ECE426 - Microprocessor Systems
Fall 2016

Final Project

I'd like you to build a system based on the discovery board and the other boards that you've been given that can act as a wireless mouse with aerial motion to control the movement of the cursor.

One board (Board 1) should act as the mouse. It should communicate information (raw sensor data or processed data) wirelessly to the other board. You should use touch buttons (preferred) or the keypad for the mouse buttons. Bear in mind that the user has to operate whatever you construct as a mouse.

The other board (Board 2) should be connected to the desktop using the USB connection. The communication with the desktop should be configured so that the input is interpreted as a mouse (see the example provided by Harsh).

Board 2 should also use the 7-segment display to provide status information about the mouse.

You have quite a lot of freedom in your design. For example, you have access to an additional board that has 9-degrees-of-freedom sensors (gyroscopes, accelerometers). This might allow you to do much smoother tracking of the "mouse" movement. It may also allow for a better user interface since you can probably more easily combine it with push-buttons.

Functional Requirements

Ask questions this week on the discussion board to make the functional requirements more concrete. I will expect at least one question from your team. Failure to ask a question will result in -0.5 mark deduction for the initial design document.

I strongly recommend having a fallback plan for the wireless. If you make a sincere effort, and document that effort, then there will be no penalty if you do not have wireless working but have a wired system instead. Decide ahead of time how many hours you will allocate to the wireless work.

Assessment

During the semester, we have introduced features that you will use in the project (timers, ADC, servo motors and RTOS). In the project, the focus will shift towards real-world problem analysis, embedded software design and conducting experiments to help configure the system towards achieving satisfactory performance. In particular, we are interested in achieving high stability,

accuracy, performance, and better coordination with the peripheral hardware. We are also interested in optimizing with regard for constrained system resources (meeting time requirements, storing and drawing graphical components, offloading CPU by using DMA whenever possible).

In grading the project, I will pay particular attention to your design process and the associated documentation. This is as important (or more) to me as the final product. I also expect to see carefully documented evidence of performance testing and validation. You should test each module (e.g. wireless, accelerometer, gyroscope). You should then test subsystems (once you have integrated modules together). All of these tests must be documented professionally – when did the test take place, what was the purpose of the test, what was the expected output, what was the output. Failed tests should be recorded too.

You have all taken ECSE 211. You know what the engineering process involves. I expect you to apply what you learned then in this course. I will be expecting to see design documentation for both the hardware and the software – flow diagrams, thread interaction, wiring diagrams, etc. I will expect to see documentation concerning project management and planning.

Deliverables

Monday Nov. 14, **before midnight**: 2 page (max) design document that provides a summary of functional requirements, specifies in more detail exactly what you will deliver, identifies how much time each member of your team will spend on the project, and how you will allocate that time. **[3 marks]**

Monday Dec. 5 (or earlier): Demonstration (book a 40 minute session with me). Since things can go wrong on demo day, I recommend making a video of an operational system. A short slide deck also generally helps you present your design decisions and to explain how your system operates. You need to have all design and testing documentation at the demo. The code must be commented – I will ask you to take me through it, and penalize you if it is not modular, clearly commented, and relatively easy to understand. **[25 marks]**

Monday Dec. 5, before midnight : Project report. There are no restrictions on the length of the report, but you should be concise in your explanations. You must explain your design decisions and provide a summary of the operation of your system and the testing. You should include diagrams illustrating the components in your system and how they interact (hardware, firmware, and software). You should not include all design or testing documentation in the main body of your report – you can include it as appendix. Code should be submitted at the same time as the report but in a separate zipped file. **[12 marks]**

Appendix: Wireless interfacing, firmware and testing

A high priority task should be establishing wireless communication. This can be quite challenging, although the testing beacon in the lab makes it easier than it has been in the past. Each group will have TI MSP430-CC2500 transceiver modules to work with. Each transceiver

should be interfaced to one of the boards. Initially, you have to physically connect the chipset to the F4 Discovery boards. You need to connect the 4-wire SPI and power to the chipset (and possibly configure and connect any interrupt pins). Consult the reference material for the F4 Discovery board and the MSP430 for schematics and pin layout to carry on the wire-wrapping. The tutorial will cover all you need to know on interfacing.

Secondly, you need to write the drivers for communicating with the transceiver via the SPI interface. The drivers need to be modular in design and function correctly at all times. The drivers are similar in concept but slightly different in implementation than the ST ones as described in the tutorial. For details on interfacing the CC2500 via SPI, refer to the cc2500 design notes. The drivers you will write must be modular. That is, you should provide a low-level set of drivers that use the SPI interface (i.e. single read or write) and high-level drivers (i.e. packet read or write) which make use of the lower-level drivers. As a reference, see how the accelerometer drivers are written. You need to write the initialisation functions to setup (configure) the cc2500 and operate it.

Thirdly, to test your drivers, you should attempt to read any of the internal status registers like VERSION or PARTNUM. You can also write into a register and then read it back (also test for burst reads or writes). At this stage, all you are testing for is SPI communication between your discovery board and the wireless module. No wireless transmission is being tested at this stage. To test for wireless communication, we will place a transmitter beacon which will continually send a pre-determined pattern. This will help you test your receiver. Once you make sure that your receiver is working correctly, then you can proceed in testing your transmitter.

Fourth, the wireless chipset configuration you need to program the chipset with is listed in the appendix. You will use this configuration as is in your project with few changes (i.e. communication channel, IOCFG register for enabling interrupts, etc.) as described in the tutorial. Do not concern yourself with the technical communication-related jargon unless covered in class or tutorial. Anything that has not been covered is out of scope of the current course. What you need to know is that each group should use a channel frequency that is different from the other groups to avoid conflicts. Each group should set up its frequency as Base frequency (**2433** MHz) + (Project group number * 8) (KHz).

Fifth, since you will be using multiple boards on the same group carrier frequency, you should enable node addressing features. This is important since transmitted data will be received by all nodes in your system. However, you might need your data to be processed by one node and discarded by the other. Enabling node addressing and setting up addresses for your transceiver modules helps solve this issue on the hardware level with minimum to no high-level software interference. When a node receives a packet that is addressed to a different node, it simply discards it. Transceivers only process the packets that are explicitly sent to them.