Devlin Brennan
Ryan Bambrough
Shenghao Xue

ECE 315 Report

Lab Goals:

There were several goals to this lab. First, we had to setup the GPIO pins as PWM peripherals. Then we had to configure these PWM peripherals. Lastly, we had to setup drivers that would allows us to make the robot move forward, backward, left, and right.

Steps to Accomplish Goals

In order to accomplish these goals, we first set up the drv8833_gpioInit() function. This is where we setup the GPIO pins for ports B, E, and F. We set up pins 4 and 5 for ports B and E as digital outputs. We also set them up as alternate functions. For port F, we had to set up pin 2 as a digital input and pin 3 as a digital output. We also set up RCGCPWM and RCC inside of the gpioInit as they are global values to be set for all of the possible pwm configurations we would have. We set the RCGCPWM to turn on both pwm module 0 and 1. We also set up the RCC so that it divided the system clock by 2. After setting up the GPIO ports as PWM peripherals, we had to then configure the pwm peripherals.

Inside of the pwm.c file we created the pwmConfig() function. Inside of this function is where we set up the load, cmpa, cmpb, gena, genb registers with the proper values to drive the motors. The values for load, gena, and genb were already given to us as defined variables at the top of the drv8833.c file. We basically set up a switch case statement to determine which pwm generators registers we needed to modify and then enabled the pwm outputs.

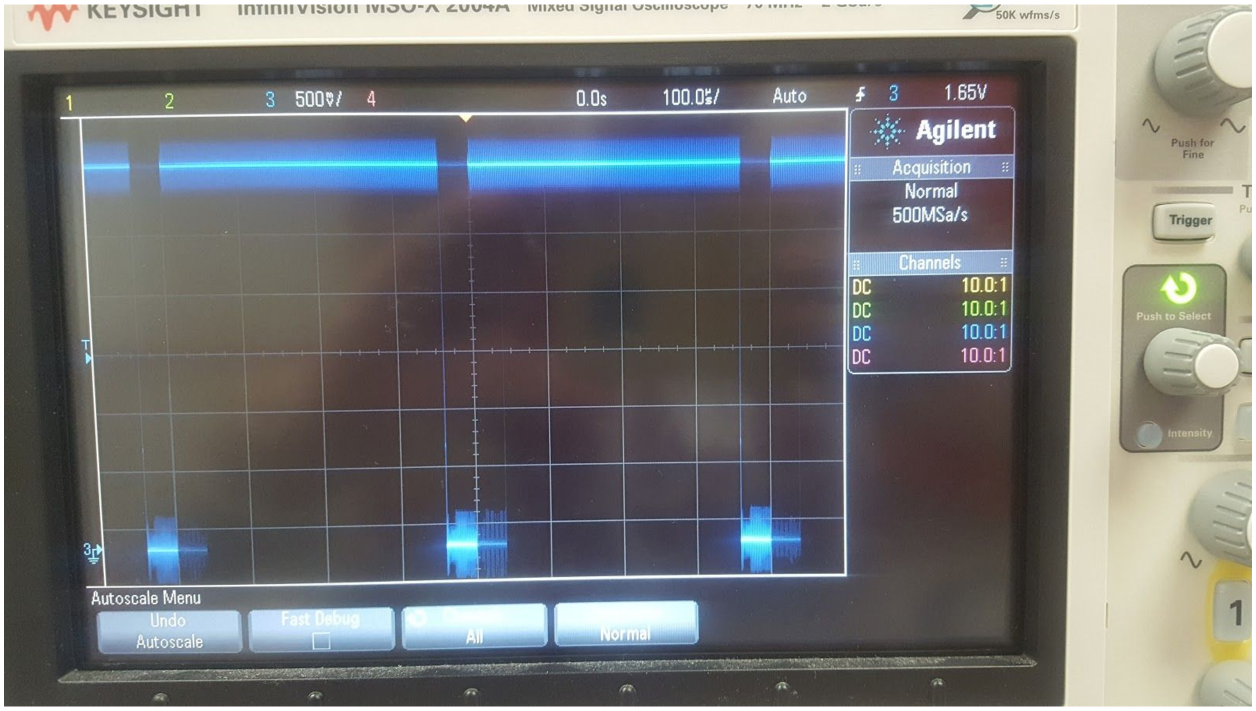
After we created the pwmConfigure() function we used it to code up the various driver functions we had such as rightForward(), rightReverse(), leftForward(), etc. Once we finished these, we just had to create a timer that counted every 1 second and then in the main.c file we added a set of if/else statements to make the robot move forward for 2 seconds, backwards for 2 seconds, and the left and right for 5 seconds each.

Problems and Solutions

We had several problems. Initially we were unable to generate the PWM signal on any of the output pins. Section 20.4 in the manual was a lot of help in resolving this issue. We went through each section in the manual and referenced it to the code that we had initially written. The main errors that we found were in enabling the specific module as well as enabling the PWM output. We ensured that we were generating the

Devlin Brennan
Ryan Bambrough
Shenghao Xue

signal that we thought we were by examining it on the oscilloscope.



After fixing our PWM generation problem we then ran into a problem with the motors not activating. This problem was minor and was resolved by examining the schematic. We found that GPIO port F was connected with the enable for the motors. It was as simple as toggling it when we wanted to use it.

Finally, our methods for driving in different directions were not correct. The robot ended up driving forward for the 14 seconds and only making slight changes in direction. This problem was resolved by examining our methods for each sides' forward and reverse movement. We ended up redoing several of the config parameters to get the robot moving in the right direction.

References

Configure GPIO for PWM:

gpio_enable_port(**BASE**)
gpio_config_digital_enable(**BASE**, **PIN**)
gpio_config_output(**BASE**, **PIN**)
gpio_config_alternate_function(**BASE**, **PIN**)
gpio_config_port_control(**BASE**, **SELECTOR**) *Selector is mux value shifted by 4*Pin

Configure PWM:

```
SYSCTL->RCGCPWM |= 0x3;        // Enable clock to both modules
SYSCTL->RCC |= USEPWMDIV;      // Enable sysclock based
SYSCTL->RCC &= DIV2; // Set sysclock modifier
```

Devlin Brennan
Ryan Bambrough
Shenghao Xue

```
myPWM = (PWM0_Type *) base;
myPWM->_0_LOAD = load;
myPWM->_0_CMPA = cmpa;
myPWM->_0_CMPB = cmpb;
myPWM->_0_GENA = gena;
myPWM->_0_GENB = genb;
myPWM->_0_CTL = 1;          // Enable Generator 0

myPWM->ENABLE |= 0xff     // Enable all outputs
```