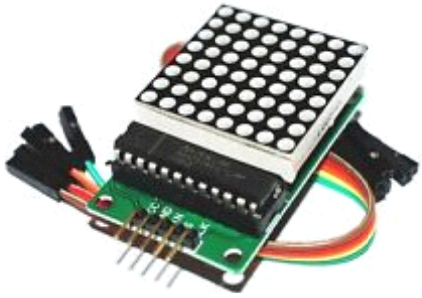
Assignment 3 SPI Device Programming and Pulse measurement

Exercise Objectives

1. To learn the basic programming technique in Linux spi and gpio kernel modules.
2. To learn the software structures of spi driver and gpio interrupt
3. To develop an application of distance measurement and animation display.

Lab Assignment

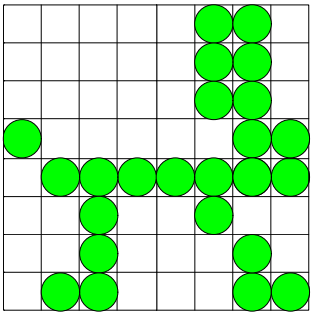
Many display devices are based on raster graphic image which consists of a rectangular matrix of pixels. To display an image, the ON/OFF (or intensity) data for all pixels is saved in a frame buffer and scanned in a fixed frame rate. In this assignment, we will use a simple 8X8 LED matrix to display 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.

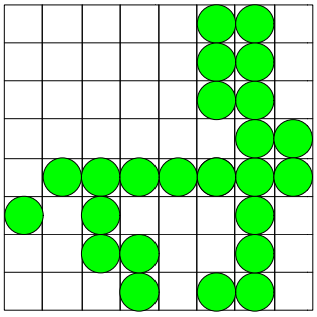


Your program needs to create an animation by displaying a sequence of patterns on the LED matrix. To display a patter, 8 bytes of pattern should be written to the driver chip via SPI interface bus. Assume we will use the example definition of a patter:

```
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.





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).

The first task of the assignment is to develop a user application (possibly multiple threads) to enable a distance-controlled animation. You can use GPIO 14 and 15 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_CS, and SPI1_MOSI pins on the digital IO connector on Galileo board.

In the 2nd task of the assignment, you will move some operations from user application to device drivers. For the SPI-based 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 measured pulse width (a 32-bit integer in micro-second), or -1 if the measurement is still on progress.

Due Date

This is assignment will be due at 11:59pm on Nov. 3.

What to Turn in for Grading

- Create a working directory to include your source files, makefiles, readme, ~~and a one-page report (for CSE 598)~~. Your source files should include the main program *main3_x.c* (the user-level program for tasks 1 and 2)*,* the drivers *spi_led.c, pulse.c,* and any other .c or .h files for the assignment. (please clean up all object code from your submission and we will recompile your code using your makefiles.)
- Comment your source files properly and rewrite the readme file to describe how to use your software.
- Compress the directory into a zip archive file named cse438(598)-lastname-assgn03.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 and time.
- Failure to follow these instructions may cause an annoyed and cranky TA or instructor to deduct points while grading your assignment.