TA Andrew Gerber 10/27/2023

Lab 07 Report

Benjamin Meads
A02323437
Nick Templeton
A02268123
Richard Snider
A02298443

Objectives

This lab was meant to introduce us to the stepper motor system. How to turn it by various methods, how to move it to a precise rotation, how to turn it quickly and at a specific rate, and how to interface with the stepper motor subsystem more generally. It was also meant to teach us how to use the buzzer to make sound using pulse width modulation.

Procedure

We started out by passing off the schematic with the TA, who had a few notes, which we addressed. Richard initially started off doing the wiring and Ben the coding, but we switched a bit later for efficiency as we were both in each others' domains a little bit.

We realized about a half hour into the process that our keypad code was connected to the same GPIO ports as we planned to connect our motor to because we had misremembered the way our keypad code worked. Since switching the keypad code seemed like a more annoying solution, we just rewired the motor to use different ports.

We were still not confident in our knowledge of the stepper motor, so we took some time to review it together. We implemented full stepping as described by the book, but we noticed some strange behavior with it. For instance, if the rotation was blocked with a finger, the motor would reverse directions, and the motor would also spend a bit of time waggling back and forth before committing to a direction. We were advised by another group to use wave stepping instead, and that did indeed solve the problems we were having. We also had trouble with the motor moving incredibly slowly and much less than expected per step, which was also solved by wave stepping.

It was at this point that we ran out of time in our second session. At this point, we could program several seconds to turn and it would go there pretty accurately. Benjamin was able to do the rest on his own time, This included making changes to the clockwise and counterclockwise functions to act properly. After that, the buzzer implementation went smoothly. Just to call a buzzer function after the clock finishes counting down inside our main function.

Near the end of the lab, I couldn't get our motor to respond to inputs 1-9. It would automatically switch to whatever number was pressed multiplied by 10. As I was attempting to debug this. I had placed the buzzer function call when each button was pressed. And because I didn't realize I needed a delay in there. Dr. Phillips passed me off even though it had the buzzer. Luckily, I was able to get a normal delay in that spot and it worked as described.

Results

The results of this lab were accurate to the specifications listed in the requirements. We were able to get the keypad, buzzer, and motor all hooked up correctly. Whenever we would press a button on the keypad and press the "#" key. It would move quickly to that position. After another "#" press, it would count down accurately (To about a 1-2 second accuracy in a full 60-second cycle). After the second hand counted down all the way. The buzzer would let off a 1-second long buzzer sound. Thus completing the lab.

Conclusion

Our lab went by without too many issues. It did have us going back to the code we wrote to re-vamp it a few times. We spend time understanding how wave-stepping, and full-stepping work inside of a motor. We understood how to modify our registers and our ODR to make the motor move smoothly instead of choppy. And we learned how to implement our buzzer. It was also helpful to learn more ways to implement the keypad into different systems. It leads to a deeper understanding of how each part fits in together. As well as the microcontroller as a whole.

Programs

Stepper.c

```
#include "stepper.h"

#include "stm32l476xx.h"
#include "keypad.h"
#include <stdio.h>
static unsigned char FullStep[4] = {0x1, 0x2, 0x4, 0x8};

void stepinit(void)
{
 //setting up clock for GPIO B
 RCC->AHB2ENR |= RCC_AHB2ENR_GPIOBEN;

 uint32_t new_moder = GPIOB->MODER;
 //sets PB 0-3 to be 01 for output mode
 new_moder &= 0xFFFFFF00;
 new_moder |= 0x00000055;
 GPIOB->MODER = new_moder;
}

void turn_seconds_ccw(int seconds)
{
 //unsigned int a, ap, b, bp;
 unsigned int odr;
 int steps = (int) (7.75 * seconds);

 ///////////////////////////CLOCKWISE ROTATION
//////////////////////////////
 for(int i = 0; i < steps; i++) //64 steps * 5.632 deg = 360 deg
 {
       for(int j = 0; j < 4; j++) //clockwise roatation
       {
             delay_ms(10); //a short delay

             //Set the value of the GPIO ODR
             odr = GPIOB->ODR;
             odr &= 0xFFFFFFF0;
             odr |= FullStep[j];
             GPIOB->ODR = odr;
       }
```

```
  }
}

void turn_seconds_cw(int seconds)
{
  unsigned int odr;
  int steps = (int) (7.75 * seconds);

  ////////////////////////COUNTER-CLOCKWISE
ROTATION////////////////////////
  for(int i = 0; i < steps; i++) //64 steps * 5.632 deg = 360 deg
  {
        for(int j = 3; j >= 0; j--) //clockwise roatation
        {
              delay_ms(10);

              //Set the value of the GPIO ODR
              odr = GPIOB->ODR;
              odr &= 0xFFFF0000;
              odr |= FullStep[j];
              GPIOB->ODR = odr;
        }
  }
}
```

Stepper.h

```
void stepinit(void);
void turn_seconds_cw(int);
void turn_seconds_ccw(int);
```

Main.c

```
#include "stm32l476xx.h"

#include "keypad.h"
#include "stepper.h"

int main(void){
      stepinit();
      keypad_init();
      int done, placeValue, counter;
      while(1) {
            done = 0;
         placeValue = 1;
```

```
            counter = 0;
            while (!done) {
                  unsigned char next = readAllRows();
                  if (48 <= next && next <= 57 && placeValue >= 0) {
                        int digit = next - 48;

                        if (placeValue == 1) {
                              counter += digit * 10;
                        }
                        else {
                              counter += digit;
                        }
                        placeValue--;
                  }

                  if (next == '#') {
                        done = 1;
                  }
            }

            turn_seconds_cw(counter);
      }
}
```