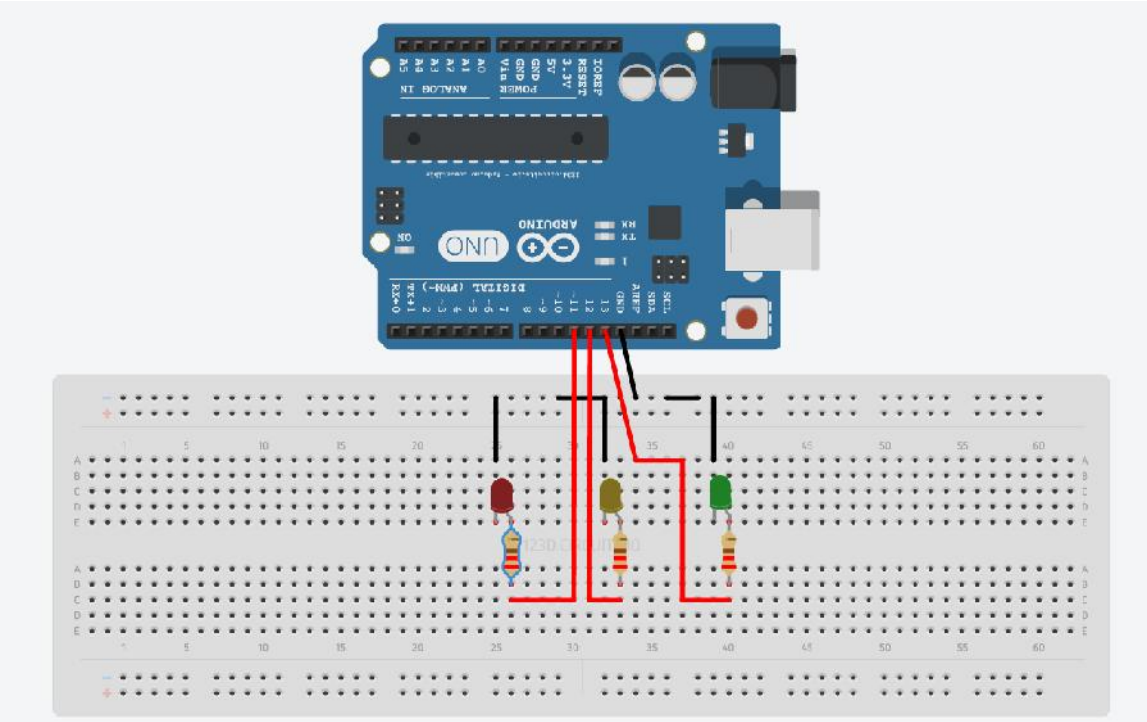
Laboratorio de Programacao Avancada TP13 – Arduino
Ludymila Lobo – 21101577

EXERCÍCIO 1

Altere o programa do semáforo para, antes de apagar o verde e acender o amarelo, o verde deve passar 2 segundos piscando. Ideal que pisque entre 5 e 10 vezes. Altere os tempos dos semáforos à seu gosto.



