Imagen que contiene edificio, cielo

Descripción generada automáticamente

Uso de RIOT-OS para programación en el ámbito de IoT

María Ariza Gamero

Alejandro Hernán Luque

Alberto Vázquez Baeza

Infraestructura Avanzada de

Redes de Sensores

Máster en Sistemas Inteligentes en

Energía y Transporte

Contenido

[1. Presentación 2](#_Toc11342908)

[2. RIOT-OS 2](#_Toc11342909)

[3. Git 2](#_Toc11342910)

[3.1. Conectarse a este repositorio 3](#_Toc11342911)

[Iniciar el repositorio de Git 3](#_Toc11342912)

[Conectarse al repositorio 3](#_Toc11342913)

[Descarga archivos del repositorio 3](#_Toc11342914)

[3.2. Realizar cambios en Git 3](#_Toc11342915)

[Estructura de Git 3](#_Toc11342916)

[Pasar de WD al área de Staging 4](#_Toc11342917)

[Commit 4](#_Toc11342918)

[3.3. Realizar subida de los cambios realizados 4](#_Toc11342919)

[4. Docker 4](#_Toc11342920)

[5. Compilar 5](#_Toc11342921)

[6. Flashear 5](#_Toc11342922)

[7. Programa leds 5](#_Toc11342923)

[8. Comunicación 5](#_Toc11342924)

# Presentación

Se desea instalar el sistema operativo RIOT-OS en las placas de Texas Instruments cc2538dk con el programador SmartRF06 y cc2650stk con el programador , conocida como Sensor Tag. Para ello, será necesito el uso de Docker. Una vez ejecutado RIOT-OS, se procederá a comprobar que funciona correctamente empleando un sencillo programa de encendido de LEDs, que primero se compilará desde RIOT-OS y después de flasheará una herramienta nativa propietaria y necesaria de Texas Instrument, Uniflash. Otro flaseador propio de esta misma marca es Flash Programmer, que también se usará. Por último, se procederá a realizar comunicación usando el protocolo 802.15.4.

