**PROJETO**

**E209 – L1**

**MATHEUS, JÔNATAS, GUILHERME E FILIPE**

**Contexto:**

**TEMA 02** - Projete um moedor que pode triturar diferentes materiais. A velocidade do motor é escolhida de maneira **inversamente proporcional** ao peso do material. Para fazer a leitura foi instalado uma célula de carga que trabalha na faixa de 0kg à 10kg. A sua saída é proporcional ao peso, que varia de 0V à 5V (pode ser representado por um potenciômetro). A leitura deve ser feita utilizando o **conversor AD**. Para escolher a velocidade do motor (pode ser representado por um LED) é necessário escolher a tensão média a ser aplicada em seus terminais (**Timer PWM**). A velocidade varia de 0% à 100% e deve ser inversamente proporcional à carga. Para uma carga de 10kg, é necessário utilizar a velocidade a 0%, para 9kg, a velocidade a ser utilizada é 10%, para 8kg, 20%, assim por diante. A velocidade é atualizada sempre que o sistema for ligado.

Para ligar a máquina, é necessário que o operador envie, utilizando a porta serial (**UART**), um comando **'L'**. Quando receber esta mensagem, o MCU deve escrever a mensagem **"SIST. LIGADO"**, utilizando também a comunicação serial. A máquina deve ser desligada assim que um sensor de presença (**GPIO**, pode ser representado por um interruptor deslizante) não estiver detectando o material na máquina. Quando isso ocorrer, utilizando a comunicação serial, o MCU deve escrever a mensagem **"SIST. DESLIGADO"**.

Deverá ser instalado também um display **LCD 16x2** que indique qual o peso do material inserido na máquina. Essa informação deve ser atualizada sempre que o sistema for ligado.

Por último, deve ser implementado também um botão de emergência, utilizando **interrupção externa**. Sempre que esse botão for pressionado o motor deve ser parado imediatamente. Quando isso ocorrer, escreva a mensagem **"SIST. PARADO!"**, utilizando a porta serial (**UART**).

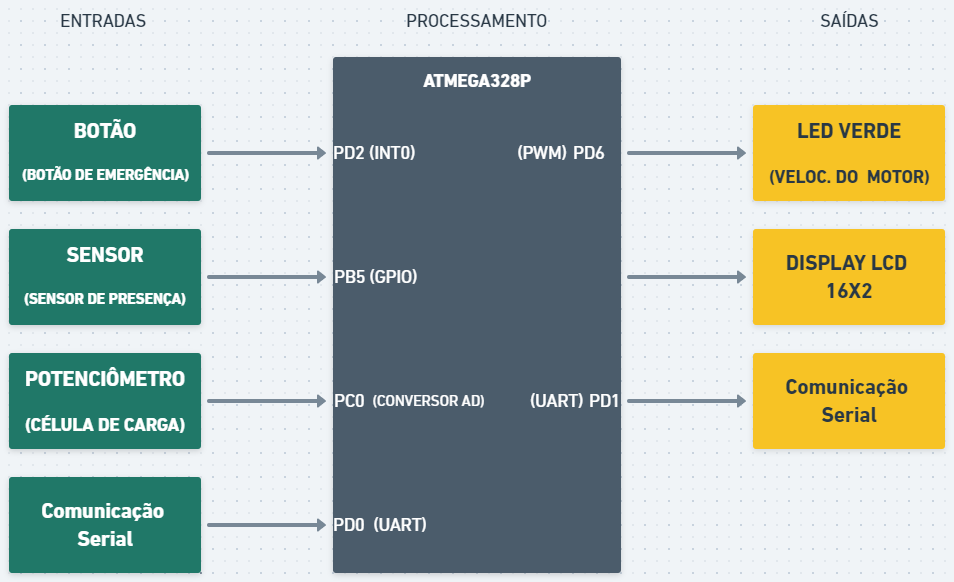
**ENTREGA:**

**1º Parte:** É necessário que o grupo (no máximo 04 pessoas) entregue o **diagrama em blocos**, a **máquina de estados** e o **código do projeto; ->** DATA: Até 27/06 no TEAMS

