Assignment 1
Real-Time Software Design: Military Tank Control

Date Due : 6 September 2013
Marks: 200 (20% of total assessment marks)
Penalty for Late Submission: 10% per day

Outline of task

You are required to design, implement and document a program for a Military Tank Control System to execute on a Windows based personal computer. The assignment does not involve any hardware connections to the PC, as the operation of the tank is simulated by a program provided for this purpose.

To complete this assessment you must submit a report documenting your system design, and submit the source files and executables of the completed (or partially completed) program by the due date. The report must include the following concise information:

• Program Listing	- Commented source file listing
• System Model	- A diagram representing the program and a brief explanation of its operation (maximum 4 pages)

System specification

Design

The application to be designed must correctly control the steering and gun targeting of the tank given user input from the keyboard. The tank configuration is detailed below. You are required to write a **state driven implementation** for a Real-Time control system to meet the following requirements:

1. Control the tank based on key strokes – **F** for forward, **B** for backward, **R** to turn right, **L** to turn left, **S** for stop. You are to assume that the tank *can only turn on the spot while stopped*, so your control is to ignore any command to turn while moving. The Tank uses a typical 2 motor control, driving a track on each side. Assume the tank has only one forward speed, one reverse speed and stop. Speed will be 1 unit/second and the turning rate is +/-15 degrees/second

2. Track the location of the tank (X,Y co-ordinates) and the heading (H). A flat working surface is to be assumed with an area of +/-20 units in each direction. Initial position will be (0,0) with a heading of 0 degrees. (use Cartesian co-ordinate system) The position and heading will also be monitored by the ‘tank simulator’ program, run in conjunction with your program to monitor the movement of the tank as commanded.

3. Stop the tank and issue a warning when it reaches the limits of work area (+/-20, X or Y).

4. On receiving the **A** keystroke, stop the tank and aim the turret at the fixed target position (5,5). This may involve turning the tank as well, see 5 below. The calculation for aiming the turret (and tank) should be based on trigonometry between the tank position and the fixed target location.

5. Control the gun turret (with respect to the forward facing direction of the tank) such that the angle never exceeds +/- 135 degrees. Issue a warning when it reaches either of these limits. Assume turret turning speed is +/-45degrees per second.

6 On receiving the **X** keystroke, fire the cannon, but *only if the tank and turret are stopped and the turret has been aimed at the target*. The 'tank simulator' program will determine how close you came to hitting the target, which is assumed to be 0.1 units wide.

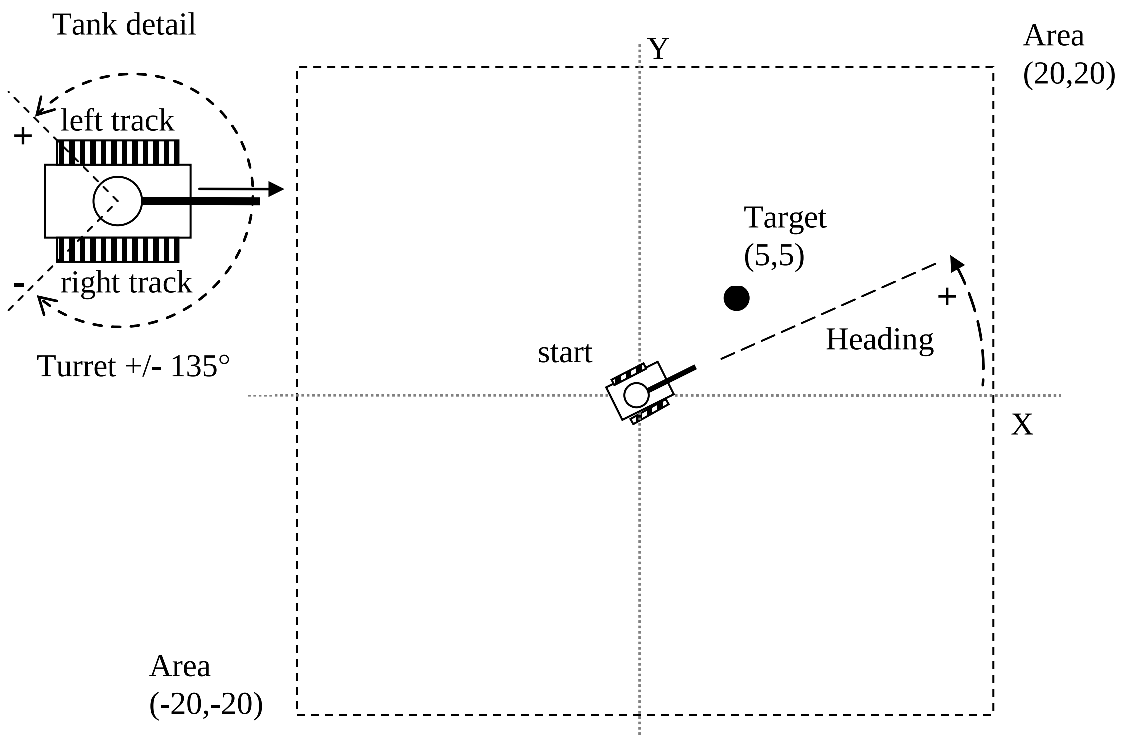

Figure 1: Tank work area

Note the size of the tank is not significant, the figure above is not necessarily to scale.