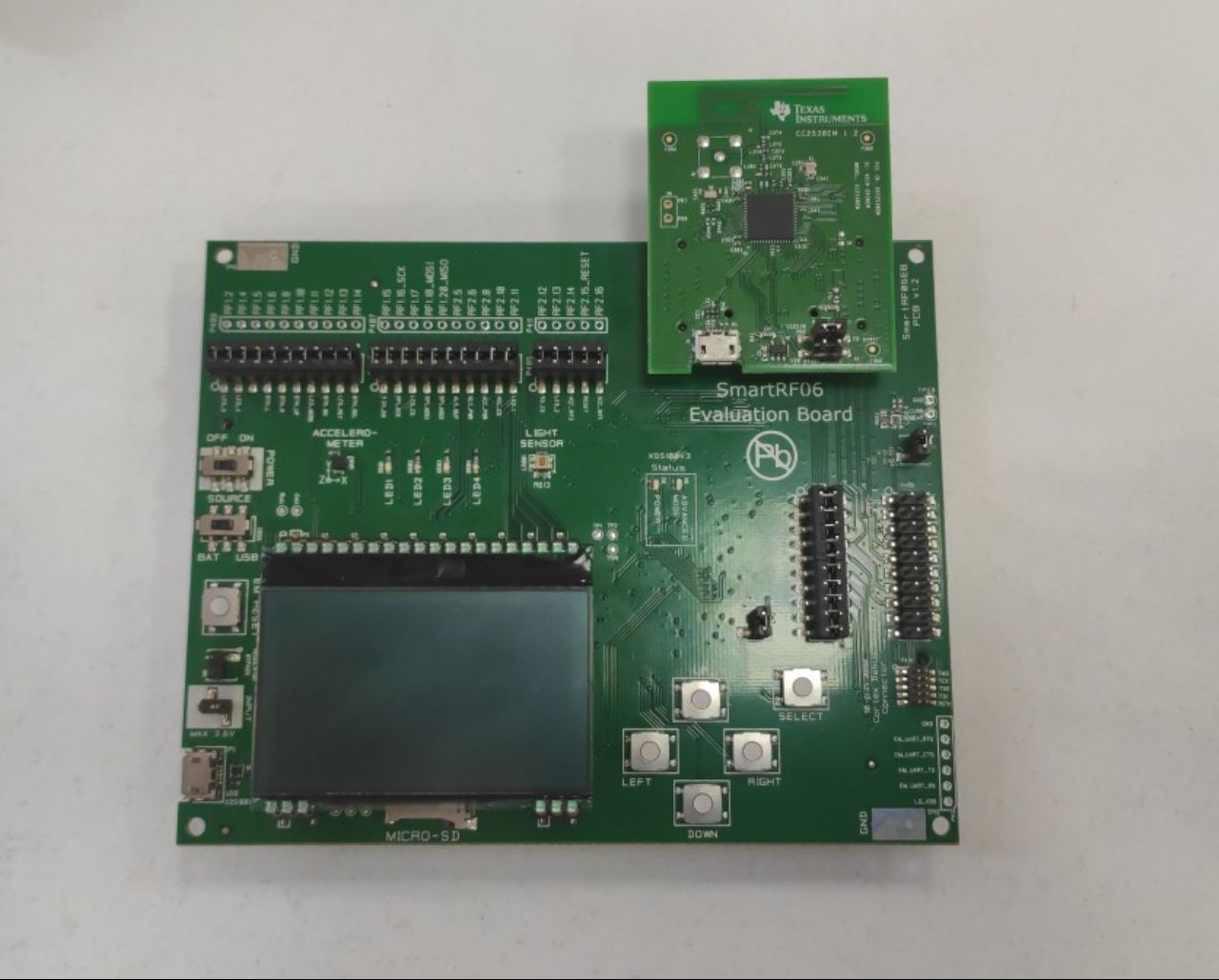


Figura 1. Microcontrolador cc2538dk con SmartRF06.

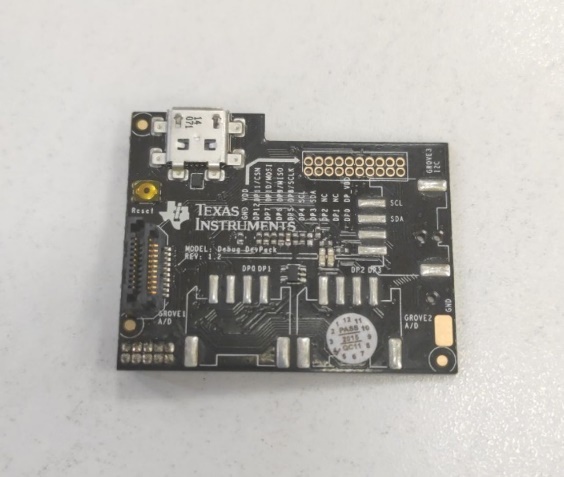


Figura 2. Microcontrolador cc2650stk (Sensor Tag).



Figura 3. Programador xds110 para Sensor Tag.

Docker está disponible para Windows y para Linux, pero en Windows necesita la versión profesional para que tenga capacidad de virtualización de forma nativa. Dado que no se posee esta versión, se ha decidido emplear Linux en su versión Ubuntu 18.04.2 LTS 64-bit desde nativo y en máquina virtual, además de la versión Ubuntu 16.04.5 LTS 64-bits.

