Emmanuel Rodriguez Lopez

CPE301 – SPRING 2016

Design Assignment 5

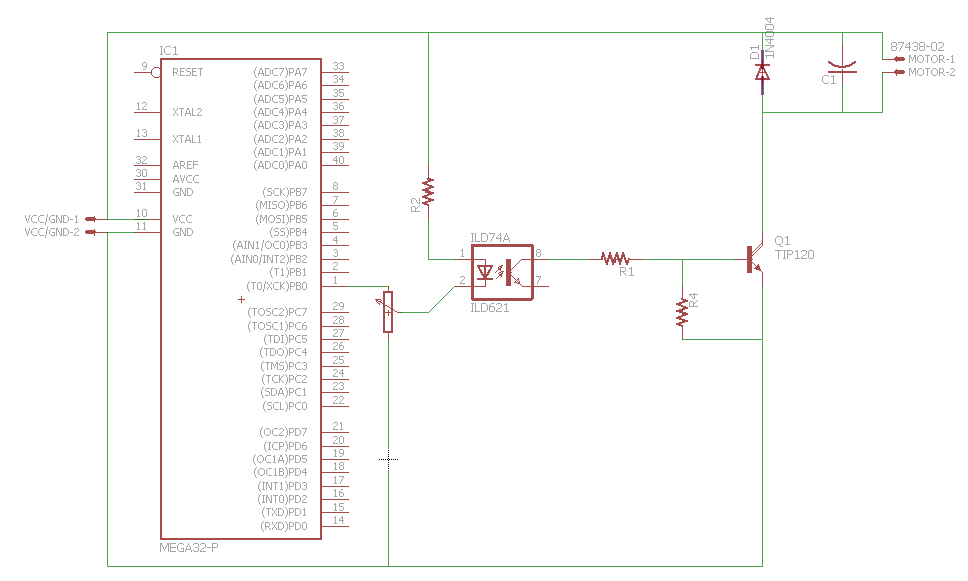
**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

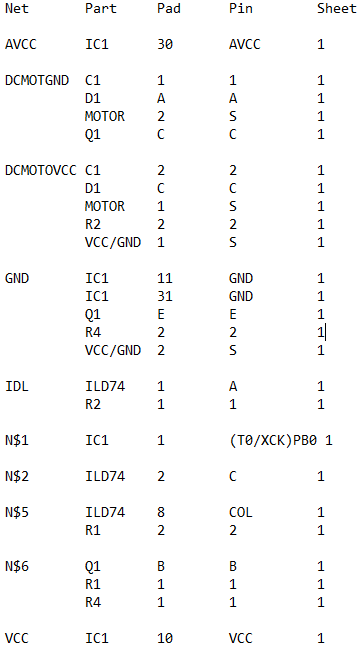
The student understands that all required components should be submitted in complete for grading of this assignment.

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SUBMISSION ITEM** | **COMPLETED (Y/N)** | **MARKS**  **(/MAX)** |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |
| 1. | INITIAL CODE OF TASK 1/A |  |  |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |
| 5. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E |  |  |
| 6. | SCHEMATICS |  |  |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 8. | SCREENSHOT OF EACH DEMO |  |  |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

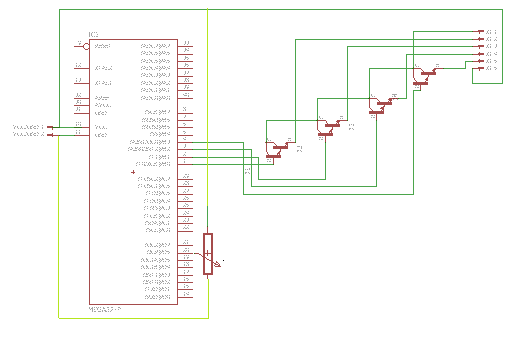
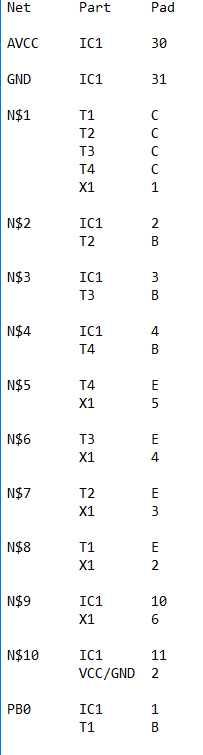
|  |  |  |  |
| --- | --- | --- | --- |
| 0. | COMPONENTS LIST AND CONNECTION DIAGRAM w/ PINS |  |  |

