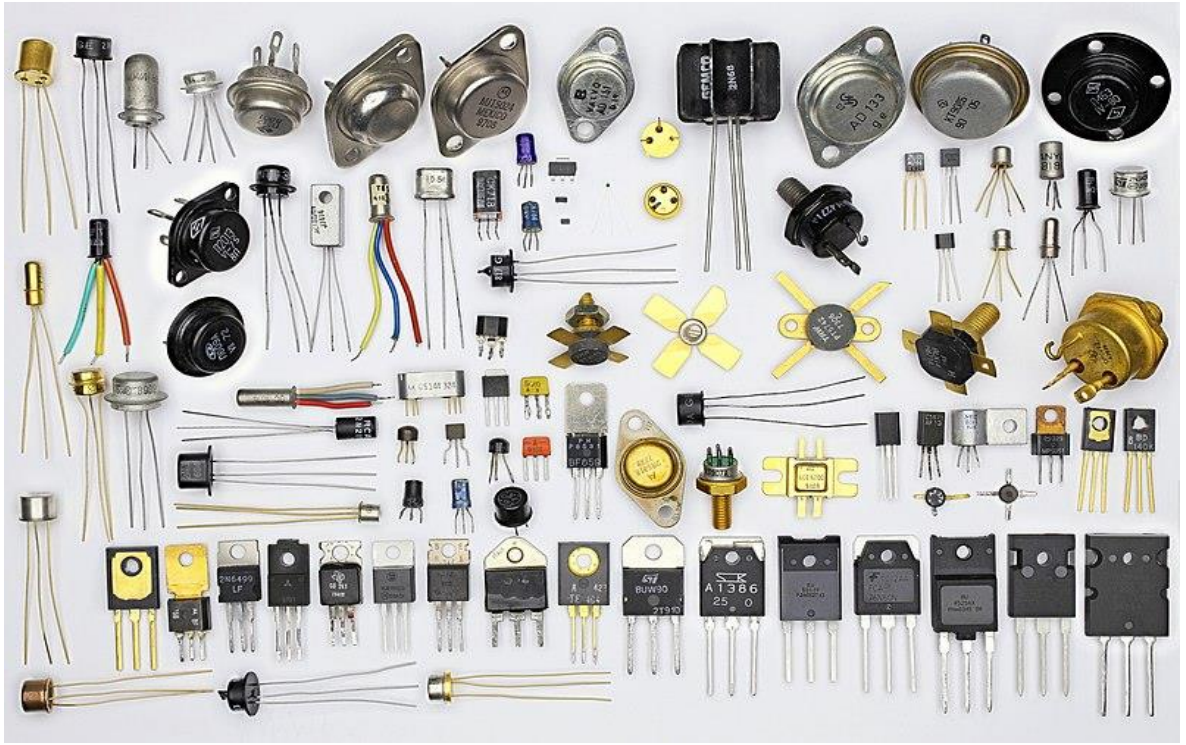
SISTEMAS DE TEMPO REAL

PROFESSOR/PROFESSEUR - VICTOR ROSETTI – PROFVICTOR.QUIROZ@FIAP.COM.BR

COMPUTADORES CLÁSSICOS

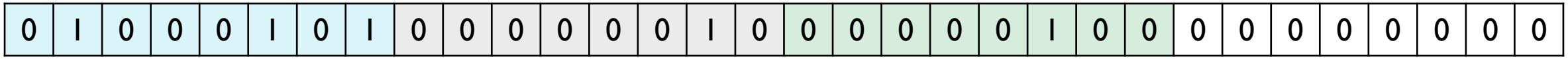
- Computadores clássicos funcionam apenas através de bits, cada bit tem 2 possibilidades (0 ou 1). O TRANSISTOR é o componente que permite aos computadores converter sinais de energia em bits (0 – nenhum sinal/energia e 1-a sinal/pulso de energia)



Various types of transistors. Reference: Wikipedia (https://commons.wikimedia.org/wiki/File:Transistor-photo.JPG)

COMPUTADORES CLÁSSICOS

- Os computadores clássicos operam através de WORDS, um comando binário com certa quantidade de bits que depende do processador. Um processador de 32 bits pode ler PALAVRAS de 32 bits.