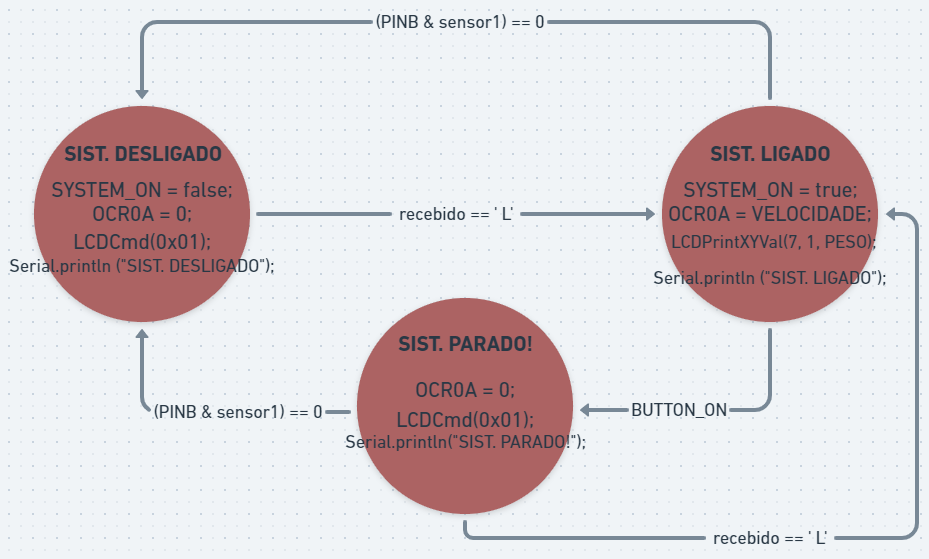
**2º Parte:** Entregar um vídeo de, em média, 4 minutos onde o grupo deve explicar como foi feito o projeto (recursos, periféricos, configurações de registros, etc); **->** DATA: Até 27/06 no TEAMS

**3º Parte:** Na semana do dia 28/06 **no horário de cada turma**, será a apresentação dos projetos, onde cada vídeo será mostrado e o monitor fará perguntas, se necessário!

