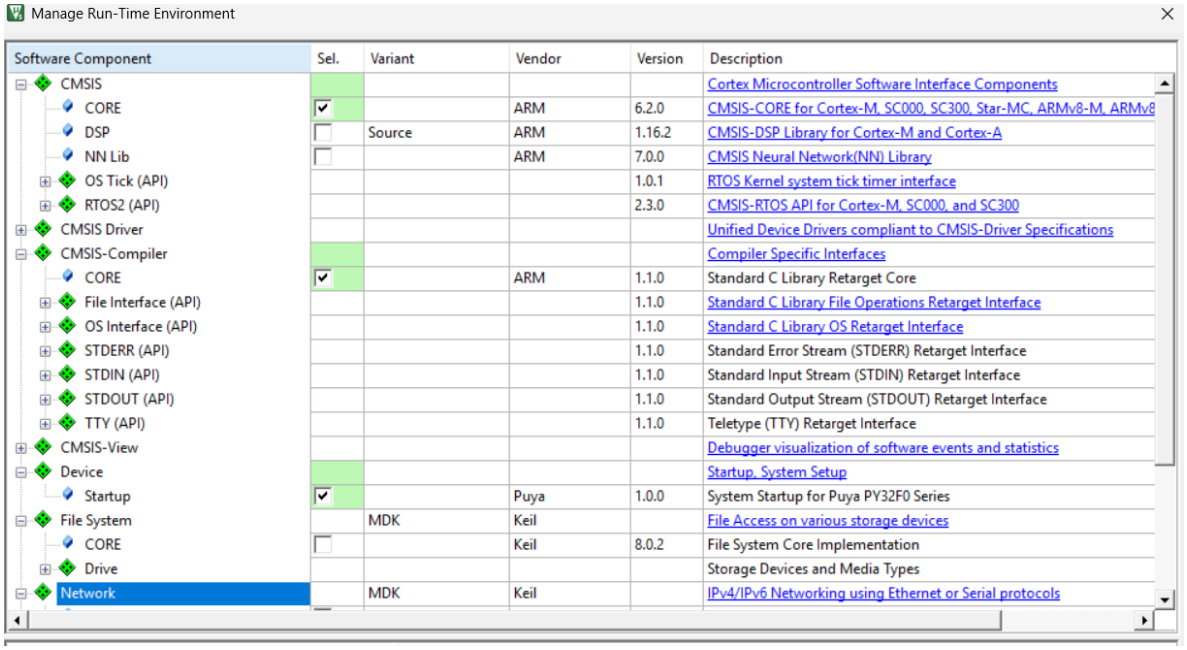
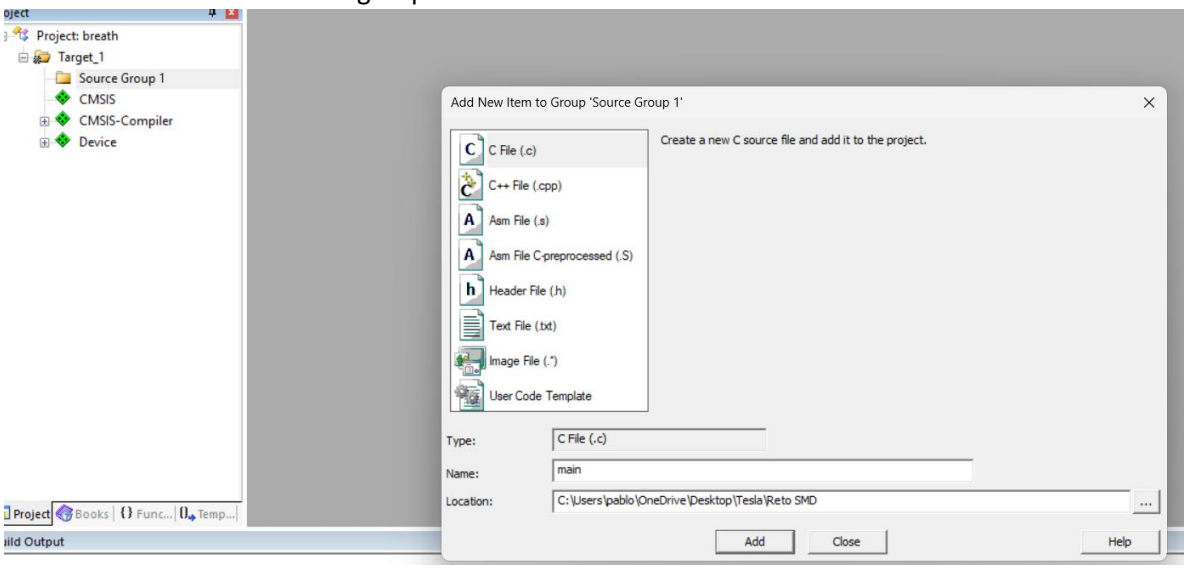
1. Instalar paquete de PUYA en Keil uVision, como usa un ARM M0+ es posible de programar básicamente igual que el STM32F303K8 o STM32F303RE, ver la hoja de datos para ver todas las funciones del microcontrolador: https://download.py32.org/ReferenceManual/en/PY32F002A%20Reference%20manual%20v1.0_EN.pdf

