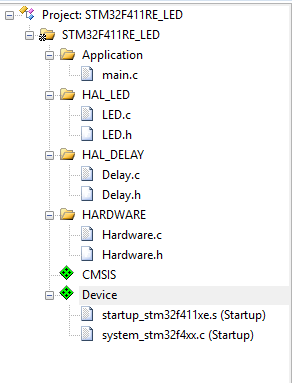
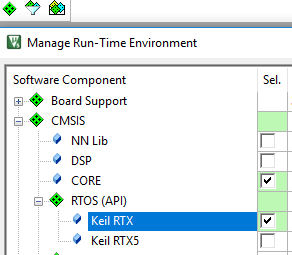
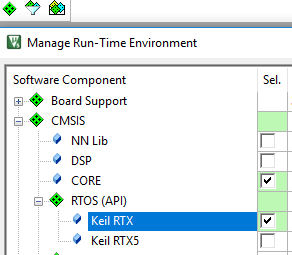
Laboratorio 2.2 – Keil RTX / CMSIS-RTOS

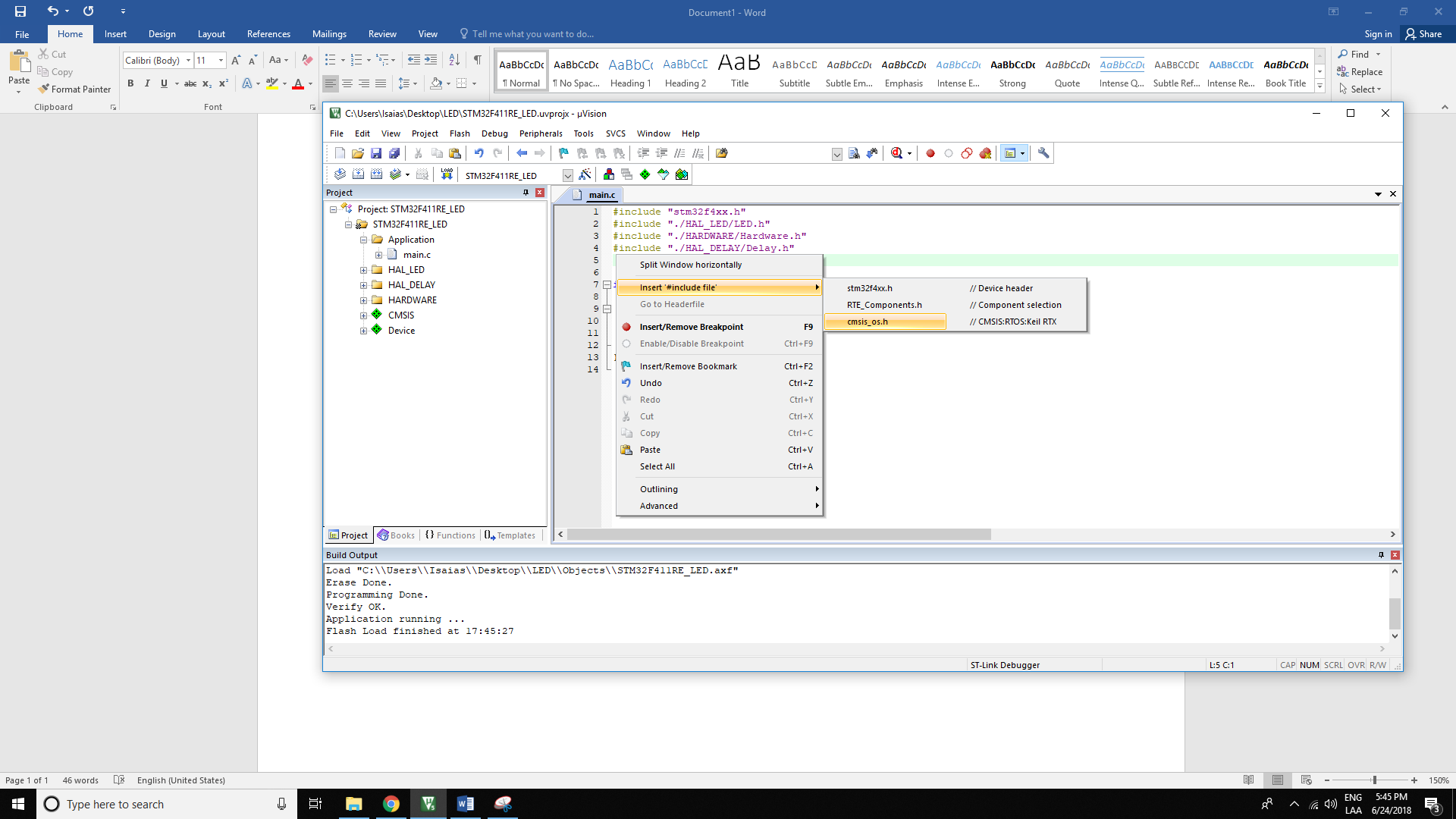
1. Ahora que poseemos toda la estructura necesaria y sabemos como inserter funciones integraremos un sistema operative. El proyecto inicial se ve como sigue:



2. Ingresaremos ahora el Sistema Operativo Keil RTX presionando aquí y seleccionandolo como dice a continuación:

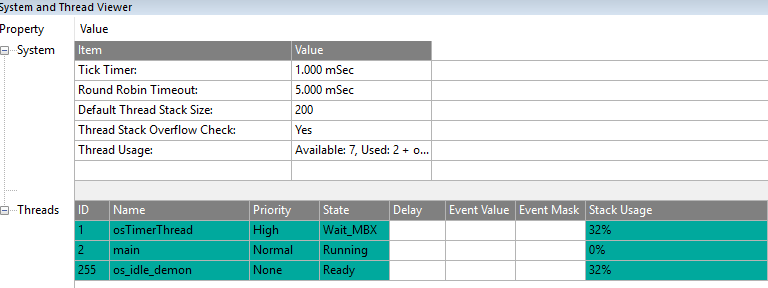
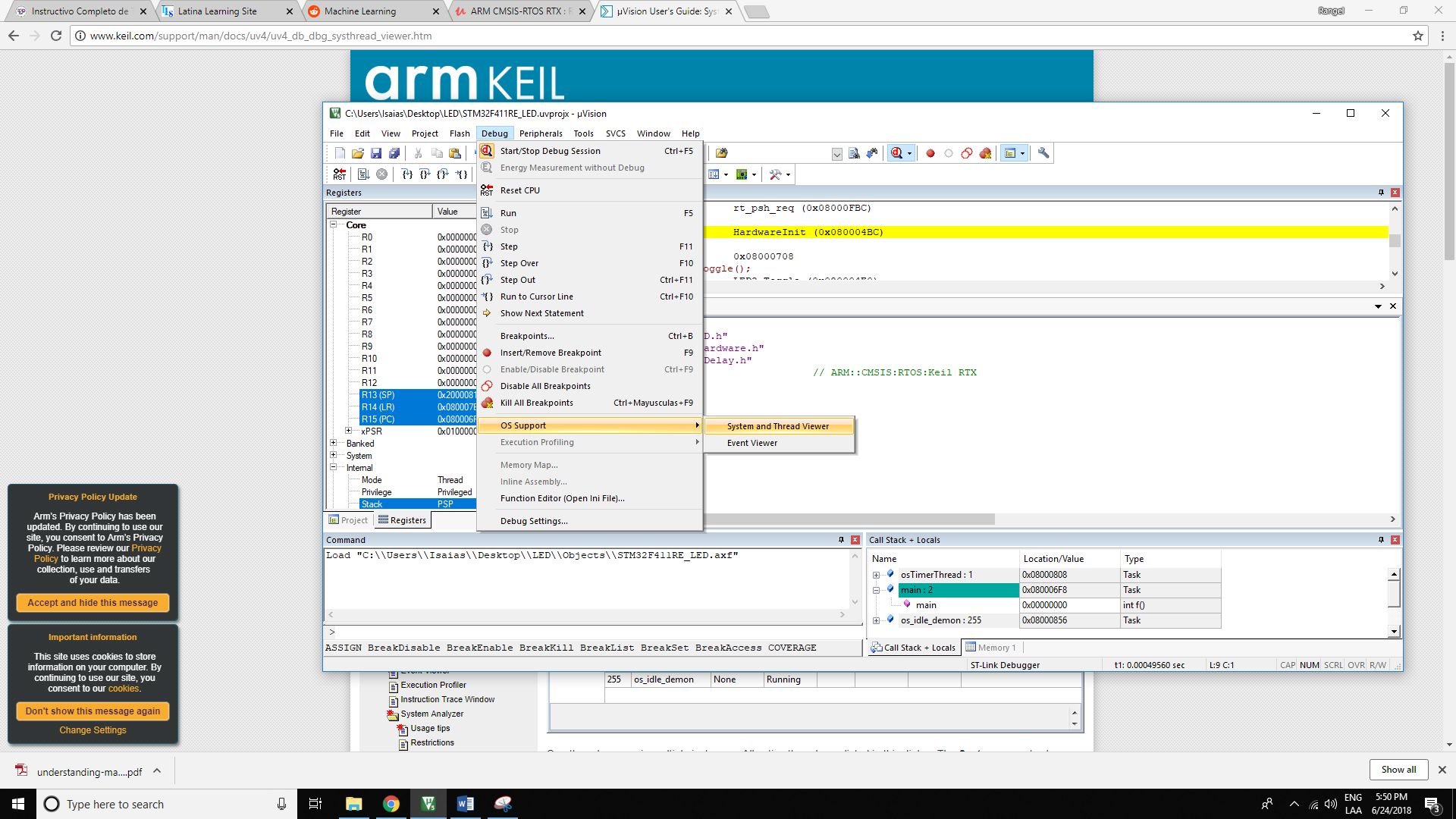