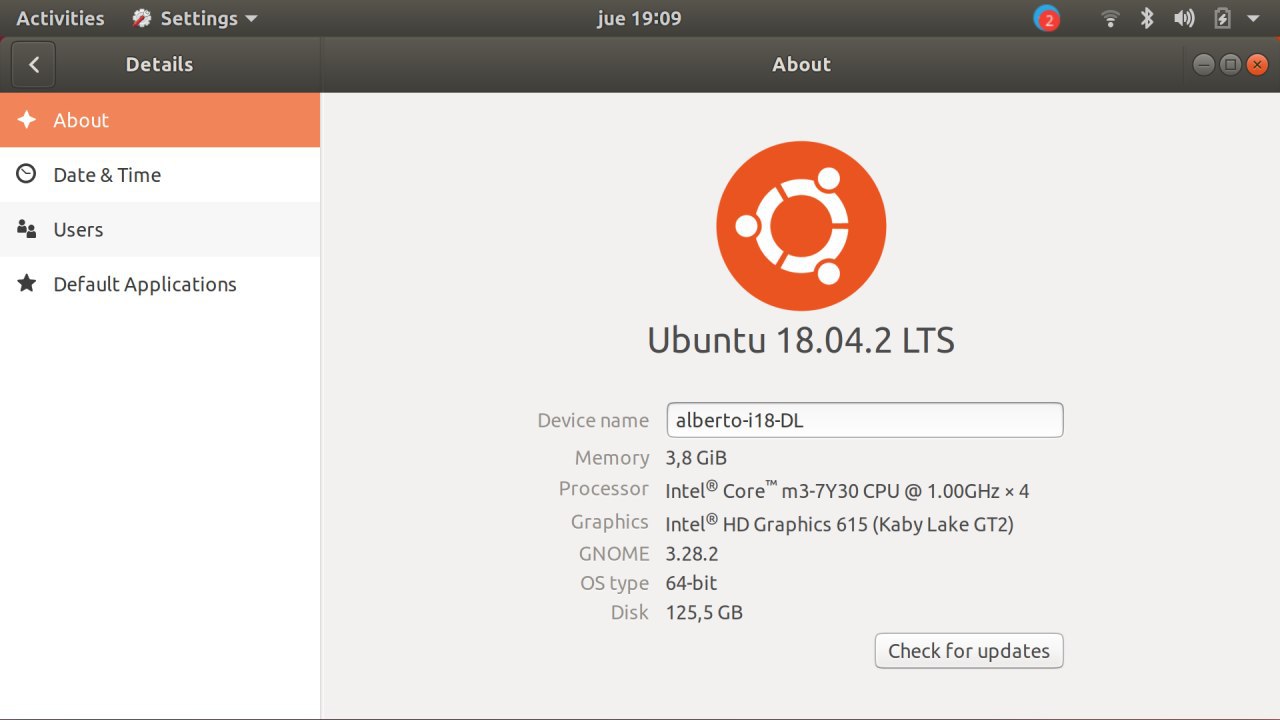


Figura 4. Detalles Ubuntu 18.04.2 en nativo.

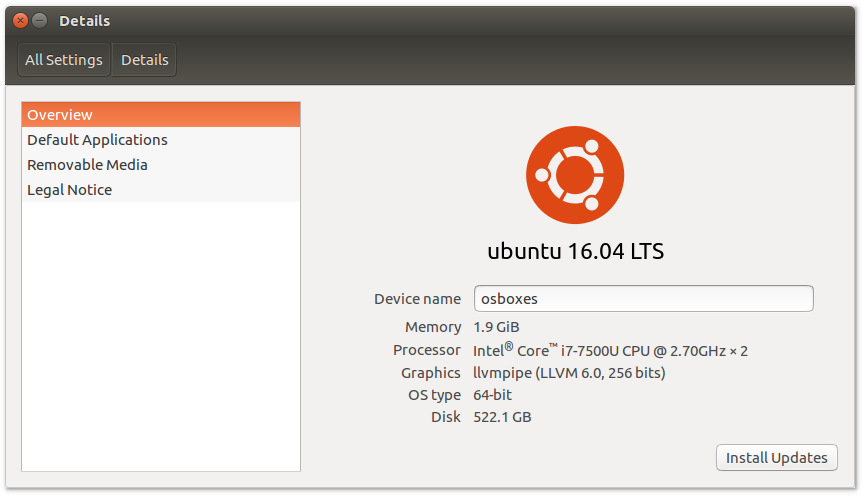


Figura 5.. Detalles Ubuntu 16.04.5 en máquina virtual.

# RIOT-OS

RIOT alimenta el Internet de las Cosas como Linux alimenta el Internet. RIOT-OS es un sistema operativo open source y gratuito desarrollado por una comunidad que reúne compañías, instituciones académicas y aficionados distribuidos por todo el mundo.

RIOT es compatible con la mayoría de los dispositivos IoT de bajo consumo y las arquitecturas de microcontroladores. RIOT tiene como objetivo implementar todos los estándares abiertos relevantes que respaldan un Internet de las Cosas conectado, seguro, duradero y amigable con la privacidad.

La semilla de RIOT fue FeuerWare, un sistema operativo que nace en 2008 para redes de sensores inalámbricos. Surge como parte del proyecto de FeuerWare donde se buscaba la monitorización de los bomberos. Para aumentar la modularidad e incluir nuevos protocolos IETF, desde el repositorio original de FeuerWare se bifurcó µkleos. El soporte para 6LoWPAN, RPL y TCP se integró en los años siguientes. En 2013 RIOT se hace público siendo el sucesor directo de µkleos, promoviendo RIOT a una comunidad mucho más grande.

