First Prize

Unattended Wireless Search Robot

Institution: **Kwangwoon University**

Participants: **Yoongoo Kim, Younggon Lee, Jeongwook Yim**

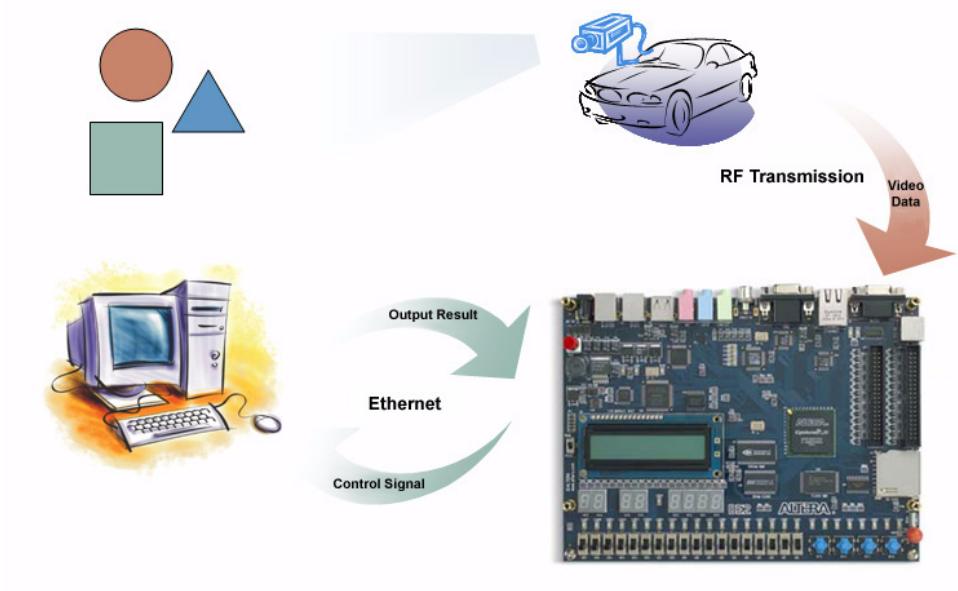
Instructor: **Yongjin Jeong**

Design Introduction

Our project, an unattended wireless search robot, implements a real-time image processor using a camera and a Nios® II-based system running on an FPGA. The robot searches for objects; when it finds one, it attempts to recognize it. It then sends a scaled 160 x 120 image via Ethernet to a PC. We used a radio frequency (RF) module, which increases the range of the robot, and the system ignores losses or errors during transfer. We used Sobel edge detection and the sonde method for faster image processing. Our implementation processes and transfers two to three frames per second.

Figure 1 shows the project's hardware system setup, including the Development and Education (DE2) board with an FPGA, National Television System Committee (NTSC) camera, 2.4-GHz RF modules, remote control (RC) car unit, and client PC. On the software side, we use the Nios II processor with a µC/OS-II RTOS, a lightweight TCP/IP network stack, and a Microsoft Foundation Class (MFC) Library. Finally, we considered various standards, e.g., NTSC, YCbCr, and RGB. We also explored concepts involving digital image processing, TCP/IP, RF theory, and RTOS.

Figure 1. System Organization



Our system operates as follows:

1. Using a client PC, the user specifies the object to be searched and transfers information about the object through the Ethernet link.

2. The RF module transfers picture frames from a camera to the DE2 board, extracts Y-pixel information from the picture, and saves it to SRAM on the DE2 board.

3. The Y-pixel picture is converted to 160 x 120, and is subjected to Sobel edge detection, which converts the picture into a binary image. Noise is removed and the picture is cropped, yielding a result picture.

4. The resulting picture is compared to the information about the search object. The picture is deemed to be that of the object if the results of the comparison are a greater than 90% match.

5. The result picture is transferred to the PC via Ethernet, where a client program displays the picture.

The project's system, named NIOS2-System, consists of an imaging device, CPU, RAM, DRAM, flash, a video codec, the Avalon® bus, an Ethernet interface, RS-232 port, and a user-defined module. We implemented the image processing module in the C language. We performed data transfer using the built-in lightweight TCP/IP (LWIP) stack configured for TCP/IP running on a µC/OS-II RTOS. We used the RS-232 connection to test the data transfer. The image recognition algorithm chooses between sonde and pixel matching, based on the simulation results.

