Lab#5: Intro Mover 4 robotic arm – Real Robot

Leonardo Fusser (1946995)

Objectives:

- Port a multi-thread Visual Studio project into a multi-thread Linux project.
- Implement a multi-thread solution to control a real robot.
- Visualize CAN packet using a sniffer.
- Test the distance measuring sensor unit.
- Use of FileZilla to transfer files to/from BBB.

Material:

Windows 10: FileZilla and Visual Studio.

Mover4 robot arm, dlink router, wi-fi usb dongle, BBB with image pre-install SD card,

RJ45 Ethernet Network cable, CAN bus adapter, DB9 can bus cable adapter.

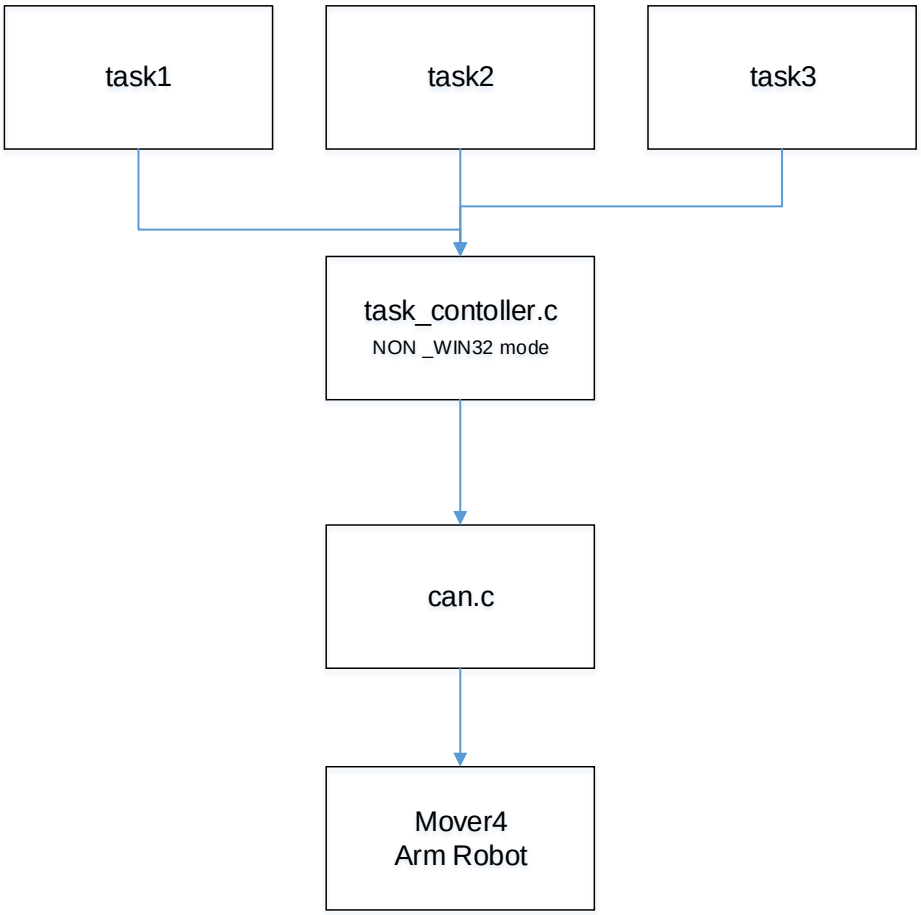
Warning:

- To avoid mechanical damage when testing, only soft object can be manipulated by the robot.

To hand in: No report to hand-in. Answers to all questions.

The structure of the lab is as follows:

- The task_controller call CAN bus APIs to communicate with the mover4 arm.
- CAN bus APIs are located inside the can.c file.
- All tasks must use APIs to communicate with the task_controller.



Lab Work

In this lab you must port your previous lab's multi-thread Visual Studio project into a multi-thread Linux project for BBB.

Setup

Follow the steps:

- Connect the wi-fi dongle – see annex.
- Enable wi-fi on your computer.
- Connect to the "mover4" router. The BBB boards have the following addresses:

  192.168.0.100
  192.168.0.101
  192.168.0.102
  192.168.0.103
  192.168.0.104
  192.168.0.105

- Using tera term log into one of the BBBs. The teacher will decide your address.
- Using FileZilla, log into the BBBs – see annex. Only one student can access the BBB at one time.
- Using FileZilla, copy all your files (**orange**) into mover4 folder. See the following hierarchy:
  - **root**
    - **can-utils**
    - **mover4_v6**
      - can.c
      - task_controller.c
      - log
      - state
      - ncurses_init.c
      - adc.c
      - **sol_mover4**
        - build
        - **mover4.c**
        - **taskKeybd.c**
        - **taskDisplay.c**
      - **header**
        - can.h
        - **public.h**
        - task_controller.h
        - ncurses_init.h
      - **demo**
        - exec

Notes: all non-orange file are already included in BBB.

