Department of Electrical Engineering
University of Washington

E E 474 Lab 3

LKMs and Shift Registers

The report and its contents are solely our team's work and cites outside references for work that is not our own.

Signature:
Xiaoyi Ling, 1428502
Tiffany Luu, 1435225
Pezhman Khorasani, 1640743

TABLE OF CONTENTS

Introduction

In this lab, we used the Beaglebone's GPIO pins and a shift register to talk to an LCD display. We implemented this by using kernel mode to implement a character device driver. When the kernel module is inserted into the Beaglebone, it will initialize the GPIOs and display the characters on the LCD by sending input data through a shift register. The LCD display in automatically initialized on character driver startup, and commands from the kernel space were used to send strings inputted from the keyboard to the kernel out to the display. We also utilized the LCD's display shifting to view longer lines, to view all 40 characters on each of the two lines of the display.
Additionally, we added a photoresistor and a temperature sensor to the ADC inputs to read data from the environment. The data about the current temperature and brightness was printed through the device driver onto the LCD screen.

Design Specification

Shift Register and Kernel Mode

Input: Keyboard
Output: LCD display
Error check: The program is able to check if a file is opened successfully. If not, the system will print error messages to the terminal and exit.

This program will send what the user inputs from keyboard to the LCD display.

- The kernel module should be inserted first and the the program file should be created in the directory dev using the .ko file (use make ko1 and for our Beaglebone, make dev1).
- Run the test program that sends input to the LCD. The terminal will display the following instructions: "Enter input to send to the LCD display. Enter \n to go to the next line, or enter \c to clear the display". The user can enter input when "Enter input:" appears on the window.
- When user enters the input, the input will be sent to lcd device driver. The program will request the next input by printing to the terminal "Enter input:" again. On the other side, kernel will receive input from the user space and send the proper command to display the words to the LCD.
- The LCD screen displays 16 characters per line at a time. When a line passes the first 16 characters, the display will shift left when a new character is entered. When it reaches 40 characters, the cursor will move to the second line to print next character and display will shift back to the first 16 characters. When the second line has reached 16 characters, the entire display will again begin to shift until the second line reaches 40 characters. Then the entire display will be cleared and this process will start over again until the user stops inputting data.
- The LCD can identify a new line command and clear display command. When the user types "\n", it will goes to the next line. When the cursor is already in the second line, it

will clear the display the start over from the first line. When the user types "\c", the entire display will be cleared.
- The program will terminate when the user inputs an EOF character (Ctrl-D on Linux).

Temperature and Light Sensor

Input: Photoresistor and temperature sensor
Output: LCD display
Error check: The program is able to check if a file is opened successfully. If not, the system will print error message to the terminal and exit

This program will read inputs from a temperature sensor and a photoresistor. It will then convert the temperature data from mV to Celsius, Kelvin, or Fahrenheit depending on the user's choice. It will convert the brightness data to an overall percentage. It will then print this data to the LCD display.

- The kernel module should be inserted first and the the program file should be created in the directory dev using the .ko file (use make ko2 and make dev2).
- Run the sensor program. The terminal will display the following instructions: "How to display temperature? K for Kelvin, F for Fahrenheit, and C for Celsius" If the user gives wrong letter, "Invalid input. K for Kelvin, F for Fahrenheit and C for Celsius" will repeat and ask for input again.
- After getting an appropriate input, the program will convert the data from the sensors to the corresponding temperature and brightness values.
- The program will send the data for temperature and brightness to the screen through the LCD driver. The first line of the LCD will display "Temp: " followed by the temperature and then unit "°C", "°F" or "°K". The second line of the LCD will display "Brightness: " followed by the percentage and "%".
- The default setup of the program will update the data for 30 times and each pair of the data will stay for about half a second. When data is updating, only the numbers on the screen should change. After 30 times, the program will terminate and clear the display. The number of repeats is adjustable.

Hardware

Shift Register and Kernel Mode

In this lab, instead of connecting the GPIO pins to every DB input needed for the LCD display, we used a shift register to send this data instead. The pinout for the SN74HC595 register we used is shown in Figure 1 below. Input from the SER line into the shift register is shifted into the outputs Q_A - Q_H. The register shifts one bit of data in every time the SRCLK pin is triggered (goes from low to high, reads a posedge). The data can then be updated at the Q outputs every time the RCLK is pulsed. The Q outputs are connected to the DB pins of the LCD, the 8-bit command is sent.
The connections between the GPIO pins of the Beaglebone and the shift register are shown below in Table 1. The connections between the LCD Display and both the Beaglebone and shift register are shown in Table 2.