Actualmente, RIOT se ejecuta en varias plataformas, incluidos dispositivos integrados y PC comunes. El código dependiente del hardware se reduce al mínimo y se abstrae del propio kernel. Esta solución diseñada pensando en IoT posee una arquitectura de microkernel y emplea estándares como 6LoWPAN que facilita su implementación.

# Git

Guit es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos en computadora y coordinar el trabajo que varias personas realizan sobre los archivos compartidos. Git modela sus datos como un conjunto de instantáneas de un sistema de archivos. Cada vez que se confirma un cambio o se guarda el estado del proyecto en Git, se hace una instantánea del aspecto de los archivos en ese momento y se guardar la referencia a dicha instantánea. Git solo almacena los archivos que se han modificado, creando un enlace al archivo anterior si el nuevo subido es idéntico. De esta forma se consigue aumentar la eficiencia del Git. Otra característica destacable de Git es su capacidad para operar de forma local, no necesita información de ningún otro ordenador de la red. Al tener la información en el disco local, la mayoría de las operaciones parecen prácticamente inmediatas.

Una vez instalado Git en nuestro dispositivo local, se crea la carpeta donde se quiere crear el repositorio. Desde dicha carpeta donde estará contenido el proyecto se ejecuta el *Git bash*.

## Conectarse a este repositorio

### Iniciar el repositorio de Git

El git init se realiza desde el terminal si se encuentra en Linux o en el bash de Git si se encuentra en Windows.

>> git init

Una vez se ejecuta este comando se crea una carpeta .git en la carpeta de trabajo.

### Conectarse al repositorio

Una vez iniciado el Git, se ejecuta la siguiente sentencia para conectarse a este repositorio. Al final de la url tiene que aparecer siempre .git.

>> git remote add origin https://github.com/blalebla/IARS.git

Se suele utilizar origin como etiqueta de carpeta de origen. Se ha conectado con el repositorio pero no se ha descargado nada todavía.

A continuación, se ejecuta el siguiente comando:

>> git fetch origin

Fetch hace una comprobación del repositorio online, enumero los objetos y las ramas presenters, en este caso solo se tiene la rama master.

### Descarga archivos del repositorio

Para descargar los archivos del repositorio se realiza un pull:

>> git pull origin master

Se ha realizado un pull de la rama master del repositorio online.

Una vez realizado lo anterior ya se tiene acceso de forma local a los archivos.

## Realizar cambios en Git

Una vez se ha descargado el repositorio en el que se desea trabajar, ya se pueden realizar modificaciones, crear archivos nuevos, eliminar algunos exitentes, crear carpetas...

### Estructura de Git

Las versiones en Git se distribuyen en 3 espacios:

* **Working Directory**: es el directorio donde se trabaja, todo lo que hay aquí se puede perder.
* **Staging**: es la zona de preparado
* **Commit**: la zona donde se guardan directamente

### Pasar de WD al área de Staging

Para pasar del working directory al área de Staging se hace uso del siguiente comando:

>> gitt add .

El punto lo que hace es agregar al área de staging todos los archivos que se han creado o modificado. También se pueden añadir los archivos uno a uno, o declarándolos de forma consecutiva tras el *add*.

>> git add <archivo1.md> <archivo2.bin>

### Commit

Para realizar un *commit* se realiza la siguiente operación:

>> git commit -m "comentario del comit"

## Realizar subida de los cambios realizados

Para subir los cambios realizados se debe realizar un *push* de la siguiente manera:

>> git push origin master

Una vez ejecutado este comando suele pedir las credenciales de Github.

# Docker

Docker es un proyecto open source para automatizar la implementación de aplicaciones como contenedores portátiles y autosuficientes que se pueden ejecutar en la nube o localmente.

Un contenedor es una unidad estándar de software que empaqueta tanto el código y como todas sus dependencias para que la aplicación se ejecute de un entorno a otro de forma rápida y fiable. Una imagen de contenedor Docker es un paquete de software ligero, independiente y ejecutable que incluye todo lo necesario para ejecutar una aplicación: código, tiempo de ejecución, herramientas del sistema, bibliotecas del sistema y configuraciones.

El software en contenedores siempre se ejecutará de la misma manera, independientemente de la infraestructura. Los contenedores aíslan el software de su entorno y garantizan que funcione de manera uniforme a pesar de las diferencias, por ejemplo, entre el desarrollo y la organización. Funcionan tanto en Windows como en Linux.