3. Ingresamos el Sistema operative con el botón derecho el archive cabecera cmsis\_os.h:



Asegurese que todo compila en orden y puede “flashear” la tarjeta.

4. Lance el debugger y seleccione como a continuación se muestra:



Como observa existen dos tareas que el Sistema operative lanza por defecto en la sección Threads os\_idle\_demon y osTimerThread, los demas parametros de System son inicializadores por defecto del Sistema operative. Aún no hemos hecho nada, solo nos muestra que el Sistema operative está integrado en la aplicación.

Para Keil y cualquier Sistema Operativo de embebidos la tarea main es la primera tarea.

5. Lo siguiente es crear la(s) tareas, declararlas y definirlas, la declaración en la parte superior de la función main y la definición por debajo de la función main.

void thread\_LED2(void const \*arg);

void thread\_LED2(void const \*arg) {

while (1) {

LED2\_Toggle();

Delay(100000);

}

}

El main ahora quedará de la siguiente manera:

int main(void) {

HardwareInit();

}

6. La manera de que el sistema operativo selecciona y cambia de una tarea a otra es por medio de el id de su tarea, crearemos estos ids fuera del main

osThreadId id\_main, id\_LED2;

7. Lo siguiente esc rear la instancia de la tarea para el sistema operative por medio de la línea:

osThreadDef(thread\_LED2, osPriorityNormal, 1, 0);

la cual irá fuera del main y segun la librería cmsis\_os.h. Examine la siguiente definición de los parámetros y haga analogía a la línea insertada anteriormente:

osThreadDef(name, priority, instances, stacksz)

// Create a Thread Definition with function, priority, and stack requirements.

// name name of the thread function.

// param priority initial priority of the thread function.

// param instances number of possible thread instances.

// param stacksz stack size (in bytes) requirements for the thread function.

