EE 472 Lab 4

Winter 2015

Introducing a Real-Time Operating System

University of Washington - Department of Electrical Engineering

Mitch Ishihara, Joshua Fromm, Tien Lee

---

1 Introduction

This lab is the next phase in development of the RoboTank system project. In previous project phases we have built a foundation of knowledge including infrastructure tools we can learn from and use, such as peripheral support and exploring different design partitioning approaches. For peripheral support we've used: five buttons via GPIO, an external LCD interfaced over GPIO, LEDs via GPIO, analog sensor input and sampling via A/D conversion, timing using General Purpose Timers, and interrupt control. For design partitioning approaches, we first implemented the functionality in a main() while loop and learned the constraints such a design imposes on a system. Next we added interrupts and moved time critical functionality into interrupt service routines while keeping time consuming non-critical functionality in the main() while loop. We learned the increasingly complex constraints imposed on such a design, particularly around sharing resources with multiple threads of execution (main loop and interrupt service routines).

Now we will take the paradigm shift of using a preemptive priority scheduler based Real-Time Operating System, FreeRTOS, which includes fundamental APIs for proper use in such a system. Using the APIs involves an understanding of their use cases, their requirements, and the ability to make design decisions. The objectives to achieve include understanding and defining the system (not just one component): requirements specification (the what), design specification (the how), implementation, and test plan. Part of an engineer's role is to be able to trace back to requirements specifications used in the design, implementation, and test plan. We will use this project to continue working with the development life cycle of an embedded system. To that end, we will begin creating formal requirements, structured specifications, modular designs, implementations, and test plan that trace back to the requirements.

2 Objectives

- Introduce a preemptive priority scheduled RTOS (FreeRTOS) which may include a number of steps to complete first. (see sections 4.1 FreeRTOS and Appendix A: Introducing FreeRTOS)
- Amend requirements specifications to reflect your groups unique RoboTank requirements that were refined and differentiate your prototype RoboTank
- Amend design specifications including the functional and architectural design that describe and define your implementation of the RoboTank
- Utilize UML diagrams to model some of the dynamic aspects of the system.
- Implement task creation as well as resources sharing and inter-task communication protection requirements with the RTOS.
- Iterate previous requirements specifications, design, and implementations of previous support for the keypad, external GPIO interfaced LCD display, timers, ISRs, A/D sensors, and other infrastructure to meet the changes required for this lab.
- Add a system clock interrupt for use by the RTOS as the OS “heartbeat” tick time base.
- Add additional A/D conversion channels for a total of four A/D channels of sampling.
- Add support for two PWMs which will be used in the final lab to drive the RoboTank's two motors via an H-bridge hardware module.

Note: The actual hardware will not be provided until the final lab. Therefore you will have to formulate and prove your testing using other methods (e.g. oscilloscope, LEDs).

- Test the new features and capabilities of the system.

3 Prerequisites

C programming proficiency introduced using the Texas Instruments Stellaris EKI-LM3S8962 evaluation board and IAR Systems Embedded Workbench integrated C / Assembler development environment. Familiarity with the ARM Cortex-M3 based on the ARMv7-M Architecture which is the processor implemented in the LM3S8962. Patience and persistence to persevere.

4 Background Information

4.1 FreeRTOS

As we are now moving our project to an RTOS, a Real-Time Operating System called FreeRTOS, dedicate some time getting to know the FreeRTOS website and related documentation.

http://www.freertos.org/

Check out the getting started and advanced information at:

- The quick start video link is worth the 20 minutes it takes to watch for the newbie.
  http://www.freertos.org/FreeRTOS-quick-start-guide.html
- Information on Official FreeRTOS Ports
  http://www.freertos.org/RTOS_ports.html
  - Information on the Stellaris LM3S8962 Port
    http://www.freertos.org/a00090.html#TI

4.2 Datasheets and References

- Stellaris LM3S8962 Evaluation Board User's Manual
  Stellaris.pdf
- Texas Instruments Stellaris LM3S8962 Microcontroller Data Sheet (Manual)
  lm3s8962.pdf
- Texas Instruments ADC Oversampling Techniques for Stellaris Family Microcontrollers
  spma001a.pdf
- Sharp GP2Y0A21YK0F Distance Sensor
  1489_Sharp_GP2Y0A21YK0F.pdf
- Optrex Dot Matrix Character LCD Module User's Manual
  OptrexMan.pdf
- Toshiba TB6612FNG dual DC motor driver IC
  TB6612FNG.pdf

