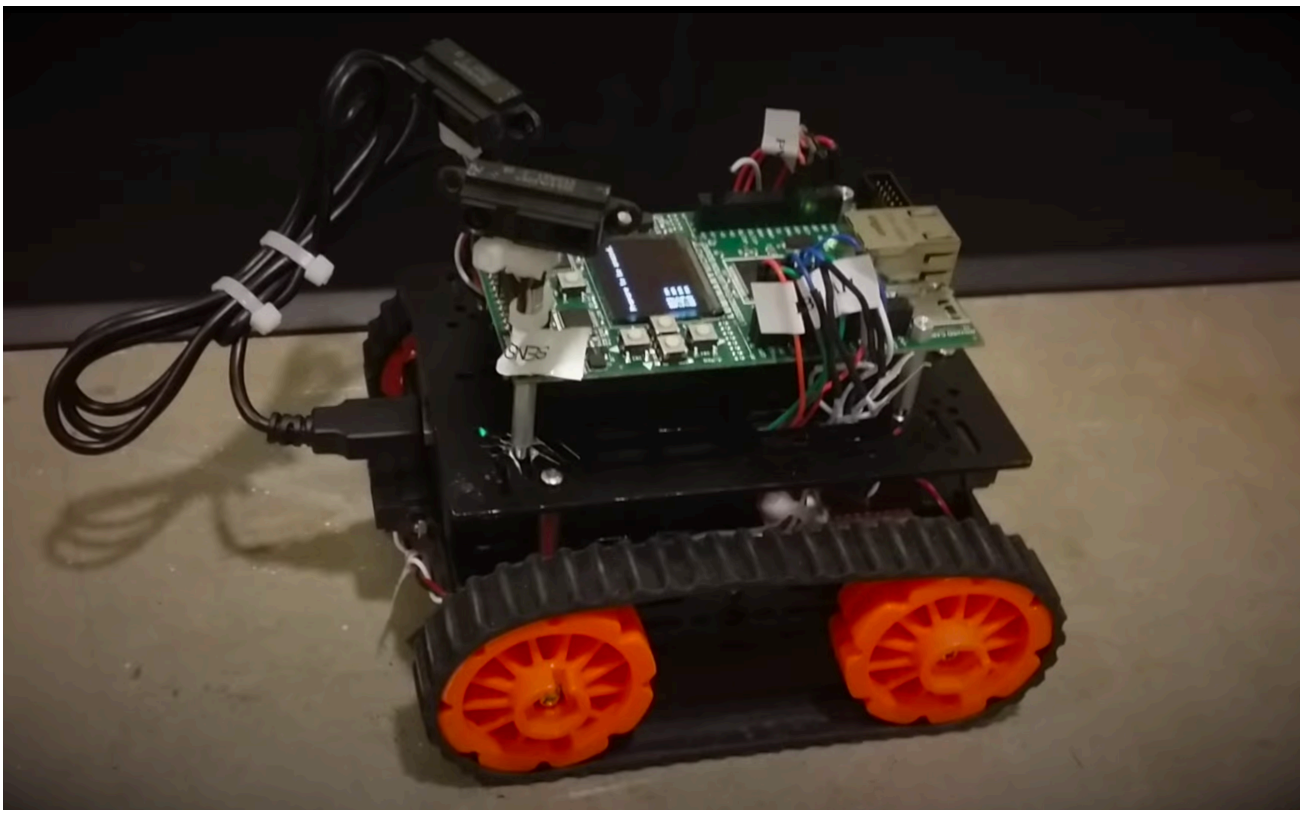
University of Washington
Department of Electrical Engineering
EE 472, Spring 2015

Report for Lab 4:
Introducing a Real Time Operating System

Beginning the RoboTank System.

Report by:
Denny Ly
Eeshan Londhe
Ruchira Kulkarni



INTRODUCTION

The purpose of this lab is to develop the RoboTank system. After understanding how keypads, menus, distance sensors, and interrupts work, the next step is to learn how to generate a PWM signal. The PWM is really important to know how to do as it crucial to driving the motor. Two separate PWM signals are needed for this lab as there are two motors in the RoboTank system. After learning how to use this, the goal is to incorporate all of the subsystems of the tank that we have been building up to all into one large system. To do so, we use a Real Time Operating System. This makes it easier to have multiple tasks running at once. FreeRTOS will be used as the Real Time Operating System. FreeRTOS is the most important part of this lab, as it will help build the entire system together. By learning how to use all of these key topics of embedded systems programming, the RoboTank system will be able to be built.

