UNIVERSIDADE FEDERAL DE SANTA MARIA

DPEE 1086 – ELETRÔNICA APLICADA E INSTRUMENTAÇÃO

Lorenzo Moreira Donatti

201820568

RELATÓRIO PROJETO INTEGRADOR

SANTA MARIA, 30 DE AGOSTO DE 2021

INTRODUÇÃO

O projeto foi desenvolvido com a proposta de fazer o controle de um motor DC em função da temperatura. A principal característica do projeto foi o uso de um microcontrolador. Foi-se utilizado a placa *Arduino UNO* e seu software de programação, juntamente com o software de simulação *PROTEUS.* O sistema é responsável pela obtenção de dados analógicos de temperatura e na conversão para uma saída PWM na casa de 20Khz, onde foi necessário um ajuste nos registradores do Arduino para obtenção de altas frequências.

ESPECIFICAÇÕES

Todos os resultados provindos do projeto vieram por meio de simulações. Infelizmente não foi possível testar com o hardware físico. Porém o software *PROTEUS* é uma ferramenta poderosa que cumpre bem com o objetivo.

Para o projeto, foi utilizado um sensor de temperatura *LM35*, a placa *ARDUINO*, um transistor *MOSFET IRF640*, um diodo *1N4007* e um motor DC com uma fonte de tensão de 12V. Além disso, foi utilizado um osciloscópio para a medição do Duty Cycle do PWM e um LCD para exibir as informações.

CÓDIGO FONTE

O Seguinte código fonte foi implementado. Na função *setup* foram modificados os registradores para a obtenção do sinal PWM de 20Khz. Além da função *Ajustetemp* que é responsável pela filtragem e obtenção do sinal analógico de temperatura.

O código possui apenas dois pinos principais utilizados no Arduino, o *A0* como input e o *10* como output, os demais foram conectados ao LCD via biblioteca *LiquidCrystal.h.*

```
#include <LiquidCrystal.h>
//declaração de pinos
int PinTemp = A0;
int PinPWM = 10;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);    // Display LCD
//declaração de variaveis
double Temperatura=0;
double PWMatual;
double analog[10];
int i;

void setup() {
    // Configuração Timer Arduino para 25KHz do PWM
  TCCR1A = 0;               // undo the configuration done by...
  TCCR1B = 0;               // ...the Arduino core library
  TCNT1  = 0;               // reset timer
  TCCR1A = _BV(COM1A1)      // non-inverted PWM on ch. A
           | _BV(COM1B1)    // same on ch; B
           | _BV(WGM11);    // mode 10: ph. correct PWM, TOP = ICR1
  TCCR1B = _BV(WGM13)       // ditto
           | _BV(CS10);     // prescaler = 1
  ICR1   = 320;             // TOP = 320
```

```
  Serial.begin(9600);
  //configura pino
  pinMode(PinPWM, OUTPUT);
  pinMode(PinTemp, INPUT);
  lcd.begin(8, 2);
}
void loop() {
  Ajustetemp();
   writeLCD();
  //PWMatual = map(Temperatura,0,100,0,255);
    PWMatual = map(Temperatura,0,100,0,320);
  if(Temperatura<=100&&Temperatura>=0){
    //analogWrite(PinPWM,PWMatual);              //pwm 490hz
    analogWrite25k(PWMatual);                   //pwm 25khz
    Serial.println("PWM");
    Serial.println(PWMatual);
  }
  else{
    PWMatual = 0;
    Serial.println("Motor desligado, temperatura fora da faixa 0-100C");
    //analogWrite(PinPWM,PWMatual);              //pwm 490hz
    analogWrite25k(PWMatual);                   //pwm 25khz
  }
}
void Ajustetemp(){
  Temperatura = 0;
  for(i = 0;i<=9;i++){
    analog[i] = analogRead(PinTemp);                              //pequeno filtro digital
    Temperatura = Temperatura + analog[i];
    delay(10);
  }
  Temperatura = Temperatura/10;                                    //faz a media de 10 amostras
  Temperatura = map(Temperatura,0,205,0,100);                      //conversão dos valores
analogicos para 0-100
  Serial.println("temperatura");                                  //LM 35 para cada 1C acresce
10mv
  Serial.println(Temperatura);
  //Serial.println(analogRead(PinTemp));
  delay(100);
}
// Função para fazer escrita do PWM no pino de saída através dos registradores
void analogWrite25k(int value)
{
  OCR1B = value;
}
// Função para escrever no display dados de PWM
void writeLCD() {
  lcd.setCursor(0, 0);
  lcd.print("%PWM:");
  lcd.print((PWMatual / 320) * 100, 0);
  lcd.setCursor(0,1);
  lcd.print("ou PWM:");
  lcd.print(PWMatual);
  lcd.print(" ");
}
```