4.3 Course Text Reading Material

Relevant chapters from the course text:

- Chapter 5: An Introduction to Software Modeling

- Chapter 8: Safety, Reliability, and Robust Design
- Chapter 9: Embedded Systems Design and Development
- Chapter 11: Real-Time Kernels and Operating Systems
- Chapter 12: Tasks and Task Management
- Chapter 16: Working Outside the Processor I: Refining the Model of Interprocess Communication

4.4 Lab Document Organization

This lab document is organized to utilize the contemporary design tools covered in the course text listed in the previous section 4.3 Course Text Reading Material. The goal is to experience hands on use of some of the concepts in the tools. This lab is not an exhaustive use of the tools; rather an exercise in using (any) tools to formulate a methodology and understanding for your benefit as an engineer. An analogy to the real world to consider is that there are various tools one can use to complete various work. Sometimes one is required to follow a certain flow, for example to meet industry certification requirements or to provide a level of certainty, and sometimes one is given freedom to choose based on own personal preference. This lab leans more toward the former in order to reduce the number of variables in the equation which becomes more critical as a project becomes larger and financial revenue is dependent on its delivery to the customers.

To that end, the sections of lab document can be described as follows:

Section 5 System Requirements Specification

This is the specification that describes and defines the requirements, the what. Often it reflects the view of the engineering *process* back to the customer. In conceptual terms, if we remove the subsequent sections of the lab, we could implement a system using completely different design choices, yet still deliver a system that meets the requirements. This is the view from the outside of the system looking into an opaque box. In a way, this distinction enables subsequent design flexibility.

There are sections in the System Requirements Specification that require engineering (your group's) amendments and updating. For example, a basic User Interface was specified in Lab 2; however you were given an opportunity to wear a different hat and influence the UI specification as part of lab 2. Your UI specification shall be rolled back into this lab however it may need to be modified accordingly. Also the Use Cases are to be supplied by your group as an exercise.

✂----------

Section 6 System Design Specification

Now we cross the boundary from the requirements of the system and go inside the design. When inside the system we begin to see how the system is put together. The Design Specification should state *how* the design will meet the requirements. One of the design tools is the functional model of the design which is a subsection in this lab. Another design tool and subsection is the Architectural Design which describes how each module will contribute to meeting the requirements. It is with these design considerations that trade-offs between different design approaches can be made.

Section Error! Reference source not found. Error! Reference source not found.

A fair amount of work is still required to implement this lab and flesh out the details. At this point it is a prototype implementation, a proof of concept and tool for understanding and confirming the system design.

4.5 Requirements terminology used:

- Shall: a requirement that is mandatory
  "The system shall calculate the time of sunrise."

- Will: something that happens without the system doing anything
  "The sun will rise in the East."
- May: an acceptable (non-mandatory) requirement
  "The system may display the results in a 24 hour clock, e.g 18:54."

It is not uncommon for a specification to require a change as a result of detailed investigations. Where a requirement or design specification is provably infeasible in this lab, provide a formal request for change along with the proof the specification cannot be met. Requests for changing specifications will only be accepted in the Class Discussion Forum.

---

5 System Requirements Specification

This specification describes and defines the basic requirements for system, a prototype RoboTank.

5.1 System Requirements

The RoboTank system shall:

- be able to move straight in the forward direction, straight in the reverse direction, move forward while turning right or left, move reverse while turning right or left, rotate clockwise, and rotate counter-clockwise (see section 5.3.2.2 Motion Control)
- support four distance sensors to detect and avoid obstacles in four directions (see section 5.3.1.1 Distance Sensor)
- be manually operated via a menu driven user interface using five buttons: up/forward, down/reverse, left, right, and select (see section 5.3.1.2 Buttons and 5.3.3 User Interface)
- have a display to provide user interface information (see section 5.3.2.1 Display)
- incorporate the real-time operating system

The RoboTank system may:

- be able to move forward or reverse with a dynamically variable turning radius while in motion (see section 5.3.2.2 Motion Control)
- support remote operation in the future via a serial wireless link (see section 5.3.4 Remote User Interface)
- Incorporate an audible feature (see section 5.3.2.3 Audible Feature)
- *(to be updated as necessary for additional extra-credit requirements)*

5.2 Specification of External Environment

The RoboTank is to operate in an indoor environment with commercial grade temperature ranges. The prototype RoboTank shall support tethered power for the microcontroller and battery operation for the motors with the ability to support completely non-tethered option in the future. The surface on which the RoboTank operates shall be flat.