Function Description

This section provides a functional description of our project.

Imaging Device Module

Figure 2 shows the system block diagram. The imaging device module, located on the right side of Figure 2, reads the NTSC video data. The 2.4-GHz RF module transfers video data to the DE2 board.

The ADV7181 video codec on DE2 board converts the NTSC video data to YCbCr. In the YCbCr standard, the original size of one frame is 720 x 525, of which only Y pixel information is required. We implemented a hardware module in Verilog HDL that saves the Y pixel data, which is then resized to 640 x 480 and stored in SRAM.

Figure 2. System Block Diagram

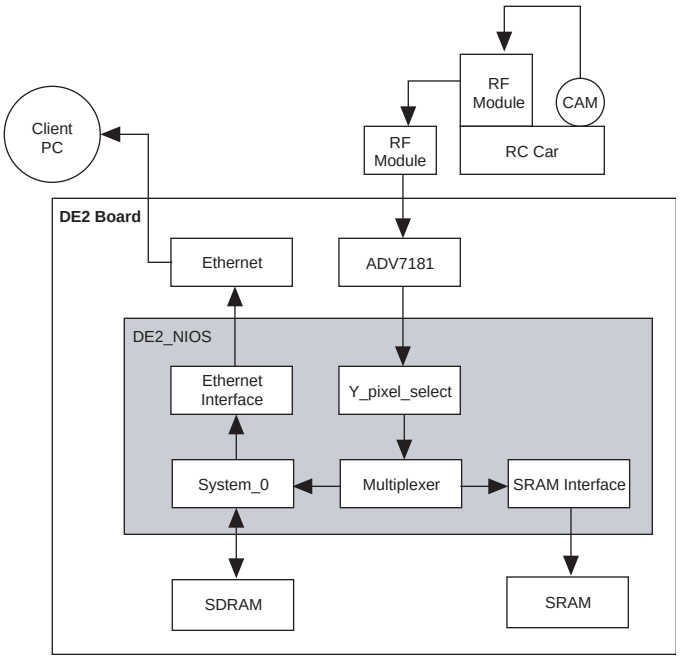


Image Preprocessing Module

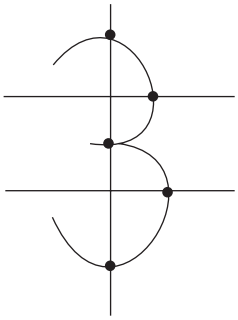
The System_0 module performs the image preprocessing. After the image data is saved in SRAM, it is copied to SDRAM using the device driver functions provided by the hardware abstraction layer (HAL). The image data is resized to 160 x 120, undergoes Sobel edge detection, and is converted into binary format.

Image Recognition Module

The sonde method provides optical character reading. Lines are drawn across the character in question and cross-points, i.e., points where the lines intersect the character, are captured to yield their characteristic position.

For example, Figure 3 shows three lines drawn across the number three. The number three has a characteristic cross-point position, so we can recognize the number by checking the cross-point positions. The sonde method is faster and more intelligent than pixel matching. However, sonde is not as effective for complex images. Therefore, we limited our test objects to 3 shapes—triangle, circle, and square—so that we could use the sonde method.

Figure 3. Sonde Method



Using the sonde method, we performed image recognition by comparing the features of triangles, squares, and circles, and checking to see if there is a match of better than 90%.

Image Transfer Module

This module transfers images and results via Ethernet to the client PC. The Ethernet chip is a DM9000a device. We modified the Ethernet driver to operate with the μC/OS-II RTOS. The robot communicates with the client PC at 100 megabits per second (Mbps) using the LWIP network stack. The PC converts the raw image data to bitmap data and displays the resulting image.

Performance Parameters

The design uses 14% of the logic elements available in the Cyclone® II device on the DE2 board and 2,094 registers, as shown in Figure 4. The DE2 board provided sufficient resources for this project.