TASK 1 DC MOTOR

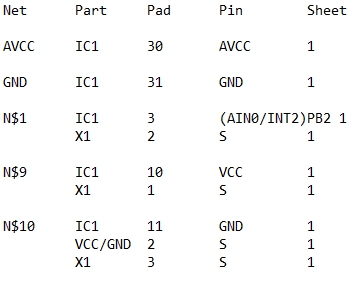
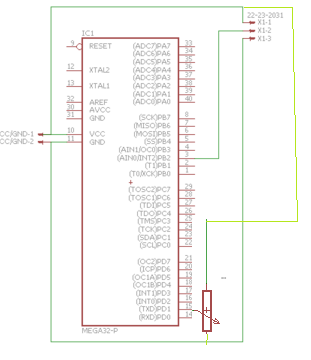




TASK 2 STEPPER MOTOR

TASK 3 SERVO



|  |  |  |  |
| --- | --- | --- | --- |
| 1 | INITIAL CODE OF TASK 1 |  |  |

#define *F\_CPU* 8000000UL

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRC = 0x20;

int ADC1 = 0;

ADC\_init(); //initializing ADC function

while(1)

{

ADC1 = adc\_read(); //get value from adc

//From here, at each of the specified rages, if the potentiometer is turned to the area //producing these values, the motor will turn at the specified speed.

if ((ADC1 >= 901) && (ADC1 < 1024))

{

PORTC = 0x20; //pc5

*\_delay\_ms*(10);

PORTC = 0x00;

*\_delay\_ms*(500);

}

if ((ADC1 >= 700) && (ADC1 < 901))

{

PORTC = 0x20; //pc5

*\_delay\_ms*(75);

PORTC = 0x00;

*\_delay\_ms*(500);

}

if ((ADC1 < 700)&&(ADC1>=500))

{

PORTC = 0x20; //pc5

*\_delay\_ms*(150);

PORTC = 0x00;

*\_delay\_ms*(500);

}

if ((ADC1 < 500)&&(ADC1>=300))

{

PORTC = 0x20; //pc5

*\_delay\_ms*(300);

PORTC = 0x00;

*\_delay\_ms*(500);

}

if ((ADC1 < 300)&&(ADC1>=150))

{

PORTC = 0x20; //pc5

*\_delay\_ms*(500);

PORTC = 0x00;

*\_delay\_ms*(600);

}

if ((ADC1 < 150)&&(ADC1>=0))

{

PORTC = 0x20; //pc5

*\_delay\_ms*(600);

PORTC = 0x00;

*\_delay\_ms*(700);

}

}

return 0;

}

int adc\_read()

{

ADMUX &= 0x00;

ADCSRA |= (1<<ADSC);

while((ADCSRA & (1<<ADSC)));

return ADC;

}

void ADC\_init(void)

{

ADCSRA = (1<<ADEN);//|(1<<ADSC);//|(1<<ADIF);

}

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | INITIAL CODE OF TASK 2 |  |  |

#define *F\_CPU* 8000000UL

#include <avr/io.h>

#include <util/delay.h>

int main()

{

DDRD = 0x0F;

int ADC1 = 0;

ADC\_init(); //initializing ADC function

while(1)

{

ADC1 = adc\_read(); //get value from adc

//From here, at each of the specified rages, if the potentiometer is turned to the area //producing these values, the motor will turn at the specified speed.

if ((ADC1 >= 901) && (ADC1 < 1024))

{

PORTD = 0x6; //fastest

*\_delay\_ms*(10);

PORTD = 0xC;

*\_delay\_ms*(10);

PORTD = 0x9;

*\_delay\_ms*(10);

PORTD = 0x3;

*\_delay\_ms*(10);

}

if ((ADC1 >= 700) && (ADC1 < 901))

{

PORTD = 0x6;

*\_delay\_ms*(15);

PORTD = 0xC;

*\_delay\_ms*(15);

PORTD = 0x9;

*\_delay\_ms*(15);

PORTD = 0x3;

*\_delay\_ms*(15);

}

if ((ADC1 < 700)&&(ADC1>=500))

{

PORTD = 0x6;

*\_delay\_ms*(20);

PORTD = 0xC;

*\_delay\_ms*(20);

PORTD = 0x9;

*\_delay\_ms*(20);

PORTD = 0x3;

*\_delay\_ms*(20);

}

if ((ADC1 < 500)&&(ADC1>=300))