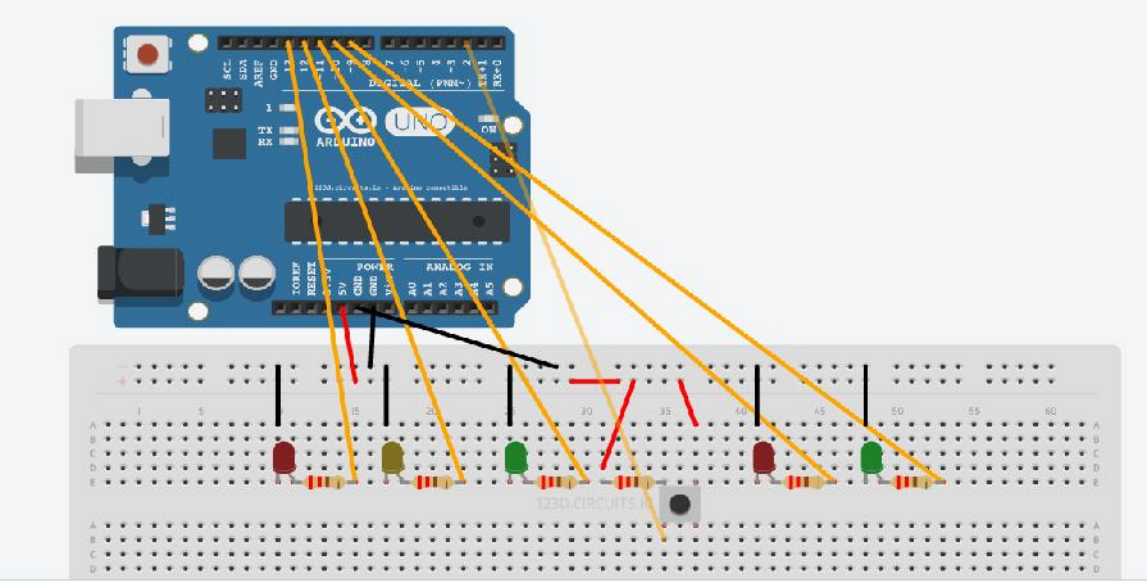
Para conectar os LEDs, liguei o Ground (GND) do Arduino na primeira linha do protoboard, e conectei todas as extremidades negativas dos LEDs nesta linha. Já a parte positiva dos LEDs, foi conectada nos pinos digitais 11, 12 e 13 por meio de resistores de 220 Oms.
O resto da lógica foi feita via código, alterando valores de delay e fazendo um loop para piscar a luz verde.

```
// Semaforo
int ledDelay = 1000;
int redPin = 11;
int yellowPin = 12;;
int greenPin = 13;
void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  digitalWrite(redPin, LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin, LOW);
```

```
}
void piscaVerde()
{
  for (int x=0; x<=5; x++){
    digitalWrite(greenPin, HIGH);
    delay(300);
    digitalWrite(greenPin, LOW);
    delay(300);
  }

}

void loop() {
  digitalWrite(redPin, HIGH);
  delay(2*ledDelay);
  digitalWrite(redPin, LOW);
  digitalWrite(greenPin, HIGH);
  delay(3*ledDelay);
  digitalWrite(greenPin, LOW);
  piscaVerde();
  digitalWrite(yellowPin, HIGH);
  delay(1.2*ledDelay);
  digitalWrite(yellowPin, LOW);
}
```

EXERCÍCIO 2

Altere o programa do semáforo interativo em duas situações: (1) o sinal verde fique piscando (entre 5 e 10 piscadas) quando faltar 2 segundos para ir para o amarelo; e (2) o sinal verde do pedestre fique piscando (entre 5 e 10 piscadas) quando faltar 1 segundo para o sinal verde do pedestre ir para o vermelho. Altere os tempos dos semáforos à seu gosto.



Para conectar os LEDs, realizei o mesmo procedimento descrito acima para conectar as extremidades negativas dos LEDs, e conectei também a tensão de 5V na segunda linha do protoboard, para conectar uma extremidade do componente de botão através de um resistor de 220 Oms. As partes positivas dos LEDs foram conectadas nos pinos digitais 9, 10, 11, 12 e 13 por meio de resistores de 220 Oms.

```
// Semaforo Interativo
int carRed = 13;
int carYellow = 12;
int carGreen = 11;
int pedRed = 10;
int pedGreen = 9;
int button = 2;
int crossTime = 4000;
unsigned long changeTime;

void setup() {
  pinMode(carRed, OUTPUT);
  pinMode(carYellow, OUTPUT);
  pinMode(carGreen, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);
  pinMode(button, INPUT);
  digitalWrite(carGreen, HIGH);
  digitalWrite(pedRed, HIGH);
}

void piscaVerde(int sinal, int delaySinal)
{
  for (int x=0; x<5; x++){
    digitalWrite(sinal, LOW);
    delay(delaySinal);
    digitalWrite(sinal, HIGH);
    delay(delaySinal);
  }
  digitalWrite(sinal, LOW);

}


void loop() {
  int state = digitalRead(button);
  if (state == HIGH &&
     (millis() - changeTime) > 3000)
     piscaVerde(carGreen, 200);
  if (state == HIGH &&
     (millis() - changeTime) > 5000)
     changeLights();
}

void changeLights() {
  digitalWrite(carGreen, LOW);
  digitalWrite(carYellow, HIGH);
  delay(2000);
```

