APÊNDICE C – CALCULAR TAXA DE AMOSTRAGEM COM FUNÇÃO DE INTERRUPÇÃO

```
/**
 * Calcular Taxa de Amostragem – INTERRUPTION
 *
 * Referência:
 * MAES, Willem. How to make Arduino fast enough to... 2018.
 * Disponível em: <http://www.optiloading.be/willem/Arduino/speeding.pdf>
 */

int numSamples = 0;
long t, t0;
float timePerSample, frequency;

void setup() {
  Serial.begin(115200);

  ADCSRA = 0; // Clear ADCSRA register
  ADCSRB = 0; // Clear ADCSRB register
  ADMUX |= (0 & 0x07); // Set A0 analog input pin
  ADMUX |= (1 << REFS0); // Set reference voltage
  ADMUX |= (1 << ADLAR); // Left align ADC value to 8 bits from ADCH
register

  // Sampling  rate  is  [ADC clock]  /  [prescaler]  /  [conversionclock
cycles]
  // For  Arduino Uno ADC clock  is  16 MHz and a  conversion  takes 13
clock  cycles
  ADCSRA |= (1 << ADPS2) | (1 << ADPS0); // 32  prescaler for  38.5 KHz
  //ADCSRA |= (1 << ADPS2) ; // 16  prescaler for 76.9 KHz
  //ADCSRA |= (1 << ADPS1) | (1 << ADPS0); // 8  prescaler  for 153.8 KHz

  ADCSRA |= (1 << ADATE); // Enable auto trigger
  ADCSRA |= (1 << ADIE); // Enable interrupts when measurement complete
  ADCSRA |= (1 << ADEN); // Enable ADC
  ADCSRA |= (1 << ADSC); // Start ADC measurements
}

ISR(ADC_vect) {
  byte x = ADCH; // Read 8 bit value from ADC
  numSamples++;
}

void loop() {
  if(numSamples >= 1000){
    t = micros() - t0; // Calculate elapsed time
    timePerSample = (float) t / numSamples;
    frequency = (float) numSamples * 1000000 / t;

    Serial.print("Tempo médio por amostra: ");
    Serial.print(timePerSample);
    Serial.println(" µs");
```

```
    Serial.print("Taxa de amostragem: ");
    Serial.print(frequency);
    Serial.println(" Hz");
    Serial.println();

    // Restart
    t0 = micros();
    numSamples = 0;
  }
}
```