5.3 System Input and Output Specification

5.3.1 System Inputs

The system shall measure the following signals.

(to be updated as necessary)

5.3.1.1 Distance Sensor

A total of four Distance Measuring Sensor Units shall be used to detect the distance in four directions and avoid obstacles. The specific angles of the sensors relative to the RoboTank chassis may be optimized to improve forward direction obstacle avoidance over the reverse direction.

Distance: *(to be updated as necessary)* cm min / max

Distance detection latency: ≤ 10 ms or in time to prevent the RoboTank from hitting a stationary obstacle

5.3.1.2 Buttons

The system shall use 5 distinct buttons (up/forward, down/reverse, left, right, and select).

Button system latency response: ≤ 100 ms

Button inputs shall be digital data

Valid button selections (motion control):

- Up (forward)
- Down (reverse)
- Right (clockwise)
- Left (counter-clockwise)
- Forward+Right (forward motion while turning right)
- Forward+Left (forward motion while turning left)
- Reverse+Right (reverse motion while turning left)
- Reverse+Left (reverse motion while turning left)
- Select

5.3.2 System Outputs

(to be updated as necessary)

5.3.2.1 Display

A display shall be used to provide information to the user that is updated a minimum rate of 3 times a second. The update rate may be faster to improve display latency.

5.3.2.2 Motion Control

The system shall control motion of the RoboTank in the forward and reverse directions with configurable variable speed. The default speed may be configured at the maximum speed.

Directional rotation control of the RoboTank shall be clockwise for a right command and counter-clockwise for a left command.

Directional forward or reverse motion while turning right or left shall be at a configurable turning radius. The turning radius configurability may be 0 (same as Left or Right selection only) to max (same as Forward or Reverse selection only).

The turning radius may be dynamically changed by the system while in motion to avoid obstacles while maintaining smooth forward or reverse motion.

(to be updated as necessary)

5.3.2.3 Audible Feature

An audible feature may be used to indicate an impending collision, acknowledgement of user input, and/or error condition.

(to be updated as necessary)

5.3.3 User Interface

The user interface, UI, shall be capable of receiving user input through the built-in 5 button keypad. The system shall use that information to manipulate a menu on the Display.

The UI shall provide a menu item to enable the RoboTank directional movement input state (e.g. forward, reverse, left, right).

The UI shall provide a menu item with the current distances from the four distance sensors that is updated at a rate of at least 3 times a second. This menu item may be combined with the directional movement input state. For example, when the forward button is pressed, the RoboTank moves forward and displays the distances while moving.

(to be updated as necessary – insert and amend your Lab 2 user interface requirements specification as required)

5.3.4 Remote User Interface

A remote user interface via a wireless link is going through requirements definition. No additional requirements are available at this time.

(to be updated as necessary – in Lab 5)

5.4 Use Cases

The following use cases express the external view of the system.

(to be added by engineering… this would be you - see Chapter 5 in the text)

5.5 Operating Specifications

The system shall operate in a standard commercial environment.

(to be updated as necessary)

