Instalar debian en SD de la Beaglebone black (BBB):

Usando Rufus formatear la siguiente imagen a una microSD de 4GB:

Debian 9.5 2018-10-07 4GB SD IoT
https://www.beagleboard.org/distros/debian-9-5-2018-10-07-4gb-sd-iot

Para bootear la BBB por primera vez con la imagen de la SD, se debe mantener el switch S2 presionado cuando se alimenta la placa, y recién se debe soltar el switch cuando los leds azules comienzan a parpadear en forma aleatoria. Ya después no es necesario hacer esto, solo alimentar la placa ya bootea desde la SD.

Encontrar IP de la BBB:

1. Conectar a modem mediante Ethernet
2. Abrir Advanced IP Scanner, y ejecutar en modo portable
3. Escanear, y buscar la IP correspondiente al dispositivo con la dirección MAC “E0:FF:F1:D8:A3:ED”, o sino tiene que decir en fabricante “Texas Instruments”

En mi modem es el 192.168.123.103

Conexión a la BBB por SSH (Ethernet en mismo modem):

1. Descargar en Visual Studio Code la extensión “Remote - SSH”
2. Conectar por Ethernet la BBB a la pc
3. Alimentar la BBB por USB
4. Escanear la red para averiguar la dirección IP de la BBB (debe ser algo como 192.168.0.8)
5. Crear el archivo de config de ssh en:

C;/Users/xxx/.ssh/config	//windows
/home/xxx/.ssh/config	//linux

Contenido de config:

```
Host bbb
     HostName 192.168.123.103
     User debian
     IdentityFile ~/.ssh/bbb_rsa
```

6. En VSC, abrir una terminal y ejecutar:

```
ssh debian@192.168.123.103
```

debian	//user por default
temppwd	//password por default

7. Debemos dejar el servidor y el driver dentro de la siguiente ruta:

```
/home/debian/tp-td3-fcc/
```

Contenido:

```
device-tree/
platform-driver/i2c_mpu6050_driver/
server/
     inc/
     src/
     Makefile
```

Conexión a la BBB por SSH (por USB compartiendo internet a la BBB):

sources:
https://fastbitlab.com/how-to-enable-internet-over-usb/
https://sge.frba.utn.edu.ar/wiki/td3/doku.php?id=bbx#internet_en_beaglebone_a_traves_de_la_pc

- **En la BBB (target):**

1. crear en /home/debian/tp-ted3-fcc/ el siguiente archivo ¨add_route_script.sh¨ con el siguiente contenido:

```
sudo ifconfig usb0 192.168.7.2
sudo route add default gateway 192.168.7.2
```

le otorgamos permisos de ejecución mediante:

```
chmod +x ~/add_route.sh
```

agregamos la ejecución automática de este script al inicio del sistema:

```
sudo crontab -e
```

Entonces agregamos la siguiente línea al final del archivo:

```
@reboot /home/debian/add_route_script.sh
```

2. Ejecutar el siguiente comando para editar resolv.conf:

```
sudo vi /etc/resolv.conf
```

Luego tocamos la tecla "i" para entrar en modo INSERT y agregamos lo siguiente:

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Tocamos ESC para salir del modo INSERT, y tipeamos :w para guarda, y luego :quit para regresar

3. Ejecutar el siguiente comando para editar interfaces:

```
sudo vi /etc/network/interfaces
```

Luego tocamos la tecla "i" para entrar en modo INSERT y agregamos al final de todo y con la misma identación lo siguiente:

```
dns-nameservers 8.8.8.8
dns-nameservers 8.8.4.4
```

Tocamos ESC para salir del modo INSERT, y tipeamos :w para guarda, y luego :quit para regresar

- **En la PC (Host):**

Crear el archivo nettobbusb.sh que se deberá ejecutar manualmente en cada inicio de la pc:

```
sudo iptables --table nat --append POSTROUTING --out-interface enp0s3 -j MASQUERADE
sudo iptables --append FORWARD --in-interface 1d6b:0001 -j ACCEPT
sudo echo 1 > /proc/sys/net/ipv4/ip_forward
```

en mi caso "enp0s3" es el ethernet que corresponde al dispositivo que me provee conectividad a internet
y "1d6b:0001" corresponde al dispositivo USB asociado a la BBB

podemos averiguar para nuestro equipo el nombre de estos dispositivos mediante alguno de estos comandos:

```
ifconfig
lsusb
```

De esta forma ahora nuestro archivo en la PC ¨/home/xxx/.ssh/config" ahora deberá ser:

```
Host bbb
  HostName 192.168.7.2
  User debian
  IdentityFile ~/.ssh/bbb_rsa
```

Agregar linux-headers para poder compilar:

se supone que solo con la siguiente linea deberia ser suficiente:

```
sudo apt install --reinstall linux-headers-$(uname -r)
```

- Pero en caso de tener error: *Unable to locate package linux-headers-xxxx*

primero hay que corregir el siguiente archivo

```
sudo vi /etc/apt/sources.list
```

Buscamos todas las líneas que tengan "deb.debian.org" y las modificamos para que sean "archive.debian.org". Esto ocurre porque la imagen de debian que usé era vieja.