Los contenedores Docker se caracterizan principalmente porque son:

* **Estándares:** Docker creó el estándar de la industria para contenedores, por lo que podrían ser portátiles en cualquier lugar
* **Ligeros:** los contenedores comparten el kernel del sistema operativo de la máquina y, por lo tanto, no requieren un sistema operativo por aplicación, lo que aumenta la eficiencia del servidor y reduce los costos del servidor y de la licencia.
* **Seguros:** las aplicaciones son más seguras en contenedores y Docker proporciona las capacidades de aislamiento predeterminadas más sólidas de la industria.

Los contenedores de Docker surgen a partir de la tecnología LXC de Linux. Docker, además de ser un sistema de virtualización ligera para ejecutar contenedores, ofrece además la capacidad para crear y diseñar contenedores o enviar imágenes y crear versiones de estas entre otras muchas funciones.

Intentar encontrar similitudes entre Docker y una máquina virtual es un error, pues son dos tecnologías diferentes que surgen a raíz de intentar resolver diferentes problemas.

Por un lado, las máquinas virtuales nacen para plantear solución a la utilización en los servidores y fundamentalmente abstraer el hardware del sistema operativo, creando un entorno aislado para los sistemas operativos y aplicaciones. De esta manera es posible tener en un único servidor físico varias máquinas virtuales. Por otro lado, los contenedores abstraen aplicaciones del sistema operativo para hacer las aplicaciones más portables. Los contenedores Docker arrancan más rápido, ocupan menos espacio en el disco y permiten ejecutar más contenedores Docker que máquinas virtuales en un servidor. Con las máquinas virtuales pueden tenerse varios sistemas operativos en un mismo servidor, existiendo además numerosas plataformas de virtualización. Dependerá del problema que se plantea el usar una solución u otra, pues son soluciones diferentes a problemas diferentes.

# Compilar

>> make sin el nombre de la placa compila en nativo para probar si furula.

>> sudo BOARD=cc2650stk make BUILD\_IN\_DOCKER=1 DOCKER=”sudo docker”

Si se pone el comando term dentro de la sentencia, abre un terminal al compilar para ver la comunicación del terminal con la placa. En este caso no es posible al no tener disposición de puerto serie.

# Flashear

Para poder flashear los microcontroladores, esto es, almacenar en la memoria flash el programa que previamente se ha compilado, pueden emplearse dos métodos diferentes: desde el propio terminar con el comando make flash o desde una de las dos herramientas de Texas Instrument, Uniflash o Flash Programmer.

## make flash

En un primer momento se plantea el uso del comando *make flash* propio de RIOT para poder compilar.

>> make flash BOARD=cc2538dk

>> make flash BOARD=cc2650stk

Al intentar flashear cualquiera de las dos placas, se descubre que los microcontroladores de Texas Instruments precisan de su propio flasheador. El primer flasheador, Uniflash, está disponible tanto para Windows como para Linux, y el segundo, Flash Programmer solo para Windows. Una vez instalado

## Uniflash

Una vez instalado Uniflash en Linux, puede ejecutarse desde el terminal o desde el propio programa Uniflash. Es necesario estar dentro de la carpeta que contiene el archivo main.c que se quiere compilar y flashear e indicar la dirección del programa Uniflash.

Para compilar desde RIOT-OS a la vez que se flashea, se emplea la siguiente sentencia:

>> sudo BOARD=cc2650stk make BUILD\_IN\_DOCKER=1 DOCKER=”sudo docker” UNIFLASH\_PATH=/home/osboxes/ti/uniflash\_5.0.0/ flash

Los comandos contenidos entre el nombre de la placa (*BOARD=cc2650stk*) y *flash* pueden cambiarse de orden, pues son comandos independientes.

Esta sentencia solo funciona para la Sensor Tag y no para la cc2538dk, que necesita ser flasheada desde el propio programa Uniflash. Funciona tanto desde Ubuntu en nativo como en máquina virtual.

Para ambas placas, puede usarse el programa Uniflash en nativo desde Ubuntu o Windows, aunque en este último caso da error de manera ocasional. El programa detecta el microcontrolador de forma automática cuando se conecta. Una vez se ha compilado el archivo main.c y se han obtenido los archivos .bin, .elf y .hex, debe introducirse uno de estos tres para flashear la placa.

