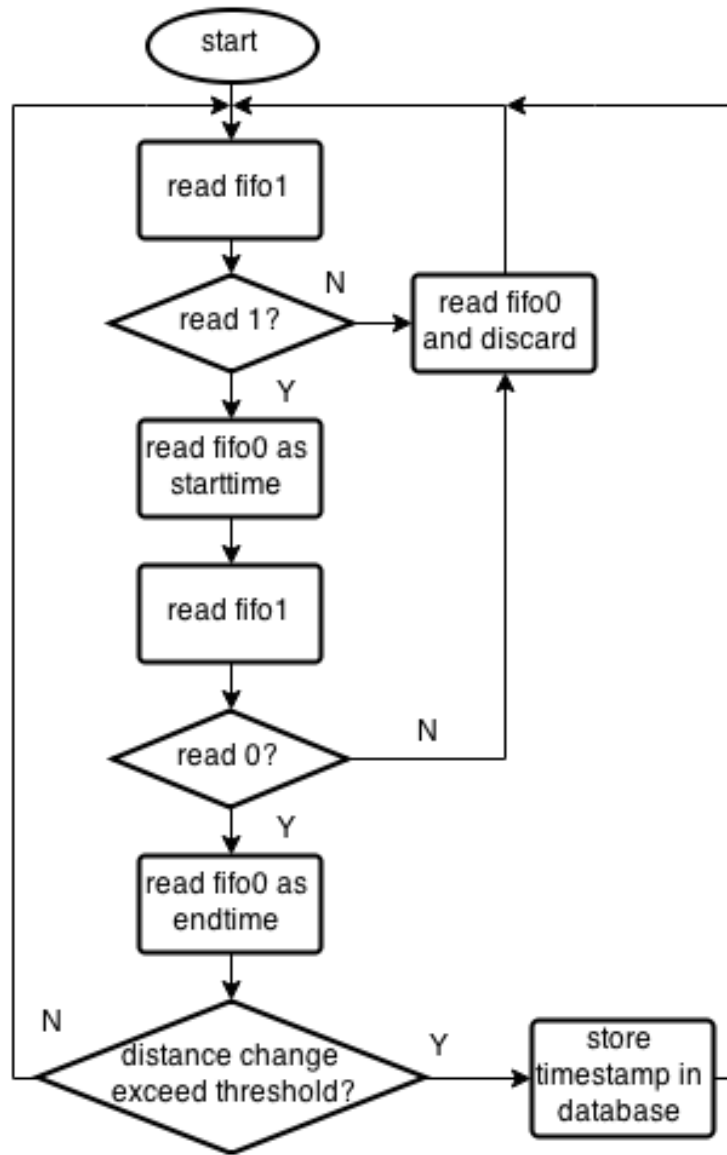


Figure 4 User Space Receiver Flow Chart

The receiver on user space is in charge of reading distance and comparing distance. It will check the flag in fifo1 first. If it read a falling edge (flag = 0) in fifo1, it means an error has occurred and falling edge trigger the interrupt first. So the process should discard first data in fifo0. It could only accept and calculate distance by start time and end time if the fifo1 has a “1”, “0” sequence. This assure the accuracy of the distance reading program.

FairCom Server Flow Chart

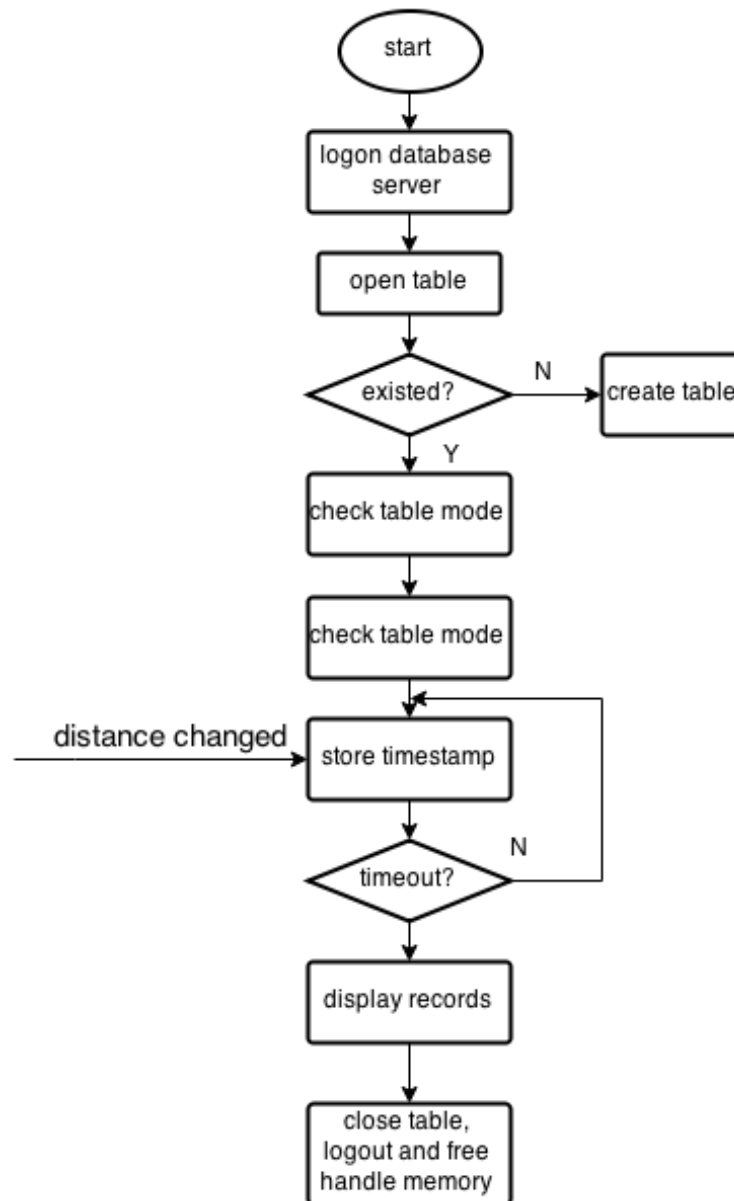


Figure 5 FairCom Server Flow Chart

Before start the FairCom program, the server should be started manually. After successfully logon the database session, handlers is allocated to maintain table, field and record. Record handler is used to store timestamp and read data for displaying. When the program ends, we must close table, session and database (if you use) then logout. Memory allocate for the handler also must be free.

Constraints

- Linux kernel on TS-7250 board is version 2.4, which is too old to support most of the nowadays USB devices. The Linux UVC (USB video class) driver is only included in the kernel later than 2.6, which can support most of the USB Webcam. This means I cannot connect the Webcam to the TS-7250 board by USB directly and read image from it. An alternative way is to connect to camera to another computer and send image and control signal by network or serial port.
- Student cannot install OpenCV or CImage library on workstations in Lab 1246 because we are not permitted to modify the system.
- My laptop doesn't come with a COM port for serial communication. So I can install related library on my laptop but cannot use serial communication to connect TS-7250 board.
- Student could only access TS-7250 board through nfs1 server. And the TS-7250 boards are fixed in Lab 1246. All TS-7250 board and nfs1 are in the same LAN, which is separate from most of LAN in engineering building except the one that PC in Lab 1246 is connected on. This means we could only access the nfs1 server and TS-7250 board by laptops connected to MizzouWireless or plug the Ethernet cable in Lab 1246 to our laptop, which need IP configure and permission of Lab manager.
- MizzouWireless is a WLAN with firewall. Based on network knowledge, we know usually a WLAN router will allocate IP address to all connected device in the subnet and the IP address is not global public. So if we send a message to establish connect with nfs1 server and TS-7250 board, the TS-7250 cannot get right IP address to send control signal back.

- TS-7250 comes with a processor that is not fast enough to detect rising edge or high level of a short pulse to generate interrupt, which means the HC-SR04 cannot work directly with the board. It could only also send data to an external computer and communicate with TS-7250 board. But also limited by constraints mentioned above. Meanwhile, the simulated pulse from push button is slow enough to trigger interrupt.

Implement Method

FairCom Database Server

FairCom Database based on c-tree and c-treeACE cross-platform database engine. This project utilizes its c-treeDB API to develop. c-treeDB offers a method of database creation and maintenance that is easier to use than the traditional c-treeACE ISAM and low-level APIs. The API provides access to sessions, databases, tables, records, fields.

Table 2 c-treeDB Equivalent Date Type

c-treeDB Field Type	c-treeACE Field Type	Equivalent Data Type	Implementation
<i>CT_DATE</i>	<i>CT_DATE</i>	<i>CTDATE</i>	Unsigned four-byte integer interpreted as date.
<i>CT_TIME</i>	<i>CT_TIME</i>	<i>CTTIME</i>	Unsigned four-byte integer interpreted as time.
<i>CT_VARCHAR</i> or <i>CT_LVC</i>	<i>CT_STRING</i>	<i>pTEXT</i>	Varying length field delimited data. Variable length string data.

FairCom Database is designed to be able to store a variety of data types. In this project, two data types are supposed to be stored into the database. When the distance value exceeds the threshold, a timestamp is recorded by calling c-treeDB function `ctdbCurrentDate()` and `ctdbCurrentTime()`; Date and time are stored separately as two fields in one record. Based on support from FairCom VP Engineer Randal Hoff, JPG and PNG images can be stored by *CT_VARCHAR* field type.

Based on what mentioned in Constraints section, the image storing part is removed from the project.

According to the Table 2 c-treeDB Equivalent Date Type, the variable/data type is different from field type. For example, if you set a field in the table as type “CT_DATE”, the variable type you declare in C code should be “CTDATE”. Data type “pTEXT” means a pointer to data type “TEXT”, which actually is char data type in C. For displaying purpose, CTIME and CTDAET are converted to string by ctdbDateToString() and ctdbTimeToString() function. So actually the timestamps are stored as CT_STRING field type, which shows as CT_VARCHAR in c-treeDB.

The table of database is stored on local disk. So a program is designed separately to open exist database and display records.

HC-SR04 Ultrasonic Sensor

HC - SR04 sensor provides 2cm - 400cm distance measurement function, which can reach to 3 mm ranging accuracy. The modules includes ultrasonic transmitters, receiver and control circuit.



Figure 6 HC-SR04 Ultrasonic Sensor

Basic principle of work:

- (1) Connected “Trig” pin to TS-7250 board 5V DIO pin and trigger for at least 10us high level signal;
- (2) TS-7250 automatically sends eight 40 kHz to measure the range of obstacle.
- (3) If the ranging signal back, a 5V high level pulse between 150us and 25ms will be generated on “Echo” pin. The duration of high level is the time from sending ultrasonic to returning. A 38ms pulse will be generate if no obstacle.
- (4) Convert the time to distance by simple mathematic calculation. For example, distance in centimeter equals to $\text{Time}(\mu\text{s}) / 58$.

Based on what mention in Constraints section, the TS-7250 board cannot detect such a fast rising edge and generate interrupt (Also, it cannot detect such a fast high level), a push button is used to simulate Echo signal. When pushed, it output high level signal. Meanwhile, the HC-SR04 sensor is connected to Arduino board and we can see correct distance on serial monitor with the Arduino IDE.

Creative Live! Cam Chat HD

Creative Live! Cam Chat HD is a 5.7MP USB Webcam. It is able to record HD 720p video up to 30 frames per second or capture image.

After connecting the camera to the TS-7250 board with USB, OpenCV or CImage library are using to access the camera and read image. For the software capability, it is possible to record a video. But considering disk usage and the purpose of this project, it just read image frame when control signal comes. The image data should be stored in FairCom database with date and time in a record.

Based on what mention in Constraints section, the camera cannot run on the TS-7250 board because the Linux kernel on the TS-7250 board is too old (version 2.4) to support most of

nowadays Webcam. So the camera cannot connect with the board directly. So the image is captured on PC and doesn't be sent and store in FairCom database.

