University of Washington
Department of Electrical Engineering
EE 472, Spring 2015

Report for Lab 4: Introducing a Real-Time Operating System

Report by: Sorav Sharma, Anthony Grigore, Rigoberto Orozco Diaz

EXECUTIVE SUMMARY

In this lab, we used a priority scheduler based Real-Time Operating System, FreeRTOS. The objective was to understand and define the system, including: requirements specification, design specification, implementation, and test plan. We were able to achieve this by using a handful of datasheets and references to our Stellaris board. We used this debugging process to continue building on the life cycle of our RoboTank.

1 LEDs

Using the datasheets provided, we were able to make an LED blink. The Stellaris.pdf and the LM3S8962.pdf were the two datasheets that were used for this part of the lab. We focused mainly on the system control and GPIO sections of the LM3S8962.pdf datasheet.

In order to set up our project, we first made a main loop which set the clock speed to run from the on board crystal by using the SysCtlClockSet function from Lab 1. Once we had a set clock, we wrote a function that made onboard status LED (LED1) blink. In order to do so, a LED_init method was written. In this method, we first enabled the GPIO port, port F, for LED1. Next, we set the direction of the pin controlling LED1 to be output. After that, we disabled alternate functionality of the pin using the GPIO. Lastly, the pin in the GPIO Digital Enable Register was enabled and the output of the pin was set to 1. This allowed LED1 to turn on.

With an LED always blinking, the next task at hand was to make it blink. In order to make the LED blink, we made a function called LED_toggle. This function virtually takes the state of the LED and inverts its state. Once the two functions were created, they were included into a loop in the main file of the project.

2 KEYPAD INPUT

We initialized the keypad which allowed us to use the arrows on the Stellaris board. In initializing the keypad, we wrote a key_init function that initialized the keypad. This function was very similar to the one in part one as we had to enable ports E, for the direction keys, and F, for the select key.

We went on to write a function named is_a_key that returned a boolean value indicating whether a key was pressed or not. We went on to write a third function called getKey that returned the value of the button or combination of buttons being pressed. With the functions is_a_key and getKey, we were able to continue on to write the fourth function called keyMaster. This function returned the value of a key if one was pressed and 0 if no key was pressed. We were able to test the keypad's behavior by including a conditional statement in our main loop. The conditional statement turned on the LED if the select key was pressed. By allowing the select key to control the LED, we were easily able to troubleshoot the keypad by observing the LED.

After initializing, we came to a conclusion that the LEDs did not toggle some of the times we pressed select. Also holding down a key caused the LED to be less bright. In order to fix the glitch of the key, we added a function called debounce that only returned true if the same key value was read continuously for 100 milliseconds. Since our code did not have a way of telling if a key was pressed is new or not, we had a make a function that did so. We went on to write a function called fresh_key that returned true if a key press was not a repeat from a previous press. This function completed our keyMaster function and completed our keypad initialization.

3 DISTANCE SENSORS

We went on to periodically sample the output of four distance sensors. We started by wiring up a distance sensor to the Stellaris board. The red and black wires were hooked up to power (5 V) and ground respectively and the white wire was plugged into their respective ports on the Stellaris board. The voltage on the white wire of each sensor started at 0 when no object was in front of it and increased as an object approached it.

Rather than having ADC interrupts, we requested a sample and then read the value after a delay. We wrote a function called ADC_init for each port that we had to initialize. Then we made a task so that for each channel, 4096 instantaneous samples were accumulated and then a copy of the accumulated value safely shared for use by the task in the system. Once 4096 samples were accumulated, the accumulated value was reset to zero and the ISR set a flag for the task to call an average function. Each time, the task alternated between requesting ADC data and reading that data. Taking in consideration that the value read from the sensors would range between 0 and 1024, we modified our task accordingly.