Implementation

You are required to implement a **state-based** real time control program in C to execute on a Windows compatible PC, using the cygwin or windows programming environment. Stated-based operation is required, NOT input based.

You will be provided with 'TankSim.exe', a separate program which you run at the same time as your control program. This program will simulate the tank's movement, track its position, heading and turret angle based on the commands you give and determine the accuracy of the cannon firing.

Note: tank simulator may or may not include a graphic of the tank movement and the area.

You will be provided with the header file ‘tank.h’ and the source file ‘tankInt.c’ which provides a set of subroutines that interface with the tank simulation program. Look for these files on the USQ Study Desk through the button of “Assessment”. A summary of the function of each subroutine is provided below. You should incorporate these files in your program compilation.

Functions of subroutines

The first two subroutines control the connection to the tank simulator program.

int tankopen(void);

Call this routine once at the top of your program to initialize the link to the tank simulator program and reset the tank position to its initial state: (0,0) heading 0. Returns 0 for success and -1 for failure to open link.

int tankclose(void);

Call this routine once at the end of your program to close the link to the tank simulator program. Returns 0 for success and -1 for failure to close link.

The next four subroutines control the tank track motors, the turret and the cannon.

void leftmotor(int onoff);

Call this routine with +1, 0 or -1 as a parameter to control the left track motor: Forward, Stop and Backward respectively.

void rightmotor(int onoff);

Call this routine with +1, 0 or -1 as a parameter to control the right track motor: Forward, Stop and Backward respectively.

int turret(int onoff);

Call this routine with +1, 0 or -1 as a parameter to control the turret motor:
Left (+ve Cartesian angle), Stop and Right (-ve Cartesian angle) respectively.
Returns 0 for success and -1 for if the turret is already at +/- 135 degrees..

double fire(void);

Call this routine to fire the canon. It returns a double precision float which gives the error in targeting in units, positive numbers are to the left of target (+ve cartesian angle). Value within +/- 0.05 indicates a hit.

double angle(void);

This routine returns the angle of the turret with respect to the tank. This angle is changed when the turret() routine is used to turn the turret.

Special Notes:

1. Cumulative errors will be expected in motion and angular calculations. Discrepancies will also occur between your position tracking and that of the tank simulation. These discrepancies can be minimized by using short loop intervals in your state machine. (=< 0.1 seconds)

2. Programs submitted must be the work of each individual student. Exchange of ideas on conceptual issues between students is expected, but students must ***write their own programs*** and not share code. Students are required to complete this assignment in the C language on a Windows compatible personal computer. Students are required to submit both source codes and executables on the StudyDesk (EASE), plus a design report.

ELE3307 Assignment 1 marking criteria

Each attribute below relates to a key element of the assignment and is marked on a scale of 25, 50, 75 or 100% for that attribute. Zero marks will be awarded for no attempt.

Attribute	Novice	Apprentice	Practitioner	Master
Source Files	Program compiles with many syntax errors and **cannot be tested**.	Program compiles with some syntax errors, but **with correction is testable**.	Program compiles without error, but testing reveals **some execution errors**.	Program compiles without error and testing reveals **no execution errors**.
Functional requirements 1, 2 and 3	The program **incorrectly uses sensor information or accumulated data**.	The program **makes limited use of sensor information or accumulated data**.	The program **adequately uses sensor information and accumulated data**.	The program **consistently uses sensor information and accumulated data.**
Functional requirements 4, 5 and 6	The program **incorrectly controls traffic flow or displays system conditions.**	The program **partially controls traffic flow or displays system conditions**.	The program **adequately controls traffic flow and displays system conditions**.	The program **reliably controls traffic flow and displays system conditions**.
Functional requirements (Timing)	The program **incorrectly uses timing elements**.	The program **makes limited use of timing elements**.	The program **adequately uses timing elements**.	The program **consistently uses timing elements**.
Modularity	The program makes **limited use of interface routines and modularity**.	The program makes **some use of interface routines and modularity**.	The program makes **adequate use of interface routines and modularity**.	The program makes **good use of interface routines and modularity**.
State Driven Approach	Program **is not state based and exhibits no uniformity** for conditional tests.	Program **is state based but exhibits limited uniformity** for conditional tests.	Program **is state based and exhibits some uniformity** for conditional tests.	Program **is state based and exhibits consistent uniformity** for conditional tests.
Program Implementation	The program is implemented with **limited use of accepted programming techniques**.	The program is implemented with **some use of accepted programming techniques**.	The program is implemented with **adequate use of accepted programming techniques**.	The program is implemented with **good use of accepted programming techniques**.
Documentation	Documentation is **poorly written** or does not explain program.	Documentation is **limited** or explanation of program is incomplete.	Documentation is **adequately written** and explains program operation.	Documentation is **well written** and explains program operation **clearly**.
System Model (diagram)	Diagram is **inappropriate for task** or does not represent program structure.	Diagram is **limited or incomplete** and does not represent program structure well.	Diagram is **adequately** drawn and represents program structure.	Diagram is **well drawn** and represents program structure **clearly**.

Commented program	The program has **limited** commenting and layout.	The program has **some** commenting and layout.	The program has **adequate** commenting and layout.	The program has **good** commenting and layout.

Comments

TOTAL MARK