Embedded Hardware Design Lab4

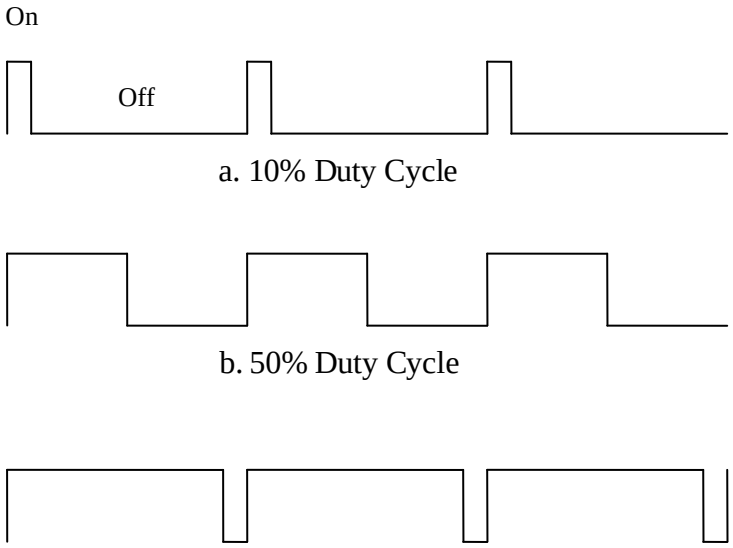
Objective: Controlling the speed of dc motor using light sensor (LDR).

In this lab, we would want to control the speed of a DC motor with the help of light sensor. This would allow us to learn basic sensing and control mechanism.

To control the speed of DC motor, we will use Pulse Width Modulation, abbreviated as PWM. PWM is a method of transmitting information on a series of pulses. The data that is being transmitted is encoded on the width of these pulses to control the amount of power being sent to a load.

PWM is very handy tool, you can use it for power delivery, voltage regulation and amplification and audio effects. In the previous lab you have gone through the basic understanding of PWM, register configuration for different modes of PWM in atmega32.

Analog voltage and current can be used to control devices directly like speed of a fan. In a simple analog controller, a knob is connected to a variable resistor. As you turn the knob, the resistance goes up or down. As that happens, the current flowing through the resistor increases or decreases. As intuitive and simple as analog control may seem, it is not always economically attractive or otherwise practical. Analog circuits can get very hot; the power dissipated is proportional to the voltage across the active elements multiplied by the current through them. Analog circuitry can also be sensitive to noise. By controlling analog circuits digitally, system costs and power consumption can be drastically reduced. What's more, many microcontrollers already include on-chip PWM controllers, making implementation easy, for an example atmega32 comes with 3 timers all can be configured to used for PWM.



a. 10% Duty Cycle

b. 50% Duty Cycle

c. 90% Duty Cycle

Figure1

Figure 1 shows three different PWM signals. Figure 1a shows a PWM output at a 10% duty cycle. That is, the signal is on for 10% of the period and off the other 90%. Figures 1b and 1c show PWM outputs at 50% and 90% duty cycles, respectively. These three PWM outputs encode three different analog signal values, at 10%, 50%, and 90% of the full strength. If, for example, the supply is 5V and the duty cycle is 10%, a 0.5V analog signal results

pwm

Explanation:
In the previous lab2 you have done pwm program which generates the square pulse on pin pd-4/pd-5.

Library Files: avr/io.h
avr/interrupt.h
avr/signal.h
inttypes.h

Library Functions:
sei

Functions: pwm1a_init()
pwm1a_setDutyCycle()

Registers: DDRD
DDRB PORTB
OCR1AL
TCCR1A
TCCR1B
ADCSRA
ADC (ADCH,ADCL)

Local Variables:
int i,j

ADC

For ADC: Do refer
Features and operation of ADC: Page. 199 onwards
Registers: Page. 212 onwards

An analog-to-digital converter (abbreviated ADC, A/D or A to D) is a device that converts a continuous quantity to a discrete time digital representation. An ADC may also provide an isolated measurement. The reverse operation is performed by a digital-to-analog converter (DAC).

Typically, an ADC is an electronic device that converts an input analog voltage or current to a digital number proportional to the magnitude of the voltage or current. However, some non-electronic or only partially electronic devices, such as rotary encoders, can also be considered ADCs.

The digital output may use different coding schemes. Typically the digital output will be a two's complement binary number that is proportional to the input, but there are other possibilities. An encoder, for example, might output a Gray code.