Puesto que Uniflash no funciona desde máquina virtual, se instala en Windows, que en ocasiones también falla, por lo que se recurre a otro flasheador también de Texas Instrument: Flash Programmer.

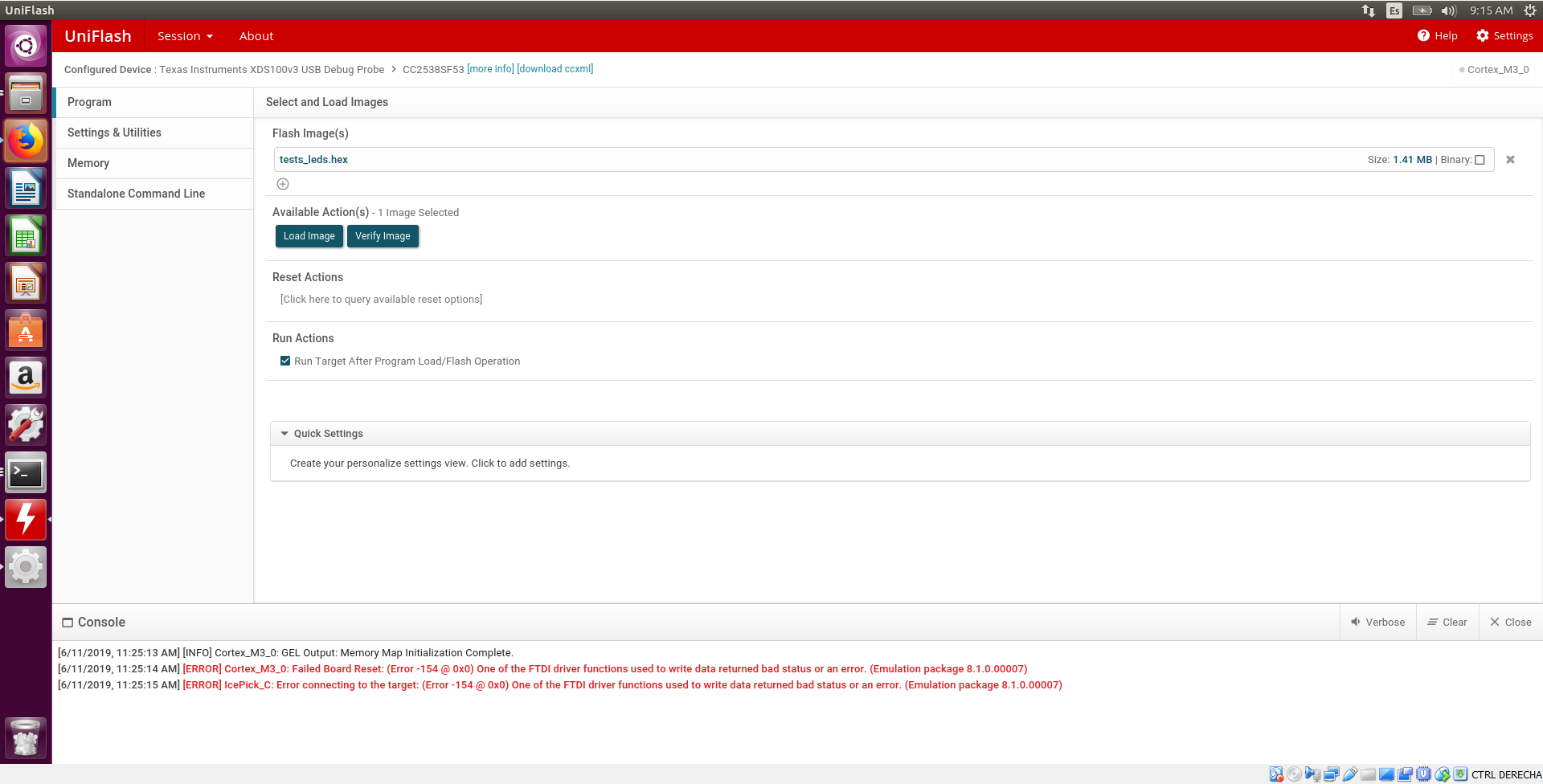


Figura 6. Uniflash en Ubuntu virtual.