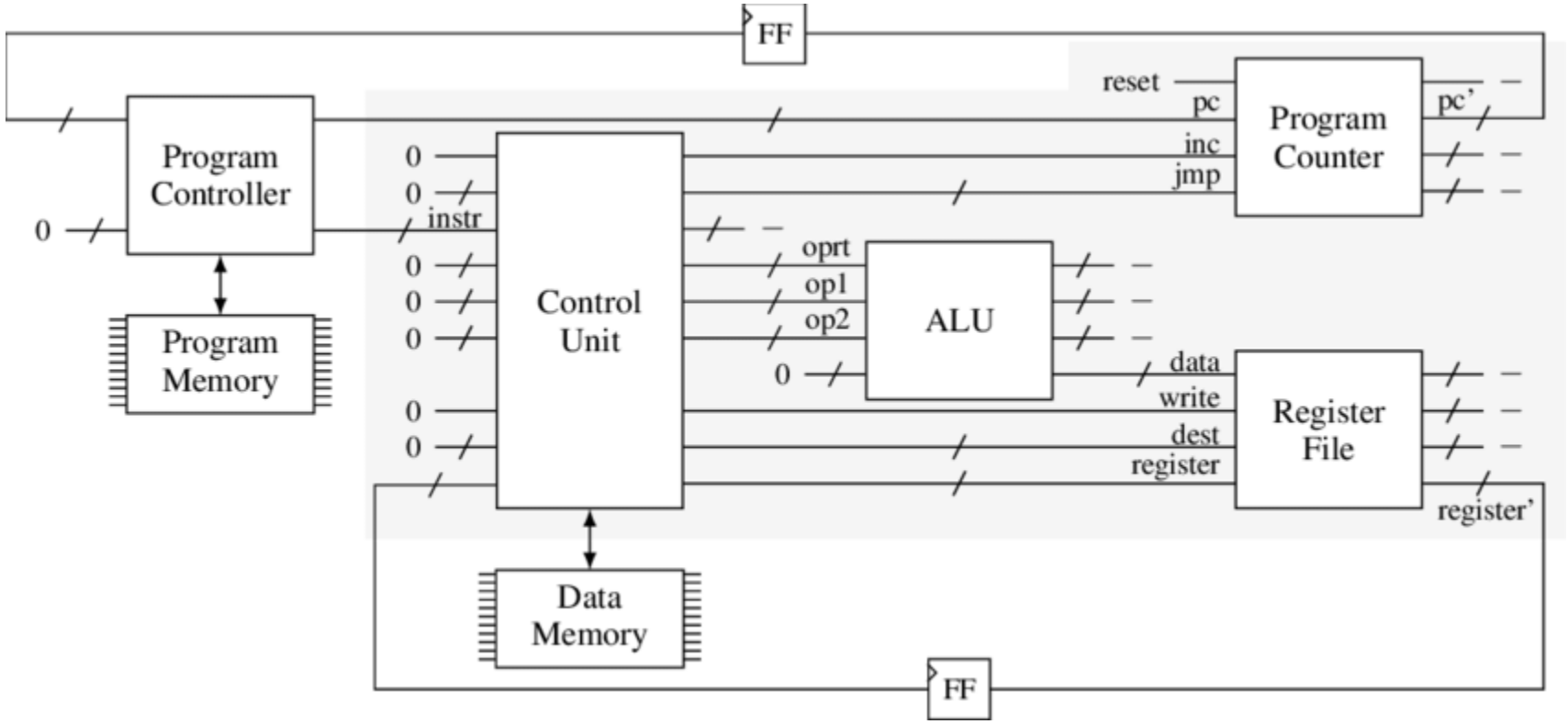
Operação/Comando(ADD) | Registrador A(RA) | Registrador B(RB) | Flags ou bits não utilizados