{

PORTD = 0x6;

*\_delay\_ms*(25);

PORTD = 0xC;

*\_delay\_ms*(25);

PORTD = 0x9;

*\_delay\_ms*(25);

PORTD = 0x3;

*\_delay\_ms*(25);

}

if ((ADC1 < 300)&&(ADC1>=150))

{

PORTD = 0x6; //slowest

*\_delay\_ms*(30);

PORTD = 0xC;

*\_delay\_ms*(30);

PORTD = 0x9;

*\_delay\_ms*(30);

PORTD = 0x3;

*\_delay\_ms*(30);

}

if ((ADC1 < 150)&&(ADC1>=0))

{

PORTD = 0x6;

*\_delay\_ms*(40);

PORTD = 0xC;

*\_delay\_ms*(40);

PORTD = 0x9;

*\_delay\_ms*(40);

PORTD = 0x3;

*\_delay\_ms*(40);

}

}

return 0;

}

int adc\_read()

{

ADMUX &= 0x00;

ADCSRA |= (1<<ADSC);

while((ADCSRA & (1<<ADSC)));

return ADC;

}

void ADC\_init(void)

{

ADCSRA = (1<<ADEN);//|(1<<ADSC);//|(1<<ADIF);

}

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | INITIAL CODE OF TASK 3 |  |  |

#define *F\_CPU* 8000000UL

#include <avr/io.h>

#include <util/delay.h>

int main()

{

DDRB = 0xFF;

ICR1 = 20000;

//Setting up PWM

TCCR1A |= (1<<COM1A1)|(1<<WGM11);

TCCR1B |= (1<<WGM12)|(1<<WGM13)|(1<<CS11);

int ADC1 = 0;

ADC\_init(); //initializing ADC function

while(1)

{

ADC1 = adc\_read();

// the actual value of ADC was used as the position; it is doubled and added to the //lower, 0 degree, value of the motor.

*\_delay\_ms*(20);

OCR1A = 600 + (2\*ADC1);

}

return 0;

}

int adc\_read()

{

ADMUX &= 0x00;

ADCSRA |= (1<<ADSC);

while((ADCSRA & (1<<ADSC)));

return ADC;

}

void ADC\_init(void)

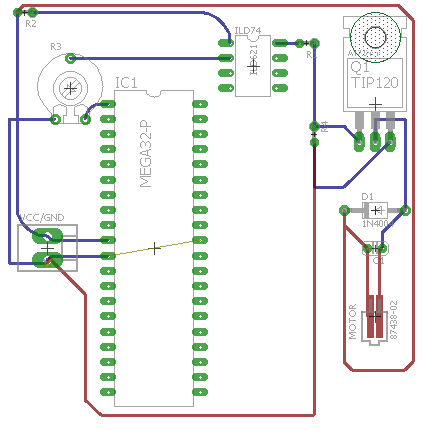
{

ADCSRA = (1<<ADEN);

