Sophomore Design Project One

Group 17

Caitlyn Svitek and Jessica Kuo

Objective and Approach

The first project entails coding a Pololu 3Pi robot to travel to a given coordinate on a 2D plane. The coordinate is decided by a user inputting the coordinate using switches to a Liquid Crystal Display (LCD). To accomplish this, we decided to use three switches; one for the X-coordinate, one for the Y-coordinate, and one to tell the robot it was time to move. The switches are mapped to two motors, which, when the third switch was pressed, would save the input and move to the desired destination.

Key Design Decisions

In general, the robot would take inputs from two switches, each one controlling the values of an X and Y direction, respectively. A third switch saves the input and signals the motors to either drive forward or turn to move to the input.

```
    uint8_t left_button_pressed = (PINB & (1<<1)); // button a
    uint8_t middle_button_pressed = (PINB & (1<<4)); //button b
    uint8_t right_button_pressed = (PINB & (1<<5)); //button c
```

Register B was chosen to store button inputs, and pins 1, 4, and 5 were used to store values dictated by left, middle, and right buttons, respectively. By using bit masking, a value would only be changed to a logical 0 if the desired button was pressed as well, making sure that the correct command was being executed.

To control motor speed, PWM (Post-Width Modulation) was used. PWM is controlled using a clock and a duty cycle. We decided to use an inverting PWM signal; when duty cycle was decreased, motor speed decreased. As seen in the following snippet, the duty cycle chosen was 30 for reversing the robot:

```
void reverse(int j){
    reverse_motorL = 30;
    reverse_motorR = 30;
    _delay_ms(j*1400);
    reverse_motorL = 0;
    reverse_motorR = 0;
    _delay_ms(500);
}
```

In the code, reverse_motorL and reverse_motorR are the registers responsible for moving the motors backwards. The _delay_ms() is responsible for controlling how long to run the motors before stopping them. The expression *j*1400* was used because for every inch the robot had to reverse, the motors would have to run for 1400ms. For turning, the robot only needed to turn 90, 180, and 270 degrees. We decided that the robot would only turn right until it reached one of those angles. In order to make sure the robot didn't move as it turned, one wheel would spin forward while the other backwards.

```
void turn(int j){
    forward_motorL = 30;
    reverse_motorR = 30;
```

```
	_delay_ms(j);
	forward_motorL = 0;
	reverse_motorR = 0;
	_delay_ms(500);
}
```

Similarly to the forward() function, the turn() function is controlled by activating two different registers, named forward_ motorL and reverse_motorR, each controlling one motor in an opposite direction. The length of the _delay_ms() is determined in our defines:

```
#define turn_90 2000
#define turn_180 3940
#define turn_270 5750
```

With the values for 90, 180, and 270 degrees specified.

Challenges

One difficult part of the project was dealing with the 3Pi's directional bias. In our case, the robot tended to turn to the left as it moved; as a result, the left motor was made to be slightly faster than the right so the two would be more in sync.

```
	void forward(int j){
	forward_motorL = 31.5;
	forward_motorR = 30;
	int i;
	for(i=0;i<j;i++){
	_delay_ms(700);
	}
	forward_motorL = 0;
	forward_motorR = 0;
	_delay_ms(500);
}
```

As seen with the forward() function, the duty cycles for the left motor (forward_motorL) and the right motor (forward_motorR) are slightly different to mitigate the directional bias. Furthermore, to minimize error, turns were done so the robot wouldn't change location as it turned by moving one wheel forward and the other backward. Additionally, the robot would come to a complete stop before completing it's next command rather than coasting. Speed was also decreased because although it took longer to reach the destination, the robot would travel smoother and wouldn't jerk around as much when starting and stopping. All duty cycles were reduced to approximately 30 because of this.

