Assignment 3 SPI Device Programming and Pulse Measurement (200 points)

Exercise Objectives

1. To learn the basic programming technique in Linux spi and gpio kernel modules.
2. To learn the application of kernel thread and interrupt handler to manage spi and gpio operations
3. To develop an application of distance measurement and animation display.

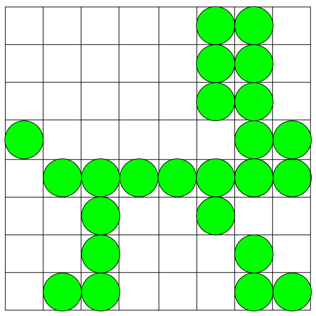
Lab Assignment

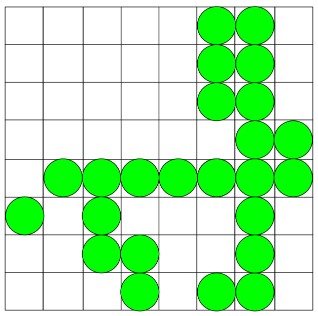
Many display devices are based on raster graphic image which consists of a rectangular matrix of pixels. To display an image, the ON/OFF (or intensity) data for all pixels is saved in a frame buffer and scanned in a fixed frame rate. In this assignment, we will use a simple 8X8 LED matrix to display patterns. The device is equipped with a MAX7219 driver. In addition to source current for LED display, the driver chip can buffer display data and scan digits using an internal oscillator.

Your program needs to create an animation by displaying a sequence of patterns on the LED matrix. To display a patter, 8 bytes of pattern should be written to the driver chip via SPI interface bus. Assume we will use the example definition of a patter:

```
typedef struct
{
        uint8 led[8];      // 8 bytes (64 bits) for 64  leds
} PATTERN;
```

By switching between the following two patterns, the LED matrix should show a running or walking dog. (This is an example display pattern. Please create something interesting, such as bouncing balls, for this assignment.)





The other device you will use in the assignment is a HC-SR04 ultrasonic sensor distance measuring module. After sending a “ping” signal on its trigger pin, the module sends 8 40khz square wave pulses and automatically detect whether receive any echo from a distance object. So the pulse on the echo pin width tells the time of sound traveled from the sensor to measured distance and back. The distance to the object is

Test distance = (pulse width * ultrasonic spreading velocity in air) / 2

To make your application interesting, you should use the measured distance to determine how fast your display object is moving (from walking to running) and the direction it moves (left or right when the distance object moves to the sensor or away from the sensor).

Part 1: A user application program for distance-controlled animation

The first task of the assignment is to develop a user application (possibly multiple threads) to enable a distance-controlled animation. You can use IO 2 and 3 of Galileo board to connect to the trigger and echo pins of HC-SR04. To detect rising and falling edges on the echo signal, you will need to “poll” or “select” GPIO interrupt events in your user-level program. To connect a SPI bus to the display module, you will use SPI1_SCK, SPI1_MOSI, and IO12 (as SPI1_SS) pins on the digital IO connector on Galileo board and spidev driver for data transfer to MAX7219.

Note that it is required to have few separate modules for IO operations, such as:

1. IO initialization
2. Control the display patterns on LED matrix
3. Measure distance from HC-SR04 sensor

Other than these modules, your main program should contain the application logic, including calculating distance and controlling display patterns in timely manner. It may be reasonable to have two threads to handle the operations on the two devices.

Part 2: A device driver for SPI-based LED matrix and pulse measurement

In the 2nd task of the assignment, you will move the code in device operation modules from user application to device drivers. For the SPI-based LED matrix display, you can develop a device driver that provides

- An IOCTL command to send in 10 patterns into a display buffer in kernel space.
- A write function to send in a display sequence (patterns 0 to 9 and durations in miliseconds), for instance, (0, 100, 3, 200, 0, 0) indicates an animation that displays pattern 0 for a duration of 100ms and then pattern 3 for a duration of 200ms. The last (0, 0) is the termination symbol of a display sequence. So, a sequence with only (0, 0) is to turn off the display on LED matrix.

For the pulse measurement using HC-SR04, a device driver should be developed in which:

- A write to trigger a measurement
- A read to receive the last measured pulse width (a 32-bit integer in micro-second), or -1 if the measurement is still on progress.

The main program in Part 1 can be revised to call the driver functions such that a similar animation based on measured distance can be performed.

Due Date

The due date for Part 1 is 11:59pm, Nov. 10. For Part 2, the due date is 11:59pm, Nov. 17.

What to Turn in for Grading

- Create a working directory to include your source files (.c and .h), makefile(s), readme file. Compress the directory into a zip archive file named cse438-teamX-assgn02-partN.zip. Note that any object code or temporary build files should not be included in the submission. Submit the zip archive to Blackboard by the due date.
- Please make sure that you comment the source files properly and the readme file includes a description about how to make and use your software. Don't forget to add each team member's name and ASU id in the readme file.
- There will be 20 points penalty per day if the submission is late. Note that submissions are time stamped by Blackboard. If you have multiple submissions, only the newest one will be graded. If needed, you can send an email to the instructor and TA to drop a submission.
- Your team must work on the assignment without any help from other teams and is responsible to the submission in Blackboard. No collaboration between teams is allowed, except the open discussion in the forum on Blackboard.
- Failure to follow these instructions may cause deduction of points.
- Here are few general rule for deductions:
  - No make file or compilation error -- 0 point for the part of the assignment.
  - Must have "–Wall" flag for compilation -- 5-point deduction for each warning.
  - 10-point deduction if no compilation or execution instruction in README file.
  - Source programs are not commented properly -- 10-20-point deduction.
- ASU Academic Integrity Policy (http://provost.asu.edu/academicintegrity), and FSE Honor Code (http://engineering.asu.edu/integrity) are strictly enforced and followed. A grade XE will be assigned to any cases of AIP violation.