- Comando em Assembly: **ADD(RA,RB)**

- Comado no código: **variableA + variableB**

COMPUTADORES CLÁSSICOS

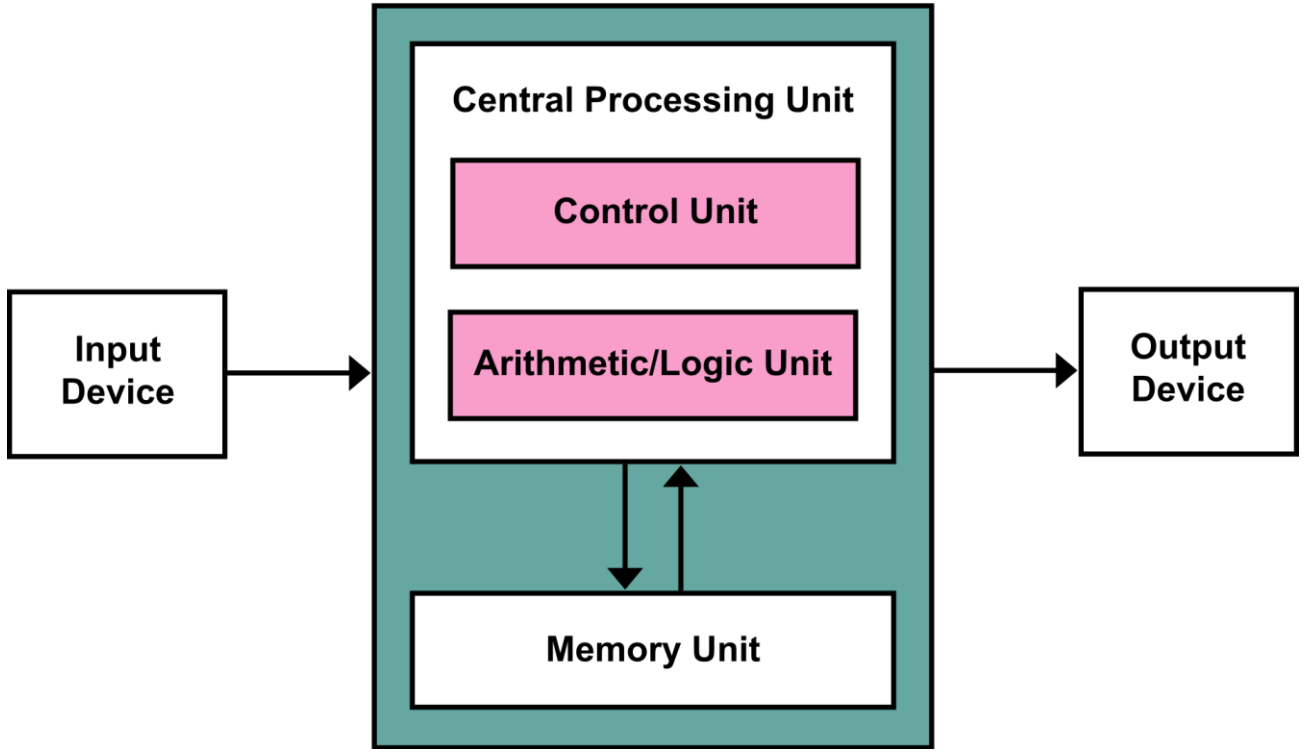
- Ao analisar a arquitetura de um processador é possível compreender como um processo é processado.



Designing a RISC CPU in reversible logic - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Schematic-diagram-of-the-CPU-implementation_fig2_224247580

COMPUTADORES CLÁSSICOS

- Um dos problemas que ainda enfrentamos é conhecido pelo gargálo de Von Neumann, onde o acesso aos dados da memória RAM é afetado pela velocidade da memória e pelo barramento.



Von Neumann architecture. Reference: Wikipedia (https://en.wikipedia.org/wiki/Von_Neumann_architecture#/media/File:Von_Neumann_Architecture.svg)

CLASSICAL COMPUTERS

- A placa mãe do PS5 possuí várias modificações para amenizar os problemas encontrados anteriormente.

