ECE426 - Microprocessor Systems
Winter 2014

Final Project

Introduction

For the final project, you will build a platform for 3D board orientation control. First, you are required to set up a closed loop feedback control system to orient one board using servo motors to match the 3D space orientation of another freely moved remote board. The reference board and platform board will communicate through a wireless link. The second application is to manually input a sequence of spatial tilt angles at run time for the base board to follow. Two sets of bonuses (one is a pre-requisite for the other) will add interesting extra features. The project will merge many of the elements you've built in the labs through the semester into a cohesive and practical objective with the addition of a few new modules which you will have to develop or use.

Since during the semester we have offloaded many of the features previously reserved for the project part, the focus will shift slightly towards embedded software design. In particular, we are interested in achieving high stability and performance, better coordination with the mechanical system components / structural design and their imperfect effects on overall system design,. We are interested in optimizing with regard for constrained system resources (i.e. in stage 1 bonus, how to achieve the longest recording period yet in the smallest memory footprint) and/or power.

Project Details and Design

Platform mechanical/structural design

Your application platform consists of two parts: a remote free board and a fixed base board. You are to construct your base board platform using any means you deem appropriate. Options include hard card board, Lego, wood, strings or pulleys among others. Unfortunately, we don't provide any of the structural/mechanical components or building blocks so you have to get them on your own. Be creative yet cost effective. Attempt to construct the base platform from any material you can readily get rather than buy any. Be a MacGyver (If you don't get it, check urban dictionary).

A fixed system where the board is rigidly attached to the controlling servo motors might be restricted in its orientation movements (unless you devise creative ways to avoid this). A system based on pulleys and strings (to move the board) controlled by servo motors might have higher freedom of movement (better). Yet, this loose structure might be slightly harder to control (find solutions). Either way, you system is only sensitive to the roll and pitch angles you have been working with in previous labs.

In any case, bear in mind that your platform should be solid in the sense that it does not contribute significant sources of error. It should not affect the speed of convergence towards a stable accurate system. You might need to investigate how your physical platform might introduce such sources of error and devise software solutions to tackle them. **You will be judged on the degree of freedom your system can achieve and its performance.**

Wireless interfacing

In designing your system, you will need to start with breaking up your project into several modules to work on. The highest priority is establishing wireless communication. To this end, each group will have two TI MSP430-CC2500 modules to work with. Initially, you have to physically connect the chipset to the F4 Discovery board. You need to connect the 4-wire SPI and power to the chipset (and possibly configure and connect any interrupt pins). Consult F4 Discovery board and the MSP430 for schematics and pin layout to carry on the wire-wrapping.

Secondly, you need to write the drivers for communicating with the transceiver via SPI interface. The drivers need be modular in design and function correctly at all times. The drivers are similar in concept but slightly different in implementation than ST ones as described in the tutorial. For details on interfacing the CC2500 via SPI, refer to the cc2500 design note.

Thirdly, you need to write the high level functions to setup (configure) the cc2500 and operate it. Finally, 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 transmissions).

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. channel, IOCFG for interrupts ... etc.) as described in the tutorial. Do not concern yourself with some of the technical communication related jargon. This 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 setup its frequency as Base frequency (2433 MHz) + (Project group number * 8) (KHz).

More on Modes of operation

As introduced earlier, your system should be able to execute in two different modes, (the method of mode switching is left up to you):

1. *Real Time board tilting:* The remote board will freely move in 3D space transmitting its angles to the base board. The base board should accurately follow the remote board. Even though this sounds simple, you might wish to revisit your calibration / filtering techniques since both boards will work in tandem and accuracy is a prime concern. You are free to research any techniques to suit your ends. Do not forget to factor in the possible added error from your base platform structural construction. The more degrees of freedom you add to board movement might possibly add more time to the board to stabilize towards the correct orientation. Remember all this should be done in real time with strict deadlines.
2. *Movement sequence input:* The second mode of operation requires that you input through the use of a keypad a sequence of strings in the form of $<\alpha, \beta, \Delta\tau>$ where α, β are the roll and pitch angle pair and $\Delta\tau$ is the time specified for the board to move from position $< \alpha_{t1}, \beta_{t1} >$ to $< \alpha_{t2}, \beta_{t2} >$

where $\Delta\tau = t_2 - t_1$. The board is always assumed to start from position $\alpha = \beta = 0$. The movement sequence terminates in the initial position. To avoid abrupt movement changes between positions, you should in your software do some sort of interpolation between the start and end points to get a vector of angular steps. This will smooth your transitions and add more stability to the system. You should investigate which interpolation techniques best fit this problem. Note that you should never exceed $\Delta\tau$. For demonstration purposes, a sequence of four to five movements is enough. Assume integer angles as input. The angles should fit within the degrees of freedom you have physically constructed your base platform to achieve.

Coding and Safety

As before, all your code should be written using RTOS threads and services. It should work in real time, be responsive, thread safe and exhibit no unintended behaviour. Your code should be event/interrupt driven.

