Código Fonte – Fase 4

```
/*
 * Desenvolvedor: Luis Fernando
Galonetti
 * Orientador: André Takeshi Endo
 *
 * Alterações: Bruno Eduardo
Esteves de Lima,
 *             Rhuan Edson
Caldini Costa,
 *             Gulherme Ricken
Mattiello.
 *
 * Criado em Outubro de 2014
 * Alterado em Maio de 2016
 * UTFPR-CP
 *
 */
//pins
int RED3 = 12;
int GREEN3 = 11;
int BtnPedestre = 10;
int RED1 = 9;
int YELLOW1 = 8;
int GREEN1 = 7;
int RED2 = 6;
int YELLOW2 = 5;
int GREEN2 = 4;
static unsigned long
last_interrupt_time01 = 0;
static unsigned long
last_interrupt_time02 = 0;
static unsigned long lastRed = 0;
int ligado = 0;
int piscando = 0;
class Semaforo{
  int vermelho, amarelo, verde;

  public:
  Semaforo(int verm, int amar,
int verd) {
    vermelho = verm;
    amarelo = amar;
    verde = verd;
```

```
  };

  void ligarVermelho(){
    digitalWrite(vermelho, HIGH);
    digitalWrite(amarelo, LOW);
    digitalWrite(verde, LOW);
  };
  void ligarVerde(){
    digitalWrite(vermelho, LOW);
    digitalWrite(amarelo, LOW);
    digitalWrite(verde, HIGH);
  };
  void ligarAmarelo(){
    digitalWrite(vermelho, LOW);
    digitalWrite(amarelo, HIGH);
    digitalWrite(verde, LOW);
  };
  void desligar(){
    digitalWrite(vermelho, LOW);
    digitalWrite(amarelo, LOW);
    digitalWrite(verde, LOW);
  };
};
class SemaforoPedestre{
  int vermelho, verde;

  public:
  SemaforoPedestre(int verm, int
verd) {
    vermelho = verm;
    verde = verd;
  };

  void ligarVermelho(){
    digitalWrite(vermelho, HIGH);
    digitalWrite(verde, LOW);
  };
  void ligarVerde(){
    digitalWrite(vermelho, LOW);
    digitalWrite(verde, HIGH);
  };
  void desligar(){
    digitalWrite(vermelho, LOW);
    digitalWrite(verde, LOW);
  };
};
```

```
Semaforo s01(RED1, YELLOW1,
GREEN1), s02(RED2, YELLOW2,
GREEN2);
SemaforoPedestre sp(RED3,
GREEN3);
void setup(){
  pinMode(RED1, OUTPUT);
  pinMode(YELLOW1, OUTPUT);
  pinMode(GREEN1, OUTPUT);
  pinMode(RED2, OUTPUT);
  pinMode(YELLOW2, OUTPUT);
  pinMode(GREEN2, OUTPUT);
  pinMode(RED3, OUTPUT);
  pinMode(GREEN3, OUTPUT);
  attachInterrupt(0,
ligarDesligar, FALLING);
  attachInterrupt(1, piscar,
FALLING);
  ligado = 0;
  piscando = 0;
}
//Interrupção para botão
liga/desliga
void ligarDesligar(){
  unsigned long interrupt_time =
millis();
  // If interrupts come faster
than 1000ms, assume it's a bounce
and ignore
  if (interrupt_time -
last_interrupt_time01 > 1000){
    ligado = !ligado;

    if(!ligado)
      naoOperar();
  }
  last_interrupt_time01 =
interrupt_time;
}
//Interrupção para botão piscar
void piscar(){
  unsigned long interrupt_time =
millis();
  // If interrupts come faster
than 1000ms, assume it's a bounce
and ignore
```

```
  if (interrupt_time -
last_interrupt_time02 > 1000){
    piscando = !piscando;
  }
  last_interrupt_time02 =
interrupt_time;
}
void loop(){
  operar();
}
//semaforo pedestre paralelo com
semaforo2
void operar(){
  if(ligado && !piscando){
    s01.ligarVerde();
    s02.ligarVermelho();
    sp.ligarVermelho();

    //3,5s ou se pedestre apertar
botao
    lastRed = millis();

while(digitalRead(BtnPedestre)==0
&& (millis()-lastRed)<3500);
    s01.ligarAmarelo();
    delay(500);
    s01.ligarVermelho();
    s02.ligarVerde();
    sp.ligarVerde();
    delay(3500);
    s02.ligarAmarelo();
    delay(500);
  } else if(ligado && piscando){
    sp.desligar();
    s01.ligarAmarelo();
    s02.ligarAmarelo();
    delay(500);
    s01.desligar();
    s02.desligar();
    delay(500);
  } else{
    naoOperar();
  }
}
void naoOperar(){
  s01.desligar();
```

```
    s02.desligar();
    sp.desligar();
  }
```