SIMULAÇÃO

Com a simulação foi possível obter resultados satisfatórios sobre o projeto. De acordo com a temperatura o Duty Cycle do sinal PWM é alterado, onde este varia de 0-320 enquanto a temperatura varia de 0-100C. A função do MOSFET é funcionar como uma chave, otimizando o circuito. E o diodo tem como função evitar a força contra eletromotriz advinda do motor. É também possível notar que o sinal PWM se mantém na faixa de um pouco mais de 3 vezes maior que o sinal analógico da temperatura.

Foi necessário a utilização a biblioteca externa Arduino.lib no software PROTEUS, pois o mesmo não possui compatibilidade com o Arduino UNO por padrão. Biblioteca encontrada no site [Arduino Library for Proteus V2.0 - The Engineering Projects]

Para a simulação, o seguinte circuito foi montado:

Figura 1 - Esquemático do circuito.

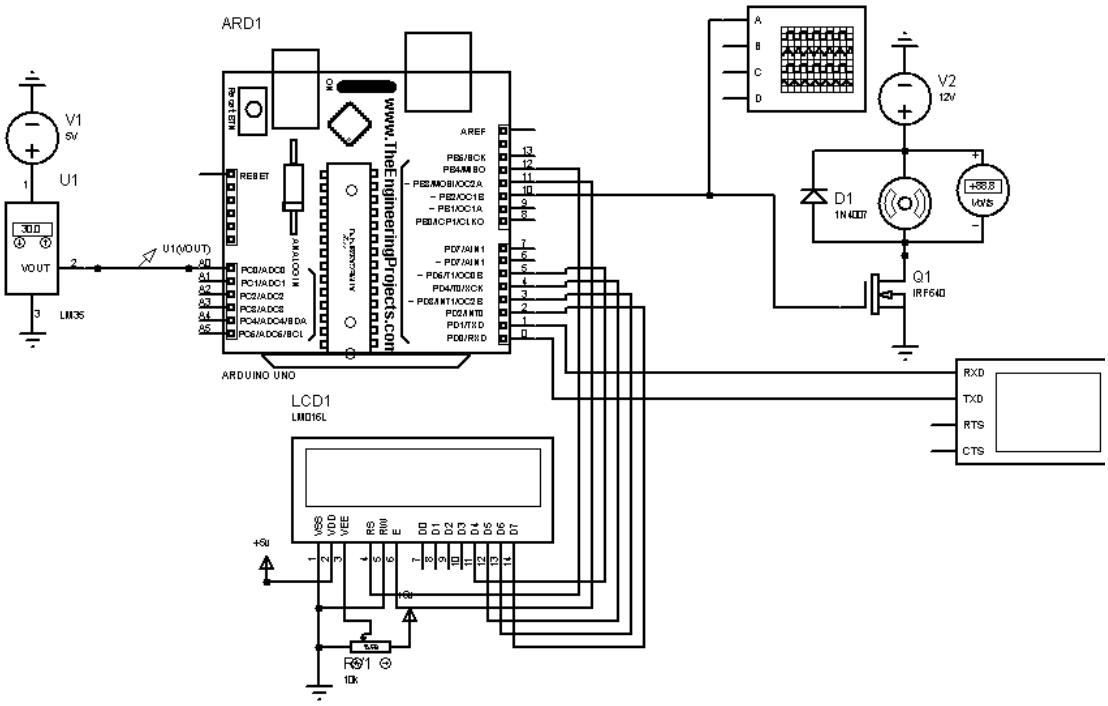


Figura 2 – Circuito simulando a temperatura de 30C.