The key subsystems of the RoboTank are:

- Distance Sensors (ADC's)
- PWM Signals (for the motors)
- User Interface
- Keypad Interface

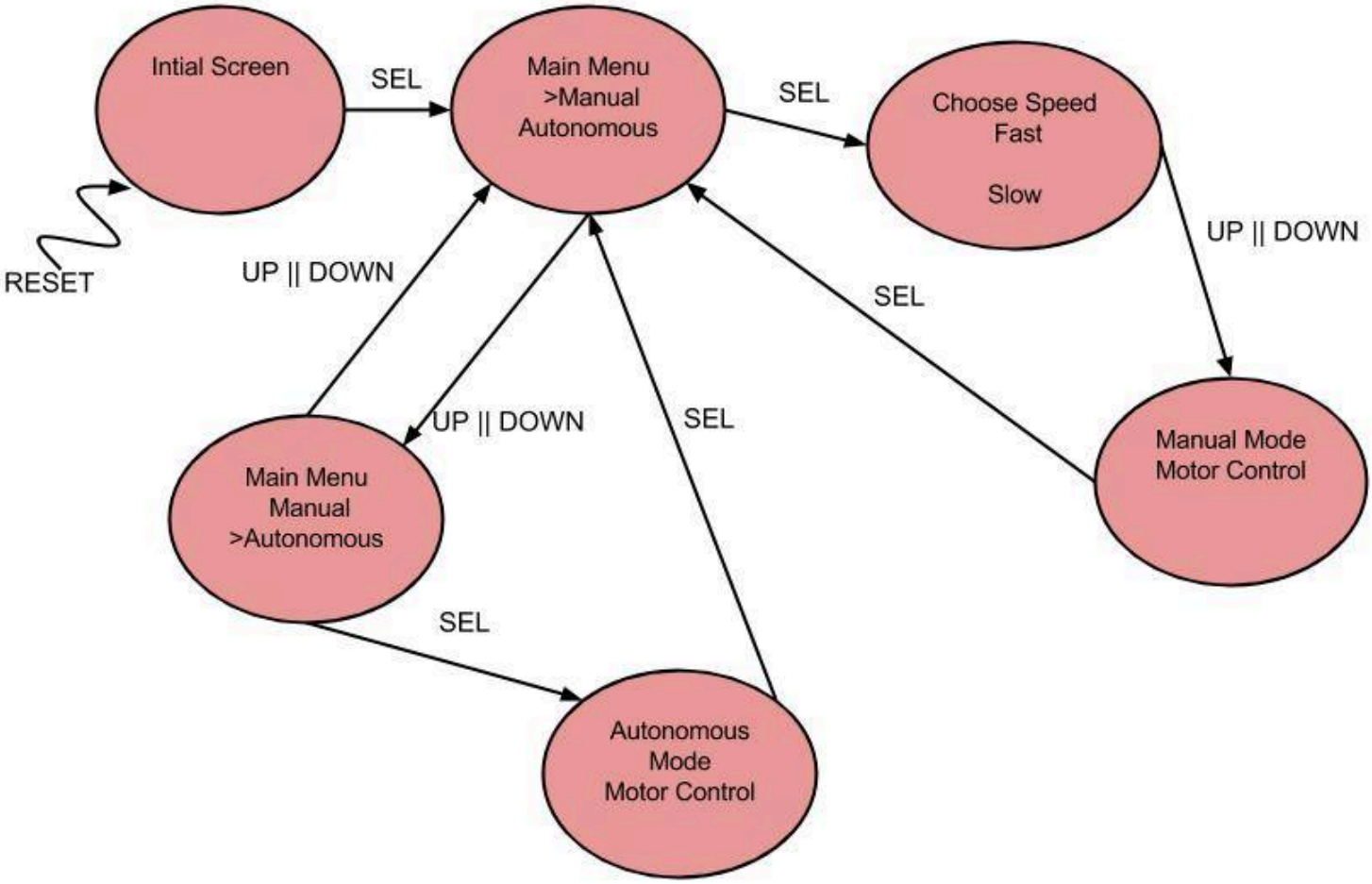
DATASHEETS AND REFERENCES

Some good references to do this lab are to look at the data sheet for both the Stellaris board and the microcontroller. Also, it will help to go on the FreeRTOS website to learn their API so that the built in functions and methods can be used to improve the overall system of the RoboTank.

