**CG1112 Engineering Principles and Practices**

**Week 6 Studio 2 – ADC**

**Answer Book**

**Student Number: A0214561M**

**Name: Zhuang Jianning**

**Pre-start Check (Check Fail: -5 mark penalty)**

**Did you name your file AxxxxxxY.docx, where AxxxxxxY is your student number? -2 marks if NO.**

**Did you fill in your student number and name above? -3 marks if NO.**

Question 1

Resistance of Sensor when placed straight = 41.50kΩ

Question 2

Resistance of Sensor when bent = 120.15kΩ

Question 3

ADC reading when Sensor is placed straight = 805

Voltage measured at A0 = 3.975V

Question 4

ADC reading when Sensor is bent = 930

Voltage measured at A0 = 4.617V

Question 5

The adc readings are mapped in the range 0 – 5 V

10 bits allows us to divide the signal range into 1024 different quantization levels where 0 represents 0V and 1023 represents 5V

For Q3, the adc reading of 805 corresponds to about 805/1023 \* 5 = 3.934V which agrees with our measured voltage

For Q4, the adc reading of 930 corresponds to about 930/1023 \* 5 = 4.545V which agrees with our measured voltage

Question 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Flex Sensor Flat |  | Flex Sensor Bent |  |  |  |
|  | ADC | Voltage/V | ADC | Voltage/V | ADC Range | Voltage Range/V |
| 1.2K | 995 | 4.922 | 1013 | 4.956 | 18 | 0.034 |
| 12K | 807 | 3.980 | 930 | 4.604 | 123 | 0.624 |
| 22K | 690 | 3.428 | 865 | 4.185 | 175 | 0.757 |
| 39K | 560 | 2.778 | 772 | 3.847 | 212 | 1.069 |

Question 7

Adcvalue =

The 1.2kΩ resistance is much smaller than the resistance of the flex when it is not bent which is about 42kΩ. When the resistor value in the denominator is much smaller than the flex resistance, an increase in the flex resistance will not cause adcvalue to increase by much as the ratio of the numerator/denominator will not increase by much. For example, for the 1.2kΩ resistor, when the resistance of the flex increases to about 120kΩ, adcvalue only increases to 1013 from an unbent value of 995. But for the 12KΩ resistor, when flex resistance increases from 42 to 120kΩ, adcvalue increases from 795 to 930. Hence, the adc range and voltage range will be much smaller with the 1.2kΩ resistance.

Question 8

The brightness of the LED is determined by the duty cycle.

\_delay\_loop\_2 is a delay loop using a 16-bit counter \_\_count, so up to 65536 iterations are possible. The loop executes four CPU cycles per iteration, not including the overhead the compiler requires to setup the counter register pair.

Thus, at a CPU speed of 16 MHz and a max adc value of 1023, delays of up to about 255 microseconds can be achieved.

The LED is toggled on and off based on this delay and the delay from conversion and polling, but since the intervals are so short, the duty cycle remains at 50% even when the flex sensor is bent as the adc values stabalise very quickly. Hence the delay when the LED is ON and the delay which the LED is OFF is almost always the same, giving us a constant duty cycle of 50%. Thus we perceive the brightness to remain consistent regardless of the flex sensor being bent.

Question 9

Lowest frequency: 1.51kHz

Highest frequency: 1.76kHz

Question 10

ISR(ADC\_vect)

{

// Provide your code for the ISR

loval = ADCL;

hival = ADCH;

adcvalue = (hival << 8) | loval;

ADCSRA |= (1 << ADSC);

}

Question 11

Lowest frequency: 2.32kHz

Highest frequency: 2.77 kHz

Question 12

For the polling method, we have to poll bit 6 of ADCSRA until it becomes 0 before we can read in the adc value from ADCL and ADCH. Hence the total period includes the conversion time, polling time and the \_delay\_loop\_2. However, if we use interrupts, the adc value is read when the ADSC bit is 1, hence there is no waiting time for polling, only conversion time and \_delay\_loop\_2. Thus the overall period is smaller than polling hence frequency is higher.

Question 13

#include "Arduino.h"

#include <avr/interrupt.h>

unsigned int adcvalue, loval, hival;

void InitPWM()

{

TCNT0 = 0;

OCR0A = 0;

TCCR0A = 0b00000001;

TIMSK0 |= 0b10;

}

void startPWM()

{

TCCR0B = 0b00000100;

}

ISR(TIMER0\_COMPA\_vect)

{

// Provide your code for the ISR

OCR0A = (adcvalue-690)\*255/175;

PORTB ^= (1 << PORTB5);

}

ISR(ADC\_vect)

{

// Provide your code for the ISR

loval = ADCL;

hival = ADCH;

adcvalue = (hival << 8) | loval;

ADCSRA |= (1 << ADSC);

}

void setup() {

// put your setup code here, to run once:

PRR &= 0b11111110;

ADCSRA = 0b10001111;

ADMUX = 0b01000000;

DDRB |= 0b00100000;

InitPWM();

startPWM();

sei();

ADCSRA |= 0b01000000;

}

void loop() {}

**For TA Use:**