# En el código Xbee\_Slave hay que definir lo siguiente:

*#include <Lights.h>*

*// Definition of Master Addresses*

*uint64\_t Master\_address = 0x0013A20040E86DC9;*

*// Trinket Pin which is connected to XBee pin 15*

*uint8\_t Trinket\_pin = 5;*

*// Effects Handler Creation*

*EffectsHandler eHandler = EffectsHandler(Master\_address, Trinket\_pin);*

*// Neopixel Creation*

*#define s\_number 2 //Number of Neopixel Strips*

*// Number of leds in each Neopixel Strip*

*#define Stick1\_pixels 52 //Tira de 52 leds*

*#define Stick2\_pixels 8 //Tira de 08 leds*

*// Trinket pins used for Neopixel Strips control*

*#define Stick1\_pin 11 //Tira de 52 leds*

*#define Stick2\_pin 12 //Tira de 08 leds*

*NeoPatterns Stripes[s\_number] = {NeoPatterns(Stick1\_pixels, Stick1\_pin, NEO\_GRB + NEO\_KHZ800), NeoPatterns(Stick2\_pixels, Stick2\_pin, NEO\_GRB + NEO\_KHZ800)};*

**(en Xbee\_Slave, el resto del código, tanto setup() como loop() se tiene que copiar sin hacer cambios en todos los trinket Slave)**

# En el código Xbee\_Master hay que definir lo siguiente:

*#include <Lights.h>*

*// Definition of Slave Addresses*

*uint64\_t Slave\_address = 0x0013A20040F9E054;*

*// Effects Manager Creation*

*EffectsManager eManager = EffectsManager();*

*// Payload of TX Message*

*uint8\_t payload[] = {StickA, EffectA, RedA, GreenA, BlueA, Interval1A, Interval2A, TotalStepsA, CoolingA, SparkingA};*

**En Xbee\_Master, el código de setup() se tiene que copiar sin hacer cambios. La parte de loop() se usa para crear la secuencia de efectos mediante el comando:**

*eManager.Change(Slave\_address, payload6, 20);*

*//Slave\_address is the address of the Slave Trinket that wants to be commanded,*

*//payload is the message to be sent*

*//20 is the length of the message. IT IS MANDATORY TO INCLUDE THIS FIELD*

# ¿Cómo se define el mensaje que pasa el master al slave (payload)?

uint8\_t payload[] = {**StickA**, **EffectA**, **RedA**, **GreenA**, **BlueA**, **Interval1A**, **Interval2A**, **TotalStepsA**, **CoolingA**, **SparkingA**};

También puede mandarse en un solo mensaje un efecto para más de una tira, por ejemplo, para dos:

uint8\_t payload[] = {**StickA**, **EffectA**, **RedA**, **GreenA**, **BlueA**, **Interval1A**, **Interval2A**, **TotalStepsA**, **CoolingA**, **SparkingA, StickB**, **EffectB**, **RedB**, **GreenB**, **BlueB**, **Interval1B**, **Interval2B**, **TotalStepsB**, **CoolingB**, **SparkingB** };

# Descripción de las componentes del mensaje:

**StickA**: Tira de Neopixel (Que puede valer 1, 2, 3…)

**EffectA**: Número de efecto (De 0 a 11)

* NONE 0 Requiere rellenar:
* FULL\_COLOR 1 Requiere rellenar: **RedA**, **GreenA**, **BlueA**
* DROP 2 Requiere rellenar: **RedA**, **GreenA**, **BlueA**, **Interval1A**
* DOUBLE\_DROP 3 Requiere rellenar: **RedA**, **GreenA**, **BlueA**, **Interval1A**
* SPARKLE 4 Requiere rellenar: **RedA**, **GreenA**, **BlueA**, **Interval1A**
* STROBE 5 Requiere rellenar: **RedA**, **GreenA**, **BlueA**, **Interval1A, Interval2A, TotalStepsA**
* FADE 6 Requiere rellenar: **RedA**, **GreenA**, **BlueA**, **Interval1A, TotalStepsA**
* RAINBOW 7 Requiere rellenar: **Interval1A, TotalStepsA**
* SWEEP 8 Requiere rellenar: **RedA**, **GreenA**, **BlueA**, **Interval1A**
* FIRE\_H 9 Requiere rellenar: **RedA**, **GreenA**, **BlueA**
* FIRE\_V 10 Requiere rellenar: **Interval1A, Interval2A, TotalStepsA**, **CoolingA**, **SparkingA**
* BREATHE 11 Requiere rellenar: **RedA**, **GreenA**, **BlueA**, **Interval1A, TotalStepsA**

**RedA**: Componente roja del color del efecto (de 0 a 255)

**GreenA**: Componente verde del color del efecto (de 0 a 255)

**BlueA**: Componente azul del color del efecto (de 0 a 255)

**Interval1A**: los ms que pasan desde que el Trinket comanda una actualización del efecto hasta la siguiente actualización (de 0 a 255)

**Interval2A**: similar a Interval1A, pero sólo con algunos efectos (de 0 a 255)

**TotalStepsA**: número de pasos que tiene el efecto (de 1 a 255)

**CoolingA**: Sólo para el efecto de fuego vertical. *Indicates how fast a flame cools down. More cooling means shorter flames, and the recommended values are between 20 and 100* (de 0 a 255)

**SparkingA**: Sólo para el efecto de fuego vertical. *Indicates the chance (out of 255) that a spark will ignite. A higher value makes the fire more active. Suggested values lay between 50 and 200, with my personal preference being 120* (de 0 a 255)

# Descripción de los efectos:

FULL\_COLOR fija el color definido en **RedA**, **GreenA**, **BlueA** instantáneamente.