TS-7250 Single Board Computers

The TS-7250 Single Board Computers (SBC's) run on a 200 MHz ARM9 processor with power as low as 1/2 Watt. It provides 20 Digital Input/Output lines and two USB 2.0 interfaces for the user. The EP9302 interrupt controller allows up to 54 interrupts to generate an Interrupt Request (IRQ). So it is supposed to read the pulse width from Echo pin of the HC-SR04 Ultrasonic Sensor.

Connecter with HC-SR04

Table 3 TS-7250 LCD Header Signals

PIN	Function	Comments
1	LCD 5V	LCD Power
2	LCD_GND	
3	LCD_RS	Register select
4	Bias	620 Ohm to GND
5	LCD_EN	Active high enable
6	LCD_WR#	Active low write
7	LCD_D1	D0 - D7: Buffered bi-directional data bus connected with Port A of EP9302
8	LCD_D0	
9	LCD_D3	
10	LCD_D2	
11	LCD_D5	
12	LCD_D4	
13	LCD_D7	
14	LCD_D6	

The HC_SR04 sensor is running on 5V, while the DIO1 pin, which contains Port bit 0 to 7 and PortF bit 1 on TS-7250 board is running on 3.3V. So we can't use DIO1 pin to connect

HC_SR04 module. Since all pins on LCD header are running on 5V except pins 3, 5, 6 and PortA bit 0 to 7 are also locate at LCD header, the HC-SR04 should connect with TS-7250 board through LCD header.

Related Register

Edge type interrupt is used on PortA to measure the pulse width on the “Echo” pin of HC-SR04. To read/write registers for interrupt, correct mapping is necessary before all start. Port A and B are combined into a single signal GPIOINTR which is connected to the system interrupt controller on VIO interrupt source 59.

Table 4 GPIOA Register Map (partial)

0x8084_xxxx	GPIO	GPIO Control Registers
0x8084_0000	PADR	GPIO Port A Data Register
0x8084_0010	PADDR	GPIO Port A Data Direction Register
0x8084_0090	GPIOIntType1	Register controlling type, level or edge, of interrupt generated by the pins of Port A
0x8084_0094	GPIOIntType2	Register controlling polarity, high/low or rising/falling edge, of interrupt generated by Port A
0x8084_0098	GPIOAEOI	GPIO Port A End of Interrupt Register
0x8084_009C	GPIOIntEn	Controlling the generation of interrupts by the pins of Port A
0x8084_00A8	GPIOADB	GPIO A Debounce Register

Kernel Module

To generate trigger pulse (at least 10us), a real-time task is created. The period of the trigger signal is set to 10ms and the duty cycle is 20%. When the kernel module is initiated, GPIOIntType2 is set so the first rising edge will be detect. GPIOIntType2 will be toggle in the interrupt handler so the TS-7250 board can detect next falling edge. When an edge is coming,

a timestamp will be recorded in the handler and put into fifo0, which communicate with user space FairCom Database process. Besides put timestamps in to fifo0, a flag to identify a rising or falling edge is also sent by fifo1.

Based on what mention in the Constraints section, the TS-7250 board cannot successfully detect such a fast pulse. So a push button is used to simulate “Echo” pin signal to generate a longer pulse for interrupt. So actually the trigger part has no use in the final implementation.

User Space Receiver

The user space process is supposed to read the timestamps from fifo that connected to the kernel module. The process will check fifo1 first for rising/falling edge flag then read timestamp in fifo0. After the distance is calculated, it will compare the latest distance data with the previous one. If the change exceed a certain threshold, it will record a timestamp and store it into the FairCom database.

Experiments and Results

Based on the proposed idea and assumptions. All parts should work fine with the TS-7250 board so no more inter-device communication is required. It was not going very well when I implemented camera module and sensor module. To figure out the reason, a lots of experiments were done for debugging purpose.

Experiments

- Connect the camera directly to the TS-7250 board by USB port
Failed. The Linux kernel 2.4 on TS-7250 board is too old to support USB Webcam. Can't not see usb device in the list
- Connect the camera to workstations in Lab 1246
Failed. The camera can be detected, but I cannot install OpenCV or CImage library on the system because I don't have permission
- Connect the camera to my laptop
Succeed. The camera works find on the laptop with OpenCV. After <ENTER> key was pressed, an image is capture and display on the screen.
- Trigger and read "Echo" pin of the sensor on PortA, toggle GPIOIntType2 in handler
Failed. The TS-7250 cannot read interrupt from short distance and cannot incorrectly from long distance.
- Trigger the sensor on PortA and read "Echo" pin on PortB, toggle GPIOIntType2 in handler

- Failed. The TS-7250 cannot read interrupt from short and cannot incorrectly from long distance.
- Trigger the sensor on PortA and read “Echo” pin on PortF, toggle GPIOIntType2 in handler
Failed. The TS-7250 cannot read interrupt from short and cannot incorrectly from long distance.
- Trigger and read “Echo” pin of the sensor on PortA, configure interrupt as rising edge sensitive
Failed. The TS-7250 cannot read interrupt from short and cannot incorrectly from long distance.
- Trigger the sensor on PortA and read “Echo” pin on PortB, configure interrupt as rising edge sensitive
Failed. The TS-7250 cannot read interrupt from short and cannot incorrectly from long distance.
- Trigger the sensor on PortA and read “Echo” pin on PortF, configure interrupt as rising edge sensitive
Failed. The TS-7250 cannot read interrupt from short and cannot incorrectly from long distance.
- Trigger the sensor on PortA and configure interrupt as falling edge sensitive
Succeed. Reading “Echo” pin on either PortA, PortB, PortF works fine. The interrupts were generated in correct period.
- Use function generator to trigger sensor and use oscilloscope to see pulse on “Echo” pin

Succeed. As long as the duty cycle is low (less than 30%), even a 100Hz (10ms) trigger pulse can trigger the sensor and receive correct distance varied pulse on oscilloscope.

- Trigger the sensor on PortA and use oscilloscope to check the “Echo” pin

Succeed. Correct waveforms for both “Trig” and “Echo” pin are shown on the oscilloscope.

- Use a push button to simulate “Trig” signal

Failed. The TS-7250 cannot read interrupt from a short push pulse trigger. Data also seems to be obviously wrong from a long push pulse trigger.

- Use a push button to simulate “Echo” signal

Succeed. TS-7250 board can generate interrupt correctly and put right timestamp into fifo. User space program shows different distance depends on the time push button is pushed, which corresponds to HC-SR04 datasheet.

- Connect the sensor with Arduino and send distance value to laptop

Succeed. Sensor is working pretty well. When I move the obstacle in front of the sensor.

The distance value returned change frequently.

- Store timestamps in FairCom database

Succeed. Local timestamp is recorded and parse correctly. The display function can display correct timestamp stored in database.

Results

Circuit Overview

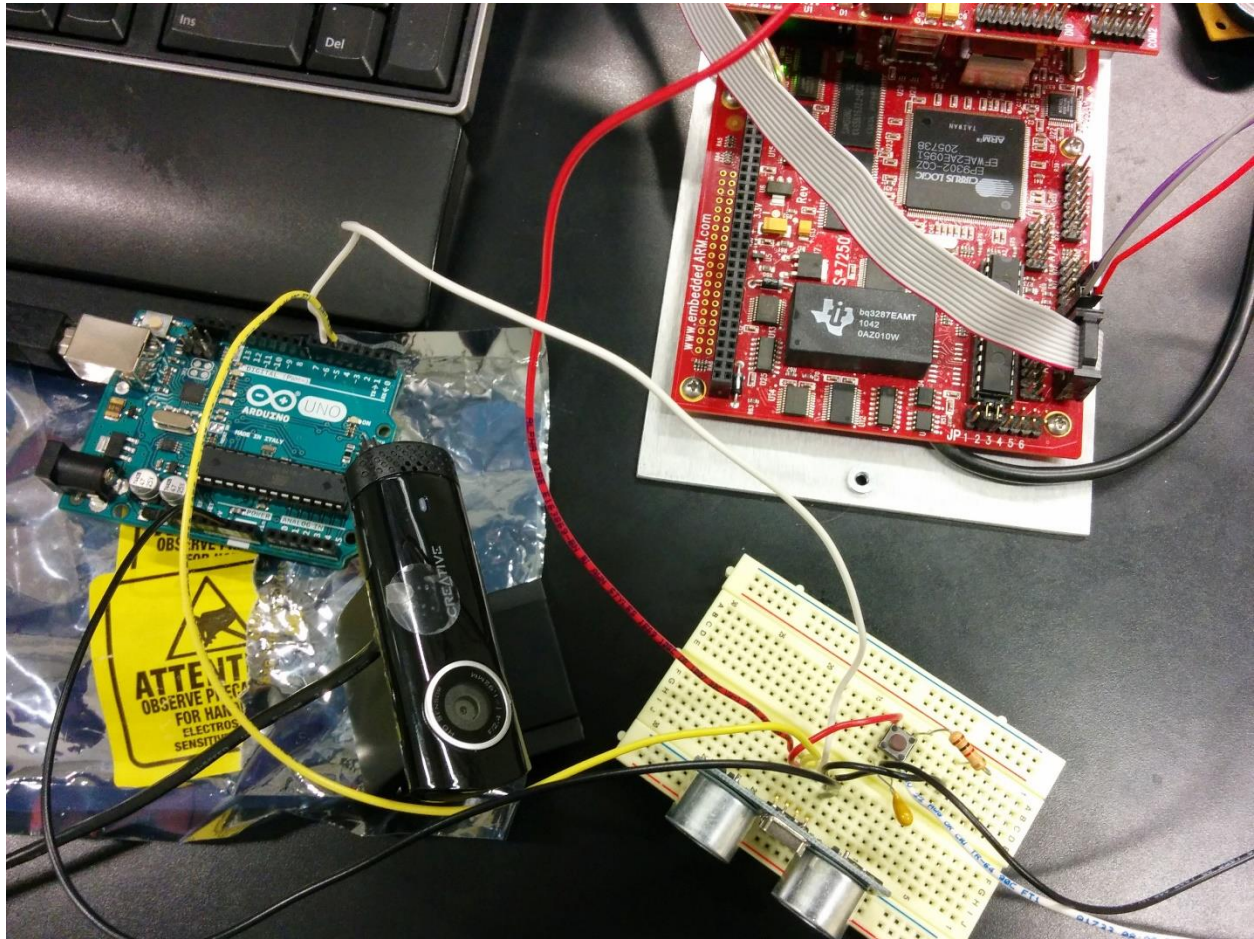


Figure 7 Circuit Overview

FairCom server runs on the TS-7250 board on ARM CPU. The HC-SR04 sensor and push button, which are on the breadboard, powered by TS-7250 board with the red cable. The white cable connected TS-7250 board is ground. Purple one connect LCD_1, which is also PORTA1 on TS-7250 board with push button output to receive simulated echo pulse. Meanwhile, “Trig” pin is connected to Arduino DIO 5 and “Echo” pin is connect to Arduino DIO6, which deal with the sensor part. The Webcam in the picture is connected to my laptop by USB 2.0 port.

