CPE403 – Advanced Embedded Systems

Design Assignment 2

Name: Samuel McCormick

Email: samuel.mccormick@gmail.com

Github Repository link (root): https://github.com/brokenboredom/tech-muffin

Youtube Playlist link (root):
https://www.youtube.com/playlist?list=PLCSfNPhZUD_NZxU7Tsm_bMTwxiQqxqpBr

Follow the submission guideline to be awarded points for this Assignment.

Submit the following for all Assignments:

1. In the document, for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the base code for the first task only.
2. Create a private Github repository with a random name (no CPE403/603/710, Lastname, Firstname). Place all labs under the root folder MSP432E4/CC1352, sub-folder named Assignment1, with one document and one video link file for each lab, place modified c files named as asng_taskxx.c.
3. If multiple c files or other libraries are used, create a folder asng1_t01 and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) with startup_ccs.c and other include files, c) text file with youtube video links (see template).
5. Submit the doc file in canvas before the due date. The root folder of the github assignment directory should have the documentation and the text file with youtube video links.
6. Organize your youtube videos as playlist under the name “ADVEMBSYS”. The playlist should have the video sequence arranged as submission or due dates.
7. Only submit pdf documents. Do not forget to upload this document in the github repository and in the canvas submission portal.

I understand the Student Academic Misconduct Policy - http://studentconduct.unlv.edu/misconduct/policy.html

“This assignment submission is my own, original work”.

Samuel McCormick

Task 1) Interface the provided DC Motor with encoder with the DRV8838 Single Brushed DC Motor Driver, provide a PWM signal to rotate the shaft of the motor both CW and CCW to a max of one rotation/revolution.

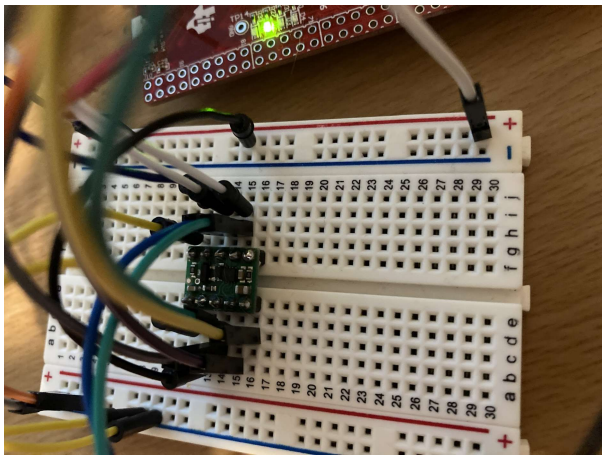
We used the provided github example *pwm_adc* and the *module_10A* slides to create a pwm configuration for a dual band signal at 250Hz with a deadband of 160. Inside *main()* the duty cycle and phase were changed every time we read the ADC so that when the joystick was below 2000 duty cycle was under 50% and phase was logical 1. Above 2100 it was over 50% and logical 0. Between 2000 and 2100 sleep was enabled.

The system clock is 120MHz instead of 16MHz so the value for the period is calculated with 480000 instead of 64000.

And the provided code enabled PWM_GEN_2 instead of PWM_GEN_0.

Altered C Code:

```
MAP_PWMGenPeriodSet(PWM0_BASE,
PWM_GEN_0, 480000);

MAP_PWMGenEnable(PWM0_BASE,
PWM_GEN_0);
```



Motor driver chip and wiring.

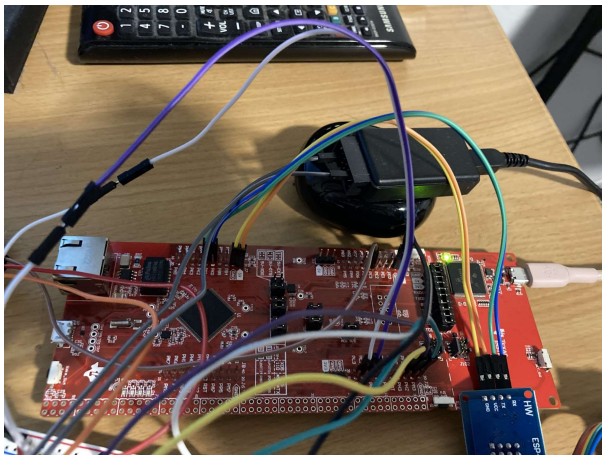
Task 2) Using the QEI module determine the number of ticks per revolution and fix the speed of the motor to a slow positional value.