UI

You should continue to use the user interface components from previous labs to convey information to the user. Feel free to use the LEDs, LCD, and the user button. In addition to that, you are required to use the keypad for your second mode of operation. A tutorial on how to use the keypad was already provided or you and you need to write the driver for that. **Interestingly, based on the wireless signal strength, the boards could tell how far apart they are from each other. You could use this as means of input to your system.**

Bonus Stage 1 (Up to 10% of the total project mark)

In addition to the core two modes of operation above, the first stage bonus is to add a third mode of operation as follows:

3. *Record and play option:* this feature lies in between the first two options. The base board will record the movement of the remote board and reproduce the movements. Since the internal memory is limited, you might need to find ways to compress the data stream stored in order to reproduce long movement sequences. One way you could borrow from above is to take snapshots of the remote board orientation and store them. You can use interpolation again to reconstruct the missing samples. Note that the sampling frequency of the snapshots affect how faithfully you reproduce the movement pattern. You don't want to miss samples. **You are encouraged to find better solutions to compress the recording stream. Better solutions will be awarded higher bonus marks.**

Bonus Stage 2 (Up to 10% of the total)

You can only attempt to do this bonus only if you implement the first bonus. Skipping the first bonus and directly implementing this stage will give you zero points.

You can choose one of the following suggestions or add your own.

Our list of extra bonus ideas:

- Use DMA to handle data from the MEMS sensor (Medium difficulty)
- Use the Cryptographic processor block in the ST32MF407G SoC to encrypt/decrypt the wireless data stream. Drivers are provided by ST. (Medium difficulty)
- Use the processor cyclic redundancy check (CRC) calculation unit to verify data transmission. At the same time, enable the Watch dog timer for fault recovery and make sure you clear it at appropriate intervals to avoid unnecessary/destructive system resets. You need to analyze where to best place your reset code. Drivers are provided by ST. (Medium difficulty)
- Use the USB interface to send control data to the board for part two from PC (High difficulty). You will need to write your own PC program to handle communication between PC and board
- Or suggest and idea of your own. You are required to come up with a new feature different from the list above or any of the project requirements and research/investigation you have to carry. Your feature should lean to be on the medium difficulty side. **No two groups should have the same idea.** This, ensures your project uniqueness. You could either:

    1. Introduce a hardware component of your own and use it in your application
    2. Add a certain feature to any mode of operations (other than the required optimizations, algorithms, stability and performance features)
    3. Add a fourth mode of operation
    4. If you have background in software testing/validation/verification, use and show literature formal methods applied to your design. Note, this is not the same as debugging. If you don't know what I am talking about, skip this part. ☺

The harder and/or more creative your idea is, the higher bonus point you will be awarded.

Components

Onboard:

USB

The STM32F4 chips have a USB 2.0 OTG peripheral with a full-speed PHY. The Discovery board exposes this through a micro-USB connector, allowing for USB-based applications to be developed. ST provides drivers for CDC (serial port) and HID (human interface device) classes but USB can be very difficult to implement and you are encouraged to explore the implementation details further before proposing it as a feature in your application. Also keep in mind that it is likely that you would have to write some PC software to accommodate.

Extras

HD44780 Character LCD

You will be provided with a Hitachi HD44780-compatible 2x16 character LCD that can be used in one of the special features requirements. They follow a simple protocol which is well documented for outputting text and very basic graphics. You are free to use these in your project.

Keypad

We have a lot of telephone-style keypads which are very useful for providing events for arm control or for precise movement input / correction.

Application notes on using HD44780 LCD and keypad have been written for you to assist in writing your drivers.

RF2500 Wireless Kits

The TI wireless kits are a required component of your final project. They don't have enough bandwidth to move large amount of data but simple control signals can be sent to coordinate the boards. One nice feature is that they can indicate the received signal power and this can provide an estimate of distance between the two systems.

Anything Else

You can add your own parts as well. There are plenty of free I/O pins to add whatever you want. For fairness, you can only be judged based on innovative design and the difficulty of implementation for whatever you may add, as of course not everybody has access to the same resources.

Demonstration (course graduation day)

All presentations will be held on the same day during the assigned hours (TBA). No early demos will be accommodated. The schedule of the final presentation will be posted on the web. Presentations will take place in one of the last days of the semester. The professor not the TAs demos the projects and grades your reports, so make sure that your presentation is clear and comprehensive.

You are also required to prepare a slides for the presentation. Your presentation should around 20 slides (±3). You demo time will be between 20-40 minutes per group.

Note that this demonstration is different than the usual ones. You will be presenting to the professor the sum of your knowledge of this course. You are expected to do so in a professional manner. You should be comprehensive in your presentation yet precise and to the point (find a balance). Never exceed your allotted time. Rehearse if needed.

Sometimes, things could go wrong on demo day (for example, a certain component fails, you break the platform ... etc.). Make sure to have a video of yourselves and your fully working project just in case.