FairCom Program with Distance Receiver

```
home/jcmx9/workspace/FinalProjectFairComDisplay/Release

[root@ts7250-19 /home/jcmx9/workspace/FinalProjectFairComRead/Release]# ./FinalProjectFairComRead
INIT
    Logon to server...
DEFINE
    Open table...
MANAGE
    Delete records...
Distance changes within threshold.
Distance changes within threshold.
Distance changes within threshold.
Distance changes within threshold.
Distance changes within threshold.
Distance changes within threshold.
Distance changes within threshold.
Distance changes within threshold.
Distance changes within threshold.
Distance changes within threshold.
Distance changes within threshold.
Distance changes within threshold.
Distance changes within threshold.
Distance changes within threshold.
Distance changes within threshold.
Distance Changed @
Date: 05/10/2015.
Time: 10:40:30 AM.

Distance Changed @
Date: 05/10/2015.
Time: 10:40:33 AM.

Distance changes within threshold.
Distance changes within threshold.

Distance Changed @
Date: 05/10/2015.
Time: 10:40:42 AM.

timeout...
    Display records...
        05/10/201510:40:30 AM
        05/10/201510:40:42 AM
        05/10/201510:40:33 AM
DONE
    Close table...
    Logout...

Press <ENTER> key to exit . . .

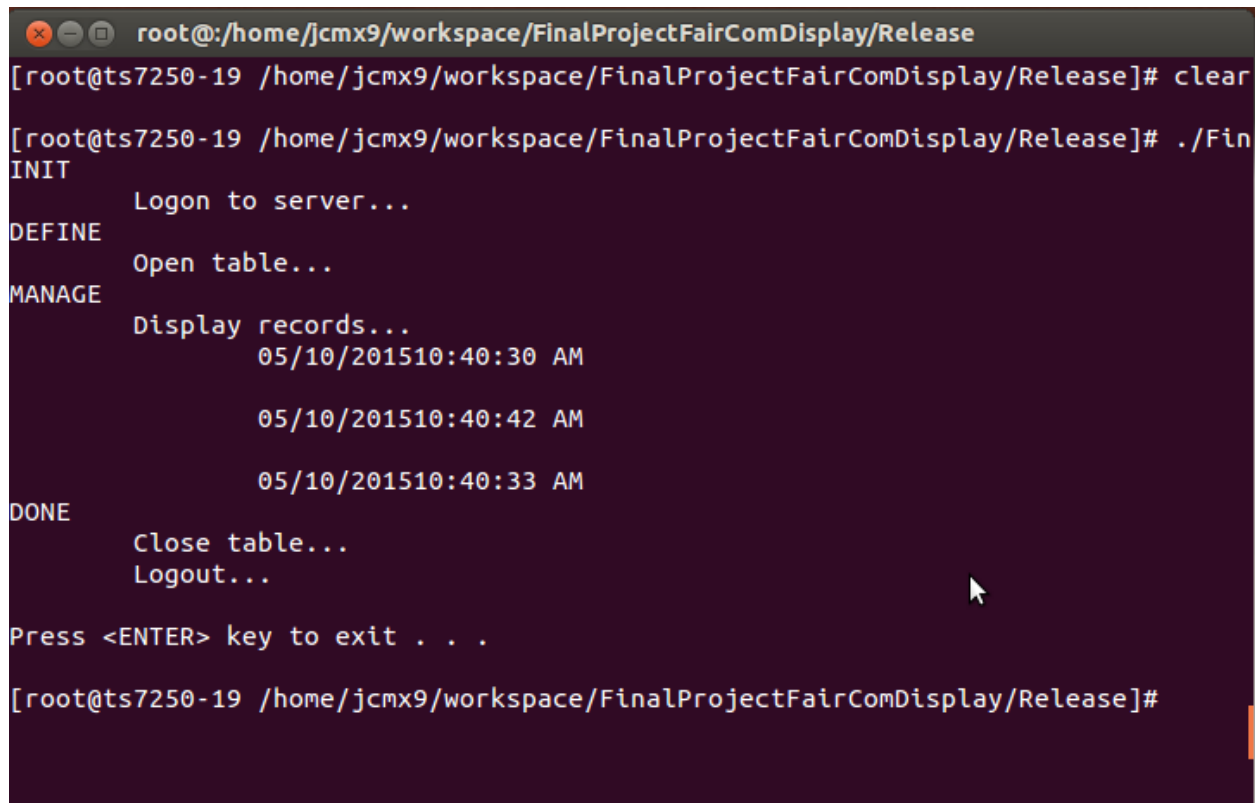
[root@ts7250-19 /home/jcmx9/workspace/FinalProjectFairComRead/Release]# cd ../../
[root@ts7250-19 /home/jcmx9/workspace]# cd FinalProjectFairComDisplay/Release/
[root@ts7250-19 /home/jcmx9/workspace/FinalProjectFairComDisplay/Release]# clear
```

Figure 8 Result of FinalProjectFairComRead

In Figure 8 Result of FinalProjectFairComRead, we can see few “Distance changes within threshold” message, which is displayed when I frequently push the buttons in a similar

duration. We can also see three timestamps were recorded when I change the duration of pushing. Check details in attached video clip and next section.

FairCom Program for Displaying Records

A terminal window with a dark background and light-colored text. The window title is 'root@:/home/jcmx9/workspace/FinalProjectFairComDisplay/Release'. The user enters 'clear' and then './Fin'. The program output shows 'INIT', 'Logon to server...', 'DEFINE', 'Open table...', 'MANAGE', 'Display records...', and three timestamps: '05/10/201510:40:30 AM', '05/10/201510:40:42 AM', and '05/10/201510:40:33 AM'. It then shows 'DONE', 'Close table...', 'Logout...', and a prompt to press the ENTER key to exit. The prompt returns to the shell.

```
root@:/home/jcmx9/workspace/FinalProjectFairComDisplay/Release
[root@ts7250-19 /home/jcmx9/workspace/FinalProjectFairComDisplay/Release]# clear

[root@ts7250-19 /home/jcmx9/workspace/FinalProjectFairComDisplay/Release]# ./Fin
INIT
    Logon to server...
DEFINE
    Open table...
MANAGE
    Display records...
        05/10/201510:40:30 AM
        05/10/201510:40:42 AM
        05/10/201510:40:33 AM
DONE
    Close table...
    Logout...

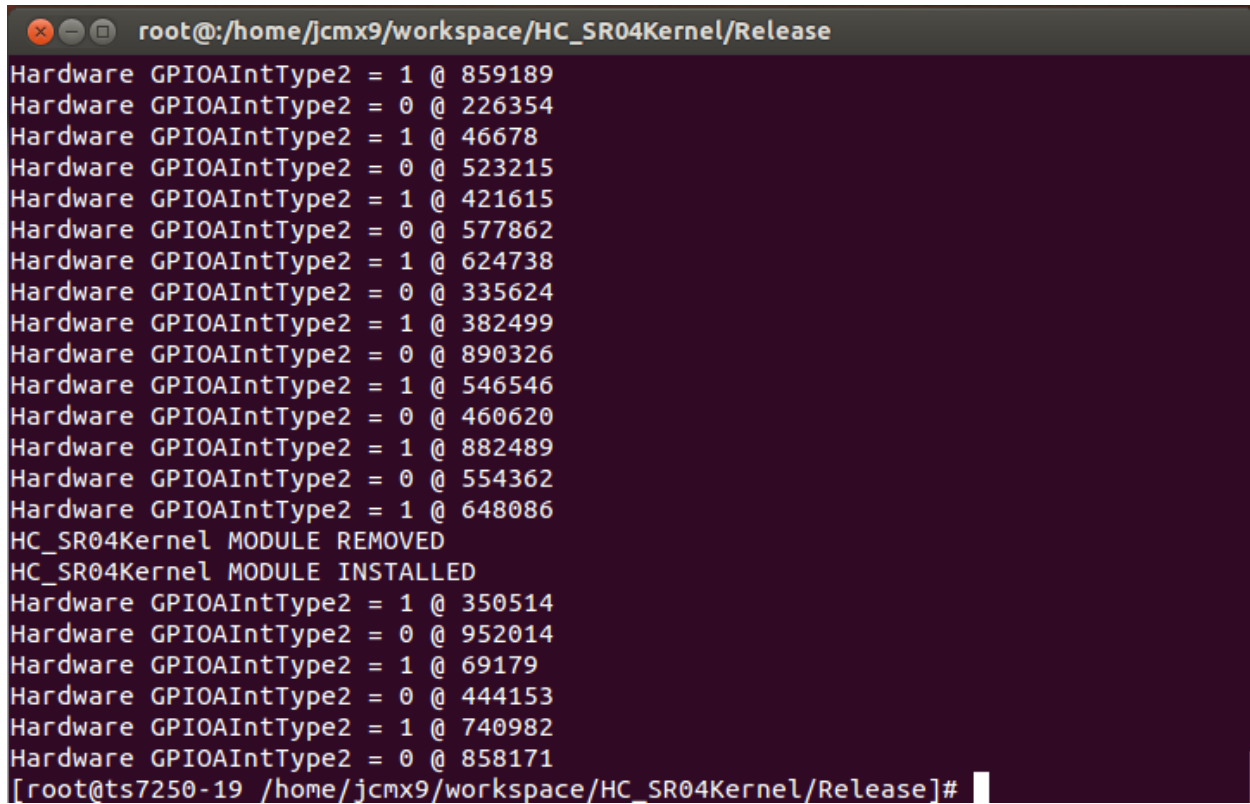
Press <ENTER> key to exit . . .

[root@ts7250-19 /home/jcmx9/workspace/FinalProjectFairComDisplay/Release]#
```

Figure 9 Result of FinalProjectFairComDisplay

In the display program, three timestamps are printed out, which are saved in the database before. It proves that the data stored in database can be access later for further use.

Toggling GPIOIntType2 in Interrupt Handler

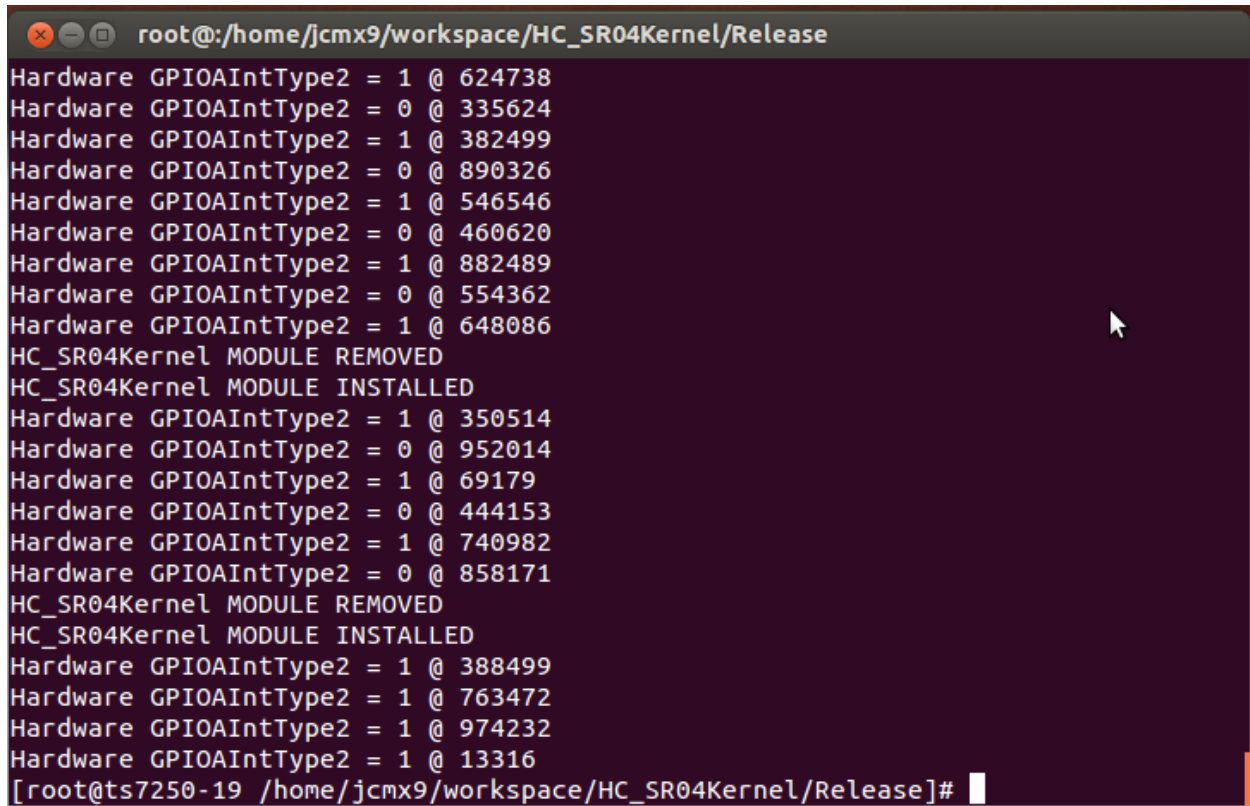
A terminal window with a dark background and light-colored text. The title bar shows a window icon, a close button, and the path 'root@:/home/jcmx9/workspace/HC_SR04Kernel/Release'. The terminal output consists of a series of 'Hardware GPIOIntType2 = 1 @' and 'Hardware GPIOIntType2 = 0 @' messages with increasing addresses. This is followed by 'HC_SR04Kernel MODULE REMOVED' and 'HC_SR04Kernel MODULE INSTALLED', then another series of similar messages. The prompt '[root@ts7250-19 /home/jcmx9/workspace/HC_SR04Kernel/Release]#' is visible at the bottom.

```
root@:/home/jcmx9/workspace/HC_SR04Kernel/Release
Hardware GPIOIntType2 = 1 @ 859189
Hardware GPIOIntType2 = 0 @ 226354
Hardware GPIOIntType2 = 1 @ 46678
Hardware GPIOIntType2 = 0 @ 523215
Hardware GPIOIntType2 = 1 @ 421615
Hardware GPIOIntType2 = 0 @ 577862
Hardware GPIOIntType2 = 1 @ 624738
Hardware GPIOIntType2 = 0 @ 335624
Hardware GPIOIntType2 = 1 @ 382499
Hardware GPIOIntType2 = 0 @ 890326
Hardware GPIOIntType2 = 1 @ 546546
Hardware GPIOIntType2 = 0 @ 460620
Hardware GPIOIntType2 = 1 @ 882489
Hardware GPIOIntType2 = 0 @ 554362
Hardware GPIOIntType2 = 1 @ 648086
HC_SR04Kernel MODULE REMOVED
HC_SR04Kernel MODULE INSTALLED
Hardware GPIOIntType2 = 1 @ 350514
Hardware GPIOIntType2 = 0 @ 952014
Hardware GPIOIntType2 = 1 @ 69179
Hardware GPIOIntType2 = 0 @ 444153
Hardware GPIOIntType2 = 1 @ 740982
Hardware GPIOIntType2 = 0 @ 858171
[root@ts7250-19 /home/jcmx9/workspace/HC_SR04Kernel/Release]#
```