4 PULSE WIDTH MODULATORS (PWMS)

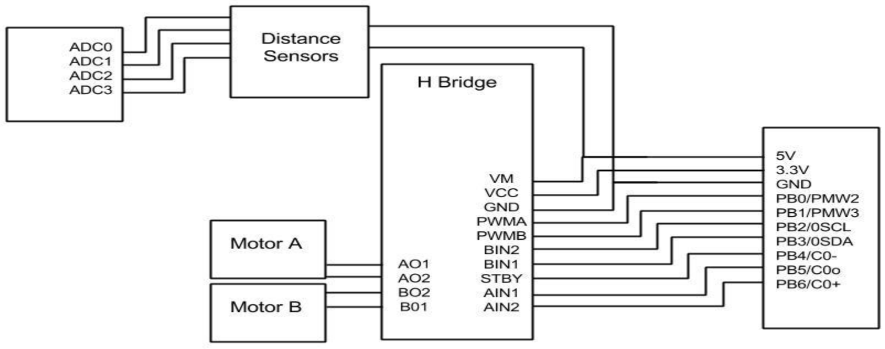
We went on to add support for two PWMs which were used to play a song, Tequila by The Champs, and to control the RoboTank's motor. Each PWM (PWM0 and PWM1) was initialized by writing an initialization function for each. In the initialization process, we enabled the ports that we were going to use it on, set the pulses to an arbitrary frequency, and set them to arbitrary duty cycles. We wrote start and toggle functions for both PWMs that would help us later on this lab.

The first PWM was put to use for using the speaker and the blinking the Status LED on the Stellaris board. We wrote a function called tequila that played a song called Tequila by The Champs.

We went on to add support for the second PWM which was used to drive the RoboTank's two motors via an H-bridge hardware module. We used the H-bridge to enable a voltage that could have been applied in any direction. By using an H-bridge, it allowed us to move the motor on RoboTank forwards or backwards.

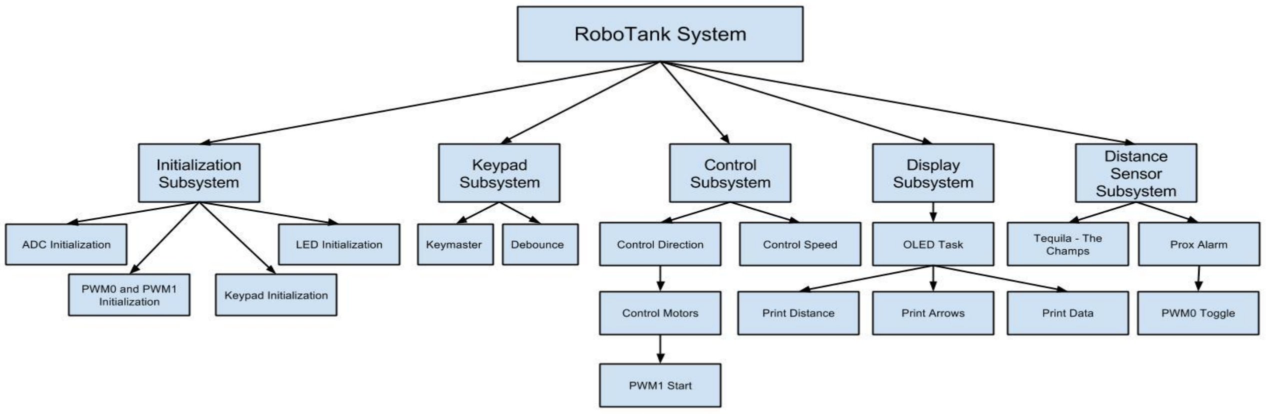
5 Schematic

The following figure displays the schematic of the RoboTank system.



6 Flow Chart

The following figure displays the logic used to run the RoboTank system.



RoboTank System
Initialization Subsystem
Keypad Subsystem
Control Subsystem
Display Subsystem
Distance Sensor Subsystem
ADC Initialization
PWM0 and PWM1 Initialization
Keypad Initialization
LED Initialization
Keymaster
Debounce
Control Direction
Control Motors
PWM1 Start
Control Speed
OLED Task
Print Distance
Print Arrows
Print Data
Tequila - The Champs
Prox Alarm
PWM0 Toggle