Figure 4. Compilation Result

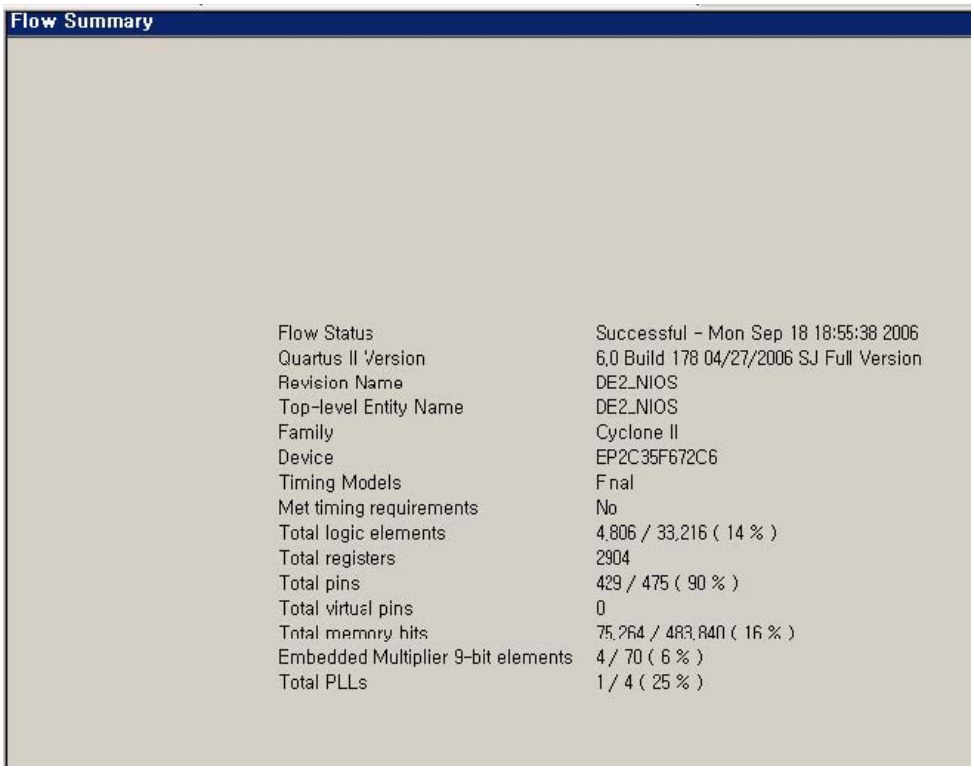


Figure 5 shows the debug messages sent from the DE2 board to the PC console through the JTAG connection. The messages show the various system processes.

Figure 5. JTAG Debug Messages

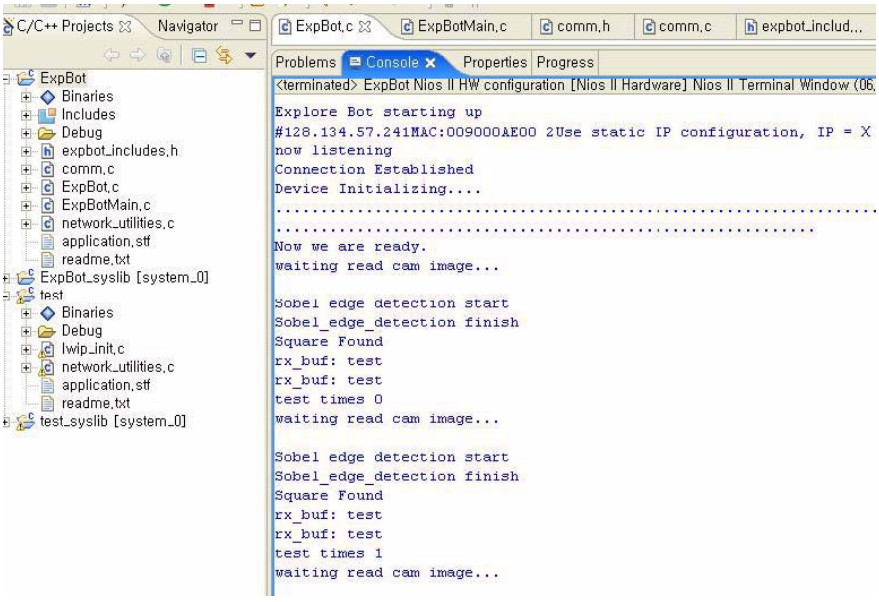
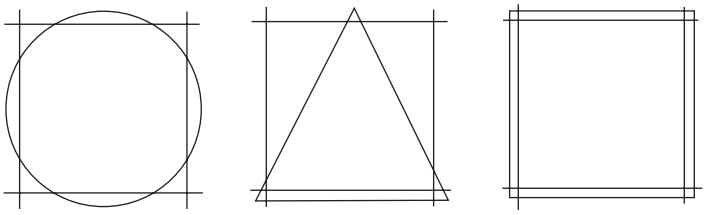