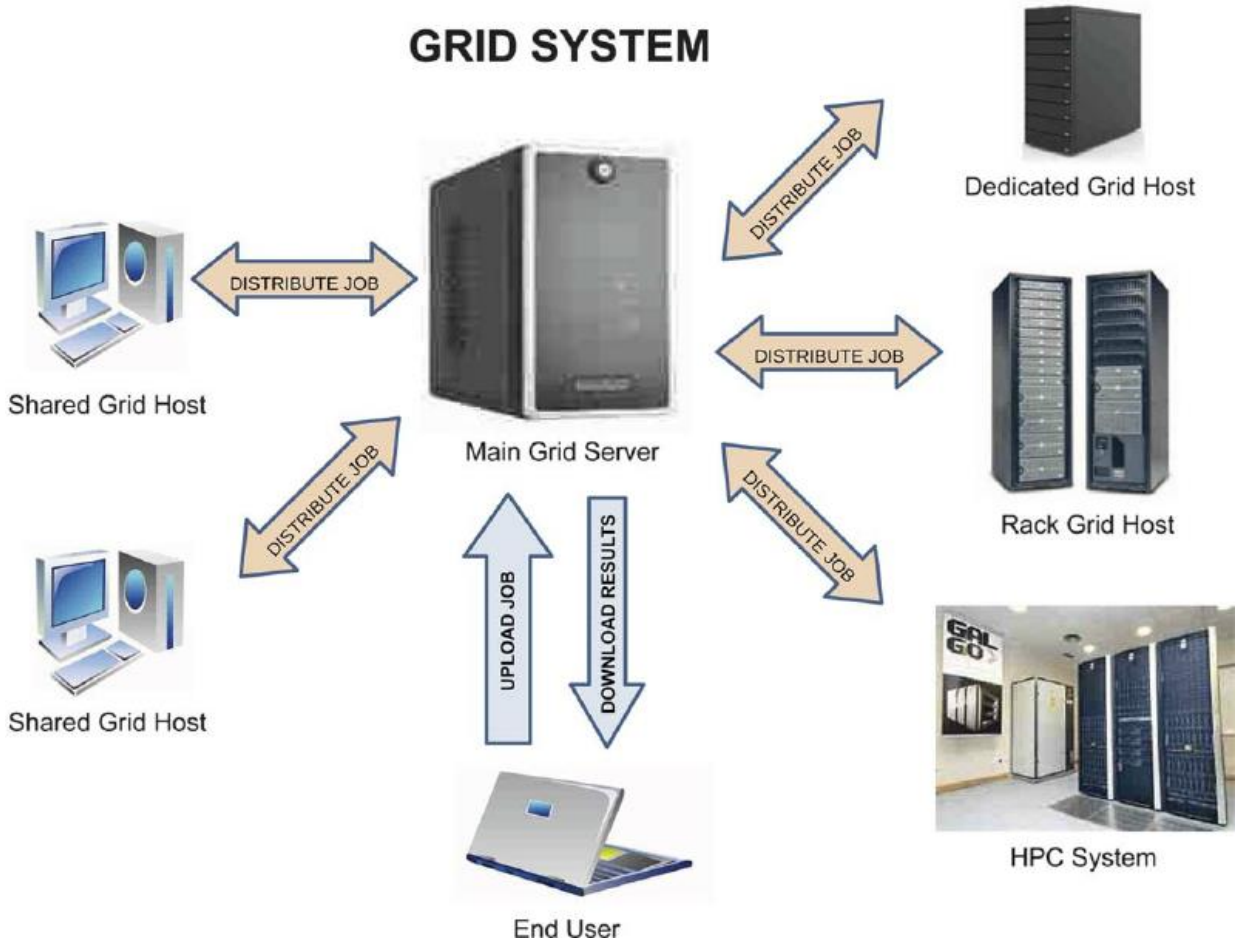
PS5 Teardown. Reference: Playstation Official Channel Youtube (https://www.youtube.com/watch?v=CaAY-jAjm0w)

COMPUTADORES CLÁSSICOS

Clusters. Reference: Wikipedia (https://en.wikipedia.org/wiki/Computer_cluster)