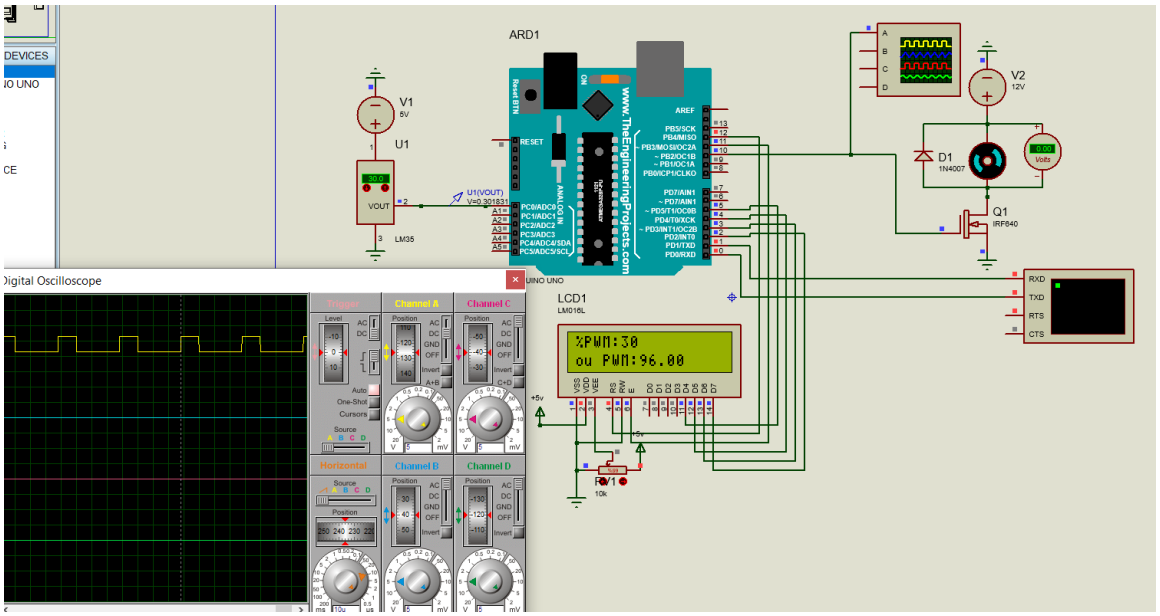


Figura 3 – Circuito simulando a temperatura de 85C.

