### Algoritmo para Placa Arduino Mega (SLAVE)

// Inclusão da biblioteca

#include <Wire.h> // Biblioteca nativa do core Arduino

// Variáveis globais

const int myAddress = 0x08; // armazena o endereço deste dispositivo (slave)

void setup() {

  Serial.begin(115200);  // Configura a taxa de transferência em bits por

  //                        segundo (baud rate) para transmissão serial.

  Serial.println();

  Wire.begin(myAddress); // inicia o dispositivo com o endereço definido anteriormente

  Wire.onReceive(receiveEvent); //registra o evento de recebimento de mensagem

  Serial.println("Software do arduino MEGA");

}

void loop() {}

void receiveEvent(int howMany) {

  String message = readString();

  Serial.print(message); // imprime a mensagem recebida

}

String readString() {

  String retorno;

  while (Wire.available()) { // Enquanto houver bytes disponíveis para leitura, ...

    char c = Wire.read(); // recebe o byte como caractere

    retorno += c;

  }

  return retorno;

Algoritmo para placa Arduino Uno (MASTER)

// Inclusão da biblioteca

**#include <Wire.h> // Biblioteca nativa do core Arduino**

// Variáveis globais

const **int** slaveAddress = *0x08*; // constante armazena o endereço do dispositivo slave

**int** x = 0; // variável de armazenagem do valor para envio

**void** setup() {

Wire.begin(); // inicia a comunicação I2C

}

**void** loop() {

Wire.beginTransmission(slaveAddress); // transmite para o dispositivo slave

// Cria e formata a mensagem para envio

String message = "O valor de x é ";

message.concat(x);

message.concat("\n");

Wire.write(message.c\_str());// envia a mensagem

Wire.endTransmission(); //para de transmitir

x++; // incremento da variável

delay(500); // pausa de 500 milissegundos

}