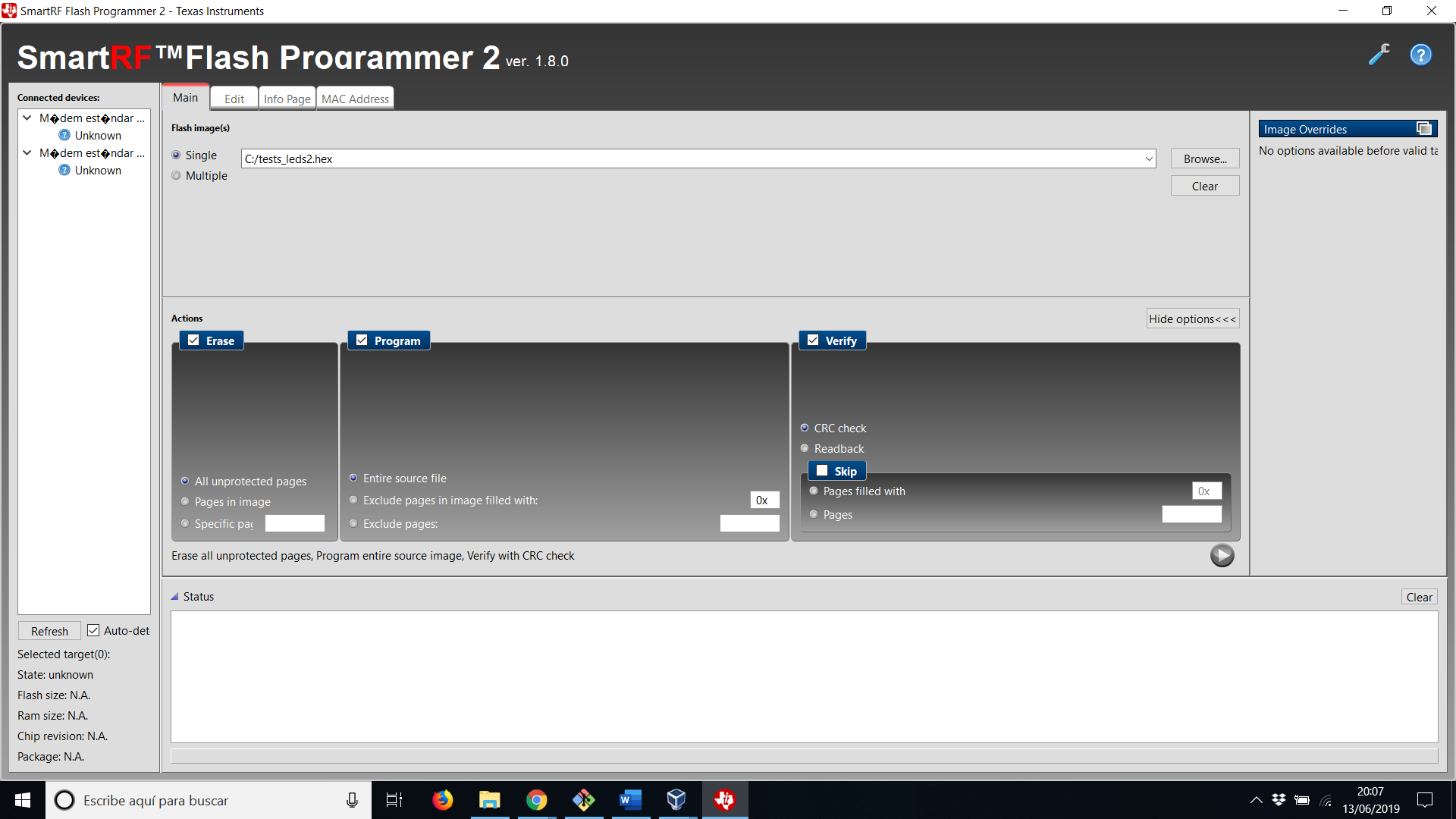


Figura 7. Flash Programmer en Windows nativo.

## Flash Programmer

Flash Programmer está disponible solo para Windows y funciona de forma similar a Uniflash. El mismo programa detecta de forma automática la placa cuando se conecta, se introduce uno de los tres archivos compilados y solo hay que dar a *play* para flashear el microcontrolador.

# Programa leds

Cc2538 NO VIENEN DEFINIDOS LOS BOTONES Y SE HA TENIDO QUE MANUAL DE USUARIO Y DE LA EXPANSION BOARD TAMPOCO

# Comunicación