5.6 Reliability and Safety Specification

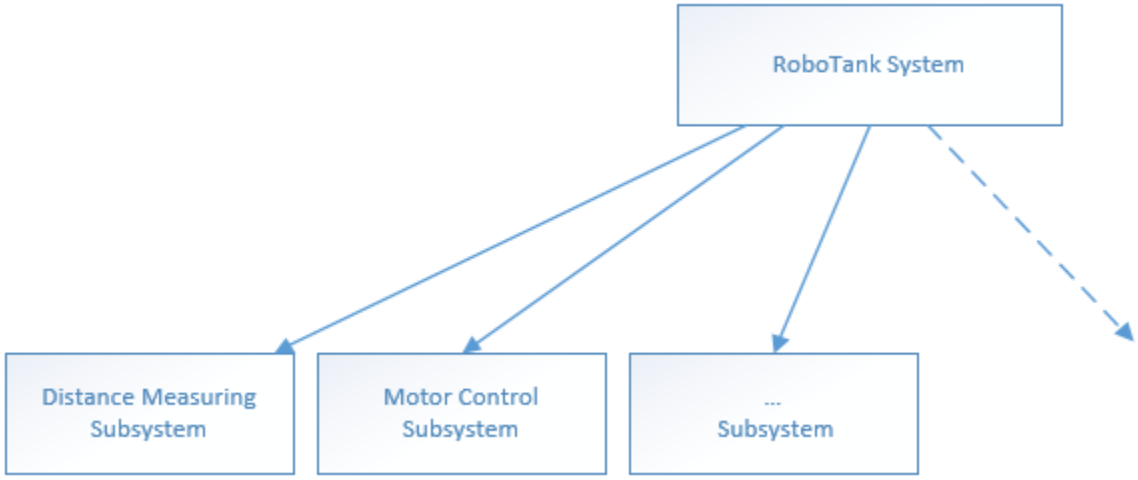
(to be updated as necessary)

6 System Design Specification

This specification describes and defines how the system is designed to implement the requirements specification of the prototype RoboTank.

6.1 Functional Design

The following diagram shows a functional model of the subsystems in the RoboTank system.



(To be updated – by engineering)

The following are the descriptions of the subsystems.

Motor Subsystem

- Responsible for RoboTank movement straight in the forward direction, straight in the reverse direction, move forward while turning right or left, move reverse while turning right or left, rotate clockwise, and rotate counter-clockwise (see section)
- may be able to move forward or reverse with a dynamically variable turning radius while in motion (see section)

Distance Measurement Subsystem

- Distance support four distance sensors to detect and avoid obstacles in four directions (see section)

Keypad Subsytem

- be manually operated via a menu driven user interface using five buttons: up/forward, down/reverse, left, right, and select (see section)

Display Subsystem

- have a display to provide user interface information (see section)

Audible Indicator Subsystem

- Incorporate an audible feature (see section 5.3.2.3 Audible Feature)
- *(to be updated as necessary – for extra credit)*

6.2 Architectural Design

6.2.1 Constraints

To reduce the scope of design choices in the prototype RoboTank, specific hardware and software will be chosen and included in the design.

The prototype RoboTank hardware will be comprised of the following:

a. four Sharp GP2Y0A21YK0F distance sensors
   Outputs analog data, voltage range 0.0 to 3.3 VDC
b. a Stellaris LM3S8962 Evaluation Board based on the ARM Cortex-M3 as the primary microcontroller
c. an Optrex LCD or alternatively the Stellaris OLED may be used
   (To be determined – by engineering)
d. a 5 VDC power shall be provided to the Stellaris LM3S8962 Evaluation Board via the USB mini connector

The prototype RoboTank software will be comprised of the following:

a. A Real-Time Operating System – Implement using FreeRTOS (see Appendix A)

6.2.1.1 RoboTank Chassis

The actual RoboTank hardware (chassis, motor driver) will not be available until the last lab, Lab 5. Functional testing that the system meets the requirements shall still be required. As such, your test plan shall consider this constraint in Lab 4.

6.2.1.2 Real-Time Operating System

FreeRTOS will be used for a pre-emptive multi-tasking design with all Inter-task Communication using FreeRTOS APIs.

1. The basis for the project may be the FreeRTOS demo directly downloadable from the freertos.org website; however, demo example software shall be cleaned to the only include what is used for this project.

Note: When a context switch is made because a running task blocks or terminates or on return from interrupt, the highest priority task that is ready to run will be selected from the list, started or resumed, and moved to the run state. For the case of an interrupt, the resumed task may or may not be the interrupted task.

6.2.1.3 Task Control Blocks

The design and implementation of the system will comprise a number of tasks. TCB creation will be done under the FreeRTOS OS utilizing xTaskCreate().