Finally, the professor will not wait for you to finish a last minute implementation of a feature or debugging. BE READY on demo day. This project is not to be taken lightly. Setting up the wireless interface on its own could take a week or 10 days. With many sources of input, there is lots of sources of error and countless hours of debugging.

Report

The final project report is supposed to be **more formal** than the lab reports which you wrote during this semester. You should consider all feedback you've received this semester while preparing your report.

In particular, we would like to stress the need for the extra following points in your report:

- The components in your system
- A timeline of work and a breakdown between team members
- A block diagram of your firmware, showing roughly how modules interact

As always, you should explain the reasoning behind your design choices.

IMPORTANT – Returning your kits

All groups should return all components to ECE labs **within one week of the final project demo.** The parts returned should exactly match the ones given to you (specs, models and part numbers). They should all be in a fully working condition. This includes all boxes, kits (discovery and wireless), tools (screwdrivers and wire wrappers), breadboards, peripheral components (LCD, 7-segment displays, keypads, and motor kits (with all brushes and horns intact and complete)) and the **80 wire connectors**. The breadboards returned should have no components mounted on them. ECE labs always tracks and checks your kits and keep us informed. On the occasion of your failure to return the kit **on time**, or having any missing components, **you will be penalized 40% of the total project grade**

Disclaimer – FINAL GRADES

We, the TAs of ECE426, have no involvement whatsoever in relation to your final course mark. Our involvement does not go beyond grading your quizzes, lab reports and demos. The course lecturer (in this case the professor) is responsible for grading the reports and final project demo out of 40+. Your grades in the quality form (A, B^+, B, B^- … etc.) are granted by the course lecturer. Therefore, all your follow up should be directed to the professor. We, the TAs, promise that we will have your grades out of 60 be available at least one week before your final project demo day. This will allow you to submit any revision requests before your grades become final and rendered unchangeable. All your request for grade revision should be within the final week before the project.

APPENDIX – Wireless Configuration

```
#ifndef SMARTRF_CC2500_H
#define SMARTRF_CC2500_H

#define SMARTRF_RADIO_CC2500

#define SMARTRF_SETTING_FSCTRL1         0x0C//0x12 //Frequency offset = 457kHz
#define SMARTRF_SETTING_FSCTRL0         0x00
#define SMARTRF_SETTING_FREQ2           0x5D // Carrier Frequency is 2.433GHz
#define SMARTRF_SETTING_FREQ1           0x93
#define SMARTRF_SETTING_FREQ0           0xB1
#define SMARTRF_SETTING_MDMCFG4   0x0E //0x2D // BW of channel = 541.666kHz
#define SMARTRF_SETTING_MDMCFG3   0x3B // Baud Rate = 125kb
#define SMARTRF_SETTING_MDMCFG2       0x73 before demodulator, MSK modulation, 16/16 sync
word bits detected
#define SMARTRF_SETTING_MDMCFG1    0x42 /
#define SMARTRF_SETTING_MDMCFG0   0xF8 // Default Channel Spacing of 200kHz
#define SMARTRF_SETTING_CHANNR      0x00 // Channel 0
#define SMARTRF_SETTING_DEVIATN    0x00 //0x01 // 1785kHz
#define SMARTRF_SETTING_FREND1     0xB6
#define SMARTRF_SETTING_FREND0     0x10
#define SMARTRF_SETTING_MCSM0      0x18 // Automatically calibrate When going from IDLE to
RX or TX (or FSTXON) check CC2500 datasheet
#define SMARTRF_SETTING_FOCCFG     0x1D // check datasheet
#define SMARTRF_SETTING_BSCFG      0x1C
#define SMARTRF_SETTING_AGCCTRL2   0xC7
#define SMARTRF_SETTING_AGCCTRL1   0x40 //0x00
#define SMARTRF_SETTING_AGCCTRL0   0xB0
#define SMARTRF_SETTING_FSCAL3     0xEA
#define SMARTRF_SETTING_FSCAL2     0x0A
#define SMARTRF_SETTING_FSCAL1     0x00
#define SMARTRF_SETTING_FSCAL0     0x19 //0x11
#define SMARTRF_SETTING_FSTEST     0x59
#define SMARTRF_SETTING_TEST2      0x88
#define SMARTRF_SETTING_TEST1      0x31
#define SMARTRF_SETTING_TEST0      0x0B
#define SMARTRF_SETTING_FIFOTHR    0x07
#define SMARTRF_SETTING_IOCFG2     0x29
#define SMARTRF_SETTING_IOCFG0D    0x06
#define SMARTRF_SETTING_PKTCTRL1   0x04
#define SMARTRF_SETTING_PKTCTRL0   0x05 //0x05 // Fixed Packet Length (0x05)
#define SMARTRF_SETTING_ADDR       0x00 // Global Broadcast Address
#define SMARTRF_SETTING_PKTLEN     0x0A // Packet Length of 10bytes (0xFF)

#endif
```