Figure 6 shows the original image and the edge-detected output resulting from the Sobel edge detection operation. Edge detection makes it easy to work with the image.

Figure 6. Sobel Edge Detection Result

We limited shapes in our test to circles, triangles, and squares. Figure 7 shows the character line and cross points for each shape.

Figure 7. Sonde Method Character Lines & Cross Points



After performing the simulation 30 times, our results showed that the error rate using the sonde method was lower than using pixel matching, as shown in Table 1. Table 2 shows statistics of the resulting 30 pictures.

Table 1. Sonde & Pixel Matching Error Rates

	Sonde	Pixel Matching
Error Rate	1 : 5	1 : 3

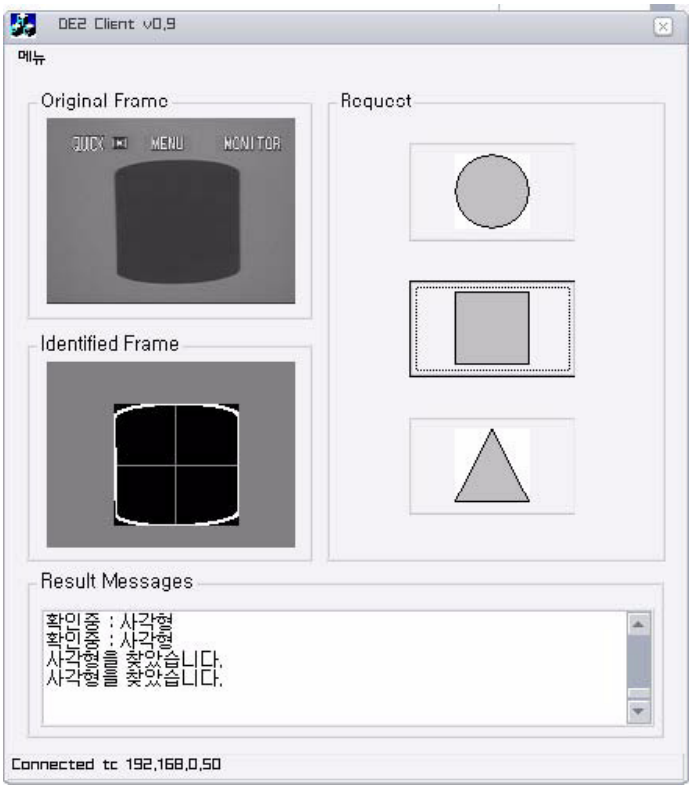
Table 2. TCP-IP & RS-232 Comparison

	640 x 480	160 x 120
RS-232	22 seconds per frame	2 seconds per frame
TCP-IP	2 to 3 seconds per frame	0.3 seconds per frame

Using an RS-232 connection, it took 22 seconds to transfer one frame, which includes the time required to process the image. In contrast, TCP-IP requires only 2 to 3 seconds to transfer 1 frame, a 10x decrease in transfer time. However, this transfer time is still far from real-time. Therefore, we resized the image to 160 x 120, which requires only 0.25 seconds to transfer 1 frame. Because 3 to 4 frames can be transferred in 1 second, we treated this rate as roughly real-time.

Figure 8 shows the resulting screen output by the program using an MFC on the client PC. The user selects the object to look for in the **Request** box. The **Original Frame** box shows a cylinder image and the **Identified Frame** box shows the processed image. The **Result Messages** box shows the messages that appear when the robot finds an object.