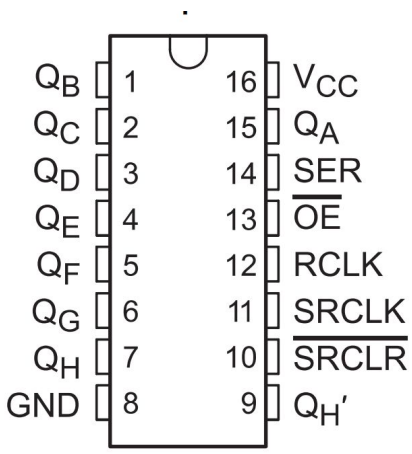

Figure 1: SN74HC595 shift register pinout

Table 1: Shift Register I/O Connections

Shift Register	Beaglebone/ LCD	Function
Q_B	LCD 8	Data 1: Output data to the LCD, DB1
Q_C	LCD 9	Data 2: Output data to the LCD, DB2
Q_D	LCD 10	Data 3: Output data to the LCD, DB3
Q_E	LCD 11	Data 4: Output data to the LCD, DB4
Q_F	LCD 12	Data 5: Output data to the LCD, DB5
Q_G	LCD 13	Data 6: Output data to the LCD, DB6
Q_H	LCD 14	Data 7: Output data to the LCD, DB7
GND	GND	Power: 0 V
$Q_{H`}$	Free	Inverse of Q_H, not connected
~SRCLR	SYS 5	Shift Register Clear, shift register is reset to 0 when the SRCLR is low, connected to 5 V
SRCLK	GPIO 20	Shift Register Clock, shifts data from SER into the shift register
RCLK	GPIO 115	Storage Register Clock (Latch), shifts data into the output register/updates the output
~OE	GND	Output Enable, shift register is enabled when the OE is low, connected to 0 V
SER	GPIO 49	Serial Input (to be shifted into the Q outputs)
Q_A	LCD 7	Output data to the LCD, DB0
V_{CC}	SYS 5	Power: 5 V

Table 2: LCD Display Connections

LCD	Beaglebone/ Shift Register	Function
1	GND	Power: V_{SS} = 0 V
2	SYS 5	Power: V_{CC} = 5 V
3	~0.05 V	Power: powers the LCD display. A potentiometer was used to set a voltage close to 0 V to make the display readable.
4	GPIO 60	Register Select: 0 to read/write to the instruction register, 1 to read/write data as characters
5	GND	Power: V_{SS} = 0 V
6	GPIO 112	Enable: Tells the LCD display when to perform an instruction, reads the posedge of the enable pulse
7	Q_A	Data 0: Input/output data to/from the LCD
8	Q_B	Data 1: Input/output data to/from the LCD
9	Q_C	Data 2: Input/output data to/from the LCD
10	Q_D	Data 3: Input/output data to/from the LCD
11	Q_E	Data 4: Input/output data to/from the LCD
12	Q_F	Data 5: Input/output data to/from the LCD
13	Q_G	Data 6: Input/output data to/from the LCD
14	Q_H	Data 7: Input/output data to/from the LCD, busy flag that tells when the LCD is running an instruction

Input from the user via keyboard is also sent from the computer to the beaglebone, which then translates to outputs to the shift register. The shift register sends the data to the LCD as described above.

Temperature and Light Sensor

We decided to add input peripherals to our project. The sensors that we used were the LM35 temperature sensor and the 200 kΩ photoresistor, shown in Figure 2 and Figure 3 respectively.

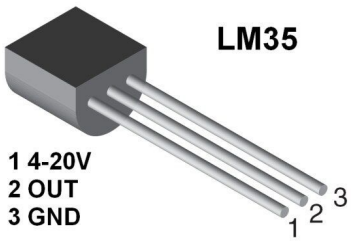

Figure 2: LM35 temperature sensor

Figure 3: 200 kΩ photoresistor

The analog input pins AIN0 and AIN1 are used for this part. The light sensor is in series with a 10kΩ resistor, and AIN0 is connected to the light sensor output across the 10kΩ resistor. The AIN1 is connected to the temperature sensor output (pin 2). A 10kΩ resistor was connected in parallel across the output and GND to stabilize the output. The model of the temperature sensor used is LM35. This connections for both sensors are shown below in Tables 3 and 4.

Table 3: LM35 Connections

LM35	Beaglebone	Function
V_{SS} (pin1)	SYS 5	Power: V_{CC} = 5 V
Out (pin2)	AIN1	Read input voltage from temperature sensor. For voltage, the value is displayed as millivolts. Output range: 0 - 1 V
GND (pin3)	~0.05 V	Power: V_{SS} = 0 V

Table 4: Photoresistor Connections

Photoresistor	Beaglebone	Function
Photoresistor/pin1	VDD_ADC	Power: V_{DD} = 1.8 V
Presistor pin2/ & 10kΩ resistor/pin1	AIN0	Read input voltage from photoresistor. For voltage, the value is displayed as millivolts. Output range: 0 - 1.8 V
10kΩ resistor/pin2	GND	Power: V_{SS} = 0 V

Software

The following is a short description of the C files and programs used to create a device driver via a kernel module, use the module and driver to implement the LCD display, and read input from a temperature sensor and photoresistor.