6.2.1.4 Data Types

Boolean

Although an explicit Boolean type was added to the ANSI standard in March 2000, the compiler we're using doesn't recognize it as an intrinsic or native type. (See http://en.wikipedia.org/wiki/C_programming_language#C99 if interested in more detail)

We can emulate the Boolean type as follows:

```
enum myBool { FALSE = 0, TRUE = 1 };
typedef enum myBool Bool;
```

Put the code snippet in an include file and include it as necessary.

Integer

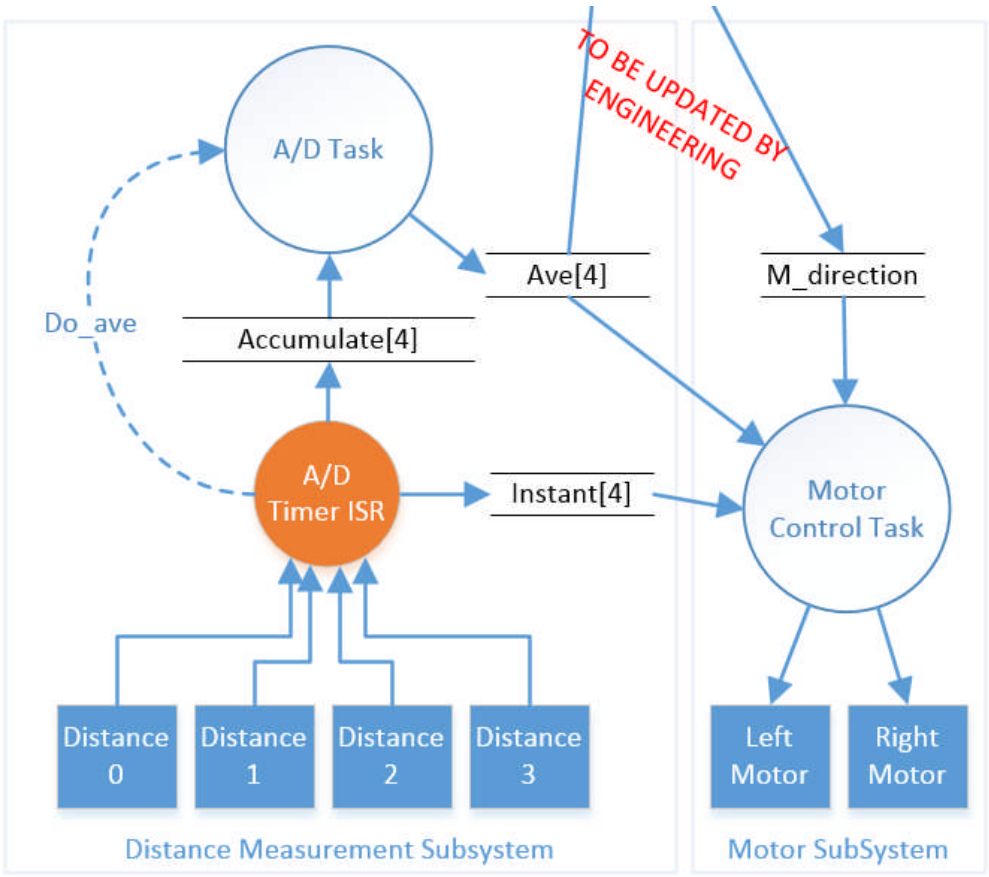
In addition, C99 standard integer types are used by FreeRTOS (at least as of 8.2.0). See the RTOSDemo, stdint.h. C99 integer types shall be used where the specific with of a data type is known and fixed.

6.2.2 System Data and Control Flow Diagram

The RoboTank system's high-level system software architecture shall support multitasking operations and real-time behavior. Information within the system shall be exchanged utilizing FreeRTOS Inter-Task Communication methods to protect shared resources. The design shall be comprised of modularized subsystems clearly defining ownership of resources, ISRs, and tasks. Following power on, tasks are executed utilizing a pre-emptive, priority based scheduling algorithm. Tasks may be scheduled and pending based on priority within the system and availability of the work to do. In addition, event driven design and interrupt service routines shall be used instead of software loops and software delay loops.

(To be updated as necessary – by engineering)

The following System Data and Control Flow Diagram includes the tasks, ISRs, inter-task communication, and hardware interfaces.