Test on the real robot

Once your program has been approved by the teacher you can proceed with testing it on the real robot.

In task_controller.c, make sure the following line is disabled. This will disable the debug mode.
```
//#define DEBUG
```

Inside the build file, tweak the gcc line according to your source file's names.

gcc -pthread -lncurses ../task_controller.c ../can.c ../../can-utils/lib.c -lm -o $OUT_FILE

The gcc line must be modified to add your c source file's names:

*gcc -pthread -lncurses **your_task1.c your_task2.c** ../task_controller.c ../can.c ../../can-utils/lib.c -lm -o $OUT_FILE*

Then compile and test your code:

root@beaglebone/mover4_v6/sol_mover4#./build

CAN packet sniffer

Open a second console and run the following process:

root@beaglebone:~# ./candump can0

Take a screenshot of the CAN packets:

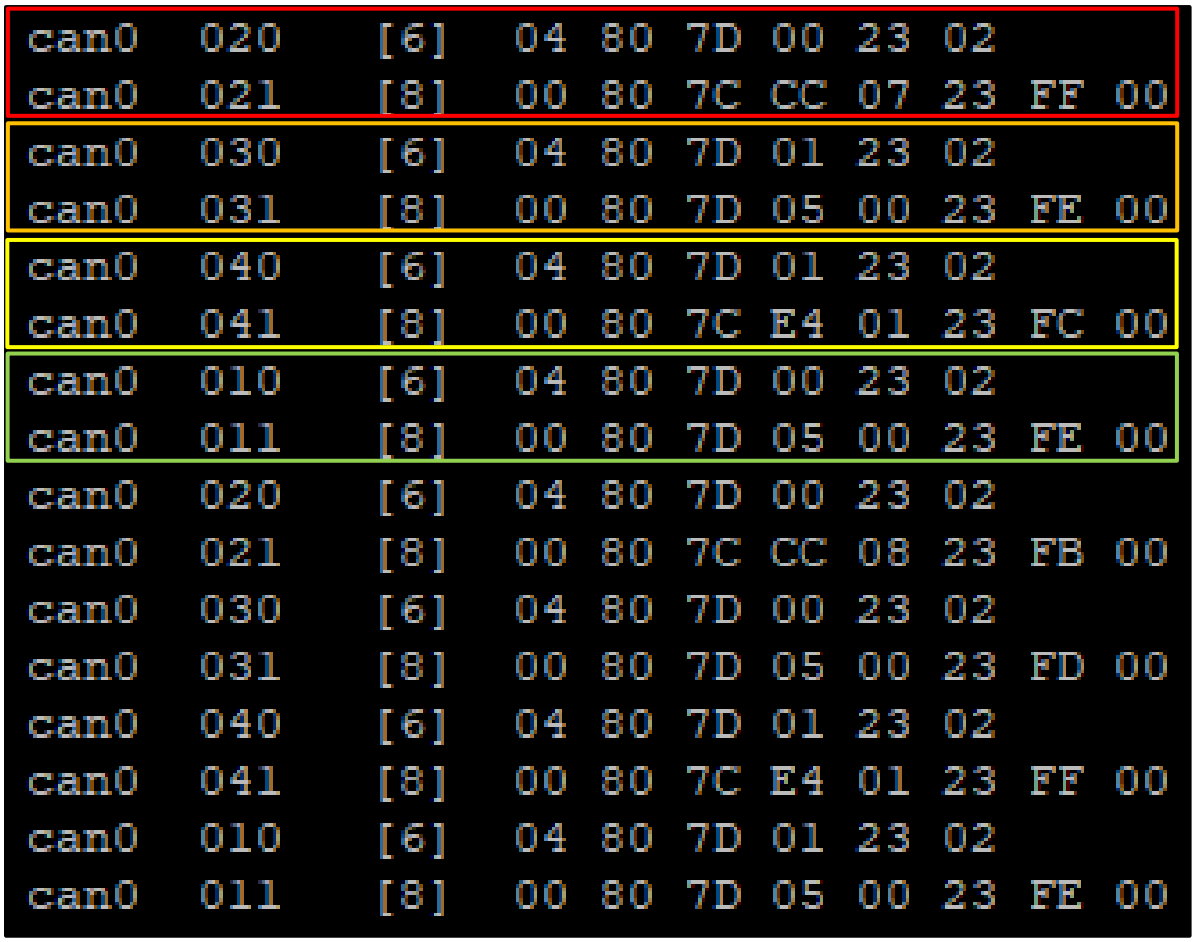


Figure 1. Screenshot of captured CAN packets from BBB. The frequency interval packets are sent and received is around 20Hz (50mS). All servo joints are shown above. In RED: shoulder joint send (0x20) and receive (0x21). In ORANGE: elbow joint sent (0x30) and receive (0x31). In YELLOW: wrist joint send (0x40) and receive (0x41). In GREEN: base joint send (0x10) and receive (0x11).