The QEI was configured using the github example *QEI_MSP* as well as *module_10A* slides to capture the position as well as the velocity with PL1 and PL2 reading the PWM outputs

I have no way to drive this motor however and have not been able to encode the position.

C code with velocity added:

```
QEIVelocityConfigure(QEI0_BASE,
QEI_VELDIV_1, systemClock);

QEIEnable(QEI0_BASE);
QEIVelocityEnable(QEI0_BASE);
```



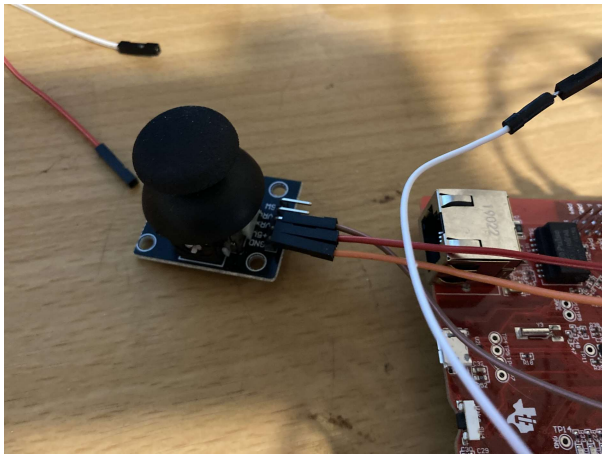
Logic analyzer and MSP432 board

Task 3) Interface a joystick to ADC0 channel and control the position of the motor shaft to move 0-180 deg or 0-360* deg based on the position of the joystick. Only use one axis of the joystick to control the position of the motor shaft. If the joystick is in the center the motor shaft should be at 90 deg or 180* deg. Moving the joystick left or right should proportionally move the motor shaft CCW or CW.

Using code from Assignment 1 as well as the github example *pwm_adc* the joystick x-axis was connected to PE3 and successfully controlled the duty cycle for the PWM signal and the phase.

The joystick reads around 2025 when idle and 4055 when fully pushed so it was divided by 41 to prevent the duty cycle from going over 100.

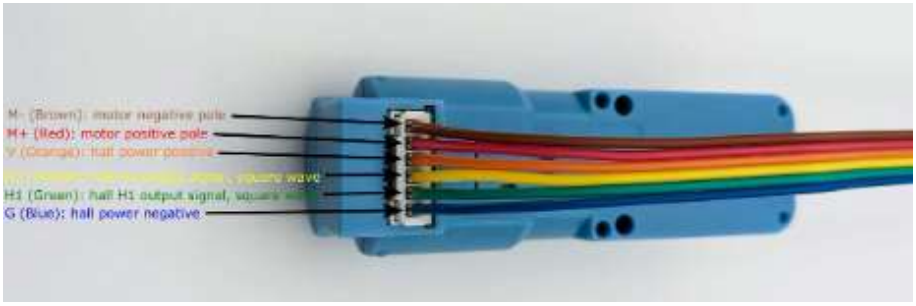
```
if(getADCValue[0] <= 2000)
    {
      // Create a condition for QEI reading

      GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0x1);
      GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0x1);
      MAP_PWMPulseWidthSet(PWM0_BASE, PWM_OUT_1,
                  MAP_PWMGenPeriodGet(PWM0_BASE,
                  PWM_GEN_0) * getADCValue[0]/41/100);
    }
    else if (getADCValue[0] >= 2100)
    {
      GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0x0);
      GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0x1);
      MAP_PWMPulseWidthSet(PWM0_BASE, PWM_OUT_1,
                  MAP_PWMGenPeriodGet(PWM0_BASE,
                  PWM_GEN_0) * getADCValue[0]/41/100);
    }
```



