**COMP3402 – Motor Control Project**

Isa Berk Geriler – 20SOFT1034  
Yağız Efe Atasever – 19COMP1035

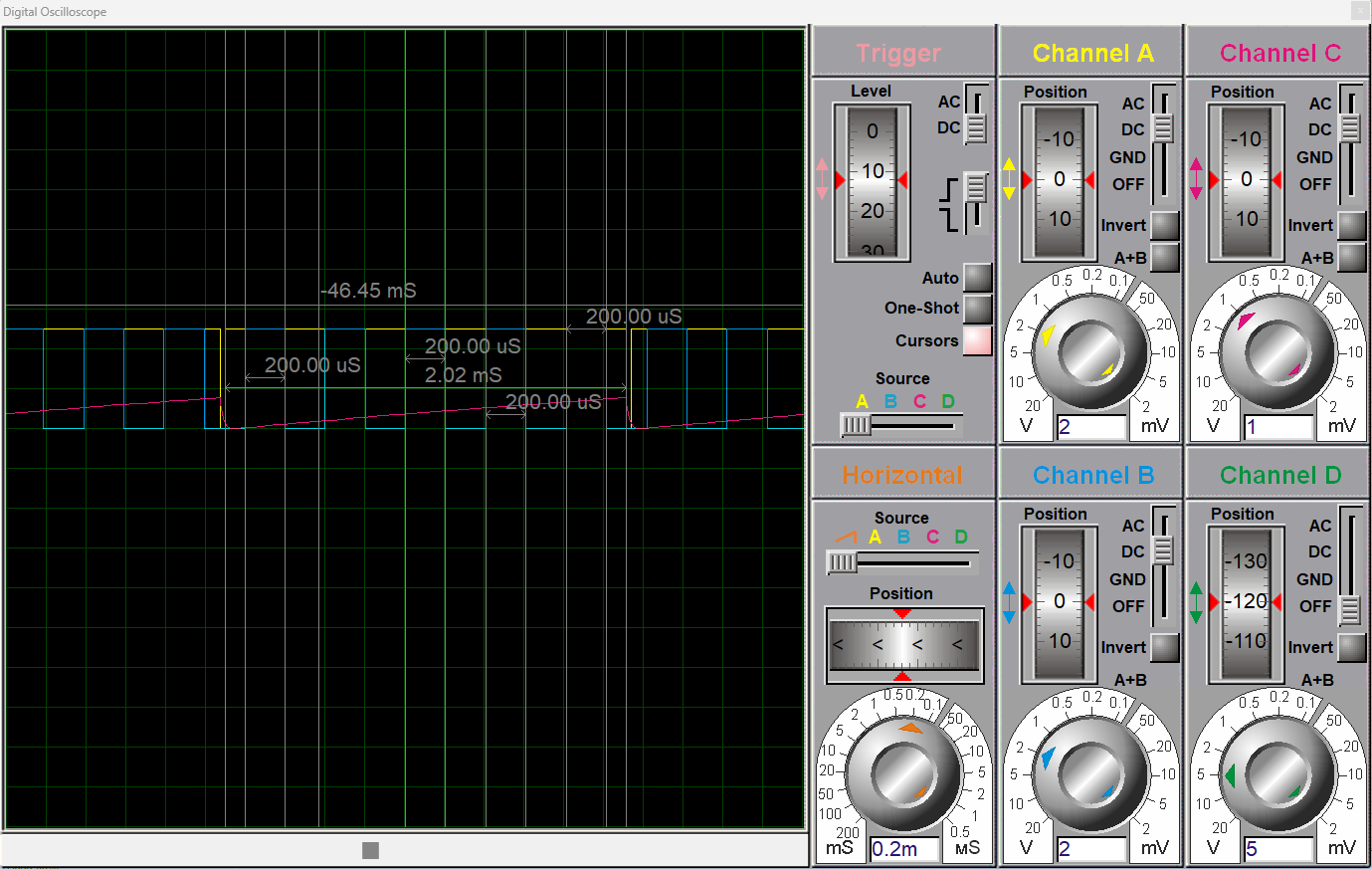
**1) Proteus Schmatic (Motor Control)** 

**2)** **Arduino Code (Motor Control)**

|  |
| --- |
| #include <LiquidCrystal.h>  const int RS = 12, EN = 11, D4 = 10, D5 = 6, D6 = 5, D7 = 4;  LiquidCrystal lcd(RS, EN, D4, D5, D6, D7);  #define adc\_pin A1  #define led 13  #define IN1 8  #define IN2 7  #define pwm\_pin 9  #define T2\_initialVal 156  const int SIZE = 10;  int adcArray[SIZE];  int elementsRead = 0;  volatile bool allElementsRead = false;  // interrupt service routine that wraps a user defined function  ISR(TIMER2\_OVF\_vect) {  TCNT2 = T2\_initialVal; // preload timer  digitalWrite(led, digitalRead(led) ^ 1); // toggle the led (^ 1 means XOR function)  if (elementsRead < SIZE) {  adcArray[elementsRead] = analogRead(adc\_pin);  elementsRead++;  } else {  allElementsRead = true;  }  }  float calculateAverageAdc(int elementsRead) {  float sum = 0;  float divider = (float)elementsRead;  float averageAdc = 0;  for (int i = 0; i < elementsRead; i++) {  sum += (float)adcArray[i];  }  averageAdc = (sum/divider);  return averageAdc;  }  void setup() {  // put your setup code here, to run once:  Serial.begin(57600);  lcd.begin(20, 4); // since we used LM044L on proteus    pinMode(led,OUTPUT);  pinMode(IN1,OUTPUT);  pinMode(IN2,OUTPUT);    digitalWrite(IN1,HIGH);  digitalWrite(IN2,LOW);  // setup Timer2  noInterrupts(); // disable all interrupts  TCCR2A = 0;  TCCR2B = 0;  TCNT2 = T2\_initialVal; // preload timer 65536-16MHz/8/1kHz  TCCR2B |= (1 << CS21)|(1 << CS20); // 32 prescaler  TIMSK2 |= (1 << TOIE2); // enable timer overflow interrupt  interrupts(); // enable all interrupts  }  void loop() {  // put your main code here, to