ECE 375 LAB 2

Introduction to AVR Development Tools

Lab Time: Mon 12pm-2pm

James Stallkamp

INTRODUCTION

This lab is focused on using the I/O ports on the board to provide some sensory input. In the lab we were to use some I/O ports as input to "detect" a collision with one of the whiskers then change the bots behavior based on which input was triggered. The main concept behind this lab seems to be getting familiar with using I/O ports on the board to understand how data gets into the device.

PROGRAM OVERVIEW

The program works by running a loop forever and waiting for inputs from an I/O port. When one of the I/O ports is activated the bot changes behavior based on the input. The important part of code is the series of conditionals that evaluates which input was triggered and changes behavior. For each I/O or whisker there is a conditional to match it and for each one there are a set of commands to make the bot stop and change direction then continue.

INITIALIZATION ROUTINE

We did not have an initialize.

MAIN ROUTINE

The main routine in this case was fairly simple. We mapped some I/O ports then went directly into an infinite loop to wait for sensory input.

ADDITIONAL QUESTIONS

1) *This lab required you to compile two C programs (one given as a sample, and another that you wrote) into a binary representation that allows them to run directly on your mega128 board. Explain some of the benefits of writing code in a language like C that can be "cross compiled". Also, explain some of the drawbacks of writing this way*

There are two main benefits to writing in C. First C is generally an easier language to understand and allows for powerful instructions that would require multiple lines in assembly. The next advantage is that C can be written and converted easily into most languages, IE you can write assembly using C but not write C using assembly. The main advantages of writing in assembly are that it is lower level giving the developer more control of what exactly the code is doing. Also that writing in direct assembly generally creates smaller and more efficient implementations.

2) *The C program you just wrote does basically the same thing as the sample assembly program you looked at in Lab 1. What is the size (in bytes) of your Lab 1 & Lab 2 output .hex files? Can you explain why there is a size difference between these two files, even though they both perform the same BumpBot behavior?*

   The lab1 hex file was 485 bytes and the lab 2 hex file is 799 bytes. The difference in size is because the c file has to be converted into assembly and during that compilation extra functions and dependencies have to added.

DIFFICULTIES

The only hard part of this lab was configuring input and outputs.

CONCLUSION

In this lab the primary focus was using the bots I/O ports to send data to the board. This allows us to user information obtained by the robot to change behavior or respond to things around it.

SOURCE CODE

```
#define F_CPU 16000000

#include <avr/io.h>

#include <util/delay.h>

#include <stdio.h>

int main(void)

{

	DDRB = 0b11110000;	// configure Port B pins for input/output

	PORTB = 0b01100000;	// set initial value for Port B outputs

	// (initially, disable both motors)

	DDRD = 0b00000000;

	PORTD = 0b11111111;

	while (1) // loop forever

	{

		PORTB=0b01100000;


		if(PIND==0b11111100)

		{

			PORTB = 0b01100000;	// make TekBot move forward

			_delay_ms(1000);	// wait for 1 s

			PORTB = 0b00000000;	// make TekBot move backwards

			_delay_ms(500);	// wait for 500 ms

		}

		else if(PIND==0b11111110)
```

```
                {
                        PORTB = 0b01100000;     // make TekBot move forward
                        _delay_ms(1000);        // wait for 1 s
                        PORTB = 0b00000000;     // make TekBot move backwards
                        _delay_ms(500);         // wait for 500 ms
                        PORTB = 0b00100000;     // turn left
                        _delay_ms(500);        // wait for 500 ms

                }
                else if (PIND==0b11111101)
                {
                        PORTB = 0b01100000;     // make TekBot move forward
                        _delay_ms(1000);        // wait for 1 s
                        PORTB = 0b00000000;     // make TekBot move backwards
                        _delay_ms(500);         // wait for 500 ms
                        PORTB = 0b01000000;     // turn right
                        _delay_ms(500);        // wait for 500 ms

                }
        }
}
```

Challenge Source Code

```
#define F_CPU 16000000
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>

int main(void)
{
        DDRB = 0b11110000;      // configure Port B pins for input/output
        PORTB = 0b01100000;     // set initial value for Port B outputs
```

```
// (initially, disable both motors)
DDRD = 0b00000000;
PORTD = 0b11111111;
while (1) // loop forever
{
	PORTB=0b01100000;

	if(PIND==0b11111100)
	{
		PORTB = 0b01100000;	// make TekBot move forward
		_delay_ms(1000);	// wait for 1 s
		PORTB = 0b00000000;	// make TekBot move backwards
		_delay_ms(500);	// wait for 500 ms
	}
	else if(PIND==0b11111110)
	{
		PORTB = 0b01100000;	// make TekBot move forward
		_delay_ms(1000);	// wait for 1 s
		PORTB = 0b00000000;	// make TekBot move backwards
		_delay_ms(500);	// wait for 500 ms
		PORTB = 0b00100000;	// turn left
		_delay_ms(500);	// wait for 500 ms

	}
	else if (PIND==0b11111101)
	{
		PORTB = 0b01100000;	// make TekBot move forward
		_delay_ms(1000);	// wait for 1 s
		PORTB = 0b00000000;	// make TekBot move backwards
		_delay_ms(500);	// wait for 500 ms
		PORTB = 0b01000000;	// turn right
		_delay_ms(500);	// wait for 500 ms
```

```
                }
        }
}
```