Distance Measuring Sensor Unit

Run the following program:

root@beaglebone:~# ./adc

It should display the sensor output.

Place a reflecting object (white paper) at different distances and fill in the following table:

Distance	**2cm**	**4cm**	**10cm**	**infinite**
ADC measure	1670	900	510	47

Give a demo to the teacher.

At the end of the demo, you must erase all your source files from BBB:

mover4.c, taskKeybd.c, taskDisplay.c and public.h

After lab questions:

Q1- Explain what you learned in this lab. What went wrong and what did you learn from your mistakes.

- The main thing that went wrong was when the code from Microsoft VS was ported to Linux on the BBB. The issue was that when the code compiled from the build file, it would not because the header files couldn't be located. After modifying the path for the header file in the code, it would compile and run properly.

  The other issue that was encountered was that if the thresholds for any of the joints on the Mover4 exceeded way past what they are supposed to be, it could lead to the entire robot locking up (even after clearing the state file, the robot sometimes remained locked up). The solution, a painful one, would require the teacher to manually reset the entire robot.

  Other than the most obvious facts (learned to port code to linux properly and to not exceed robot thresholds), the main takeaway of this lab was how to utilize pthreads to control a robot arm and how to use the Sharp photoelectric sensor to read distances.

Q2- What is the frequency at which packets are **sent**?

- The frequency at which the CAN packets are sent are around 20Hz (or around every 50mS). This is the absolute minimum frequency to avoid the CAN controller on the Mover4 robot to enter into an error state.

Q3- Explain the CAN packets sent and received – explain the meaning in term of angles, servomotors, etc:

- For the RED box in the screenshot in "Figure 1" above:

  [Send command]
  ID is the shoulder joint on the Mover4 (0x20).
  SetJoint command used (first parameter = 0x04).
  Set point (SP) is 0x7D00 (0 tics or 0°).

  [Receive command]
  ID is the shoulder joint on the Mover4 (0x21).
  No errors being reported (first parameter = 0x00).
  Current position (PV) is 0x7CCC (-52 tics or -0.8°).

- For the ORANGE box in the screenshot in "Figure 1" above:

  [Send command]
  ID is the elbow joint on the Mover4 (0x30).
  SetJoint command is used (first parameter = 0x04).
  Set point (SP) is 0x7D01 (1 tic or 1°).

  [Receive command]
  ID is the elbow joint on the Mover4 (0x31).
  No errors being reported (first parameter = 0x00).
  Current position (PV) is 0x7D05 (5 tics or 5°).

- For the YELLOW box in the screenshot in "Figure 1" above:

  [Send command]
  ID is the wrist joint on the Mover4 (0x40).
  SetJoint command is used (first parameter = 0x04).
  Set point (SP) is 0x7D01 (1 tic or 1°).
  The gripper is set to be closed or is closed already (last parameter = 0x02).

  [Receive command]
  ID is the wrist joint on the Mover4 (0x41).
  No errors being reported (first parameter = 0x00).
  Current position (PV) is 0x7CE4 (-28 tics or -0.43°).

- For the GREEN box in the screenshot in "Figure 1" above:

  [Send command]
  ID is the base joint on the Mover4 (0x10).
  SetJoint command is used (first parameter = 0x04).
  Set point (SP) is 0x7D00 (0 tics or 0°).

  [Reveive command]
  ID is the base joint on the Mover4 (0x11).
  No errors being reported (first parameter = 0x00).
  Current position (PV) is 0x7D05 (5 tics or 5°).

Appendix

Wi-Fi connection and router setting

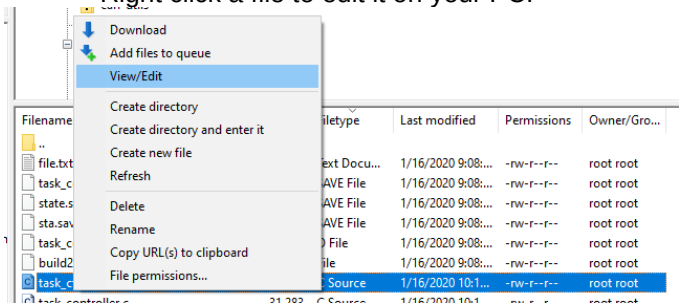
- Insert the installation CD.
- Install the drivers for the RTL8188CU model (5 to 10 minutes).
- Open the Realtek 11n USB Wireless Utility App.
- Connect BBB to mover4 router.

File transfer and file editing

- Open FileZilla.
- Connect to BBB. See screenshot below:

- Transfer all the files by dragging them into your mover4 folder. You must respect the proper hierarchy specified above.

- Right click a file to edit it on your PC:



- To associate .c and .h files click on the edit->setting tab: