The University of Queensland
School of Information Technology and Electrical Engineering
Semester One, 2013
CSSE3010 / CSSE7301 - Project 2
Due for demonstrations: your prac session - week 10 (6,7,10 May), 2013
Milestones Due – week 8 & 9.
Weighting: 20% of your overall mark (CSSE3010)

Project 2 – Automated Tracking System

Objectives:

- To apply your knowledge and skills developed in practicals 1 to 5 into a collective design.
- To accurately and quickly track a fiducial marker with your pan and tilt camera system.
- To learn and use the SD card library to store files.
- To calculate real time geometrical coordinates of fiducial markers to estimate distance data.
- To learn how to integrate the use of the Command Line Interface (CLI) library.

Introduction:

This assignment will build on your knowledge from practicals 1-5. You will be working in individually for tasks 1 – 7 but you will need to work in groups of two if you want to attempt challenge task 2B. Project 2 is designed to have you demonstrate that you have understood practical content and that you can additionally integrate some new functionality into your program consisting of fiducial marker tracking, the uSD card and CLI. The basic requirements will give you 15% of 20%. To get full marks, you must attempt two challenges.

Milestones:

There are two milestones that you must pass in weeks 8 and 9. The milestones require you to implement basic functionality for the project and are used to track your progress with project 2.

Milestone 1 (Week 8) Pass/Fail: A) Able to demonstrate Reactivision control of your servo. B) Able to answer questions stated in tasks 6.

Milestone 2 (Week 9) Pass/Fail: A) Able to demonstrate basic CLI commands and uSD card usage. B) able to answer questions stated in task 7.

Required Tasks:

NOTE: Each individual requirement as listed is marked as 0, 1 or 2 representing little to no functionality, reasonable functionality and complete functionality of task respectively. If you wish to implement an additional feature, consult your tutor and ask if this can be included as a challenge task.

IMPORTANT: This project **MUST** be implemented using FreeRTOS.

(2 marks) Task 1: Fiducial Marker Tracking - You should be able to track a fiducial marker with your pan and tilt camera. This will use Reactivision. Your pan and tilt should not overshoot the marker and you may wish to use a bounding box to reduce jitter. The speed at which you are able to track your marker contributes to the marking criteria for this task.

(2 marks) Task 2: SD Card Logging – Your platform should log all incoming data packets and store them in a human readable text file named studentNumber_vlogNumber.txt. Note the studentNumber consists of the last 4 digits of your student ID Number. Every time your board is rebooted your system must check for existing log files and create a new log file, incrementing the logNumber as appropriate. For example, if your student number is 41234567 and you have rebooted your system 4 times resulting in 3 existing log files, your new log file should be named "4567_v4.txt".

The format of the file is fairly open for your adjustment, however, should include the following entries as a minimum.
timestamp: x_position, y_position, pan_degrees, tilt_degrees \n

The timestamp should simply relate to the number of seconds since the platform has been turned on, however, should be formatted as "Hours-Minutes-Seconds".

You are expected to explore the SD functionality without the aid of a practical. Example usage of the library is included in the source directory from the repository (main.c file under the SD card folder). **NOTE: there is a 1-mark deduction if you cannot use the ls and tail commands to display your SD card files.**

(3 marks) Task 3: Distance Estimation – When two markers are introduced to Reactivision, it relays information to your platform consisting of both markers. When two markers are at a fixed distance apart (printed on a single A4 page), this can be used to geometrically calculate the 'z-axis' distance from the camera. You will be asked to explain your geometric logic and check the real versus estimated measurements forming the basis of the mark for this task. You should use the terminal interface to relay real time calculations.

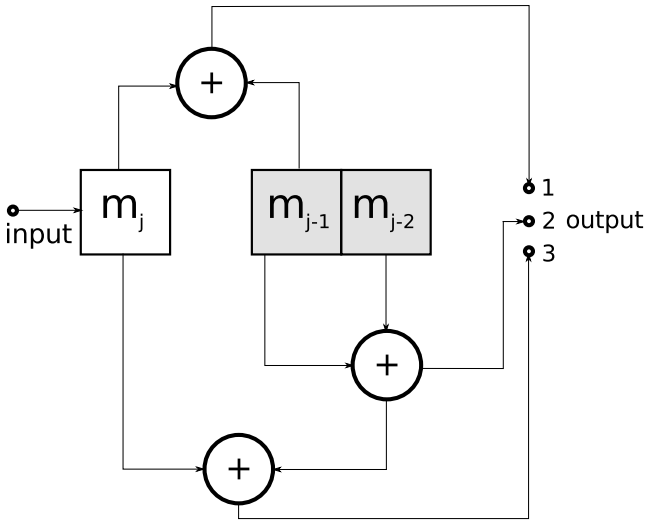
(5 marks) Task 4: CLI (Command Line Interfacing) - You must implement the following commands using the CLI FreeRTOS Library:

Command	Description
laser on/off	Turn the laser on or off
tail	Display the last 10 lines of a file on the uSD card.
ls	List the directory contents of the uSD card
task_usage	List the current number of tasks running and the respective task state and stack high water-mark usage.[1]

[1]**HINT**: The FreeRTOS API for task management can be found at www.freeRTOS.org

(2 marks) Task 5: Positional Replay – Task 2 requires you to store pan and tilt angles in a log file, together with a timestamp. This task requires you to read the data within these files and 'play it back' recreating the pan and tilt movements. You must replay these movements as per the timestamp embedded in your log file.

(2 marks) Task 6: Convolutional Encoding – Take a byte (8 bits) worth of data, and use the convolutional encoding described below to output 3 bytes (24 bits) worth of output. The encoder is a 1/3 code rate convolutional encoder, which means for every bit input (m_j), there are 3 bits output. Also, this encoder has two bits of state information (m_{j-1} and m_{j-2}). The initial state information of a convolutional encoder is assumed to be 0, 0. To ensure reusability of this function, put it into its own function.



An example of output for the convolutional code is shown below. The input sequence of bits is 1, 0, 0. Assuming least significant bit first, this would be for the first three bits of input 0x1

B0	B1	B2	B3	B4	B5	B6	B7
1	0	0	0	0	0	0	0

Here is the sequence of states and outputs for this bit sequence.

Input	(m_{j-1})	(m_{j-2})	Out[0]	Out[1]	Out[2]	Out	Comment
X	0	0	X	X	X	X	Initial State
1	0	0	1	0	1	0x5	
0	1	0	1	1	1	0x7	
0	0	1	0	1	1	0x6	

These bits are combined such that Out[n] is the nth output bit, and they can be assembled into a large word (long) as follows

B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11
1	0	1	1	1	1	0	1	1	.	.	.

As hex: 0x1BD

The command to be **implemented in the CLI** needs to take data in either ASCII or hex form. Hex data will be prefixed with '0x'. You need to only assume a single byte of ASCII.
Hence, the following two commands should give equivalent output:
 # encode b
 # encode 0x62

2a8de8 (the encoded output is 24 bits)

Note: for this task, for each byte sent, the convolutional encoder is reset back to the initial state.

Task 6 Milestone Questions:

Q1: Using the input and the 2 state information bits, what are the calculations required to get the three outputs?

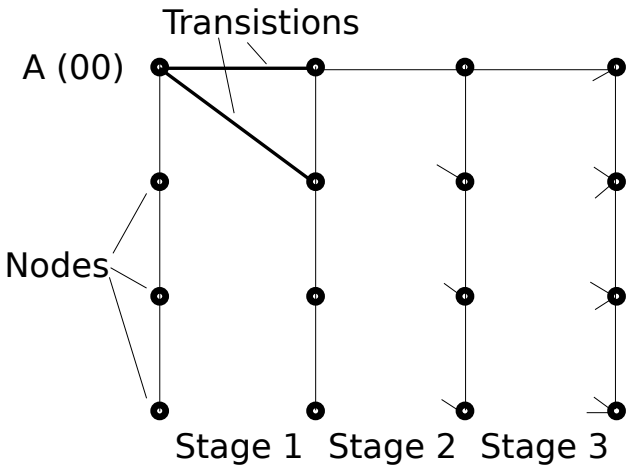
Q2: How many internal states and how many types of transitions/outputs are possible in this encoder? Draw a state machine for all of the states.

(3 marks) Task 7: Convolutional Decoding – Decoding the convolutional code can be done in many ways, however in this task you are required to use the Viterbi algorithm. It produces the best possible BER for a given input sequence.

The command to be **implemented in the CLI** needs to take hex encoded data, ie 0x2a8de8. It is highly unlikely that ASCII data would be a valid codeword.
eg **decode 0x2a8de8**
0x62 'b'

When each group of bits (in this case 3), is received, we call it one stage of decoding. In each stage the hamming distance between the received bits and the possible bits for each transition is calculated. The hamming distance is the number of bits which would need to be flipped to make one code word another, and is shown in the table below.

Codeword 1	Codeword 2	Hamming Distance
111	000	3
101	110	2
110	111	1



In Stage1, only two possible transitions are possible, as the original stage is known to be ‘0 0’. Therefore the hamming distance between these two transitions and the actual received bits are stored. So there are two paths, with two different path lengths (where the path length is the hamming distance for each of them).

In Stage 2, four possible transitions are possible. The length of the four paths can be calculated by taking the sum of the hamming distance of each of the four transitions plus the previous path length from the node that they came from.

In Stage 3 and future stages there are more transitions than nodes. This is where some tree pruning can be done, for each node we only keep the transition and its associated path with the lowest path length. This is a key part of the Viterbi algorithm. For example, if the inputs to “node A’ have path lengths 5 and 4, then the path corresponding to the path length of 4 is stored.

It should be noted that although a path may have optimum length at stage n, depending on future bits, it may not be the final selected path.

At a given stage if two paths have equal length, then the “upper” path is kept.

HINT: when coding this, the paths can be implemented as arrays. The number of these arrays is just 2 times the number of nodes.

Task 7 Milestone Questions:

Q1: Draw the Viterbi decoding tree for the input data 0x1BD. (Same as the encoder example)

Q2: Introduce a single error in the input sequence, and see what the most likely output is by drawing the Viterbi decoding tree for this new example.

(3 mark) Code Commenting and Style – Your code should conform to the following style and to the **FreeRTOS formatting style** as presented in Lecture 6:

(1 mark) Comments: Comments should be placed at the **front of each function** and **while/for loop**. Key sections of code should be commented.

(1 mark) Variable and Function Names: single letter names (ie. i, j, k) should only be used for counters. Each variable should have a name relevant to its use. **You must use the same variable name and function name format as specified by the FreeRTOS syntax presented in Lecture 6,**
i.e. usValue (unsigned short usValue)
i.e. vMyFunction (void vMyFunction())

(1 mark) Spacing: your code should follow the correct 4-space tab indentation for functions and loops.

(2 marks) Workbook– Your work will be marked according to the format outlined:
(0.5 mark) Objectives of Project
(0.5 mark) Hardware Implementation
(0.5 mark) Software Implementation
(0.5 mark) Testing Procedure

Total: 25 marks (15% of course marks)

Challenge Tasks:

NOTE: You will ONLY be assessed on the challenges, if you passed all tasks with marks 2 (or 6). With other words: it is pointless to spend time on challenges before you got all the tasks working very well.

NOTE: these challenges are made to ***challenge*** you and they will NOT be marked easily. If you are not 100% on the rest of your project, it is strongly suggested you work on those first as challenge marks are comparatively much harder to obtain. **You only need to do two challenges: Challenge 1 and either Challenges 2A or 2B. You have a choice of doing either Challenge 2A or 2B.**

(4 Marks) Challenge 1: Remote logging – Send data to your PC for logging. The logging software on the PC is available inside the binaries folder. It listens on port 2000, and expects input that is encoded in the convolutional code introduced in this project. Please note that in this task, you should not reset the convolutional encoder state to the initial state, until the end of each line sent. You can add a new CLI command that enables and disables logging, as well as lets you configure the remote logging host and port. When logging is enabled, all CLI commands and their responses will be send to the remote logging system. By sending a byte full of zeros, you can clear the state of the server binary.