COMPUTADORES CLÁSSICOS

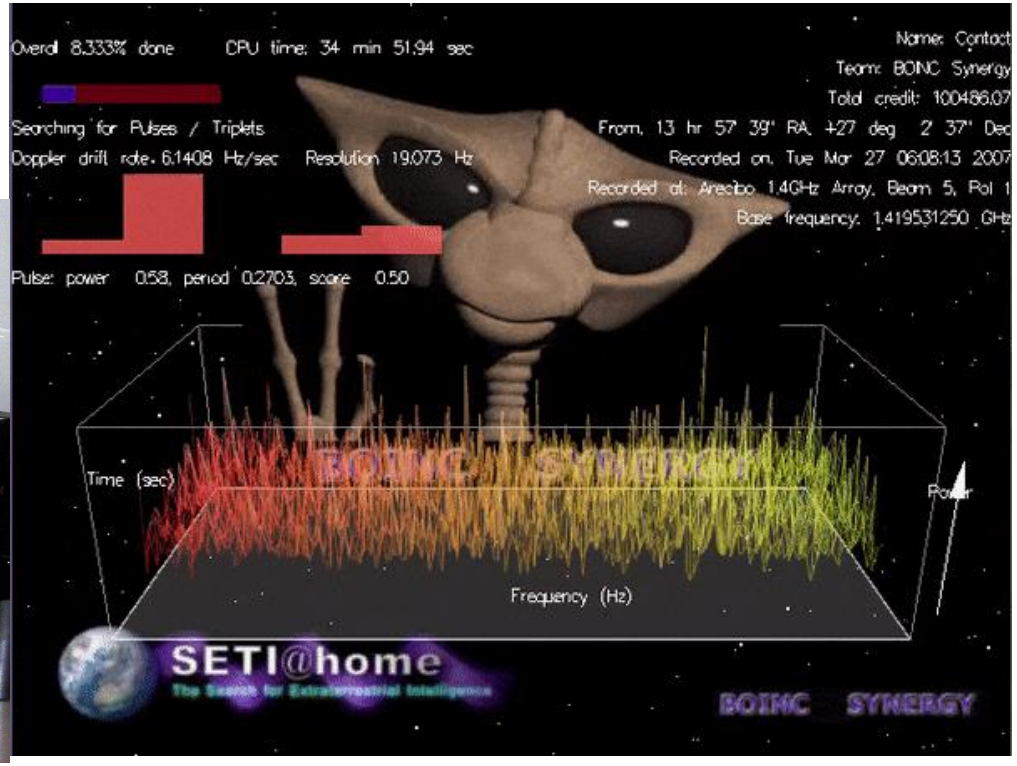
- No slide anterior temos Clusters, conjuntos de máquinas homogêneas, conectadas por redes e um sistema operacional distribuído.
- Ao lado um Grid, um conjunto de máquinas heterogêneas conectadas via software.



Precipitation estimates for hydroelectricity - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Grid-computing-architecture-The-computing-load-is-distributed-by-the-main-grid-server_fig3_257084882

COMPUTADORES CLÁSSICOS

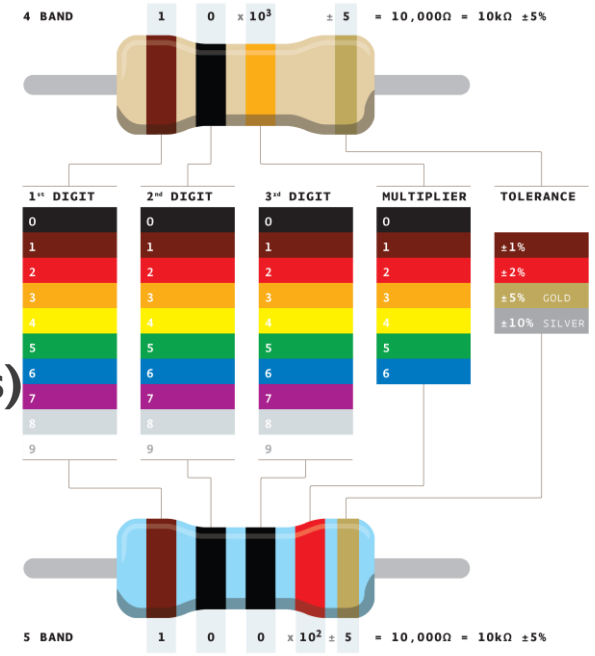
- Dois projetos de Grid, SETI@HOME e FOLDING@HOME