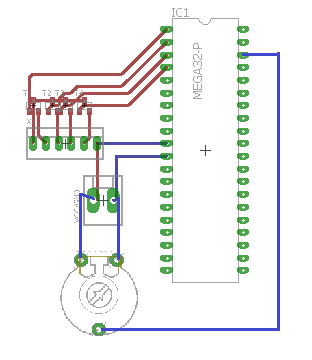
}

|  |  |  |  |
| --- | --- | --- | --- |
| 6. | SCHEMATICS |  |  |

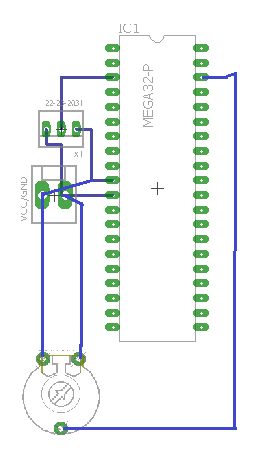
TASK 1



TASK 2



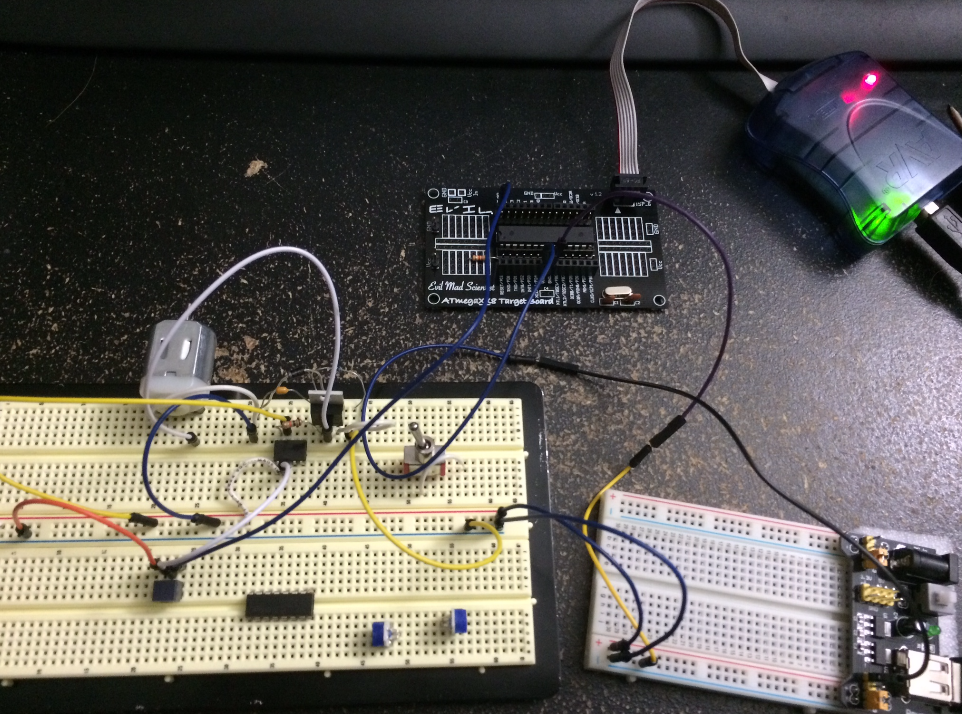
TASK 3

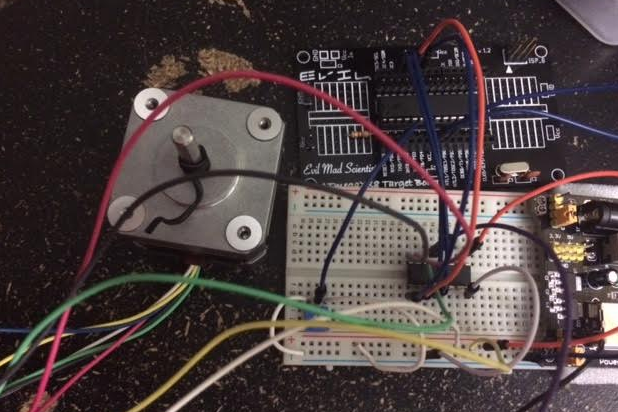


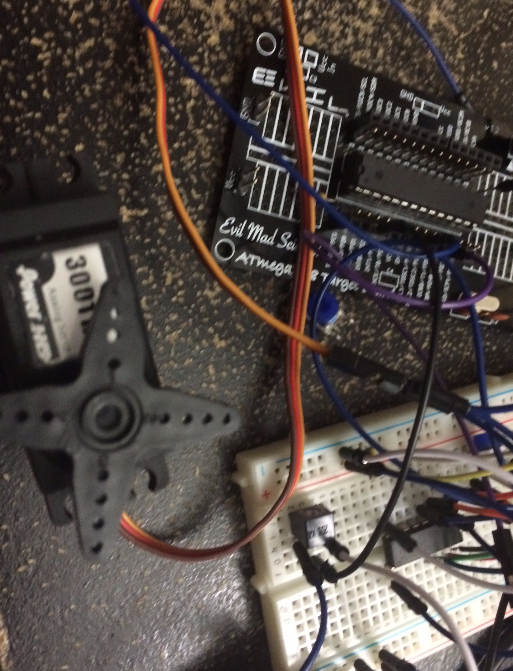
|  |  |  |  |
| --- | --- | --- | --- |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |

Simulations do not anything useful.

|  |  |  |  |
| --- | --- | --- | --- |
| 8. | SCREENSHOT |  |  |

TASK 1

TASK2

TASK3

|  |  |  |  |
| --- | --- | --- | --- |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| 1: <https://www.youtube.com/watch?v=3u6KKWqgKnc>  2: <https://www.youtube.com/watch?v=KRxYHFTMmOI> 3: <https://www.youtube.com/watch?v=YkqNfeDtbZg> | | | |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
| http:// @svn or github repository link | | | |

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Emmanuel Rodriguez Lopez