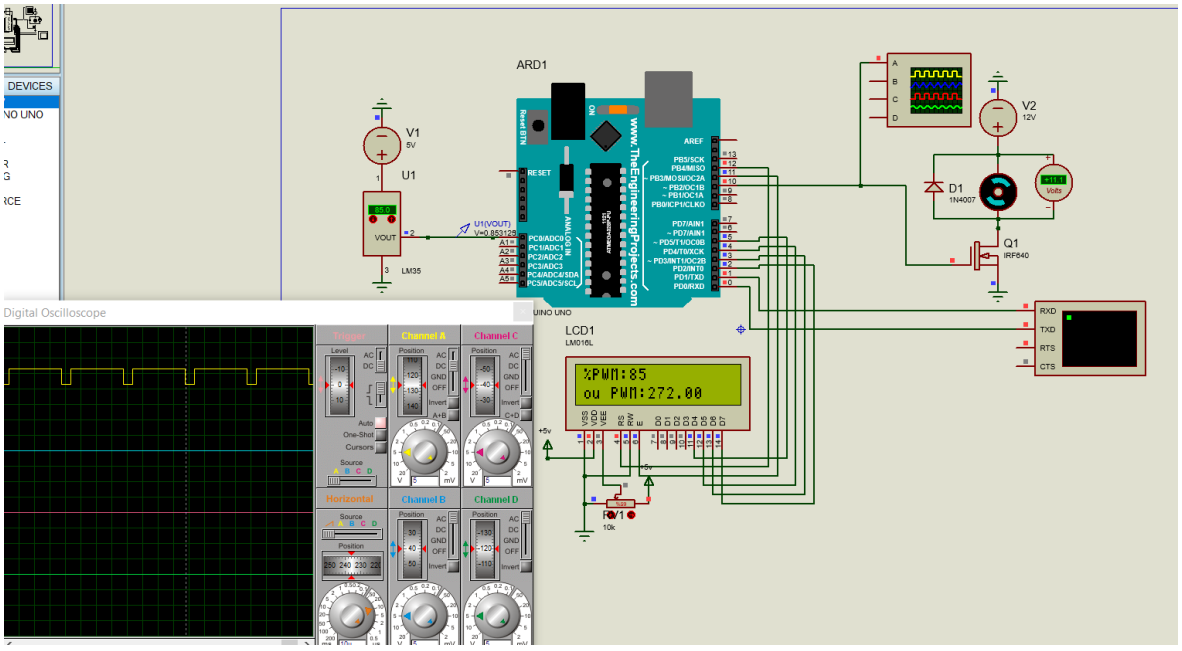
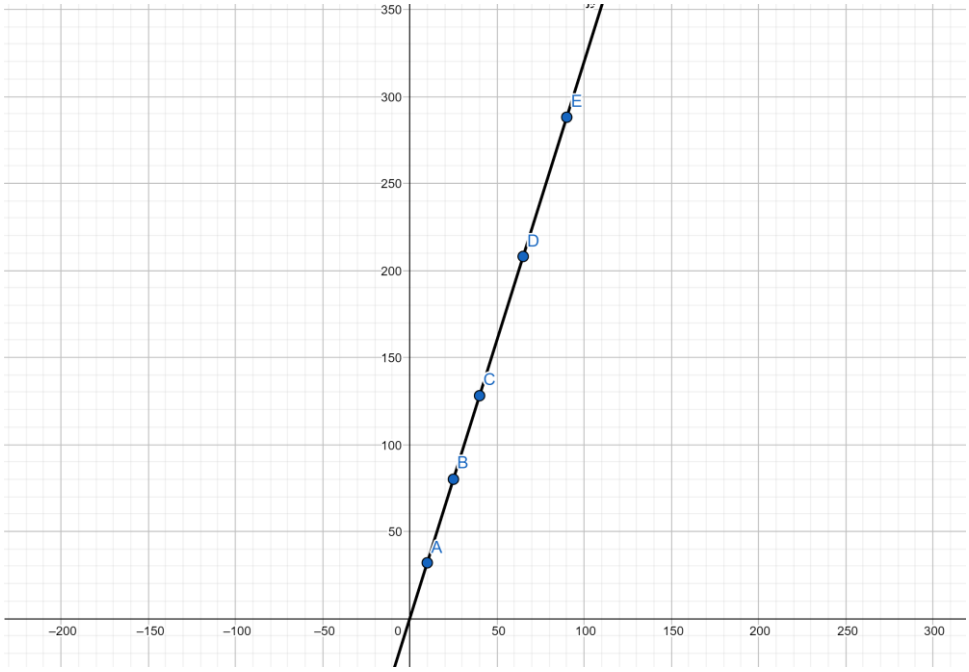


GRÁFICO ENTRADA X SAÍDA

Colocando os dados da entrada de temperatura e os dados de saída PWM no software GeoGebra foi possível traçar um gráfico que relaciona ambos. Sendo o eixo x caracterizado pela temperatura e o eixo y a saída PWM.

Figura 4 - Gráfico Temperatura X PWM.



CONCLUSÃO

É possível concluir que o projeto cumpriu com o papel de simular um controle de velocidade de um motor DC pela função de temperatura, onde que, mesmo em um sistema simulado, os resultados foram satisfatórios. Foi possível visualizar com clareza o impacto da temperatura na variação do duty Cycle do PWM, sendo comprovado pelo gráfico plotado acima. No mundo real alguns fatores como ruídos interfeririam no sistema, porém o filtro digital implantado na parte programada teria um papel relevante neste quesito.