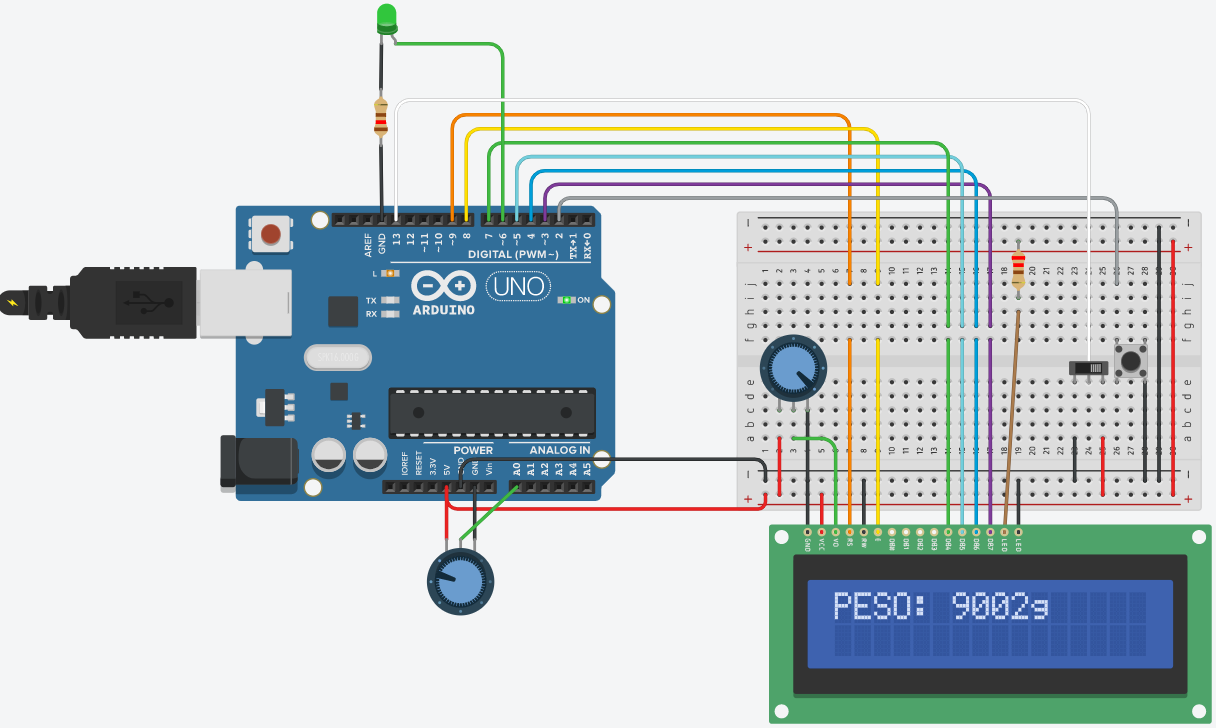
DIAGRAMA EM BLOCOS



MÁQUINA DE ESTADOS



CÓDIGO DO PROJETO



|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #define FOSC 16000000U //Clock Speed  #define BAUD 9600  #define MYUBRR FOSC / 16 / BAUD - 1  #define sensor1 (1 << PB5)  //Data Pins  #define LCD\_D4 (1 << PD7)  #define LCD\_D5 (1 << PD5)  #define LCD\_D6 (1 << PD4)  #define LCD\_D7 (1 << PD3)  //Control Pins  #define LCD\_RS (1 << PB1)  #define LCD\_EN (1 << PB0)  #define LCD\_EN\_ON PORTB |= LCD\_EN  #define LCD\_EN\_OFF PORTB &= ~LCD\_EN  #define LCD\_RS\_ON PORTB |= LCD\_RS  #define LCD\_RS\_OFF PORTB &= ~LCD\_RS  #define LCD\_D4\_ON PORTD |= LCD\_D4  #define LCD\_D4\_OFF PORTD &= ~LCD\_D4  #define LCD\_D5\_ON PORTD |= LCD\_D5  #define LCD\_D5\_OFF PORTD &= ~LCD\_D5  #define LCD\_D6\_ON PORTD |= LCD\_D6  #define LCD\_D6\_OFF PORTD &= ~LCD\_D6  #define LCD\_D7\_ON PORTD |= LCD\_D7  #define LCD\_D7\_OFF PORTD &= ~LCD\_D7  #define LCD\_DATA\_DIR DDRD |= (LCD\_D4 + LCD\_D5 + LCD\_D6 + LCD\_D7)  #define LCD\_DATA\_OFF PORTD &= (~(LCD\_D4 + LCD\_D5 + LCD\_D6 + LCD\_D7))  //Valor de THRESHOLD  #define THRES 500  //FUNÇÕES - DISPLAY LCD  //Send a command to the display.  void LCDCmd(uint8\_t cmd)  {  LCD\_RS\_OFF;  sendnibble(cmd >> 4);  sendnibble(cmd & 0x0F);  }  //Routine to work with four bits  void sendnibble(uint8\_t data)  {  LCD\_DATA\_OFF;  if ((data & 0x01) == 0x01)  LCD\_D4\_ON;  if ((data & 0x02) == 0x02)  LCD\_D5\_ON;  if ((data & 0x04) == 0x04)  LCD\_D6\_ON;  if ((data & 0x08) == 0x08)  LCD\_D7\_ON;    LCD\_EN\_ON;  delayLCD();  LCD\_EN\_OFF;  }  //Initializes display  void InitLCD(void)  {  LCD\_DATA\_DIR;  LCD\_EN\_OFF;  LCD\_RS\_OFF;    sendnibble(0x30 >> 4);  delayLCD();  sendnibble(0x30 >> 4);  delayLCD();  sendnibble(0x30 >> 4);  delayLCD();  sendnibble(0x20 >> 4);  delayLCD();  LCDCmd(0x28);  delayLCD();  LCDCmd(0x08);  delayLCD();  LCDCmd(0x0C);  delayLCD();  LCDCmd(0x01);  delayLCD();  }  //Delay to set up the display  void delayLCD(void)  {  \_delay\_ms(0.2);  }  //Send a message to the display  void LCDPrintStr(const char \*data)  {  while (\*data != 0)  LCDChar(\*data++);  }  //Send a message to the display in the position X,Y  void LCDPrintXYStr(uint8\_t x, uint8\_t y, const char \*data)  {  uint8\_t pos;  pos = x - 1;  if (y == 1)  {  pos = pos + 0x80;  LCDCmd(pos);  }  else if (y == 2)  {  pos = pos + 0xc0;  LCDCmd(pos);  }  LCDPrintStr(data);  }  //SEND A DATA TO THE DISPLAY.  