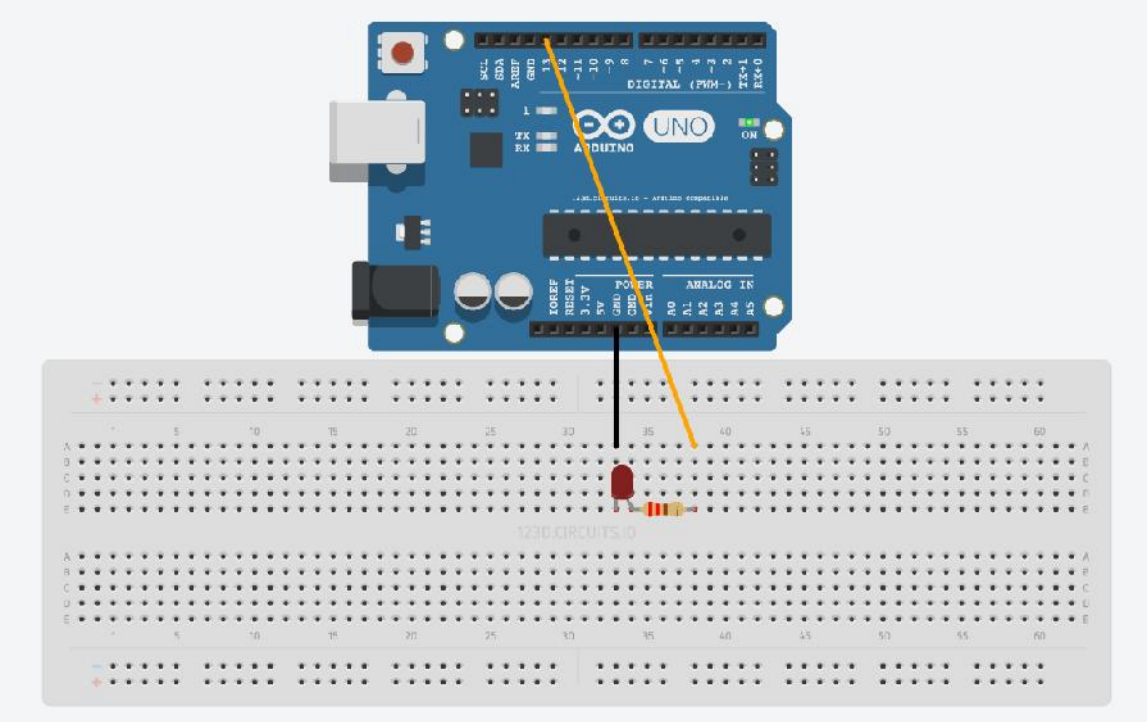
```
  digitalWrite(carYellow, LOW);
  digitalWrite(carRed, HIGH);
  delay(500); // da um tempinho

  digitalWrite(pedRed, LOW);
  digitalWrite(pedGreen, HIGH);
  delay(crossTime-1000);
  piscaVerde(pedGreen,100);
 // acende o vermelho do pedestre
  digitalWrite(pedRed, HIGH);
  // da um tempinho
  delay(500);

  digitalWrite(carRed, LOW);
  // da um tempinho soh para nao confundir
  delay(200);
  digitalWrite(carGreen, HIGH);
  changeTime = millis();
}
```

Exercicio 3

Faça um programa que emita uma mensagem usando o código Morse, que é baseado em pontos (sinal curto) e traços (sinal longo). O mais comum é que a codificação seja enviada por áudio, mas nada impede que também seja visual. Neste caso, sugiro que você use somente 1 LED. Nesse exemplo considere somente as 26 LETRAS. Considere que o sinal curto seja 200ms e que o sinal longo seja 500ms; o espaçamento entre os sinais seja 100 ms; e que o espaçamento entre as letras seja de 150 ms. A palavra a ser transmitida deve ser entrada diretamente no código em uma variável. Arduino aceita switch/case.

A ·–	J ·–––	S ···	2 ··–––
B –···	K –·–	T –	3 ···–––
C –·–·	L ·–··	U ··–	4 ····–
D –··	M ––	V ···–	5 ·····
E ·	N –·	W ·––	6 –····
F ··–·	O –––	X –··–	7 ––···
G ––·	P ·––·	Y –·––	8 –––··
H ····	Q ––·–	Z ––··	9 ––––·
I ··	R ·–·	1 ·––––	0 –––––

Foi utilizada esta tabela como base para o código Morse de cada letra.



A parte do circuito do projeto trata-se apenas de um LED, cujo polo negativo esta conectado no Ground (GND) e o polo positivo está conectado no pino digital 13. O input é lido e passado caractere por caractere para a codificação Morse, que faz o LED piscar de acordo com os tempos definidos pela questão para pontos, traços e intervalos de tempo entre sinais e entre letras.