8. Seguido tenemos en el main que tomar los id del main y también arrancar las tareas que queramos en el sistema operative, recuerde que no todas las tareas deben arrancar inmediatamente solo las que necesitemos y otras tareas pueden ser arrancadas por una tarea.

He así como quedaría nuestro main

int main(void) {

HardwareInit();

id\_main = osThreadGetId();

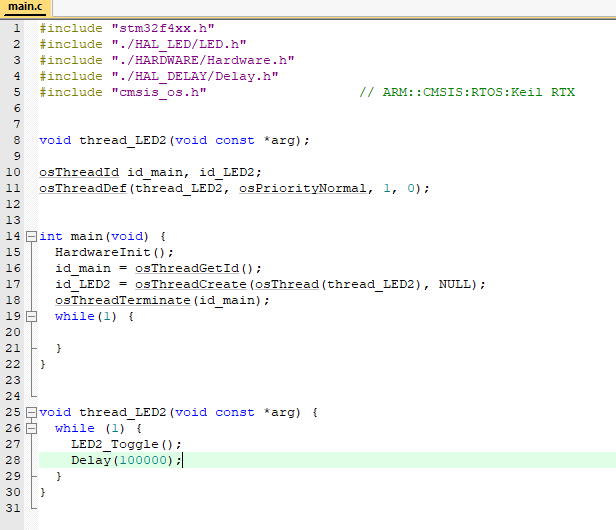
id\_LED2 = osThreadCreate(osThread(thread\_LED2), NULL);

}

9. Debajo de la línea donde cramos la tarea del LED2 (id\_LED2 = …) vamos a terminar la función main porque toma recursos que no queremos

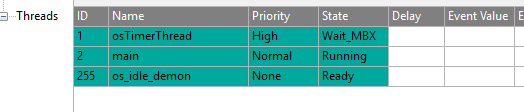
osThreadTerminate(id\_main);

y también crearemos debajo de el un loop infinito, recuerde que el sistema operativo administra todo lo demas



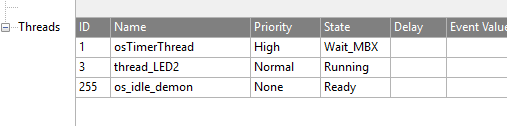
10. Si ahora procede a compilar y “flashear” la tarjeta verá que en tiempo real tiene el mismo comportamiento que nuestra aplicación de LED, al final para el usuario es indiferente pero nosotros hemos logrado inserter un sistema operativo (Keil RTX) en nuestra tarjeta.

11. Lance el Debugger y observe la ventana de “System and Thread Viewer”



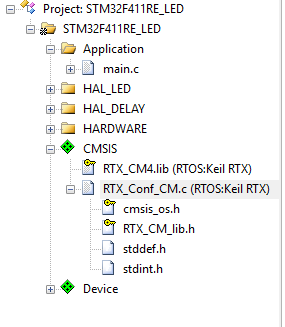
La tarea os\_idle\_demon está en el sistema operativo y no está corriendo pero es normalmente utilizada cuando ubicamos al sistema en bajo consume, es allí provechosa.

12. Ejecute con Step Over (F10) una y otra vez y observe en esta ventana como añade las tareas, elimina y ejecuta, esta es la función básica de un sistema operativo.



13. Si obtuvieramos más de 1 tarea podríamos ver como el sitema operativo pausa una tarea y sigue con otra.

14. Hecharemos un vistazo ahora a la configuración del Sistema Operativo hacienda doble click en el archivo RXT\_Conf\_CM.c



OS\_TASKCNT es el máximo de tareas que podemos correr concurrentemente, este número puede cambiarse (aumentar o disminuirse)