DROP hace caer un píxel de color definido por **RedA**, **GreenA**, **BlueA**. La velocidad de caída depende de **Interval1A**, que indica los ms que tarda la luz en llegar a un led nuevo.

DOUBLE\_DROP hace caer un píxel de color definido por **RedA**, **GreenA**, **BlueA**. Cuando llega al final, rebota una vez y vuelve hasta el principio. La velocidad de caída depende de **Interval1A**, que indica los ms que tarda la luz en llegar a un led nuevo.

SPARKLE enciende un pixel aleatorio con el color definido por **RedA**, **GreenA**, **BlueA** cada **Interval1A** ms y apaga todos los demás. Al utilizar valores pequeños de **Interval1A** se consigue que parezca que hay más de uno simultáneamente encendidos.

STROBE enciende toda la tira con el color definido por **RedA**, **GreenA**, **BlueA** y después la apaga. Se repite de forma continuada. El trinket realiza el cambio de encendido a apagado con el número de pasos intermedios definido por **TotalStepsA**. **Interval1A** indica los ms que pasan desde que el trinket ejecuta uno de los pasos de color hasta que ejecuta el siguiente. Por lo tanto, el paso de encendido a apagado tarda un tiempo total de **Interval1A** (ms) x **TotalStepsA.** El trinket realiza el cambio de apagado a encendido con el número de pasos intermedios definido por **TotalStepsA**. **Interval2A** indica los ms que pasan desde que el trinket ejecuta uno de los pasos de color hasta que ejecuta el siguiente. Por lo tanto, el paso de apagado a encendido tarda un tiempo total de **Interval2A** (ms) x **TotalStepsA.**

FADE cambia el color actual de la tira por el definido por **RedA**, **GreenA**, **BlueA.** El trinket realiza el cambio de un color a otro con el número de pasos intermedios definido por **TotalStepsA**. **Interval1A** indica los ms que pasan desde que el trinket ejecuta uno de los pasos de color hasta que ejecuta el siguiente. Por lo tanto, el paso de un color a otro tarda un tiempo total de **Interval1A** (ms) x **TotalStepsA.**

RAINBOW hace un barrido continuo por los colores del arcoíris. El trinket realiza el paso de un color a otro con el número de pasos intermedios definido por **TotalStepsA**. **Interval1A** indica los ms que pasan desde que el trinket ejecuta uno de los pasos de color hasta que ejecuta el siguiente. Por lo tanto, el paso de un color a otro tarda un tiempo total de **Interval1A** (ms) x **TotalStepsA.**

SWEEP pasa toda la tira al color definido por **RedA**, **GreenA**, **BlueA** en forma de cascada. Es como un DROP que no apaga la estela. La velocidad de caída depende de **Interval1A**, que indica los ms que tarda la luz en llegar a un led nuevo.

FIRE\_H simula un fuego horizontal del color definido por **RedA**, **GreenA**, **BlueA.**

FIRE\_V simula un fuego vertical el que las llamas ascienden por la tira de leds. Los parámetros, **CoolingA** y **SparkingA**, tal y como se ha definido antes, permiten configurar la probabilidad de que aparezcan llamas nuevas y lo rápido que se extinguen las ya existentes. **Interval1A** indica los ms que pasan desde que el trinket ejecuta una actualización de la llama hasta que ejecuta la siguiente. **Interval2A** se utiliza en este efecto para indicar el color del fuego entre las siguientes posibilidades:

* **Interval2A = 1 ->** blanco – amarillo – verde (siendo el blanco el origen de la llama y el verde la parte más alejada)
* **Interval2A = 2 ->** blanco – cyan – verde (siendo el blanco el origen de la llama y el verde la parte más alejada)
* **Interval2A = 3 ->** blanco – rosa – azul (siendo el blanco el origen de la llama y el azul la parte más alejada)
* **Interval2A = 4 ->** blanco – cyan – azul (siendo el blanco el origen de la llama y el azul la parte más alejada)
* **Interval2A = 5 ->** blanco – rosa – rojo (siendo el blanco el origen de la llama y el roja la parte más alejada)
* **Interval2A = cualquier otro valor válido ->** blanco – amarillo – rojo (siendo el blanco el origen de la llama y el rojo la parte más alejada)

Para una tira de 50 leds parámetros aceptables son:

* Interval1A = 50
* CoolingA = 55
* SparkingA = 120

BREATHE simula un encendido y apagado continuo de los leds a ritmo de respiración humana. El color usado es el definido por **RedA**, **GreenA**, **BlueA.** El trinket realiza el cambio de encendido a apagado con el número de pasos intermedios definido por **TotalStepsA**. **Interval1A** indica los ms que pasan desde que el trinket ejecuta uno de los pasos de color hasta que ejecuta el siguiente. Por lo tanto, el paso de encendido a apagado tarda un tiempo total de **Interval1A** (ms) x **TotalStepsA.**

# Conexiones del Trinket Master:

* Pin TX de Trinket a Pin RX de XBee.
* Pin RX de Trinket a Pin TX de XBee.
* Pin G de Trinket a Pin GND de XBee.
* Pin 5V de Trinket a Pin 5v de XBee.

# Conexiones del Trinket Slave:

* Pin TX de Trinket a Pin RX de XBee.
* Pin RX de Trinket a Pin TX de XBee.
* Pin G de Trinket a Pin GND de XBee.
* Pin 5V de Trinket a Pin 5v de XBee.
* Pin *Trinket\_pin* (en el código se sugiere el 5) de Trinket al pin ASC de XBee (pin 15).
* Pin de Trinket a elección al terminal de control de dicha tira (tantas veces como tiras).
* Conectar GND de Trinket+XBee con GND de la fuente y de los NeoPixel.