Figure 10 Kernel Message Buffer when Toggling GPIOIntType2

If I lay the breadboard with HC_SR04 on the table, nothing will be printed as kernel message after “HC_SR04Kernel MODULE INSTALLED”. If I turn the sensor towards a faraway surface like ceiling few data are printed but not continuously and frequently as period of trigger signal.

Configure as Rising Edge Sensitive

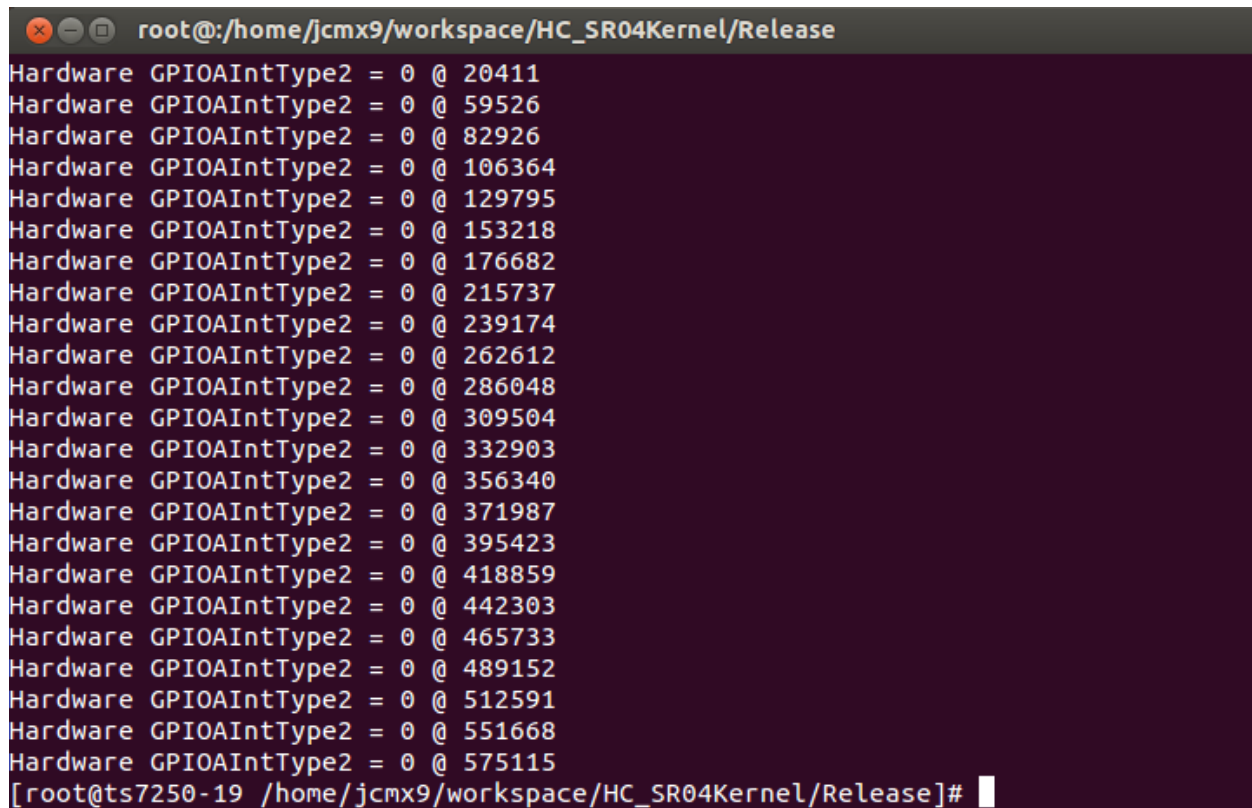
A terminal window with a dark background and light-colored text. The title bar shows a window icon, a close button, and the path 'root@:/home/jcmx9/workspace/HC_SR04Kernel/Release'. The terminal output consists of a series of 'Hardware GPIOAIntType2' messages, each followed by an equals sign, a value, and an '@' symbol with a number. These messages are interspersed with 'HC_SR04Kernel MODULE REMOVED' and 'HC_SR04Kernel MODULE INSTALLED' messages. The sequence of messages is: 1. Hardware GPIOAIntType2 = 1 @ 624738, 2. Hardware GPIOAIntType2 = 0 @ 335624, 3. Hardware GPIOAIntType2 = 1 @ 382499, 4. Hardware GPIOAIntType2 = 0 @ 890326, 5. Hardware GPIOAIntType2 = 1 @ 546546, 6. Hardware GPIOAIntType2 = 0 @ 460620, 7. Hardware GPIOAIntType2 = 1 @ 882489, 8. Hardware GPIOAIntType2 = 0 @ 554362, 9. Hardware GPIOAIntType2 = 1 @ 648086, 10. HC_SR04Kernel MODULE REMOVED, 11. HC_SR04Kernel MODULE INSTALLED, 12. Hardware GPIOAIntType2 = 1 @ 350514, 13. Hardware GPIOAIntType2 = 0 @ 952014, 14. Hardware GPIOAIntType2 = 1 @ 69179, 15. Hardware GPIOAIntType2 = 0 @ 444153, 16. Hardware GPIOAIntType2 = 1 @ 740982, 17. Hardware GPIOAIntType2 = 0 @ 858171, 18. HC_SR04Kernel MODULE REMOVED, 19. HC_SR04Kernel MODULE INSTALLED, 20. Hardware GPIOAIntType2 = 1 @ 388499, 21. Hardware GPIOAIntType2 = 1 @ 763472, 22. Hardware GPIOAIntType2 = 1 @ 974232, 23. Hardware GPIOAIntType2 = 1 @ 13316. The prompt '[root@ts7250-19 /home/jcmx9/workspace/HC_SR04Kernel/Release]#' is at the bottom with a cursor.

```
root@:/home/jcmx9/workspace/HC_SR04Kernel/Release
Hardware GPIOAIntType2 = 1 @ 624738
Hardware GPIOAIntType2 = 0 @ 335624
Hardware GPIOAIntType2 = 1 @ 382499
Hardware GPIOAIntType2 = 0 @ 890326
Hardware GPIOAIntType2 = 1 @ 546546
Hardware GPIOAIntType2 = 0 @ 460620
Hardware GPIOAIntType2 = 1 @ 882489
Hardware GPIOAIntType2 = 0 @ 554362
Hardware GPIOAIntType2 = 1 @ 648086
HC_SR04Kernel MODULE REMOVED
HC_SR04Kernel MODULE INSTALLED
Hardware GPIOAIntType2 = 1 @ 350514
Hardware GPIOAIntType2 = 0 @ 952014
Hardware GPIOAIntType2 = 1 @ 69179
Hardware GPIOAIntType2 = 0 @ 444153
Hardware GPIOAIntType2 = 1 @ 740982
Hardware GPIOAIntType2 = 0 @ 858171
HC_SR04Kernel MODULE REMOVED
HC_SR04Kernel MODULE INSTALLED
Hardware GPIOAIntType2 = 1 @ 388499
Hardware GPIOAIntType2 = 1 @ 763472
Hardware GPIOAIntType2 = 1 @ 974232
Hardware GPIOAIntType2 = 1 @ 13316
[root@ts7250-19 /home/jcmx9/workspace/HC_SR04Kernel/Release]#
```

Figure 11 Kernel Message Buffer with Rising Edge Sensitive Interrupt

As we see in the Figure 11, after last “HC_SR04Kernel MODULE INSTALLED” message. Only few interrupts are caught. That is also generated when I turn the sensor towards the ceiling. However, User space program show the distance measure is wrong. It also got nothing back from a short distance.

Configure as Falling Edge Sensitive

A terminal window with a dark background and light text. The title bar shows a window icon and the text 'root@:/home/jcmx9/workspace/HC_SR04Kernel/Release'. The terminal output consists of 20 lines, each starting with 'Hardware GPIOAIntType2 = 0 @' followed by a number. The numbers are: 20411, 59526, 82926, 106364, 129795, 153218, 176682, 215737, 239174, 262612, 286048, 309504, 332903, 356340, 371987, 395423, 418859, 442303, 465733, 489152, 512591, 551668, and 575115. The last line is '[root@ts7250-19 /home/jcmx9/workspace/HC_SR04Kernel/Release]#'.

```
root@:/home/jcmx9/workspace/HC_SR04Kernel/Release
Hardware GPIOAIntType2 = 0 @ 20411
Hardware GPIOAIntType2 = 0 @ 59526
Hardware GPIOAIntType2 = 0 @ 82926
Hardware GPIOAIntType2 = 0 @ 106364
Hardware GPIOAIntType2 = 0 @ 129795
Hardware GPIOAIntType2 = 0 @ 153218
Hardware GPIOAIntType2 = 0 @ 176682
Hardware GPIOAIntType2 = 0 @ 215737
Hardware GPIOAIntType2 = 0 @ 239174
Hardware GPIOAIntType2 = 0 @ 262612
Hardware GPIOAIntType2 = 0 @ 286048
Hardware GPIOAIntType2 = 0 @ 309504
Hardware GPIOAIntType2 = 0 @ 332903
Hardware GPIOAIntType2 = 0 @ 356340
Hardware GPIOAIntType2 = 0 @ 371987
Hardware GPIOAIntType2 = 0 @ 395423
Hardware GPIOAIntType2 = 0 @ 418859
Hardware GPIOAIntType2 = 0 @ 442303
Hardware GPIOAIntType2 = 0 @ 465733
Hardware GPIOAIntType2 = 0 @ 489152
Hardware GPIOAIntType2 = 0 @ 512591
Hardware GPIOAIntType2 = 0 @ 551668
Hardware GPIOAIntType2 = 0 @ 575115
[root@ts7250-19 /home/jcmx9/workspace/HC_SR04Kernel/Release]#
```

Figure 12 Kernel Message Buffer with Falling Edge Sensitive Interrupt

Figure 12 shows that TS-7250 board works fine to detect the falling edge from “Echo” pin of HC-SR04 sensor. Without moving or turning the sensor, a bunch of interrupts are created based on the trigger signal period.