Após esta primeira solução, foi requerido pelo enunciado que a solução seja criativa e que não use switch/case para cada variável. Portanto, pesquisei sobre a lógica do código Morse e vi que o mesmo poderia ser convertido através de uma função se estiver em formato ASCII. Transformei em código ASCII e chamei esta função de conversão.

```
int led = 13;
int tempoEspaco = 100;
int tempoPonto = 200;
int tempoTraco = 500;
int tempoLetras = 150;
String palavra;
char received;


void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
   Serial.begin(9600);
}
//codigo antes da modificacao para a abordagem criativa
/*void codigoMorse(char received){
```

```
switch (received){
  case ('a'):
      ponto();
      traco();
  break;
  case ('b'):
      traco();
      ponto();
      ponto();
      ponto();
  break;
  case ('c'):
      traco();
      ponto();
      traco();
      ponto();
  break;
  case ('d'):
      traco();
      ponto();
      ponto();
  break;
  case ('e'):
      ponto();
  break;
  case ('f'):
      ponto();
      ponto();
      traco();
      ponto();
  break;
  case ('g'):
      traco();
      traco();
      ponto();
  break;
  case ('h'):
      ponto();
      ponto();
      ponto();
      ponto();
  break;
  case ('i'):
      ponto();
      ponto();
  break;
  case ('j'):
      ponto();
      traco();
      traco();
      traco();
  break;
  case ('k'):
      traco();
      ponto();
      traco();
  break;
```

```
case ('l'):
    ponto();
    traco();
    ponto();
    ponto();
break;
case ('m'):
    traco();
    traco();
break;
case ('n'):
    traco();
    ponto();
break;
case ('o'):
    traco();
    traco();
    traco();
break;
case ('p'):
    ponto();
    traco();
    traco();
    ponto();
break;
case ('q'):
    traco();
    traco();
    ponto();
    traco();
break;
case ('r'):
    ponto();
    traco();
    ponto();
break;
case ('s'):
    ponto();
    ponto();
    ponto();
break;
case ('t'):
    traco();
break;
case ('u'):
    ponto();
    ponto();
    traco();
break;
case ('v'):
    ponto();
    ponto();
    ponto();
    traco();
break;
case ('w'):
    ponto();
```

```
        traco();
        traco();
    break;
    case ('x'):
        traco();
        ponto();
        ponto();
        traco();
    break;
    case ('y'):
        traco();
        ponto();
        traco();
        traco();
    break;
    case ('z'):
        traco();
        traco();
        ponto();
        ponto();
    break;
    default:
    break;
  }
  delay(tempoLetras);
}*/
void traco() {
digitalWrite(led, HIGH);
delay(tempoTraco);
digitalWrite(led, LOW);
delay(tempoEspaco);
}

void ponto() {
digitalWrite(led, HIGH);
delay(tempoPonto);
digitalWrite(led, LOW);
delay(tempoEspaco);
}

static char a2m_alpha[26] = {
    0x05, /* A */
    0x18, /* B */
    0x1a, /* C */
    0x0c, /* D */
    0x02, /* E */
    0x12, /* F */
    0x0e, /* G */
    0x10, /* H */
    0x04, /* I */
    0x17, /* J */
    0x0d, /* K */
    0x14, /* L */
    0x07, /* M */
    0x06, /* N */
    0x0f, /* O */
    0x16, /* P */
```

```
    0x1d, /* Q */
    0x0a, /* R */
    0x08, /* S */
    0x03, /* T */
    0x09, /* U */
    0x11, /* V */
    0x0b, /* W */
    0x19, /* X */
    0x1b, /* Y */
    0x1c  /* Z */
};



//transformando para ascii
int ascii2morse_translate(int c)
{
    int caractere;

    if ( (c >= 'A') && (c <= 'Z') )
        caractere = a2m_alpha[c - 'A'];
    else if ( (c >= 'a') && (c <= 'z') )
        caractere = a2m_alpha[c - 'a'];

    return caractere;
}
//transformando de ascii para morse
int ascii2morse_sendChar(int c)
{
    int code, cnt;

    code = ascii2morse_translate(c);
    if (((char) -1) == code) {

        return -1;
    }
    for (cnt = 0; (!(code & 0x80)) && (cnt <= 7); cnt++ )
        code <<= 1;
    for ( ; cnt <= 6; cnt++ ) {
        code <<= 1;
        if (code & 0x80)
            traco();
        else
            ponto();
    }
    return 0;
}

void loop() {
   int aux;
  while (Serial.available() > 0)
    {
        received = Serial.read();
        palavra += received;

        //codigoMorse(received);
```

```
        aux = ascii2morse_translate(received);
        ascii2morse_sendChar(aux);
        delay(tempoLetras);

          // Process message when new line character is recieved
          if (received == '\n')
          {
              Serial.print(palavra);

                    palavra = ""; // Clear recieved buffer
          }
    }
}
```