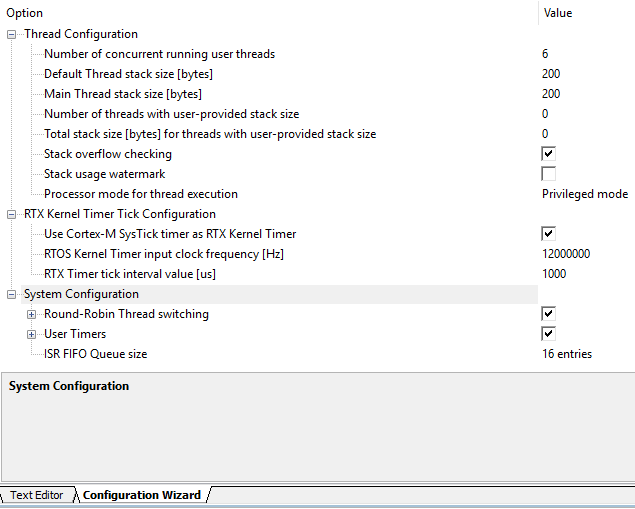
OS\_STKSIZE es el valor que dictaminará si llamamos a este macro y puede cambiarse

El valor por defecto se da cuando ponenos esta variable en nuestra rutina de creación de tareas como 0, lo cual nos dice que tiene 200.

También podemos cambiar el valor de OS\_MAINSTK que es la cantidad de stack reservado por la rutina principal.

Existen otras variables importantes como OS\_CLOCK, OS\_SYSTICK, OS\_TICK, examínelas.

Si no estamos a gusto en cambiar estos parámetros en esta ventana podemos realizarlo con la pestaña inferior de configuration wizard:



14. Para el laboratorio su asignación es lo siguiente UTILIZANDO UN SISTEMA OPERATIVO:

a. Tomar cada grupo alguno de los siguientes programas (no debe repetirse, cada grupo debe hacer un programa diferente):

Hardware:

(Resistencia de 1k, 3 LEDs). Si no desea comprar el hardware puede corroborar los estados de los LEDs con un multímetro y el tiempo con un osciloscopio.

Salidas: LED1 = PA4, LED2 = PA5, LED3 = PA6

Diego:

Enciende LED1, espera un tiempo t1

Enciende LED2, espera un tiempo t2

Enciende LED3, espera un tiempo t3

Apaga LED1, espera un tiempo t1

Apaga LED2, espear un tiempo t2

Apaga LED3, espera un tiempo t3

Repite Indefinidamente

Hellen

Enciende LED1, espera un tiempo t1

Apaga LED1, espera un tiempo 1

Enciende LED2, espera un tiempo t2

Apaga LED2, espera un tiempo t2

Enciende LED3, espera un tiempo t3

Apaga LED3, espera un tiempo t3

Repite Indefinidamente

Laura:

Enciende LED1 y LED3 espera un tiempo t1

Apaga LED1, espera un tiempo 1

Enciende LED2 y LED1 espera un tiempo t2

Apaga LED2, espera un tiempo t2

Enciende LED3 y LED2, espera un tiempo t3

Apaga LED1, espera un tiempo t3

Repite Indefinidamente

Grupo 4:

Enciende LED3 espera un tiempo t1

Apaga LED3, espera un tiempo 1

Enciende LED2 espera un tiempo t2

Apaga LED2, espera un tiempo t2

Enciende LED1, espera un tiempo t3

Apaga LED1, espera un tiempo t3

Repite Indefinidamente

Grupo 5:

Enciende LED3 espera un tiempo t1

Enciende LED2 espera un tiempo t2

Enciende LED1, espera un tiempo t3

Apaga LED1, espera un tiempo t3

Apaga LED2, espera un tiempo t2

Apaga LED3, espera un tiempo t1

Repite Indefinidamente

Grupo 6:

Enciende LED1 espera un tiempo t1

Enciende LED2 espera un tiempo t2

Enciende LED3, espera un tiempo t3

Apaga LED3, espera un tiempo t3

Apaga LED2, espera un tiempo t2

Apaga LED1, espera un tiempo t1

Repite Indefinidamente