Reading HC-SR04 sensor by Arduino

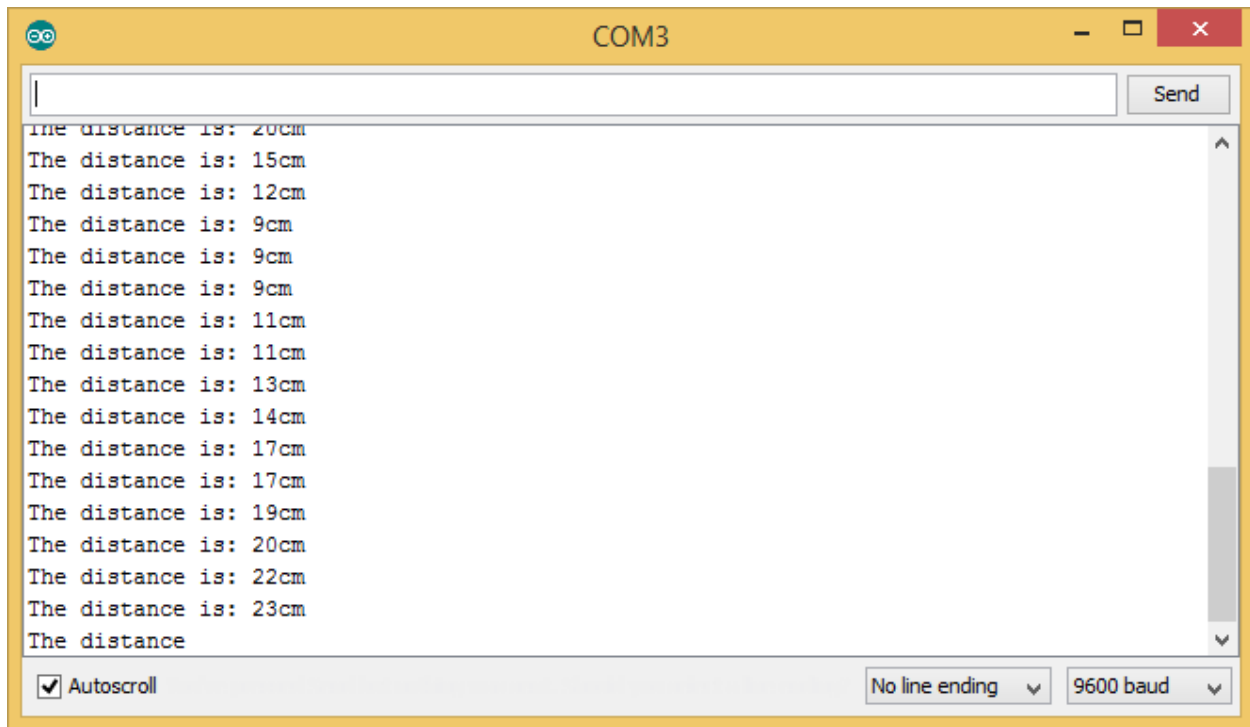


Figure 13 Result of SensorReading by Arduino

Figure 13 Result of SensorReading is the serial monitor of Arduino IDE. Arduino use USB to connect PC and it serve as a virtual serial port. We can clearly see that distance value is changing in the serial monitor.

Camera Image on Laptop

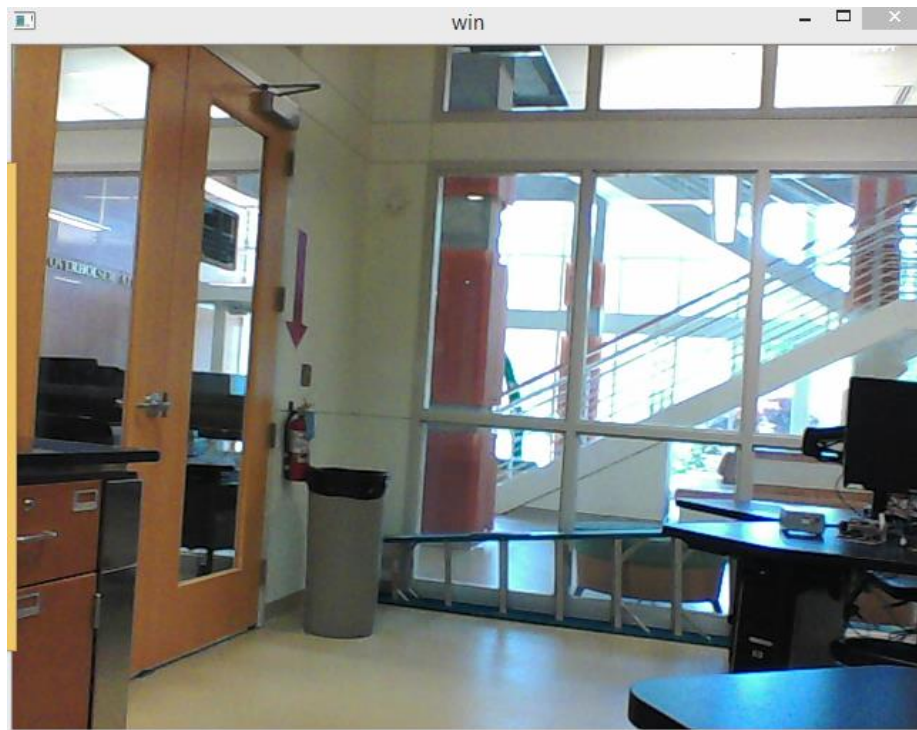


Figure 14 Camera Image on Laptop before Moving



Figure 15 Camera Image on Laptop after Moving

Discussion and Conclusions

FairCom Program with Distance Receiver

As the characteristic of the HS-SR04 sensor, the distance will be represented by the pulse length generated on “Echo” pin. The length of the pulse is exactly the time between sensor generate ultrasonic signal and receiving. So it is easy to use the speed of sound to calculate the distance. The pulse length will between $150\mu\text{s}$ and 25ms , which correspond to 2.59cm and 431cm . Due to the incapability of TS-7250 board, such a short pulse width cannot trigger either the high level or rising edge interrupt. So I could only use a push button to generate a longer pulse as “Echo” signal to trigger the interrupt on TS-7250 board. Figure 8 Result of FinalProjectFairComRead shows that the database and distance comparer works pretty fine. The distance change is set to 20000cm because a push button is much longer than actual “Echo” pin signal. In another world, if I want to trigger a distance change event and make program record a timestamp to store in database, the duration I push button should differ at least 1.16 second from last site.

FairCom Program for Displaying Records

Figure 8 shows that the database is stored in the disk but not RAM. So even the main process has been terminated. We can still pull up the timestamps by opening and accessing the database. This is the biggest different from using a database or not and administrator can easily get timestamp by displaying process no matter the reading process has been ended or not.

Toggling GPIOIntType2 in Interrupt Handler

Toggling the GPIOIntType2 register is the basic way to detect and measure a pulse width from a port. The GPIOIntType2 is initiated as logic HIGH in `init_module()`. When a

rising edge of pulse comes, an interrupt should be generated and call the interrupt handler. In the handler a timestamp will be record and then the GPIOIntType2 register is going to be toggled so the kernel will detect next falling edge and record another timestamp. After receiving these two timestamp in user space by fifo, the user space program will calculate the distance.

From the Figure 10, we can see it doesn't work as design. Instead of printing continuously message in kernel buffer (which means the handler is called), we could only notice few messages after kernel was installed. Actually, the presence of these messages are because I turned the sensor towards the ceiling, which resulted in an increase in distance so the pulse width is slow enough to generate interrupt. In the datasheet, it also mentioned "Please make sure the surface of object to be detect should have at least 0.5 meter² for better performance." ("Product User's Manual – HCSR04 Ultrasonic Sensor V1.0", Cytron Technologies). Due to the complexity of ceiling in C1246 Lab, the return pulse width contains lots of incorrect data, which probably caused by the uneven ceiling.

In other word, the sensor cannot work with the TS-7250 board interrupt mechanism.

Configure as Rising Edge Sensitive

To figure out the real problem the system has with the sensor, few experiments of register configuration was taken. In this experiment, I keep the GPIOIntType2 register as logic HIGH regardless of data correctness, which means the kernel could only have interrupt when rising edge is coming. In Figure 11, we can also see there isn't too much messages in the kernel message buffer and the only four messages there are also caught when I turned the sensor towards the ceiling. Combined with next experiment, we can see the malfunction part.

Configure as Falling Edge Sensitive

I keep the GPIOIntType2 register as logic LOW and got results in Figure 12. We happily see there are bunch of messages printed into the kernel message buffer, which means a lots of interrupt successfully occurred. Combined with Configure as Rising Edge Sensitive experiment, we can conclude that it might need a certain time for port to stay in logic HIGH before the kernel can detect and generate a rising edge interrupt.

Also the same experiment was done with high level interrupt configuration (clear the corresponding bit on GPIOIntType1) and got same result. But the same program works fine with a push button, which generated a much longer pulse than the “Echo” pin.

Reading HC-SR04 sensor by Arduino

As the HC-SR04 sensor cannot work directly with TS-7250 before as the reasons in Constraints section and the explanation above, I used an Arduino board as the alternative device for reading data from HS-SR04 sensor. In Figure 13, we clearly see it works pretty good with Arduino. The distance printed in the serial port is consistent and are varying when I was moving my hand in front of the sensor. This proves that the sensor circuit and hardware is good but problems happen on the TS-7250 board.

Camera Image on Laptop

We can clearly see the camera can grab image and output on my laptop by OpenCV library. When <ENTER> key is pushed, the window will show next frame captured by the camera. Noticed that if I moved the camera and hit <ENTER> it will still output one image at original place because the image has already been caught and saved in the buffer before I hit <ENTER> button. The OpenCV can also run on Linux, so theoretically the TS-7250 with an ARM CPU and Linux kernel is able to read image from WebCam. But due to what I mentioned

in the Constraints section, it cannot be implemented on the TS-7520 in the Lab1246 which must be accessed by nfs1 server.

Conclusion

Implemented a hardware component in this project is the most interesting and challenging part, which require the ability to look up datasheet for appropriate information. And we must now how to do interposes communication (including pipe, named pipe, stream pipe, etc.) since usually the process which communicated with the hardware component (it is a distance sensor in this project) is not supposed to be the main process and get stuck when we are retrieving and waiting data (I used fifo to send edge flags and timestamps in this project from kernel to user space program). Interrupts are also possible to be used to detect hardware input as the highest priority so the board can still operate other processes. Furthermore, the usage of FairCom database is also a criteria of student's comprehension ability and self-studying ability since the FairCom database comes with limited tutorials, resources and closed-source APIs.

In another word, the project is a epitome of all knowledge we learnt in this semester from both lecture (theoretical) and Lab (practical), which need comprehensive understand of this knowledge. It is also a great platform to present student's ability and develop their self-study ability with unknown problems out of classes. Student also can get experience on handling a project from proposal to final report, which will be used for in job interview and required by career in this field.

Although I didn't implement all features in the proposal for this project because hardware limitation on TS-7250 board mentioned in Constraints sections, all effort I put are worth. I learnt the way to debug a program with circuit and how to finish a big project steps by steps. Even one part of the whole system failed, we can switch works on another part instead of getting stuck in

one problem. Finally, it is still possible to solve all problems with increasing resources and knowledge. Also, I am pretty sure I can implement all feature with an appropriate platform.