void LCDChar(uint8\_t data)  {  LCD\_RS\_ON;  sendnibble(data >> 4);  sendnibble(data & 0x0F);  }  void LCDPrintVal(unsigned int dado)  {  if (dado >= 10000)  LCDChar((dado / 10000) + 0x30);  if (dado >= 1000)  LCDChar(((dado % 10000) / 1000) + 0x30);  if (dado >= 100)  LCDChar((((dado % 10000) % 1000) / 100) + 0x30);  if (dado >= 10)  LCDChar(((((dado % 10000) % 1000) % 100) / 10) + 0x30);  LCDChar(((((dado % 10000) % 1000) % 100) % 10) + 0x30);  }  void LCDPrintXYVal(unsigned char x, unsigned char y, unsigned int dado)  {  unsigned char pos;  pos = x - 1;  if (y == 1)  {  pos = pos + 0x80;  LCDCmd(pos);  }  else  {  pos = pos + 0xc0;  LCDCmd(pos);  }  LCDPrintVal(dado);  }  //FUNÇÕES - CONVERSOR AD.  void ADC\_init(void)  {  //Configurando Vref para VCC = 5V  ADMUX = (1 << REFS0);  /\*  ADC ativado e preescaler de 128  16MHz / 128 = 125kHz  ADEN = ADC Enable, ATIVA O ADC  ADPSx = ADC Prescaler Select Bits  1 1 1 = CLOCK / 128  \*/  ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);  }  int ADC\_read(u8 ch)  {  char i;  int ADC\_temp = 0; //ADC temporário, p/ manipular LEITURA.  int ADC\_read = 0; //ADC\_read  ch &= 0x07;    //Zerar os 3 primeiros bits e manter o resto.  ADMUX = (ADMUX & 0xF8) | ch;  //ADSC (ADC Start Conversion)  ADCSRA |= (1 << ADSC);//FAÇA UMA CONVERSÃO.  //ADIF (ADC INTERRUPT FLAG) É SETADA QUANDO O ADC pede interrupção.  //E RESETADA QUANDO o vetor de interrupção é tratado.    while (!(ADCSRA & (1 << ADIF)))  ; //Aguarde a conversão do sinal.  for (i = 0; i < 8; i++) //Fazendo a conversão 8 vezes p/ maior precisão.  {  ADCSRA |= (1 << ADSC); //FAÇA UMA CONVERSÃO.  while (!(ADCSRA & (1 << ADIF)))  ; //Aguarde a conversão do sinal  ADC\_temp = ADCL; //lê o registro ADCL.  ADC\_temp += (ADCH << 8); //lê o registro ADCH.  ADC\_read += ADC\_temp; //ACUMULA O RESULTADO (8 amostras) PARA MÉDIA.  }  ADC\_read = ADC\_read >> 3; //MÉDIA DAS 8 AMOSTRAS.  return ADC\_read;  }  bool SYSTEM\_ON = false;  void setup()  {  DDRD |= (1 << PD6); //CONFIGURA A SAÍDA PARA O PWM (PD6) - LED VERDE (VELOCIDADE DO MOTOR)  PORTD |= (1 << PD2); //HABILITA O RESISTOR DE PULL-UP no pino 2(PD2).  PORTD &= ~(1 << PD6); //DESLIGA A SAÍDA PD6.(PWM INICIA desligado).    EICRA = 0b00000010; //CONFIGURA A INTERRUPÇÃO EXTERNA 0 P/ TRANSIÇÃO DE DESCIDA.  EIMSK = 0b00000001; //HABILITA A INTERRUPÇÃO EXTERNA 0 (INT0).    TCCR0A |= (1 << WGM01) | (1 << WGM00) | (1 << COM0A1); //CONFIGURA o modo FAST PWM e modo do comparador A.  