Makefile: Compiles the user input programs for the project and contains phony targets for some linux commands. Contains the following commands:

- make lcd: Compiles the program that talks to the LCD
- make sensor: Compiles the program that displays temperature and brightness to the LCD
- make run1: Runs the program lcd
- make finput: Runs the program sensor
- make ko1: Inserts the lcd kernel module
- make ko2: Inserts the sensor lcd kernel module
- make rko1: Removes kernel module for lcd program
- make rko2: Removes kernel module for sensor program
- make dev1: Makes the driver device file for lcd program
- make dev2: Makes the driver device file for sensor program
- make rdev1: Removes the driver device file for lcd program
- make rdev2: Removes the driver device file for sensor program
- make clean: deletes all the executable files, .o files, and text editor files

Shift Register and Kernel Mode

lcd.c: Contains functions for building the character device driver, including allocating space, opening and closing device, etc. Also contains functions for reading and writing to the driver, which sends commands to write to the LCD display. Includes the initialization process to get a blinking cursor, send what reads from the buffer to the LCD, return home, clear display, etc.

lcd.h: Header file for lcd.c. Initializes the functions in lcd.c, includes libraries and default values and instructions.

lcd.ko: Built from lcd.c. This is the kernel module file that is inserted into the Beaglebone.

test_lcd.c: Receives input from user by typing from keyboard (stdin). It sends the input to the character device driver provided from the lcd.ko module.

Temperature and Light Sensor

slcd.c: Contains functions for building the character device driver, including allocating space, opening and closing device, etc. Also contains functions for reading and writing to the driver, which sends commands to write to the LCD display. Includes the initialization process, send what reads from the buffer to the LCD, return home, clear display, etc.

slcd.h: Header file for slcd.c. Initializes the functions in slcd.c, includes libraries and default values and instructions.

slcd.ko: Built from slcd.c. This is the kernel module file that is inserted into the Beaglebone.

sensor.c: Contains functions for displaying the temperature and brightness, including asking temperature types, converting temperature from input voltage to degrees, converting brightness from input voltage to percentage, etc.
The input values from AIN pins of Beaglebone are in millivolts. For the LM35, the scale is 10mV/℃, so the conversion would be temperature = value/10 in Celsius. This can then be converted to Kelvin or Fahrenheit. For brightness, the overall voltage is 1.8V, so the percentage is value/1800.

Testing

Shift Register and Kernel Mode

Difficulties we met during the lab:

When we were trying to write characters to the LCD display, the characters would not show until the next time the program was run. For example, if the program was run with ./run, the screen would initialize properly, but the characters we tried to write to the screen afterwards would not appear. The second time we ran ./run again, the letters we tried to print previously would appear before the initialization process happened again. All of the other commands worked fine, such as initialization and clear display. We spent multiple days trying to debug using several methods such as checking the hardware by using multimeter to measure output voltage of each pin and shift register, changing breadboards, Beaglebone and LCD screen. We checked the software by changing timing delays, matching them to the previous code, checking the diff between the Lab 2 and Lab 3 code, and trying another group's code on our display. None of this worked. Eventually, we discovered that we had to add an extra function set and entry mode set command that was not required from the last lab in order to get it working.

Figure 4 shows an example of text successfully displayed on the LCD display. The user inputted "Hello, world!" from the keyboard, as shown below.

Figure 4. LCD displays "Hello, world!"

Temperature and Light Sensor

Figure 5 below shows an example of the sensor program. The LCD displays the temperature from the lab room in Kelvin (chosen by the user) and also it shows the brightness percentage of the environment around the photoresistor.

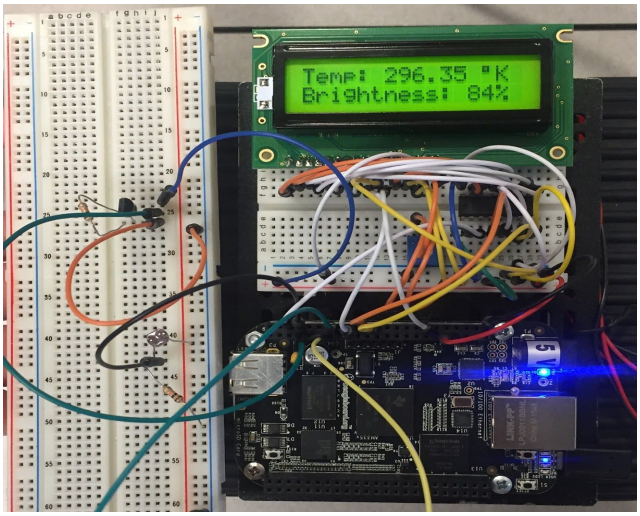

Figure 5. LCD displays temperature and brightness

Conclusion

Throughout this lab, we learned to create a device driver for our devices by writing and implementing a kernel module. We also learned to use a shift register to initialize an LCD display and send different commands to write and shift characters to the display, by programming GPIO pins from the Beaglebone with C code. Additionally, we practiced using the AIN pin by reading data from temperature sensor and photoresistor, and combining that function with commands to the LCD display. Although it was frustrating, through the testing we did to try to get characters to print on the LCD display, we practiced several different ways of debugging when we were trying to run our program and get the required output. Overall, this lab taught us another way to implement hardware, tested our debugging methods, and explore other peripherals for input.