- 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
- http://www.freertos.org/

CONTROL FLOW

This chart shows the flow of our code throughout the RoboTank System. The system begins with a simple menu screen welcoming the user asking it to press the select key to begin. Once the select key is pressed, the user is prompted with two modes of operation: Manual Mode and Autonomous mode (to be implemented later in lab 5). The user can cycle through these options using the up and down arrow keys on the keypad interface. To choose an option they press select. If the Autonomous mode is chosen, the tank will begin moving by itself, autonomously (will be implemented later). If the Manual mode is chosen, then the user is prompted with the option of choosing a fast mode or a slow mode. They can press Up for fast mode or down for slow mode. Once either of these two modes is chosen, then the tank is ready to operate using all 4 keypad inputs. The user can press up to go up, left to go left, right to go right, and down to go down. In addition to these 4 individual key presses, the user can also press up-right, up-left, down-left, and down-right. These additional 4 key presses allow the tank to move forward while turning right, etc. When the tank is in either manual or autonomous mode, the OLED display is displaying 4 readings for each of the distances from the distance sensor in millimeters refreshing at a near instantaneous rate. One feature we added for extra credit is that the speaker system beeps loudly if any of the distance values drops below a certain threshold, which means that the tank is about to hit something. Also while in these modes, if they user presses select, they are brought back to the initial menu screen prompting the user for whether they want to select the manual or autonomous mode. It is important to note that the tank is not operational once autonomous or manual mode is initiated. This way the key presses control the menu, not the tank, and vice-versa.


Intial Screen
SEL
Main Menu
>Manual
Autonomous
SEL
Choose Speed
Fast
Slow
UP || DOWN
RESET
UP || DOWN
SEL
UP || DOWN
SEL
Manual Mode
Motor Control
Main Menu
Manual
>Autonomous
SEL
Autonomous
Mode
Motor Control