Example 1 , Reading ADC with 500 msec delay

```
#include<avr/io.h>
#include <avr/interrupt.h>
#include <inttypes.h>

SIGNAL (SIG_ADC)
{
        PORTB = ADCH;
        insert a For loop of 500 msec delay.
        ADCSRA = ADCSRA | 0x40; // start the next conversion
}

void main(void)
{
        DDRB = 0xFF;  // least signficant 3 bits for output
        ADMUX = ( 1<< ADLAR);
        ADCSRA = ( 1 << ADEN )| (1 << ADIE) | ( 1 <<ADPS2);
sei ();
while (1);
}
```

Program that generates the square pulse using the timer1:

```
#include <avr/io.h>
#include <avr/delay.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
#define F_CPU 1000000UL
void pwm1a_init();
void pwm1a_setDutyCycle(uint8_t dc);
void pwm1a_init(void)
{
DDRD = DDRD | (1 << PD5)|( 1 << PD4);
OCR1AL = 0x00;
TCCR1A = TCCR1A | (1 << WGM10) | (1 << WGM11) | (1 << COM1A1);
TCCR1B = TCCR1B | (1 << CS10)| (1<<CS12);
return;
}

void pwm1a_setDutyCycle(uint8_t dc)
{
OCR1AL = dc;
return;
}

int main(void)
{
DDRB = 0xFF;
DDRD =0x00;
PORTB=0xFF;

uint8_t i=0;
uint16_t j=0,k=0;
pwm1a_init();
pwm1a_setDutyCycle(125);
sei();

while (1)
{
        for(i=0;i<255;i=i+5)
{
        //pwm1a_setDutyCycle(125);
        for(j=0;j<50000;j++);
        for(k=0;k<=50000;k++);
}
PORTB=~PORTB;
}
```

Circuit-Kit Setup:

To connect circuit-kit for above experiment we need 4 cables.
(1) Serial cable to connect serial port of computer to programmable serial port of kit.
(2) 8-wire cable to connect port-b to LEDs of kit.
(3) Cable to select target microcontroller slot to program.
(4) 2-wire cable to connect port-d (pd4 or pd5) to analog input from light sensor.
(5) For motor and LDR connection follow figure 2 and 3 respectively. You may be able to use MOSFET for driving Motors also.

Compile the program and burn on the kit.

Expected Output:

On proper cable connection and program upload you can see square pulse of configured duty cycle on pd4/pd5 pin. Now you change the duty cycle in program and again check the duty cycle changes. As of that you can see the change on the pin (connect them to port-B).

Your task:

1) In this lab manual you have been given a program that generates the square pulse using the timer1. Now as told previously, Atmega32 provides various modes to generate different waveforms. Using various setting of WGM1-0:3, try to explore different waveforms [minimum 4, CTC-page 96, Fast PWM-page 97, Phase Correct PWM-page 99, Phase and Frequency Correct PWM-page 101]. For each waveform you need to calculate the PWM frequency. You can get the required register setting for the above mode from page 107, Table-47.

2) In this lab your task is to control the speed of the motor based on the light intensity given by LDR. Connect Motor and LDR to microcontroller as per circuit given in figure 2 and 3. Write code which reads the LDR, light intensity and based on that light intensity it rotates the motor very fast, fast, slow or very slow.

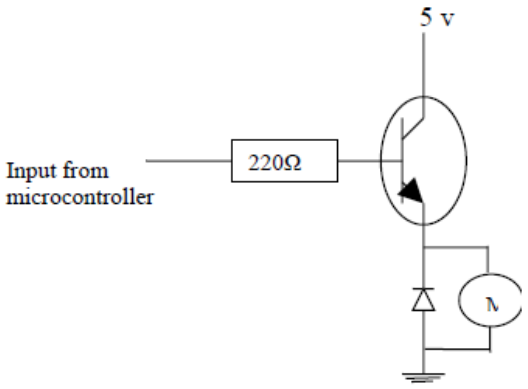


Figure 2: Connecting motor to microcontroller

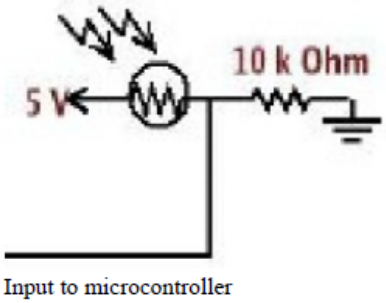


Figure 3: Connecting LDR to microcontroller