Lab6

Character Device & MQTT

Objective

In this experiment, students will be exposed to:

- Different approaches for creating periodic tasks in Kernel space.
- Learn about communication between *Kernel space* and *User space*.
- Use the **MQTT** protocol to communicate between Server and Client to pass message over the network.

Prelab

For this prelab, you should investigate and include a brief description for the following

- kthread create, kthread stop, or ktime_get, ktime_set, hrtimer_forward, hrtimer_init
- broker/server, client, topic, publisher, subscriber.

Lab Procedure

Week-1

Part 1: Speaker function

For this part of the lab, you are to create a module that contains a “speaker function”. The purpose of the speaker function is to create a square wave that will activate the speaker on the auxiliary board. The speaker is connected to *GPIO 6* on the RPi. You can use one of these two different approaches, illustrated by the following examples:

1. **kthread test.c**: This module implements a “Kernel thread”. Examine the code, then compile and install it. Use the dmesg command to see the messages printed.

   Use this example as a basis for a kernel thread based speaker function. Experiment with different delays until you hear a nice sound. You may keep the printk statements for debugging purposes, but you should remove them eventually. You should not keep printks inside loops that go relatively fast.

2. **hrtimer test.c**: This module implements a “high resolution timer”. As before, examine the code, compile and install it. Check the messages printed using dmesg.

   Use this example to create your own timer for the speaker.

Post-lab Questions

Once the speaker module is ready, you will add an *Interrupt Service Routine (ISR)* to your module. The purpose of the *ISR* is to handle one of five events. The handler will change the frequency of the square wave being played by the speaker function. You will associate a different frequency to each of the buttons (lower frequency associated to button 1, highest frequency associated to button 5). Make sure that the sound produced for each case is distinct.

You could reuse your Lab-1 ISR.

Part 2: Character Device

In this part, you are going to write a device driver to communicate between user space program and the kernel module. The user space program may send five different notes, such as ”A”, ”B”, ”C”, ”D”, and ”E” to the kernel module. And the kernel module will change the frequency of the tone based the notes.

Check the files **Lab6 cdev kmod.c** and **Lab6 cdev user.c**. The first one is a Kernel module that shows how to create a “*Character Device*”, which is one way of achieving communication between kernel and user space. The module code also shows the functions that will be invoked when reading the *Character Device* and writing to it from user space. The second file is a template for a user space program that opens and uses the *Character Device*. Use these files as a reference for your programs.

Week-2: MQTT

In this part, you are going to use MQTT protocol to communicate between machines over the network. As you did in Lab-5, you are going to determine a winner-looser relationship with the other students’ boards in the lab. This time, however, the winner board sends the current note that it is playing to all the looser boards, so that all of them play the same note as the winner’s board. This requires a few steps:

1. First, you are going to start a broker (TAmachine @ “128.206.19.11”). And then write a client program which subscribes to a topic called Election. On this topic, you will receive messages like ”WHOIS”, ”VOTE”, etc from the *Lab6 ElectionOfficer* which will be the TAmachine.

2. You will follow the same steps as Lab-5 to decide the winner in the election. In addition to that, you must program your client to accept messages that begin with @. These messages represent one of the five notes to be played: @A, @B, @C, @D, @E.

3. If your board is a looser and it receives one of those messages, it must change the frequency of the note being played. To do this, your *user-space* program needs to communicate with the *kernel-module* that you created before. Your module should still change the notes when the buttons are pressed. In other words, your module should handle the GPIO interrupt AND be able to get data from user space.

4. If your board is the master, and a button is pressed, it sends the corresponding note to all the slave boards in the network.

Useful Commands

Run the commands on a terminal as per your needs.

- Test installation – `mosquitto -v`
- Check Mosquitto Broker – `ps ax | grep mosq`

References

- How to Install Mosquitto Broker on Raspberry Pi
- Using The Mosquitto pub and Mosquitto sub MQTT Client Tools