Figure 8. Client PC Captured Result



Design Architecture

Figure 9 shows a photograph of our demo and the DE2 board setup. The demo shows an RC car image captured through the camera during a demonstration of our project. The board setup shows the DE2 board and RF modules working together. We ignored any errors caused by the differences in the illumination made for demonstration.

Figure 9. Demo and DE2 Board

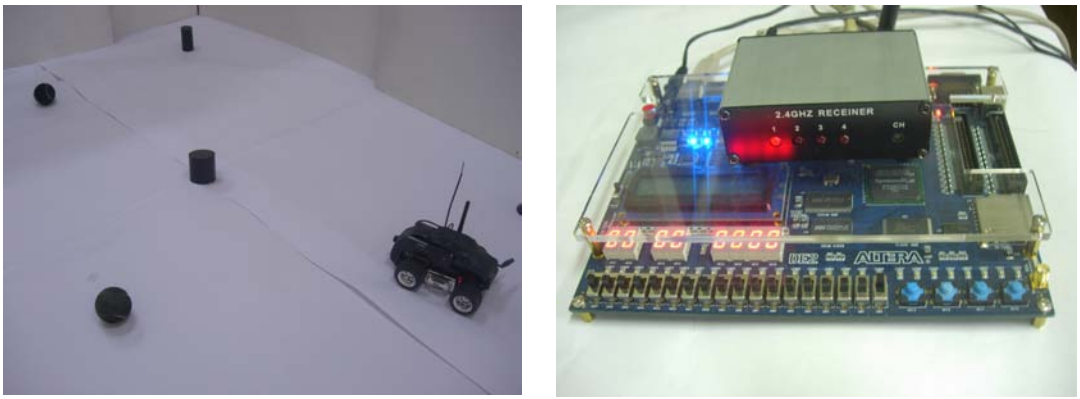


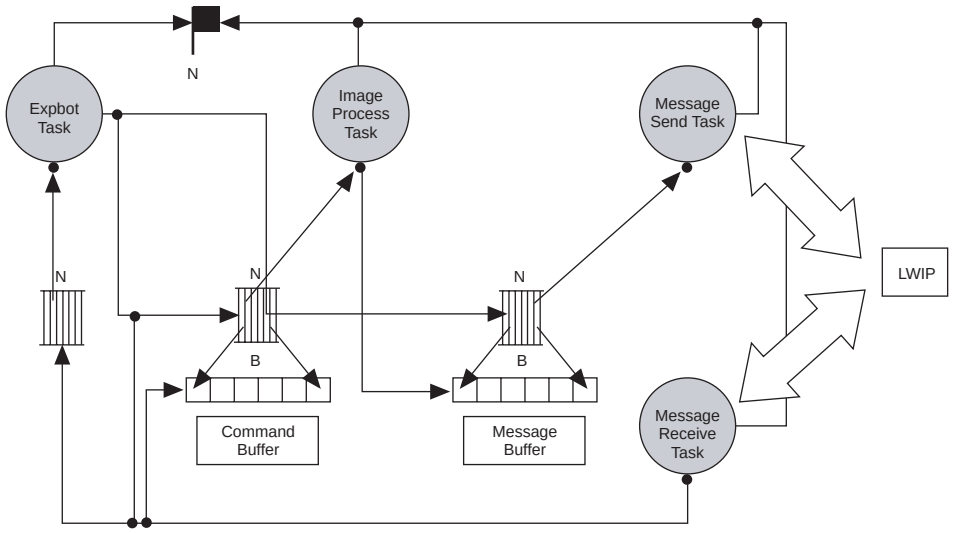
Figure 10 shows SOPC Builder and its components for the Nios II system. The project has one CPU, an LED, and a switch for simulation.