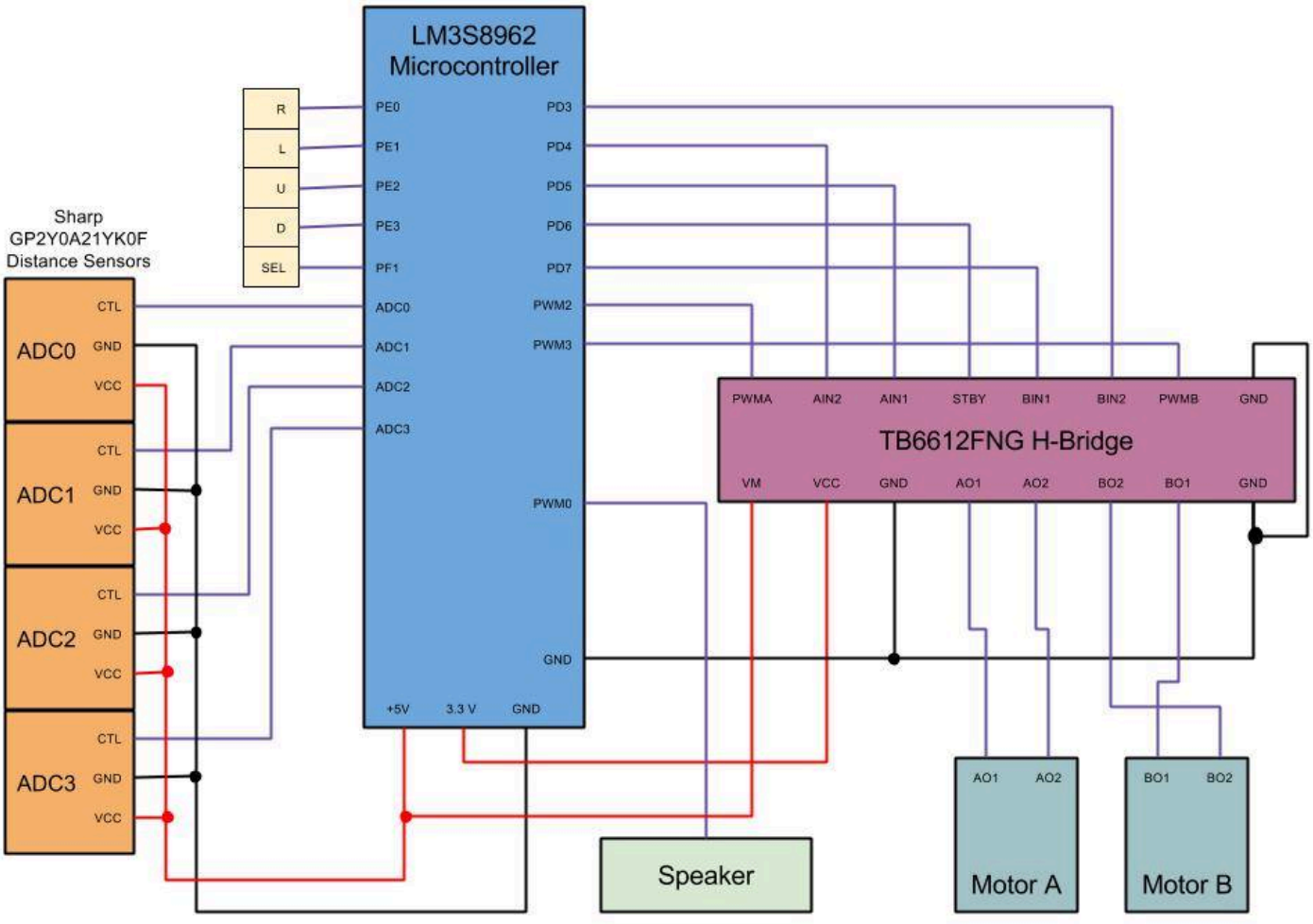
HARDWARE SCHEMATIC

The microcontroller takes in five inputs from the keypad. The four directional key buttons are wired to GPIO Ports E 0 – 3. The select key is wired to Port F 1. There are four distance sensors also wired to the Stellaris board.

The control/data wire is connected to the ADC0 port of the Stellaris board. The power is given to the sensor with a 5V pin on the board, and ground is connected to ground on the board as well. Each CTL pin is wired to ADC 0 – 3. V_CC is connected to 5 V, and GND is connected to the GND pin on the Stellaris board.