2. Crear proyecto de puya activando CMSIS y device



3. Crear main dentro del source group



4. NOTA: CREAR PROYECTO EN UNA CARPETA APARTE!! PARA EVITAR CUALQUIER TIPO DE IRREGULARIDAD

5. Añadir en este orden los componentes

main.c

```c
#include "RTE_Components.h"
#include "py32f0xx.h"

```

6. Si no compila por la definición del micro PUYA agregar esta definición

```c
#include "RTE_Components.h"

#ifndef PY32F002Ax5
#define PY32F002Ax5
#endif
#include "py32f0xx.h"

```

7. Curva de respiración tipo seno

```c
// Tabla de efecto (no se fue de chat para el efecto)
const uint8_t respiracion[50] = {
    0,  3,  6,  10, 15, 20, 26, 32, 39, 46,
    53, 60, 67, 74, 81, 87, 92, 96, 99, 100,
    100, 99, 96, 92, 87, 81, 74, 67, 60, 53,
    46, 39, 32, 26, 20, 15, 10, 6,  3,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0
};
```

8. Función delay simple

```c
//funcion delay normal
void delay_ms(uint32_t ms)
{
    for(uint32_t i = 0; i < ms * 2000; i++)
    {
        __NOP();
    }
}
```

9. Función de pin y brillo

```
void led_pwm(uint32_t pin, uint8_t brillo)// funcion recibe pin y brillo
{
    // Genera 100 pulsos con PWM
    for(uint8_t i = 0; i < 100; i++)
    {
        if(i < brillo)
            GPIOA->ODR |= (1 << pin);    // LED ON
        else
            GPIOA->ODR &= ~(1 << pin);   // LED OFF

        // Delay para el pwm, poner mas para mas separacion y lentitud jsjs
        for(volatile uint8_t j = 0; j < 15; j++) __NOP();
    }
}
```

10. Configuraciones de registros

```
int main(void)
{
    // 1. Habilitar reloj del GPIOA
    RCC->IOPENR |= RCC_IOPENR_GPIOAEN;

    // 2. Configurar PA1, PA5, PA6, PA7 como salidas
    GPIOA->MODER &= ~((3 << (1*2)) | (3 << (5*2)) | (3 << (6*2)) | (3 << (7*2)));
    GPIOA->MODER |=  ((1 << (1*2)) | (1 << (5*2)) | (1 << (6*2)) | (1 << (7*2)));

    // 3. Configurar como push-pull, velocidad baja
    GPIOA->OTYPER &= ~((1<<1) | (1<<5) | (1<<6) | (1<<7));
    GPIOA->OSPEEDR &= ~((3 << (1*2)) | (3 << (5*2)) | (3 << (6*2)) | (3 << (7*2)));
    GPIOA->PUPDR &= ~((3 << (1*2)) | (3 << (5*2)) | (3 << (6*2)) | (3 << (7*2)));
```

11. Ciclo while

```
    while (1)
    {
        for(uint8_t i = 0; i < 50; i++)
        {
            uint8_t brillo = respiracion[i];//recorro el array de respiraciones con el brillo asignado

            // Todos los pwm tienen el mismo brillo
            led_pwm(1, brillo);  // PA1
            led_pwm(5, brillo);  // PA5
            led_pwm(6, brillo);  // PA6
            led_pwm(7, brillo);  // PA7

            // Delay entre pasos (ajusta para cambiar velocidad)
            // no bajar de 10ms para que se vea, aumentar segun velocidad
            delay_ms(15);
        }
    }
}
```