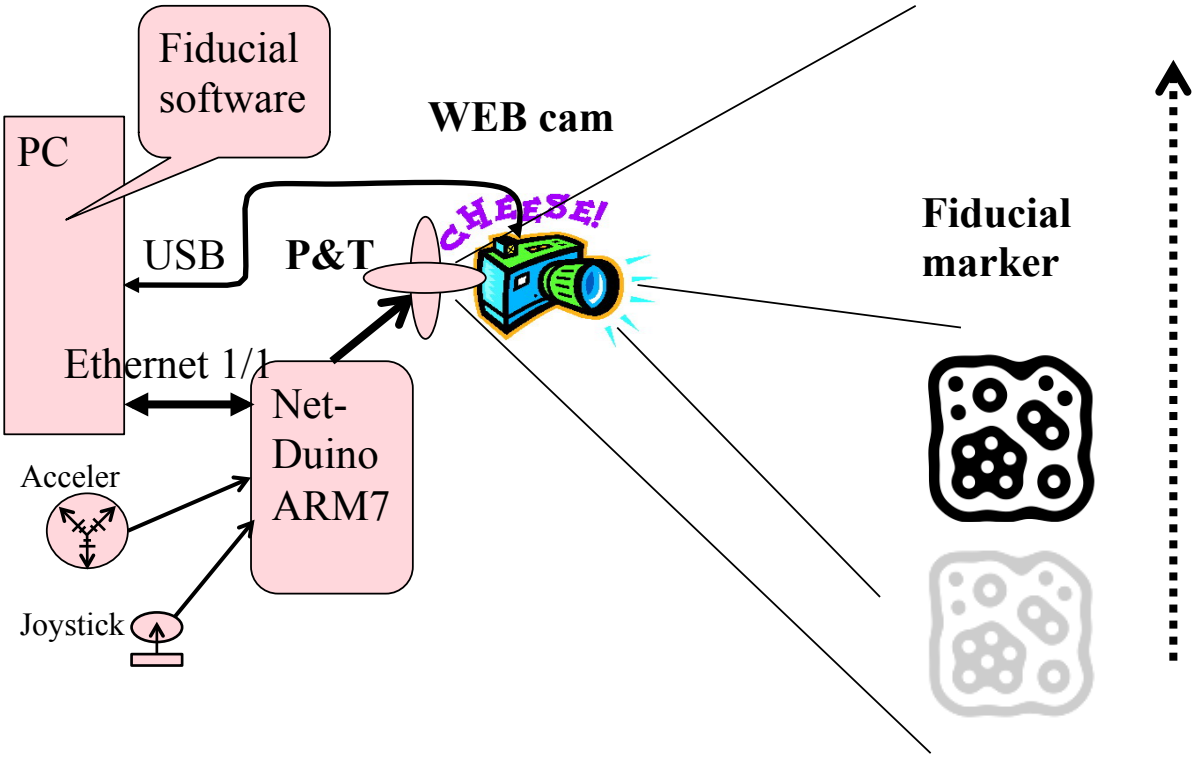
(4 marks) Challenge 2A: Web Interface – Create a webpage hosted by the Netduino Plus that can be accessed over the Ethernet connection by your computer. It is to simply list the files on your SD card. To receive full marks for this functionality, you must be able to **control** your pan and tilt servo, through the hosted webpage. You are allowed to implement a CLI command that can connect and disconnect the connection to the Fiducial Marker tracking program or enable and disable the webserver.

(4 marks) Challenge 2B: Send files via radio link / laser – You will need a **partner** for this challenge. Send a file stored on your SD card to your partners system. You may use ***either*** the radio or laser (USART) to send files; however, the files must match exactly when received by your partners system. To achieve full marks for this challenge, you must be able to list files from the SD card and select the file for transfer using either the CLI or regular USB interface.

Total: 8 marks (5% of course marks)

Mark Deductions
-1 Mark: You should not need to reprogram your board to show additional functionality. If you need to reprogram your board whilst being assessed, you will lose 1 mark.
-1 Mark: If your servo pan and tilt motion is not smooth, then you will lose 1 mark.
-1 Mark: If you cannot show SD card functionality using the ls and tail CLI commands.
-1 Mark: If you did not implement the backspace or delete key for the CLI.

Make sure you keep updated on **the newsgroup** for any of hints or answers to questions that many students are having issues with. Also be sure to update **SVN** regularly.



Project 2 Overview