(To be updated – by engineering with specific inter-task communication methods, control flow (if control not facilitated by FreeRTOS inter-task communication methods), and other subystems of the RotoTank system)

The following are descriptions of the subsystem modules in the diagram.
(To be updated with additional subsystems – by engineering)

6.2.2.1 Distance Measurement Subsystem

Sensor Outputs

Four Sharp GP2Y0A21YK0F distance sensors will be used to measure distance in four directions.

The distance sensors produce:

- analog data
- voltage range 0.0 to 3.3 VDC

This is within range of the specification for the LM3S8962 A/D inputs.

Input A/D Sample Conversion ISR

The analog voltage from the distance sensors shall be sampled using the four LM3S8962 on SoC A/D converters, one channel per sensor. The conversion for each analog channel shall be sampled at a steady periodic rate of 4096 samples per second by the A/D converter driven by a timer ISR.

The LM3S8962 A/D converter reference is 3.0 V which sets the 10-bit full scale value at 0x3FF (0b0011_1111_1111). The A/D Timer ISR provides the instantaneous values to the subsystem left shifted 6 bits (0b1111_1111_1100_0000) for a full-scale 16-bit value of 0xFFC0. Thus, 0xFFC0 represents 3.0 VDC. The instantaneous samples (the latest samples) for each channel shall be safely shared for use by the tasks in the system that require the instantaneous values.

For each channel, 4096 instantaneous samples shall be accumulated (summed) and then a copy of the accumulated value safely shared for use by the A/D Task in the system. Once 4096 samples have accumulated, the accumulated value shall be reset to zero and the ISR shall set a flag for the A/D Task to call an average function.

A/D Task

The average function shall average the accumulated samples. The averaged sample shall retain the residual lower 6 bits of resolution for a pseudo extra 6 bits of resolution.

Once the averages have been produced for each channel, a transform lookup table function shall be called by the A/D Task and convert the averages to a distance values in millimeters. The distances shall be safely shared for use by the other subsystems that require the oversampled and averaged distances.

Example of 4096 full scale samples (original 10-bit samples unshifted):
0x3FF x 4096d = 0x003F_F000 (This is within the range of a uint32_t)
0x003F_F000 >> 6 = 0xFFC0

0xFFC0 represents 3.0 VDC

0x3FF << 6 = 0xFFC0
0xFFC0 x 4096d = 0x0FFC_0000 (This is within the range of a uint32_t)
0x0FFC_0000 >> 12 = 0xFFC0
0xFFC0 represents 3.0 VDC

uint16_t LookupDistanceTable(uint16_t distance_sensor_v)

The purpose of this function is to efficiently translate the input voltage parameter distance_sensor_v into a return value distance in millimeters. This translation is done via a lookup table and linearly interpolated values.

To correlate the analog voltage to the 16-bit digital value, 65535/3.0 VDC = 21845/VDC

The valid input voltage values and their corresponding distances are:

- Minimum input voltage:
  0x0000 = 0.0 VDC = 800 millimeters (capped at max. distance)
- 0x2222 = 0.4 VDC = 800 millimeters (maximum measurable distance)
- Lookup table with *(engineering defined)* number of data points and interpolated values
- 0xFFFF = 3.0 VDC = 70 millimeters (minimum measurable distance)
  Maximum input voltage:
  0xFFFF = 3.3 VDC = 70 millimeters (capped at min. distance)

Calibration is required to set the minimum and maximum distances that can be measured by the sensor. *(To be updated as necessary – by engineering)*

6.2.2.2 Motor Control Subsystem

Motor Control Task

The Motor Control Task takes three inputs:

1. A direction command
2. The instantaneous distance[4], one for each distance sensor
3. The oversampled and averaged distance[4], one for each distance sensor

The Motor Control Task controls two motors on the RoboTank, one for each track:

1. the left motor
2. the right motor

A Toshiba TB6612FNG dual motor H-bridge driver IC is connected to the RoboTank's motors. For each motor, the task shall produce variable speed via two independent PWM outputs which are connected to the motor driver IC. The system shall use five motor control GPIO outputs: Standby, AIN1, AIN2, , BIN1, BIN2. AIN1 and AIN2 control the direction of motor A which controls the left drive motor on the tank. BIN1 and BIN2 control the direction of motor B which controls the right drive motor on the tank. PWMA and PWMB control the speed of motor A and motor B respectively. Standby puts the motor controller in standby or in a state ready to respond to the direction and speed inputs.