Appendices (Code)

Reading Sensor Kernel

```
/*
 * HC_SR04Kernel.c
 *
 * Created on: May 1, 2015
 * Author: jcmx9
 */

#ifndef MODULE
#define MODULE
#endif

#ifndef __KERNEL__
#define __KERNEL__
#endif

#include <linux/module.h>
#include <linux/kernel.h>
#include <asm/io.h>
#include <rtai.h>
#include <rtai_sched.h>
#include <rtai_hal.h>
#include <rtai_fifos.h>

MODULE_LICENSE("GPL");

static RT_TASK trigger;
RTIME period;

//use hardware interrupt Line 59, refer to GPIOINTR, the combined interrupt from any bit in
ports A or B
int hrd_irq_num = 59;

//registers pointers
unsigned long *PADR; //Port
A Data Register
unsigned long *PADDR; //Port
A Data Direction Register
unsigned long *GPIOAIntEn; //Interrupt
Enable Register A
```

```

unsigned long *GPIOAIntType1;           //Interrupt
Type 1 Register A
unsigned long *GPIOAIntType2;           //Interrupt
Type 2 Register A
unsigned long *GPIOAIntEOI;             //End-
Of_Interrupt Register A
unsigned long *GPIOADB;
    //debouncing register of PortA
unsigned long *IntStsA;
    //Interrupt Status Register

//interrupt handler
static void hwdHandler(unsigned irq_num, void *cookie){

    struct timeval tv;
    //timeval variable to record time
    int rising = 1;                       //rising
edge flag for fifo1
    int falling = 0;                       //falling edge
flag for fifo1
    rt_disable_irq (irq_num);             //disable
interrupt when handler is executed

    do_gettimeofday(&tv);                 //get
timestamp

    if ( (*GPIOAIntType2 >> 1) & 1)        //if the
interrupt caused by rising edge
        rtf_put (1, &rising, sizeof (int)); //put a rising edge
flag, 1, in the fifo1
    else
        rtf_put (1, &falling, sizeof (int)); //else, put a falling edge flag,
0, in the fifo1

    rtf_put (0, &tv, sizeof(struct timeval)); //put timestamp in fifo0

    *GPIOAIntType2 ^= (1 << 1);           //toggle so it will switch between rising and
falling edge sensitive
    *GPIOAIntEOI |= 1 << 1;               //clear hardware interrupt on portB1

    rt_enable_irq (irq_num);               //enable interrupt after handler has been executed
}

static void rt_process(int t){
    while (1){
        *PADR |= 1 << 0;                 //set HIGH to trigger signal

```

```

        rt_sleep (period);                                //trigger pulse should be at lease
10us, 2ms in this case
        *PADR &= ~(1 << 0);                             //set low to end the pulse

        rt_task_wait_period ();                           //wait for predefined real-time
period, 10ms in this case
    }
}

int init_module(void){
    //map address
    unsigned long *GPOI_base;                            //GPIO Control Registers
    GPOI_base = (unsigned long *)__ioremapped(0x80840000, 4096, 0); //map address to
virtual memory

    //char is 1 byte long, this conversion make me easy to calculate the offset from datasheet
value
    PADR = (unsigned long *)((char *)GPOI_base + 0x00);
    PADDR = (unsigned long *)((char *)GPOI_base + 0x10);
    GPIOAIntEn = (unsigned long *)((char *)GPOI_base + 0x9C);
    GPIOAIntType1 = (unsigned long *)((char *)GPOI_base + 0x90);
    GPIOAIntType2 = (unsigned long *)((char *)GPOI_base + 0x94);
    GPIOAIntEOI = (unsigned long *)((char *)GPOI_base + 0x98);
    GPIOADB = (unsigned long *)((char *)GPOI_base + 0xA8);
    IntStsA = (unsigned long *)((char *)GPOI_base + 0xA0);

    //modify registers
    *PADDR &= ~(1 << 1);                                //set bit 1 to input for receiving
"Echo" pulse
    *PADDR |= 1 << 0;                                    //set bit 0 to output for "Trig" pulse

    *PADR &= ~(1 << 0);                                    //clear 0 bit to make sure no
incorrect pulse before trigger

    *GPIOAIntType1 |= 1 << 1;                             //initiate portA 1 (echo) as edge sensitive
    *GPIOAIntType2 |= 1 << 1;                             //initiate portA 1 (echo) as rising edge
sensitive
    *GPIOADB |= 1 << 1;                                    //enable portA 1 (echo)
debouncing

    rtf_create(0, sizeof(struct timeval));                //create a timeval size fifo0 for timestamp
    rtf_create (1, sizeof (int));                          //create a int size fifo1 for edge flag

    //initiate real-time task
    rt_set_periodic_mode();                               //set the real-
time process to periodic mode

```

```

        period = start_rt_timer(nano2count(2000000));    //internal period is 2ms, minimum
counter 100000
        rt_task_init(&trigger, rt_process, 0, 512, 0, 0, 0);    //bind the real-time
task with handler
        rt_task_make_periodic (&trigger, rt_get_time(), 5 * period);    //distance measuring
internal is 10ms

        //initiate interrupt
        rt_request_irq (hrd_irq_num, hwdHandler, 0, 1);    //request irq line for
the
        *GPIOAIntEOI |= 1 << 1;
        //clear portA1 interrupt
        *GPIOAIntEn = 0x02;
        //enable interrupt on portA1
        rt_enable_irq (hrd_irq_num);    //enable
interrupt on this irq line

        printk ("HC_SR04Kernel MODULE INSTALLED\n");
        return 0;
}

void cleanup_module(void){
        *GPIOAIntEOI |= 1 << 1;
        //clear interrupt on bit 1 of Port A
        rt_disable_irq (hrd_irq_num);    //disable
interrupt on this irq line
        rt_release_irq (hrd_irq_num);    //release the
requested irq line
        rtf_destroy(0);
        //destroy fifo0
        rtf_destroy(1);
        //destroy fifo1
        rt_task_delete(&trigger);    //delete
the realtime process
        stop_rt_timer();
        //stop timer for realtime process
        printk ("HC_SR04Kernel MODULE REMOVED\n");
}

```


FairCom Server with Distance Receiver and Comparer

```
/*
 * FinalProjectFairComRead.c
 *
 * Created on: May 9, 2015
 * Author: jcmx9
 */

#ifdef _WIN32_WCE
#undef UNICODE
#undef _UNICODE
#define main my_main
#endif

/* Preprocessor definitions and includes */

#include "ctdbsdk.h"          // c-tree headers
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <sys/types.h>
#include <sys/stat.h>         //for open()/write() mode
#include <fcntl.h>            //for open()/write()
#include <err.h>              //for open() error test
#include <sys/time.h>         //for time() function
#include <unistd.h>
#include <sys/select.h>       //for select() function

#define END_OF_FILE INOT_ERR /* INOT_ERR is ctree's 101 error. See cterr.h */

/* Global declarations */

CTHANDLE hSession;
CTHANDLE hDatabase;
CTHANDLE hTable;
CTHANDLE hRecord;

/* Function declarations */

#ifdef PROTOTYPE
VOID Initialize(VOID), Define(VOID), Manage(VOID), Done(VOID);
VOID Display_Records(VOID), ReadDistanceChange (VOID);
```

```

VOID Delete_Records(CTHANDLE), Check_Table_Mode(CTHANDLE);
VOID Handle_Error(CTSTRING);
VOID error (const char *msg);
#else
VOID Initialize(), Define(), Manage(), Done();
VOID Add_Records(), Display_Records(), Add_Timestamps_Record (), ReadDistanceChange
();
VOID Delete_Records(), Check_Table_Mode();
VOID Handle_Error();
VOID error (msg);
#endif

/*
 * main()
 *
 * Call functions in steps
 */

#ifdef PROTOTYPE
NINT main (NINT argc, pTEXT argv[])
#else
NINT main (argc, argv)
NINT argc;
TEXT argv[];
#endif
{
    Initialize();

    Define();

    Manage();

    Done();

    printf("\nPress <ENTER> key to exit . . .\n");
#ifdef ctPortWINCE
    getchar();
#endif

    return(0);
}

/*
 * Initialize()

```

```

*
* Perform the minimum requirement of logging onto the c-tree Server
*/

#ifdef PROTOTYPE
VOID Initialize(VOID)
#else
VOID Initialize()
#endif
{
    CTDBRET retval;

    printf("INIT\n");

    if ((retval = ctdbStartDatabaseEngine()))                //using the Server DLL model
        to start the underlying Server.
        Handle_Error("Initialize(): ctdbStartDatabaseEngine()");

    /* allocate session handle */
    if ((hSession = ctdbAllocSession(CTSESSION_CTREE)) == NULL)
        //allocate session
        Handle_Error("Initialize(): ctdbAllocSession()");

    hDatabase = hSession; /* database not used in this program */

    /* connect to server */
    printf("\tLogon to server...\n");
    if (ctdbLogon(hSession, "FAIRCOMS", "ADMIN", "ADMIN"))    //login
        database by default id and passcode
        Handle_Error("Initialize(): ctdbLogon()");
}

/*
* Define()
*
* Open the table, if it exists. Otherwise create and open the table
*/

#ifdef PROTOTYPE
VOID Define(VOID)
#else
VOID Define()
#endif
{
    CTDBRET retval;

```

```

        CTHANDLE hField1, hField2;

printf("DEFINE\n");

/* allocate a table handle */
if ((hTable = ctdbAllocTable(hDatabase)) == NULL)
    Handle_Error("Define(); ctdbAllocTable()");

/* open table */
printf("\tOpen table...\n");
if (ctdbOpenTable(hTable, "TimeStamp", CTOPEN_NORMAL))           //Table
Name: TimeStamp
{
    /* define table fields, two fields, */
    printf("\tAdd fields...\n");
    hField1 = ctdbAddField(hTable, "Date", CT_STRING, 15);        //CT_STRING, char*
type to store date string
    hField2 = ctdbAddField(hTable, "Time", CT_STRING, 15);        //CT_STRING, char*
type to store time string

    if (!hField1 || !hField2)
        //error check
        Handle_Error("Define(); ctdbAddField()");

    /* create table */
    printf("\tCreate table...\n");
    if (ctdbCreateTable(hTable, "TimeStamp", CTOPEN_NORMAL))      //if the table
does not exist, create one
        Handle_Error("Define(); ctdbCreateTable()");

    if (ctdbOpenTable(hTable, "TimeStamp", CTOPEN_NORMAL))
        Handle_Error("Define(); ctdbOpenTable()");
}
else
{
    Check_Table_Mode(hTable);
}

}

/*
* Manage()
*
* This function performs simple record functions of add, delete and gets
*/

```

```

#ifdef PROTOTYPE
VOID Manage(VOID)
#else
VOID Manage()
#endif
{
    printf("MANAGE\n");

    /* allocate a record handle */
    if ((hRecord = ctdbAllocRecord(hTable)) == NULL)           //allocate a record
        handler to operate record
        Handle_Error("Manage(): ctdbAllocRecord()");

    /* delete any existing records */
    Delete_Records(hRecord);                                   //clean
    records before store

    ReadDistanceChange ();                                     //read
    distance from kernel by fifo and compare

    /* display contents of table */
    Display_Records();
    //display records after record finish
}

/*
 * Done()
 *
 * This function handles the housekeeping of closing tables and
 * freeing of associated memory
 */

#ifdef PROTOTYPE
VOID Done(VOID)
#else
VOID Done()
#endif
{
    printf("DONE\n");

    /* close table */
    printf("\tClose table...\n");
    if (ctdbCloseTable(hTable))                               //close table first
        Handle_Error("Done(): ctdbCloseTable()");
}