run repeatedly:  if (allElementsRead) {  noInterrupts(); // Disable interrupts while calculating average ADC  int currentElementsRead = elementsRead; // Save the value of elementsRead  float adcAverage = calculateAverageAdc(currentElementsRead); // Use the saved value on averaging ADC  interrupts(); // Enable interrupts after calculating average ADC    lcd.clear();  lcd.setCursor(0, 0);  lcd.print("AVG MOTOR CURRENT");  lcd.setCursor(0, 1);  lcd.print(adcAverage);  lcd.print("mA");  for (int i = 0; i < currentElementsRead; i++) {  Serial.print(adcArray[i]);  if (i < currentElementsRead - 1) {  Serial.print(", ");  }  }  Serial.println();  elementsRead = 0; // Reset read element count, for ISR  allElementsRead = false; // reset the state, for ISR and loop()  TCNT2 = 0; // preload timer 0  TCCR2B |= (1 << CS21); // 8 prescaler  }  analogWrite(pwm\_pin, abs(analogRead(A0)/4-1));  delay(500);  } |

**3) Explanation**

This Arduino project reads the analog values from an ADC pin and stores them of an array sized of 10. Then, the average of these values is calculated (average current), and then displayed on an LCD screen. PWM is also controlled based on the input. It also periodically prints the ADC readings to the serial monitor.



*Figure 3.a – Waveform of the motor current. The duration between two spikes is approximately 2ms. ADC readings from A1 channel are also made at each 200us.*

A screenshot of a computer

Description automatically generated

*Figure 3.b – The values read from the ADC pin are stored in an array, and are printed out, periodically.*

A screen shot of a graph

Description automatically generated with medium confidence

*Figure 3.c – Drawn Serial Plot of the project, by running “Serial Plotter” on Arduino Uno, from the COM2 connector (by usage of VSPE).*