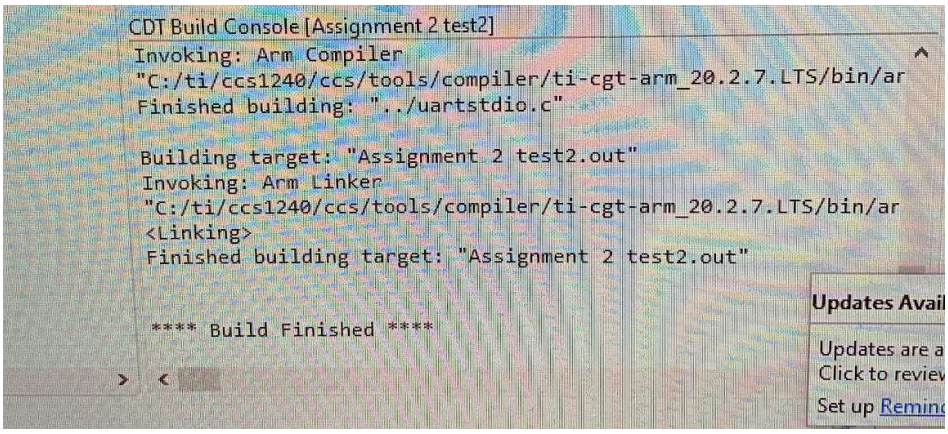
Joystick with x-axis connected.

PHASE	ENABLE	SLEEP	OUT1	OUT2	operating mode
0	PWM	1	PWM	L	forward/brake at speed *PWM* %
1	PWM	1	L	PWM	reverse/brake at speed *PWM* %
X	0	1	L	L	brake low (outputs shorted to ground)
X	X	0	Z	Z	coast (outputs floating/disconnected)

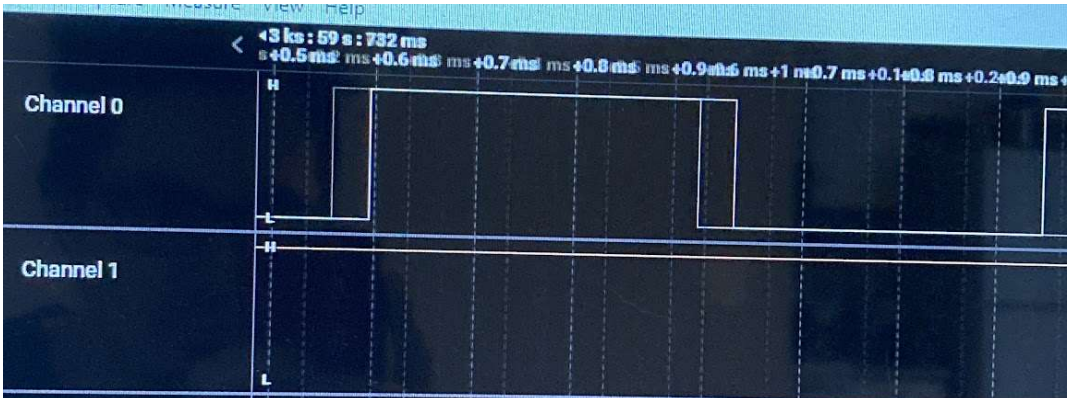
Pinout used to interface motor with driver chip. Could not drive motor through chip even with the most basic code. Motor still runs when connected to voltage.



Motor pinout was incorrect. It is the inverse of what is shown here. Green and blue are M+ and M- etc.



Assignment 2 compiled.



Logic analyzer connected to PF1. Shows changing duty cycle controlled by joystick.

Youtube Video: https://youtu.be/X56IL9WD81A?si=htEzi1Wr-qpiSLTS