Appendix

```
#include <stdint.h>
#include <stdio.h>
#include <util/delay.h>
#include <avr/io.h>
#include "lcd_driver.h"
#include "port_macros.h"

#define buttons_in DDRB
#define buttons_out PORTB
#define motor1 DDRD
#define motor2 DDRB
#define reverse_motorL OCR0A
#define reverse_motorR OCR2A
#define forward_motorR OCR2B
#define forward_motorL OCR0B
#define PWM_motorL TCCR0A
#define PWM_motorR TCCR2A
#define clock_motorL TCCR0B
#define clock_motorR TCCR2B
#define turn_90 2000
#define turn_180 3940
#define turn_270 5750

//functions
char* int_to_string(int i){
      char* str;
      if(i==25){str = "25";}
      if(i==24){str = "24";}
      if(i==23){str = "23";}
      if(i==22){str = "22";}
      if(i==21){str = "21";}
      if(i==20){str = "20";}
      if(i==19){str = "19";}
      if(i==18){str = "18";}
      if(i==17){str = "17";}
      if(i==16){str = "16";}
      if(i==15){str = "15";}
      if(i==14){str = "14";}
      if(i==13){str = "13";}
      if(i==12){str = "12";}
      if(i==13){str = "13";}
      if(i==12){str = "12";}
      if(i==11){str = "11";}
      if(i==10){str = "10";}
      if(i==9){str = "9";}
      if(i==8){str = "8";}
      if(i==7){str = "7";}
      if(i==6){str = "6";}
      if(i==5){str = "5";}
      if(i==4){str = "4";}
      if(i==3){str = "3";}
      if(i==2){str = "2";}
      if(i==1){str = "1";}
      if(i==0){str = "0";}
      if(i==-1){str = "-1";}
      if(i==-2){str = "-2";}
      if(i==-3){str = "-3";}
      if(i==-4){str = "-4";}
      if(i==-5){str = "-5";}
      if(i==-6){str = "-6";}
      if(i==-7){str = "-7";}
      if(i==-8){str = "-8";}
      if(i==-9){str = "-9";}
      if(i==-10){str = "-10";}
      if(i==-11){str = "-11";}
      if(i==-12){str = "-12";}
      if(i==-13){str = "-13";}
      if(i==-14){str = "-14";}
      if(i==-15){str = "-15";}
      if(i==-16){str = "-16";}
      if(i==-17){str = "-17";}
      if(i==-18){str = "-18";}
      if(i==-19){str = "-19";}
      if(i==-20){str = "-20";}
      if(i==-21){str = "-21";}
      if(i==-22){str = "-22";}
      if(i==-23){str = "-23";}
      if(i==-24){str = "-24";}
      if(i==-25){str = "-25";}
      return str;
}

int printer(int i){
      char* str;
      if(i==26){i=-25;}
      str = int_to_string(i);
      LCD_print_String(str);
      return i;
}
```

```
void forward(int j){
      forward_motorL = 31.5;
      forward_motorR = 30;
      int i;
      for(i=0;i<j;i++){
      _delay_ms(700);
      }
      forward_motorL = 0;
      forward_motorR = 0;
      _delay_ms(500);
}

void turn(int j){
      forward_motorL = 30;
      reverse_motorR = 30;
      _delay_ms(j);
      forward_motorL = 0;
      reverse_motorR = 0;
      _delay_ms(500);
}

void reverse(int j){
      reverse_motorL = 30;
      reverse_motorR = 30;
      _delay_ms(j*1400);
      reverse_motorL = 0;
      reverse_motorR = 0;
      _delay_ms(500);
}

void moving_motors(int x, int y){
      LCD_move_cursor_to_col_row(0x02,0x03);
      printer(y);//check if you need y= here
      LCD_move_cursor_to_col_row(0x02,0x00);
      printer(x); // check if you need x= here
      if(y>=0){
            forward(y);
            if(x>=0){
                  turn(turn_90);
                  forward(x);
            }
            else{
                  turn(turn_270);
                  x = 0-x;
                forward(x);
            }
        }
        if(y<0){
            y = 0-y;
            turn(turn_180);
            forward(y);
            if(x>0){
                turn(turn_270);
                forward(x);
            }
            else{
                x = 0-x;
                turn(turn_90);
                forward(x);
            }
        }


    }

    int main(){
      buttons_out |= (1<<1) | (1<<4) | (1<<5);
      buttons_in &= ~(1<<1) & ~(1<<4) & ~(1<<5);
      motor1 |= (1<<6) | (1<<5) | (1<<3);
      motor2 |= (1<<3);
      PWM_motorL = PWM_motorR = 0xF3;
      clock_motorL = clock_motorR = 0x02;

      forward_motorL = forward_motorR = reverse_motorL = reverse_motorR=0;

      int x_coord =0, y_coord=0;

      while(1){
            uint8_t left_button_pressed = (PINB & (1<<1)); // button a
            uint8_t middle_button_pressed = (PINB & (1<<4)); //button b
            uint8_t right_button_pressed = (PINB & (1<<5)); //button c

    // LCD screen
      initialize_LCD_driver();
      LCD_execute_command(TURN_ON_DISPLAY);
      LCD_move_cursor_to_col_row(0X00,0X03);
      LCD_print_String("Y=");
      LCD_move_cursor_to_col_row(0X00,0X00);
      LCD_print_String("X=");
```

```
//setting up the buttons
if(left_button_pressed  == 0x00){
      x_coord++;
      _delay_ms(500);
}
if(middle_button_pressed == 0x00){
      y_coord++;
      _delay_ms(500);
}
if(right_button_pressed == 0x00){
      _delay_ms(1000);
      moving_motors(x_coord,y_coord);
}
 LCD_move_cursor_to_col_row(0x02,0x03);
 y_coord = printer(y_coord);//check if you need y= here
LCD_move_cursor_to_col_row(0x02,0x00);
x_coord = printer(x_coord); // check if you need x= here
}

      return 0;
}
```