TCCR0B |= (1 << CS00); //CONFIGURA O DIVIDOR DE CLOCK. (CLK = 1)    Serial.begin(9600); //INICIA A PORTA SERIAL - CONFIGURA A TAXA DE DADOS P/ 9600 BPS.  ADC\_init(); //INICIALIZA ADC  InitLCD(); //INICIALIZA LCD  sei(); //HABILITA A INTERRUPÇÃO GLOBAL.  }  void loop()  {  //conversor AD  u16 adc\_result0;  unsigned int T, PESO, VELOCIDADE;  unsigned long int aux1;    adc\_result0 = ADC\_read(ADC0D); //lê o valor do ADC0 = PC0    //CÁLCULO DA TENSÃO  aux1 = (long)adc\_result0 \* 5000; //5000ml ou 5V  aux1 /= 1023;  T = (unsigned int)aux1;  //CÁLCULO DO PESO  PESO = (T \* 2);    //CÁLCULO DA VELOCIDADE (VARIA de 0% à 100% e é IVERSAMENTE PROPORCIONAL AO PESO DO MATERIAL).  VELOCIDADE = (10000 - PESO) / 10000.0 \* 255;    //UART  if (Serial.available() > 0) //LÊ SERIAL  {  char recebido = Serial.read();  if(recebido == 'L')  {  SYSTEM\_ON = true; //Se recebeu L, LIGA O SISTEMA.  Serial.println ("SIST. LIGADO"); //ENVIA PELA SERIAL A MENSAGEM "SIST. LIGADO".  //PWM  //ATUALIZA A VELOCIDADE DO MOTOR.  OCR0A = VELOCIDADE;  //LCD  //ATUALIZA O PESO DO MATERIAL (MOSTRANDO-O NO DISPLAY).  LCDCmd(0x01);//ANTES DE ATUALIZAR O PESO, LIMPA O DISPLAY.  LCDPrintXYStr(1, 1, "PESO: ");  LCDPrintXYVal(7, 1, PESO);    if(PESO < 10)  LCDPrintXYStr(8, 1, "g");  else if(PESO < 100)  LCDPrintXYStr(9, 1, "g");  else if(PESO < 1000)  LCDPrintXYStr(10, 1, "g");  else if(PESO < 10000)  LCDPrintXYStr(11, 1, "g");  else LCDPrintXYStr(12, 1, "g");  }  }  //GPIO  if(SYSTEM\_ON == true) //SE O SISTEMA ESTIVER LIGADO, VERIFICA O SENSOR  if((PINB & sensor1) == 0) //SENSOR DE PRESENÇA DESATIVADO (NÃO DETECTANDO MATERIAL)?  {  SYSTEM\_ON = false; //DESLIGA O SISTEMA.  Serial.println ("SIST. DESLIGADO"); //ENVIA PELA SERIAL A MENSAGEM "SIST. DESLIGADO".  OCR0A = 0; //O MOTOR É DESLIGADO (0 DE PWM).  LCDCmd(0x01); //LIMPA O DISPLAY.  }  }  //INTERRUPÇÃO EXTERNA 0 (INT0).  ISR(INT0\_vect)  {  if(SYSTEM\_ON == true)  {  Serial.println("SIST. PARADO!"); //ENVIA PELA SERIAL A MENSAGEM "SIST. PARADO!".  OCR0A = 0; //O MOTOR É DESLIGADO (0 DE PWM).  LCDCmd(0x01); //LIMPA O DISPLAY.  }  \_delay\_ms(500);  } |

LINKS:

<https://drive.google.com/file/d/1MTy-DwOLXfo5pvu5V6CQLQzapMrIvksW/view?usp=sharing>

<https://whimsical.com/projeto-2nj6LWyPkPK9qwEyXXPac3>

<https://www.tinkercad.com/things/2n84SR1RlM1-projeto-e209l1-tema02/editel?sharecode=gh2YluyXZUeQFF05o49qEiOX3_wf65xWZaBlwyA_gr4>