Figure 10. SOPC Builder Components

Use	Module Name	Description	Input Clock	Base
☑	cfi_flash_0	Flash Memory (Common Flash Interface)		0x0000000C
☑	sram_0	SRAM_16Bits_512K	clk	0x00400000
☑	cpu_0	Nios II Processor - Altera Corporation	clk	
	instruction_master	Master port		
	data_master	Master port		IRQ 0
	jtag_debug_module	Slave port		0x00480000
☑	epcs_controller	EPCS Serial Flash Controller	clk	0x00480800
☑	uart_0	UART (RS-232 serial port)	clk	0x00481000
☑	timer_0	Interval timer	clk	0x00481020
☑	timer_1	Interval timer	clk	0x00481040
☑	lcd_16207_0	Character LCD (16x2, Optrex 16207)	clk	0x00481060
☑	led_red	PIO (Parallel I/O)	clk	0x00481070
☑	led_green	PIO (Parallel I/O)	clk	0x00481080
☑	button_pio	PIO (Parallel I/O)	clk	0x00481090
☑	switch_pio	PIO (Parallel I/O)	clk	0x004810A0
☑	jtag_uart_0	JTAG UART	clk	0x004810B0
☑	dm9000a_0	DM9000A	clk	0x004810B8
☑	SEG7_Display	SEG7_LUT_8	clk	0x004810C8
☑	control	PIO (Parallel I/O)	clk	0x004810D0
☑	start	PIO (Parallel I/O)	clk	0x004810E0
☑	done	PIO (Parallel I/O)	clk	0x004810F0
☑	sdram_0	SDRAM Controller	clk	0x00800000
☑	tri_state_bridge_0	Avalon Tristate Bridge	clk	

Our software used multi-threading and an event-driven interrupt mechanism for speed and flexibility, as shown in Figure 11.

Figure 11. Software Flow Chart



Design Features

Our design features are as follows:

- *Hardware/Software Co-Design Accelerated the Design Process*—Image processing implemented with the DE2 board and Nios II processor is faster than general PC applications. The Nios II processor, Quartus® II software, SOPC Builder, and Nios II Integrated Development Environment (IDE) tools make hardware/software co-design fast and easy to implement and customize.

- *TCP/IP Increases the Image Transfer Rate*—It takes about 20 seconds to transfer data using an RS-232 connection. In contrast, three frames can be transfered per second using TCP-IP. We were able to provide a real-time image display by reducing the image size.

- *Wireless Network Using 2.4-GHz RF Link*—Wired networks did not work for long distances. We used a 2.4-GHz RF module to overcome distance restrictions. The wireless link makes the search robot ideal for dangerous unmanned search operations.

- *Wide Application*—Many unmanned search robots exist for specific purposes, e.g., military or space applications. However, this project is designed primarily for commercial applications because of its reasonable price and small size. For example, it could be used for a cleaning robot, security machine, etc. Using FPGAs and a wireless network makes processing fast and efficient.

Conclusion

We designed a system that provides image recognition and transfers three to four frames per second. By resizing the image to 160 x 120, we shortened the processing time and designed a hardware module to capture the Y-pixel information. Sobel edge detection makes image processing easier and the sonde method works quickly and enables accurate recognition. Our program is fast and flexible because it is multi-threaded and event driven. Anyone can easily use the application via the interface we designed with MFC.

We learned the following things while working on this project:

- For those who are skilled in software development, it is not easy to design a hardware system using an FPGA. But we learned how to implement our design one step at a time and it was fun.

- We spent a long time studying clock synchronization and implementing data path design control logic using Verilog HDL.
- Because of the small SRAM size, we could not save full-color images and we had to reduce the image size to work with the SRAM size. We also studied the YCbCr, NTSC, RGB, and bitmap standards.
- We found that it was not easy to use Ethernet to perform real-time transfers. The RS-232 connection was not easy to control because of the baud rate, flow control, etc. There was no easy solution, but we concentrated on the project with all of our efforts and were able to find a solution that worked for us.
- This project was an opportunity to study about image processing and RF theory.
- We found that we seldom had time for solving problems. When we did, we solved them one at a time and are proud of ourselves.
- We had to find a new algorithm for image recognition and image processing. We made many mistakes and did not have enough information for the task. The sonde algorithm worked well for us. It is the best algorithm for the project.
- We have not finished yet. We want to use a new RC car and use multiple CPUs for improved performance.

The Nios II processor-based system provided good performance for use in commercial applications and it offered us a multitasking environment. SOPC Builder is useful for implementing components in hardware easily, including a CPU and other system components. The Quartus II software and Nios II IDE made it easy to co-design hardware and software. The DE2 board is for educational purposes, but it could be used for small commercial projects.