```

```

/* logout */
printf("\tLogout...\n");
if (ctdbLogout(hSession))
    Handle_Error("Done(): ctdbLogout()");
open in this case so don't need close

/* free handles */
ctdbFreeRecord(hRecord);
handler
ctdbFreeTable(hTable);
handler
ctdbFreeDatabase(hDatabase);
handler
ctdbFreeSession(hSession);

//if we are linked to the Server DLL, then we should stop our Server at the end of the program.
ctdbStopDatabaseEngine();
Server DLL before
}

/*
 * Check_Table_Mode()
 *
 * Check if existing table has transaction processing flag enabled.
 * If a table is under transaction processing control, ctdbGetError()modify the
 * table mode to disable transaction processing
 */

#ifdef PROTOTYPE
VOID Check_Table_Mode(CTHANDLE hTable)
#else
VOID Check_Table_Mode(hTable)
CTHANDLE hTable;
#endif
{
    CTCREATE_MODE mode;

    /* get table create mode */
    mode = ctdbGetTableCreateMode(hTable);

    /* check if table is under transaction processing control */
    if ((mode & CTCREATE_TRNLOG))
    {
        /* change file mode to disable transaction processing */
        mode ^= CTCREATE_TRNLOG;
    }
}

```

```

        if (ctdbUpdateCreateMode(hTable, mode) != CTDBRET_OK)
            Handle_Error("Check_Table_Mode(); ctdbUpdateCreateMode");
    }
}

/*
 * Delete_Records()
 *
 * This function deletes all the records in the table
 */

#ifdef PROTOTYPE
VOID Delete_Records(CTHANDLE hRecord)
#else
VOID Delete_Records(hRecord)
CTHANDLE hRecord;
#endif
{
    CTDBRET  retval;                                //retval of function, for error
    checking purpose
    CTBOOL  empty;

    printf("\tDelete records...\n");

    empty = NO;
    retval = ctdbFirstRecord(hRecord);                //point the record handler to the first record
    if (retval != CTDBRET_OK)
    {
        if (retval == END_OF_FILE)
            empty = YES;                                //no record is in the table
        else
            Handle_Error("Delete_Records(): ctdbFirstRecord()");
    }

    while (empty == NO) /* while table is not empty */
    {
        /* delete record */
        if (ctdbDeleteRecord(hRecord))
            Handle_Error("Delete_Records(): ctdbDeleteRecord()");

        /* read next record */
        retval = ctdbNextRecord(hRecord);
        if (retval != CTDBRET_OK)
        {
            if (retval == END_OF_FILE)

```

```

        empty = YES;
    else
        Handle_Error("Delete_Records(): ctdbNextRecord()");           //check error
every time
    }
}
}

/*
 * ReadDistanceChange ()
 *
 * For all testing purposes
 */

#ifdef PROTOTYPE
VOID ReadDistanceChange (VOID)
#else
VOID ReadDistanceChange ()
#endif
{
    //faircom variable
    CTDBRET retval;
    CTDATE nowDate;
    CTTIME nowTime;
    TEXT nowDateString[15+1];    //string to
    TEXT nowTimeString[15+1];

    double prevDist = 0;          //initiate previous distance as 0 for comparing
convenience
    double newDist;                //variable to store incoming distance
    int edgeFlag = 2;              //invalid number for if statement
    bool startFlag = false;        //double protect from receiving out of order
timestamps
    bool endFlag = false;          //double protect from receiving out of order
timestamps
    //ts_start to store start timestamp, ts_end to store end timestamp, time_discard to store out
of order timestamp
    struct timeval ts_start, ts_end, time_discard;
    double duration;                //variable to store the duration of rising edge
and falling edge of "Echo" pin
    int fifo0 = open("/dev/rtf/0", O_RDWR);    //fifo0 to receive edge
flag from kernel
    int fifo1 = open("/dev/rtf/1", O_RDWR);    //fifo1 to receive
timestamps kernel

```



```

//for timeout option, timeout is set for demonstration purpose
fd_set set;
struct timespec timeout;
int rv;

FD_ZERO(&set);                                //clear the set
FD_SET(fifo1, &set);                          //add file descriptor, the fifo1 to the set,
timeout.tv_sec = 5;                            //5 seconds timeout for reading fifo1
timeout.tv_nsec = 0;

while (1){
    startFlag = false;                        //set the flag back to false at the beginning of
    endFlag = false;
    //set timeout for fifo1
    rv = pselect(fifo1 + 1, &set, NULL, NULL, &timeout, NULL);
    if (rv == -1)
        err(1, "ReadDistanceChange(): pselect()");                //error occur in
pselect()
    else if (rv == 0){
        printf("timeout...\n");                //when timeout, return the read
distance function
        return;
    }
    else{
        //if no error and not timeout yet, read the edge flag fifo1
        if (read (fifo1, &edgeFlag, sizeof(int)) != sizeof (int))
            err(1, "ReadDistanceChange (): read() fifo1");
    }

    if (edgeFlag == 0){                //error, should
start from rising edge
        //read fifo0 and discard this timestamp because it start with falling edge
        if (read (fifo0, &time_discard, sizeof(time_discard)) != sizeof
(time_discard))
            err(1, "ReadDistanceChange (): read() fifo0");
        startFlag = false;
        endFlag = false;
    }
    else if (edgeFlag == 1){                //correct, start
recording from rising edge
        //get timestamp of rising edge
        if (read (fifo0, &ts_start, sizeof(ts_start)) != sizeof (ts_start))
            err(1, "ReadDistanceChange (): read() fifo0");
        startFlag = true;

        //read flag fifo 1 to make sure next timestamps is for falling edge

```

```

        if (read (fifo1, &edgeFlag, sizeof(int)) != sizeof (int))
            err(1, "ReadDistanceChange (): read() fifo1");

        if (edgeFlag == 1){                                //error, cannot be two
continuously rising edge
            if (read (fifo0, &time_discard, sizeof(time_discard)) != sizeof
(time_discard))
                err(1, "ReadDistanceChange (): read() fifo0");
                startFlag = false;
                endFlag = false;
            }
        else if (edgeFlag == 0){                            //correct, record timestamp for
falling edge
            if (read (fifo0, &ts_end, sizeof(ts_end)) != sizeof (ts_end))
                err(1, "ReadDistanceChange (): read() fifo0");
                endFlag = true;
            }
        }

        if (startFlag & endFlag){
            //calculate in integer to prevent decimal number (microsecond) lost in rounding
up, for accuracy
            duration = abs(ts_end.tv_sec * 1000000 + ts_end.tv_usec - ts_start.tv_sec *
1000000 - ts_start.tv_usec) / 1000;                        //in ms
            newDist = duration / 1000 * 34000 / 2;            //speed of sound:
34000cm/millisecond
        }

        if (abs(newDist - prevDist) > 20000){                //20000cm for threshold
because I am using push button to simulate the "Echo" pulse
            printf ("\nDistance Changed @\n");
            retval = 0;
            retval |= ctdbCurrentDate (&nowDate);            //ctdb function to get local
date
            retval |= ctdbCurrentTime (&nowTime);            //ctdb function to get local
time
            if (retval != CTDBRET_OK){
                Handle_Error("ReadDistanceChange (): ctdbCurrentDate
(), ctdbCurrentTime (");
            }

            //parse date to string in MM/DD/CCYY format
            retval = ctdbDateToString (nowDate, CTDATE_MDCY, nowDateString,
sizeof (nowDateString));
            //parse time to string in HH:MM:SS AP format

```

```

        retval = ctdbTimeString (nowTime, CTTIME_HMSP, nowTimeString,
sizeof (nowTimeString));

        printf ("Date: %s.\n", nowDateString);
        printf ("Time: %s.\n\n", nowTimeString);

        if (retval != CTDBRET_OK){
            Handle_Error("ReadDistanceChange (): ctdbDateToString(),
ctdbTimeString() ");
        }

        /* clear record buffer */
        ctdbClearRecord(hRecord);

        //populate record buffer with data
        retval = 0;
        //set the new inserted record field 0 as String to match the table
        retval |= ctdbSetFieldAsString(hRecord, 0, nowDateString);
        //set the new inserted record field 1 as String to match the table
        retval |= ctdbSetFieldAsString(hRecord, 1, nowTimeString);

        if (retval)
            Handle_Error("ReadDistanceChange(): ctdbSetFieldAsString()");

        /* add record */
        if (ctdbWriteRecord(hRecord))
            Handle_Error("ReadDistanceChange(): ctdbWriteRecord()");
    }
    else
        printf ("Distance changes within threshold.\n");

        prevDist = newDist;                                //save the incoming
distance as previous distance for next loop

    }
}

/*
 * Display_Records()
 *
 * This function displays the contents of a table. ctdbFirstRecord() and
 * ctdbNextRecord() fetch the record. Then each field is parsed and displayed
 */

#ifdef PROTOTYPE

```

```

VOID Display_Records(VOID)
#else
VOID Display_Records()
#endif
{
    CTDBRET  retval;
    TEXT    nowdate[15+1];
    TEXT    nowtime[15+1];

    printf("\tDisplay records...");

    /* read first record */
    retval = ctdbFirstRecord(hRecord);           //find the first record in the
table
    if (retval != CTDBRET_OK)
        Handle_Error("Display_Records(): ctdbFirstRecord()");           //if the table is empty

    while (retval != END_OF_FILE)
    {
        retval = 0;
        //retrieve the data in table as string
        retval |= ctdbGetFieldAsString(hRecord, 0, nowdate, sizeof (nowdate));
        retval |= ctdbGetFieldAsString(hRecord, 1, nowtime, sizeof (nowtime));
        if (retval)
            Handle_Error("Display_Records(): ctdbGetFieldAsString()");

        printf("\n\t\t%s\n", nowdate, nowtime);

        /* read next record */
        retval = ctdbNextRecord(hRecord);           //point the record
handler to the next record
        if (retval == END_OF_FILE)
            break; /* reached end of file */

        if (retval != CTDBRET_OK)
            Handle_Error("Display_Records(): ctdbNextRecord()");
    }
}

/*
 * Handle_Error()
 *
 * This function is a common bailout routine. It displays an error message
 * allowing the user to acknowledge before terminating the application
 */

```

```

#ifdef PROTOTYPE
VOID Handle_Error(CTSTRING errmsg)
#else
VOID Handle_Error(errmsg)
CTSTRING errmsg;
#endif
{
    printf("\nERROR: [%d] - %s \n", ctdbGetError(hSession), errmsg);
    printf("*** Execution aborted *** \nPress <ENTER> key to exit...");

    ctdbLogout(hSession);

    ctdbFreeRecord(hRecord);
    ctdbFreeTable(hTable);
    ctdbFreeSession(hSession);

    getchar();

    exit(1);
}

/*
 * error()
 *
 * This function handles error message in ReadDistanceChange();
 *
 */
#ifdef PROTOTYPE
VOID error (const char *msg)
#else
VOID error (msg)
#endif
{
    perror(msg);
    exit(0);
}

/* end of FinalProjectFairComRead.c */

```

FairCom Server Displaying Program

```
/*
 * FinalProjectFairComDisplay.c
 *
 * Created on: May 9, 2015
 * Author: jcmx9
 */

#ifdef _WIN32_WCE
#undef UNICODE
#undef _UNICODE
#define main my_main
#endif

/* Preprocessor definitions and includes */

#include "ctdbsdk.h"          // c-tree headers
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <sys/types.h>
#include <sys/stat.h>         //for open()/write() mode
#include <fcntl.h>            //for open()/write()
#include <err.h>              //for open() error test
#include <sys/time.h>         //for time() function
#include <unistd.h>
#include <sys/select.h>       //for select() function

#define END_OF_FILE INOT_ERR /* INOT_ERR is ctrees's 101 error. See ctree.h */

/* Global declarations */

CTHANDLE hSession;
CTHANDLE hDatabase;
CTHANDLE hTable;
CTHANDLE hRecord;

/* Function declarations */

#ifdef PROTOTYPE
VOID Initialize(VOID), Define(VOID), Manage(VOID), Done(VOID);
VOID Display_Records(VOID);
VOID Delete_Records(CTHANDLE), Check_Table_Mode(CTHANDLE);
```

```

VOID Handle_Error(CTSTRING);
VOID error (const char *msg);
#else
VOID Initialize(), Define(), Manage(), Done();
VOID Add_Records(), Display_Records(), Add_Timestamps_Record (), ReadDistanceChange
();
VOID Delete_Records(), Check_Table_Mode();
VOID Handle_Error();
VOID error (msg);
#endif

/*
 * main()
 *
 * The main() function implements the concept of "init, define, manage
 * and you're done..."
 */

#ifdef PROTOTYPE
NINT main (NINT argc, pTEXT argv[])
#else
NINT main (argc, argv)
NINT argc;
TEXT argv[];
#endif
{
    Initialize();

    Define();

    Manage();

    Done();

    printf("\nPress <ENTER> key to exit . . .\n");
#ifdef ctPortWINCE
    getchar();
#endif

    return(0);
}

/*
 * Initialize()