The GPIO outputs will be:

- Digital data
- Voltage range 0.0 to 3.3 VDC

The PWM outputs will be:

- Digital data
- Voltage range 0.0. to 3.3 VDC
- Duty cycle: 0 to 100%

Two independent PWMs shall be implemented to control the speed of the motors.

1. Port F0/PWM0 Port G1/PWM1, Port B0/PWM2, Port B1/PWM3, Port E0, PWM4, Port E1/PWM5 may be used for the PWM output pins.
   (To be updated – by engineering design choice)
2. To test the PWM, the EE 472 dev platform's motor and/or an oscilloscope may be used.

Five Motor Control GPIO outputs shall be used.

1. One GPIO output for Standby shall be implemented. A logic low output will indicate standby mode to the motor driver. A logic high output will indicate to the motor controller standby mode is disabled and the motor driveris ready to respond to the direction and speed inputs.
2. GPIO outputs for AIN1, AIN2, BIN1, BIN2 shall be implemented. When IN1 is low and IN2 is high, the motor is driven counter-clockwise. When IN1 is high and IN2 is low, the motor is driven clockwise. When both IN1 and IN2 are low, the motor driver is in a high impedance state. When both IN1 and IN2 are high, the motor driver is in a short-circuit brake mode.

See the Toshiba TB6612FNG datasheet for the full truth table.
(To be updated as necessary – by engineering)

6.2.2.3 Keypad Subsystem

(To be updated – by engineering)

Button inputs will be:

- Digital data
- Voltage range 0.0 to 3.3 VDC

1. Five buttons shall be implemented using the onboard Stellaris buttons.
   a. The buttons shall be debounced in software.
   b. The buttons shall be active low.
   c. The buttons shall be sampled at a 10 Hz rate.

6.2.2.4 Display Subsystem

The LCD user interface may be either the external Optrex LCD or the onboard OLED. If the external Optrex LCD is used, the 4-bit interface may be used to free up port pins for other use.
(To be updated as necessary – by engineering)

6.2.3 Shared System Data Structures

(To be supplied – by engineering)

6.2.4 StartupTask

(To be updated as necessary – by engineering)

Startup Task – The task creation and initialization process when the system starts as required by the operating system.

The startup task shall run one time at startup and is to be the first task to run. It shall not be part of the task queue. The task shall,

- Configure and activate the system time base that is to be the reference for timing all warning and alarm durations.
- Configure and initialize all hardware subsystems.
- Create and initialize all statically scheduled tasks.
- Enable all necessary interrupts.
- Perform any other startup operations.
- Start the system.
- Exit.

The static tasks are to be assigned the following priorities:

(To be supplied by engineering)

6.2.5 Vector Table

<specification of the vector table>

(To be supplied – by engineering)

6.2.6 InitializeHardware()

The InitializeHardware() function shall be called as the first function in main() to initialize all the software visible registers required by the system's static functionality. Dynamic initialization may be done elsewhere, where appropriate.

- The Cortex-M3 System Clock shall be configured for 50MHz. This requires enabling the PLL.

(To be updated as necessary – by engineering)

6.2.7 InitializeSysTick()

The Cortex-M3 system timer tick shall be used to drive the FreeRTOS system timer interrupt. The OS tick shall be a periodic rate of 100 Hz (10 ms).

(To be updated as necessary – by engineering)

6.2.8 SysTickIsr()

<specification of the ISR>

(To be supplied – by engineering)

6.2.9 <insert name> ISR

<specification of the ISR>

(To be supplied – by engineering)

6.2.10 <insert name of function>

<specification of major function>

(To be supplied – by engineering)

6.2.11 <insert name of data structure>

<specification of major data structure>

(To be supplied – by engineering)

7 Recommended Approach

This project involves specifying, designing, developing, and integrating a number of software components. On any such project, the approach one takes can greatly affect the ease at which the project comes together and the quality of the final product. To this end, we strongly encourage you to follow these guidelines:

1. Develop all of your UML diagrams first. This will give you both the static and dynamic structure of the system.
2. Block out the functionality of each module. This analysis should be based upon your use cases. This will give you a chance think through how you want each module to work and what you want it to do.
3. Do a preliminary design of the tasks and associated data structures. This will give you a chance to look at the big picture and to think about how you want your design to work before writing any code. Where a FreeRTOS API is used, clearly indicate the API.
   This analysis should be based upon your UML class/task diagrams.
4. Write the pseudo code for the system and for each of the constituent modules.
5. Develop the high-level flow of control in your system. This analysis should be based upon your activity and sequence diagrams. Then code the top-level structure of your system with the bodies of each module stubbed out.
   This will enable you to verify the flow of control within your system works and that you are able to invoke each of your procedures and have them return the expected results in the expected place.
6. When you are ready to create the project in the IAR IDE. It is strongly recommended that you follow these steps:
   a. Build your project.
   b. Understand, and correct if necessary, any compiler warnings.
   c. Correct any compile errors and warnings.
   d. Test your code.
   e. Repeat steps a-d as necessary.
   f. Write your report
   g. Demo your project.
   h. Go have a beer.

Caution: Interchanging step h with any other step can significantly affect the successful completion of your design / project.

This project, project report, and program are to be done as a team. The work may be divided among team members to balance the workload; however all members are expected to be proficient in each other members' work and collaborate where needed.

8 Project Report

Write up your project report following the guideline on the EE 472 web page.

You are welcomed and encouraged to use any of the example code on the system either directly or as a guide. For any such code you use, you must cite the source.

Warning: You will be given a failing mark on the project if you do not cite your sources in your listing - this is not something to be hand written in after the fact, it must be included in your source code. This is an easy step that you should get in the habit of doing.

Do not forget to use proper coding style; including proper comments. Please see the coding standard on the class web page under documentation.

Please include in your project report an estimate of the number of hours you spent working on each of the following:

- Design
- Coding
- Test / Debug
- Documentation

Please include the items listed below in your project report:

1. Empirically measured individual task execution time.
2. Design document including UML diagrams
3. If you were not able to get your design to work, in addition to the design document, include a contingency section describing the problem you are having, an explanation of possible causes, a discussion of what you did to try to solve the problem, and why such attempts failed. If your design has major design flaws it is difficult to evaluate where the flaw was missed to award partial design criteria credit.
4. The final report must be signed by team members attesting to the fact that the work contained therein is their own and each must identify which portion(s) of the project she or he worked on.

NOTE: If any of the above requirements is not clear, or you have any concerns or questions about you're required to do, please do not hesitate to ask us.

9 Grading Guidelines

You are expected to demo your work to a TA, showing that it does all the required features along with anything extra you coded. Be prepared to explain how the software or hardware components used in the lab works as well as any issues encountered and solutions to the issues you encountered.

You will need to turn in a report comprised of the content described in this lab. Clearly indicate the contributions of each team member in the report. As always, you should also turn in your **thoroughly** documented source code.

Appendix A: Introducing FreeRTOS

This appendix covers the steps used to create a clean FreeRTOS project. If you haven't done so already, skim through the FreeRTOS website and related documentation at http://www.freertos.org to familiarize yourself with what is available. Also make sure to go through the FreeRTOS Quick Start Guide and videos at:

- The quick start video link is worth the 20 minutes it takes to watch for the newbie. http://www.freertos.org/FreeRTOS-quick-start-guide.html

The FreeRTOS port was downloaded from the FreeRTOS website.

- Information on Official FreeRTOS Ports http://www.freertos.org/RTOS_ports.html
    - Information on the Stellaris LM3S8962 Port http://www.freertos.org/a00090.html#TI

It was then stripped to the minimal platform support for the LM3S8962 as suggested in the quick start videos. Also the web server was removed that we cannot really use due to potential code size limitations on the evaluation version of IAR.

The following steps are suggested to run the demo:

1. Download the stripped minimal demo from the EE472 web page.
2. Build the demo (should be zero errors)
3. Run the demo in the debugger
4. Study how the system is initialized, tasks are created, the scheduler is started. You will have to replicate this functionality in your own RTOS project.

At this point, you will want to create your own IAR project with FreeRTOS.