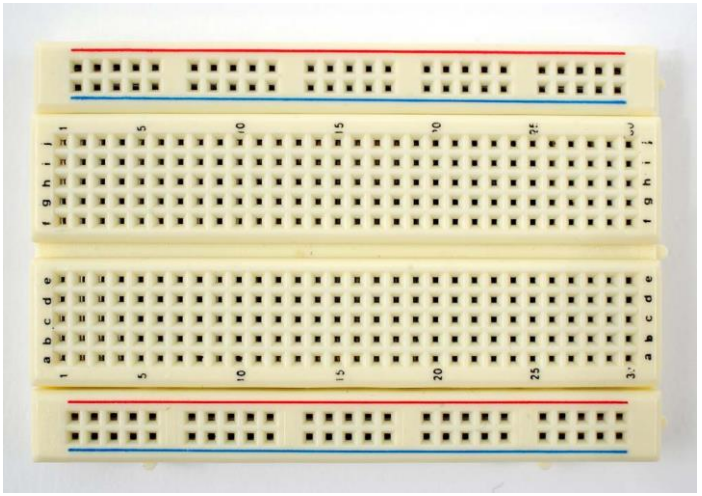


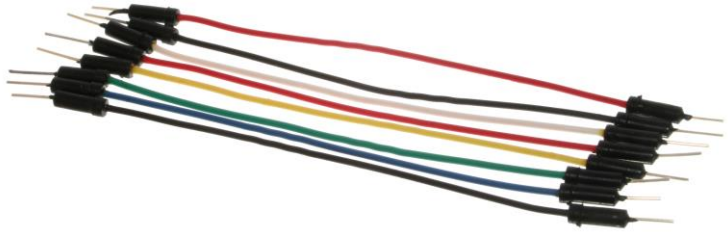
Precipitation estimates for hydroelectricity - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Grid-computing-architecture-The-computing-load-is-distributed-by-the-main-grid-server_fig3_257084882

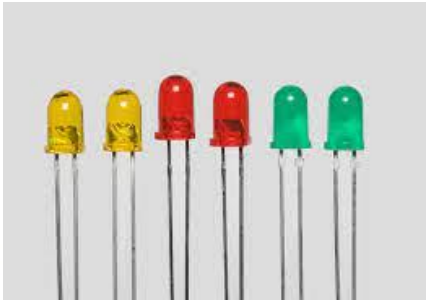
EXPERIÊNCIA RTOS

- **O que você precisará:**
- **1 Arduino Duo**
- **1 Protoboard**
- **2 Leds**
- **2 Resistores (+/-330 ohms)**
- **Jumpers**









EXPERIÊNCIA RTOS

- 1-Crie um código que permita dois leds piscarem
- 2-Tente piscar os leds ao mesmo tempo
- 3-Tente piscar os leds em tempos diferentes
- 4-Tente piscar os leds em tempos diferentes sem interferência de tempo dos “delay"

EXPERIÊNCIA RTOS

- 1-Baixar todos os arquivos de: https://github.com/adamb314/ThreadHandler
- 2-Baixar Arduino IDE de: https://www.arduino.cc/en/software
- 3-Abra o Arduino e clique em SKETCH > INCLUDE LIBRARY > MANAGE LIBRARIES
- 4-Procure por TimerOne e instale a versão mais recente
- 5-Clique novamente em SKETCH > INCLUDE LIBRARY > ADD .ZIP LIBRARY
- 6-Procure o arquivo que você baixou no passo 1
- 7-Use o seguinte código como referência para o seu projeto:

EXPERIÊNCIA RTOS

```
#include "ThreadHandler.h"
//Set Thread interrupt interval
SET_THREAD_HANDLER_TICK(1000)

//Initialize ThreadHandler
THREAD_HANDLER(InterruptTimer::getInstance())

//Create functions of what you want to execute
void function1()
{
  //Code to be executed here
}
void function2()
{
  //Code to be executed here
}
```

```
//Create a Thread for each function
//createThread(priority,period/time to be executed, offset or processor focus, function to be executed)
//priority from 0 to n where 0 is the highest.
//Time is measured in ms so 6000 = 6ms
//Offset is usually devided equally to make the start of the thread even, also the CPU load (ms * (number_of_threads/10)
//where 10 is a number you choose based in how many parts you want to divide the processor load/focus

Thread* thread1 = createThread(1,6000,1000*(2/10),function1);
Thread* thread2 = createThread(2,1000000,1000*(2/10),function2);

void setup()
{
        //Initialize the Threads
        ThreadHandler::getInstance()->enableThreadExecution();
}
```

DÚVIDAS?