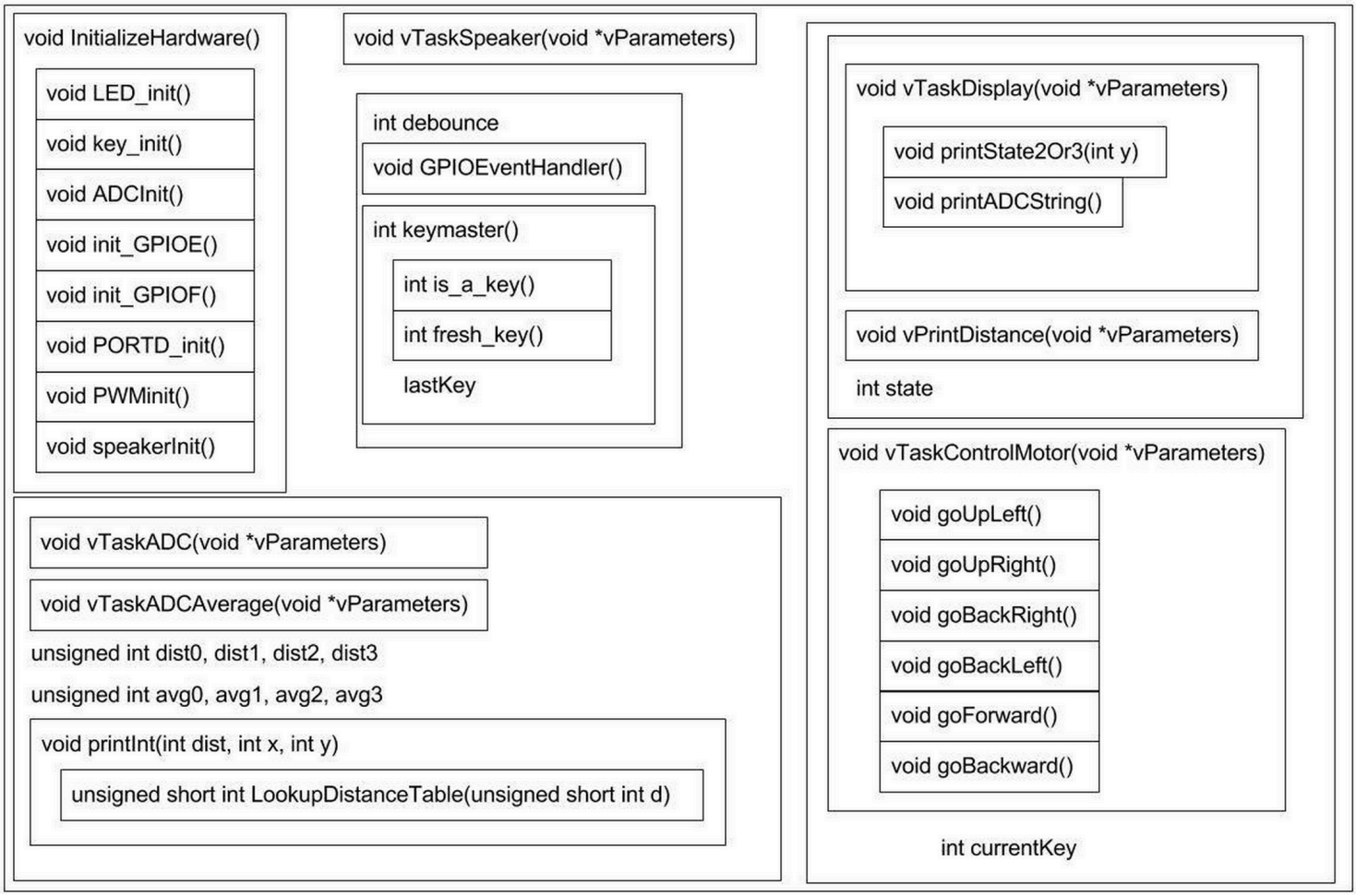
We have also initialized the speaker on the Stellaris board that we can control using PWM0.

To control the two motors, we use an H-bridge. PWMA and PWMB pins on the H-bridge are connected to the PWM2 and PWM3 pins on the Stellaris board respectively. AIN2 is connected to Port D4, AIN1 is connected to Port D5, STBY is connected to Port D6, BN1 is connected to Port D7, and BN2 is connected to Port D3. Vm is connected to 5 V on the Stellaris board, V_CC is connected to 3.3 V on the Stellaris board, and all 3 GND pins on the H-bridge are connected to GND on the Stellaris board. The A01 and A02 pins are connected to the A01 and A02 pins on one motor. The B01 and B02 pins are connected to the B01 and B02 pins on the other motor.



MEMORY DIAGRAM

This memory diagram shows how all of the functions relate to each other. It displays the hierarchy Top-Down to explain how and when a function would be called. The global variables are also displayed and this shows how the functions interact with the global variables to get/set their values.



TIME SPENT

- Design: 5 hours
- Coding: 30 hours
- Test/Debug: 10 hours
- Documentation: 5 hours

Total: 50 hours

Our team members spent an equal amount of time doing equal amount of work. See README for work distribution.

CONCLUSION

In conclusion, this lab was very valuable to learn about PWM signals and Real Time Operating Systems. We learned a lot about how to incorporate lots of subsystems and peripherals into one large cohesive system. We used FreeRTOS to manage multiple tasks running in our system. We learned core concepts of Real Time Operating Systems, including how to use a scheduler to manage tasks, and use interrupts to handle events. Overall, so far we have a working version of the RoboTank system that works in Manual Mode. We can drive the tank using all 4 directional keypads. In addition, for extra credit we implemented a speaker attribute that plays a loud noise when a collision is about to happen. We look forward to adding additional features in Lab 5.