```

```

*
* Perform the minimum requirement of logging onto the c-tree Server
*/

#ifdef PROTOTYPE
VOID Initialize(VOID)
#else
VOID Initialize()
#endif
{
    CTDBRET retval;

    printf("INIT\n");

    if ((retval = ctdbStartDatabaseEngine()))                //using the Server DLL model
        to start the underlying Server.
        Handle_Error("Initialize(): ctdbStartDatabaseEngine()");

    /* allocate session handle */
    if ((hSession = ctdbAllocSession(CTSESSION_CTREE)) == NULL)    //allocate
        session
        Handle_Error("Initialize(): ctdbAllocSession()");

    hDatabase = hSession; /* database not used in this tutorial */

    /* connect to server */
    printf("\tLogon to server...\n");
    if (ctdbLogon(hSession, "FAIRCOMS", "ADMIN", "ADMIN"))        //login
        database by default id and passcode
        Handle_Error("Initialize(): ctdbLogon()");
}

/*
* Define()
*
* Open the table, if it exists. Otherwise create and open the table
*/

#ifdef PROTOTYPE
VOID Define(VOID)
#else
VOID Define()
#endif
{
    CTDBRET retval;

```



```

        CTHANDLE hField1, hField2;

printf("DEFINE\n");

/* allocate a table handle */
if ((hTable = ctdbAllocTable(hDatabase)) == NULL)
    Handle_Error("Define(); ctdbAllocTable()");

/* open table */
printf("\tOpen table...\n");
if (ctdbOpenTable(hTable, "TimeStamp", CTOPEN_NORMAL))           //Table
Name: TimeStamp
{
    /* define table fields */
    printf("\tAdd fields...\n");
    hField1 = ctdbAddField(hTable, "Date", CT_STRING, 15);        //CT_STRING, char*
type to store date string
    hField2 = ctdbAddField(hTable, "Time", CT_STRING, 15);        //CT_STRING, char*
type to store time string

    if (!hField1 || !hField2)                                     //error check
        Handle_Error("Define(); ctdbAddField()");

    /* create table */
    printf("\tCreate table...\n");
    if (ctdbCreateTable(hTable, "TimeStamp", CTOPEN_NORMAL))      //if the table
does not exist, create one
        Handle_Error("Define(); ctdbCreateTable()");

    if (ctdbOpenTable(hTable, "TimeStamp", CTOPEN_NORMAL))
        Handle_Error("Define(); ctdbOpenTable()");
}
else
{
    Check_Table_Mode(hTable);
}

}

/*
 * Manage()
 *
 * This function performs simple record functions of add, delete and gets
 */

```

```

#ifdef PROTOTYPE

```

```

VOID Manage(VOID)
#else
VOID Manage()
#endif
{
    printf("MANAGE\n");

    /* allocate a record handle */
    if ((hRecord = ctdbAllocRecord(hTable)) == NULL)           //allocate a record
        handler to operate record
        Handle_Error("Manage(): ctdbAllocRecord()");

    //no need to delete record because we are going to display records

    /* display contents of table */
    Display_Records();
}

/*
 * Done()
 *
 * This function handles the housekeeping of closing tables and
 * freeing of associated memory
 */

#ifdef PROTOTYPE
VOID Done(VOID)
#else
VOID Done()
#endif
{
    printf("DONE\n");

    /* close table */
    printf("\tClose table...\n");
    if (ctdbCloseTable(hTable))                               //close table first
        Handle_Error("Done(): ctdbCloseTable()");

    /* logout */
    printf("\tLogout...\n");
    if (ctdbLogout(hSession))                                  //logout the session
        Handle_Error("Done(): ctdbLogout()");                //no "database" is created or
                                                                open in this case so don't need close

    /* free handles */

```

```

    ctdbFreeRecord(hRecord);                                //free record
handler
    ctdbFreeTable(hTable);                                //free table
handler
    ctdbFreeDatabase(hDatabase);                          //free databaes
handler
    ctdbFreeSession(hSession);                            //free session handler

    //if we are linked to the Server DLL, then we should stop our Server at the end of the program.
    ctdbStopDatabaseEngine();                             //stop database engine because we are linked to the Server
DLL before
}

/*
 * Check_Table_Mode()
 *
 * Check if existing table has transaction processing flag enabled.
 * If a table is under transaction processing control, ctdbGetError()modify the
 * table mode to disable transaction processing
 */

#ifdef PROTOTYPE
VOID Check_Table_Mode(CTHANDLE hTable)
#else
VOID Check_Table_Mode(hTable)
CTHANDLE hTable;
#endif
{
    CTCREATE_MODE mode;

    /* get table create mode */
    mode = ctdbGetTableCreateMode(hTable);

    /* check if table is under transaction processing control */
    if ((mode & CTCREATE_TRNLOG))
    {
        /* change file mode to disable transaction processing */
        mode ^= CTCREATE_TRNLOG;
        if (ctdbUpdateCreateMode(hTable, mode) != CTDBRET_OK)
            Handle_Error("Check_Table_Mode(); ctdbUpdateCreateMode");
    }
}

/*
 * Display_Records()

```

```

*
* This function displays the contents of a table. ctdbFirstRecord() and
* ctdbNextRecord() fetch the record. Then each field is parsed and displayed
*/

#ifdef PROTOTYPE
VOID Display_Records(VOID)
#else
VOID Display_Records()
#endif
{
    CTDBRET  retval;
    TEXT    nowdate[15+1];
    TEXT    nowtime[15+1];

    printf("\tDisplay records...");

    /* read first record */
    retval = ctdbFirstRecord(hRecord);           //point the record handler to the first record
    if (retval != CTDBRET_OK)
        Handle_Error("Display_Records(): ctdbFirstRecord()");           //if the table is empty

    while (retval != END_OF_FILE)
    {
        retval = 0;
        //retrieve the data in table as string
        retval |= ctdbGetFieldAsString(hRecord, 0, nowdate, sizeof (nowdate));
        retval |= ctdbGetFieldAsString(hRecord, 1, nowtime, sizeof (nowtime));
        if (retval)
            Handle_Error("Display_Records(): ctdbGetFieldAsString()");

        printf("\n\t\t%s\n", nowdate, nowtime);

        /* read next record */
        retval = ctdbNextRecord(hRecord);           //point the
record handler to the next record
        if (retval == END_OF_FILE)
            break; /* reached end of file */

        if (retval != CTDBRET_OK)
            Handle_Error("Display_Records(): ctdbNextRecord()");
    }
}

/*

```

```

* Handle_Error()
*
* This function is a common bailout routine. It displays an error message
* allowing the user to acknowledge before terminating the application
*/

#ifdef PROTOTYPE
VOID Handle_Error(CTSTRING errmsg)
#else
VOID Handle_Error(errmsg)
CTSTRING errmsg;
#endif
{
    printf("\nERROR: [%d] - %s \n", ctdbGetError(hSession), errmsg);
    printf("*** Execution aborted *** \nPress <ENTER> key to exit...\n");

    ctdbLogout(hSession);

    ctdbFreeRecord(hRecord);
    ctdbFreeTable(hTable);
    ctdbFreeSession(hSession);

    getchar();

    exit(1);
}

/*
* error()
*
* This function handles error message in ReadDistanceChange();
*
*/
#ifdef PROTOTYPE
VOID error (const char *msg)
#else
VOID error (msg)
#endif
{
    perror(msg);
    exit(0);
}

/* end of FinalProjectFairComDisplay.c */

```

Reading Sensor by Arduino

```
/*
 * SensorReading.ino
 *
 * Created on: May 10, 2015
 * Author: jcmx9
 */

//constant for pin number
const int triggerPin = 5;          //to trigger sensor to range
const int echoPin = 6;            //receive pulse to measure distance

void setup() {
    //set serial communication baud rate as 9600:
    Serial.begin(9600);
}

void loop() {
    //duration: the width of the "Echo" pulse
    //cm_result: the calculated result of distance in centimeter
    long duration, cm_result;

    //based on the data sheet of the HC-SR04, the trigger should be at least 10 us
    //a short LOW pulse (2us) was set beforehand to ensure a clean HIGH pulse
    pinMode(triggerPin, OUTPUT);
    digitalWrite(triggerPin, LOW);
    delayMicroseconds(2);
    digitalWrite(triggerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW);

    //echoPin is used to read pulse from "Echo" pin on the sensor
    //the duration of the pulse is the time (in microseconds) from
    //sending measuring signal to the reception of its echo
    pinMode(echoPin, INPUT);
    //pulseIn functino is defined in wiring_pulse.c file. It will
    //return a long data to state the width of a pulse on specific pin
    duration = pulseIn(echoPin, HIGH);

    //convert the time into a distance in centimeter unit
    cm_result = microsecondsToCentimeters(duration);

    //serial output the result to PC
    Serial.print("The distance is: ");
    Serial.print(cm_result);
}
```

```
Serial.print("cm");  
Serial.println();  
  
delay(100);  
}  
  
long microsecondsToCentimeters(long microseconds)  
{  
    //the speed of sound is 340 m/s or 29 microseconds per centimeter.  
    //also we should divide the time by two because the ultrasonic wave went twice the distance  
    (back and forth)  
    return microseconds / 29 / 2;  
}
```

Reading Image from Webcam by OpenCV

```
/*
 * main.cpp
 *
 * Created on: May 10, 2015
 * Author: jcmx9
 */

#include "cv.h"
#include "highgui.h"
#include <stdio.h>

int main(int argc, char** argv) {
    //initiate a openCV Window and name it as "Capture Image"
    cvNamedWindow("Capture Image");

    //allocates and initialized the CvCapture structure for reading a video stream from the camera.
    CvCapture* capture = cvCreateCameraCapture(0);
    //The IplImage is taken from the Intel Image Processing Library, in which the format is native.
    //supported by OpenCV
    IplImage* frame;

    while(1) {
        printf ("\nPress <ENTER> key to capture.\n");
        //block the program until <ENTER> key press
        //when <ENTER> key press, a line will be read by getchar()
        getchar();
        //grabs a frame from camera, decompresses it
        frame = cvQueryFrame(capture);
        //if no frame was caught, there is an error, bread the while loop incase
        //future pointer error
        if(!frame)
            break;
        //shows the image in the specified window
        //image is scaled to fit the window by default
        cvShowImage("Capture Image", frame);

        //NECESSARY! highgui is never given time to process the draw requests
        //if no cvWaitKey is applied here

        //waits for key event for 50 milliseconds,
        //return -1 if no key is pressed
        char c = cvWaitKey(50);
        //when <ECS> is hit in the window "Capture Image",
        //bread while loop to stop the program
    }
}
```



```
        if(c==27)
            break;
    }

    //releases allocated CvCapture structure
    cvReleaseCapture(&capture);
    //destroy the window with a giver name "Capture Image"
    cvDestroyWindow("Capture Image");
    return 0;
}
```

Bibliography

- [1] Coles, Christopher F. "Security system with method for locatable portable electronic camera image transmission to a remote receiver." U.S. Patent No. 6,181,373. 30 Jan. 2001.