Comandos que tuve que hacer después:

```
sudo apt-get update
sudo apt upgrade --fix-missing
sudo apt-get install apt-transport-https
sudo apt-get update
sudo apt install --reinstall linux-headers-$(uname -r)
```

En mi caso los instaló en "/usr/src/linux-headers-4.14.71-ti-r80", así que tuve que ajustar a donde apunta el Makefile cuando va a buildear el módulo

---

Cambiar el device-tree de la BBB para que use nuestro driver:

Device tree para la BBB ya viene con todo configurado para esa placa/hardware, nosotros vamos a agregar nuestro driver, luego vamos a recompilar este device tree, de esta forma Linux va a saber que existe un nuevo dispositivo conectado.

Al conectarnos por SSH a la BBB vamos a encontrar los directorios de:

```
platform-driver
device-tree
```

dtb (Device Tree Blob o Binary) -> Lo debemos convertir a dts (Device Tree Source) ubicado en:

/boot/dtbs/<kernel_version>
<kernel_version> = 4.14.71-ti-r80

am335x-boneblack.dtb	// archivo que nos interesa

Para ver cual es el que se usa en el linux del BBB, vamos a /boot hacemos:

```
nano uEnv.txt
```

//Otra forma de saber de cual usa es con pararnos en boot/dtbs/4.14.71-ti-r80/ y ejecutar:

```
uname -a
```

este archivo .dtb primero lo vamos a copiar a nuestra PC, para luego decompilar, modificar, volver a compilar, y subirlo de nuevo a la BBB

lo copiamos con:

```
sudo cp am335x-boneblack.dtb /home/debian/tp-td3-fcc/device-tree
```

nos paramos donde dejamos la copia, y con el comando dtc (Device Tree Compiler) para decompilar este archivo binario extraído:
(previo instalar: sudo apt install device-tree-compiler)
(NO CAMBIAR LAS MAYUS/minus de estas lineas)

```
dtc -I dtb -O dts -o <nombre-archivo-destino>.dts <nombre-archivo-origen>.dtb
```

```
dtc -I dtb -O dts -o am335x-boneblack-td3.dts am335x-boneblack.dtb
```

Abrimos este archivo .dts y modificamos la línea "compatible" agregando nuestro driver ->

```
/dts-vq/;

/ {
    compatible = “ti,am335x-bone-black”, “ti,am335x-bone”, “ti,am33xx”, “td3_i2c_fcc”;
    ...
```

en la sección "aliases" podemos observar el mapa de memoria de los I2C:

```
/dts-vq/;

/ {
     ...
     aliases {
          i2c0 = “/ocp/i2c@44e0b000”,
          i2c1 = “/ocp/i2c@4802a000”,
          i2c2 = “/ocp/i2c@4819c000”,         //Este es el canal de ic2 que voy a usar
          ...
     };
...
```

luego más abajo se debe modificar donde se describe el canal seleccionado del ic2, y le cambiamos el nombre/alias y el compatible:

```
td3_i2c@4819c000 {
     compatible = “td3_i2c_fcc”;
     ...
}
```

En el bloque anterior también se pueden realizar otras configuraciones de esta descripción de hardware

Una vez terminados los cambios, creamos el blob con:
(PRIMERO MOVER EL ORIGINAL PARA NO PISARLO CON LA VERSION MODIFICADA)

```
dtc -I dts -O dtb -o am335x-boneblack.dtb am335x-boneblack-td3.dts
```

Si el comando sale bien, entonces el archivo dtb generado cuando se abra es de tipo binario

y lo volvemos a copiar el device tree con:

```
sudo cp "am335x-boneblack.dtb" /boot/dtbs/4.14.71-ti-r80/
```

Ahora para cambiar el device tree del booteo por este modificado vamos a /boot y hacemos:

```
sudo nano uEnv.txt
```

y para que apunte a nuestro nuevo dtb, descomentamos y le cambiamos estas líneas por:

```
uname_r=4.14.71-ti-r80
dtb=am335x-boneblack.dtb
```

guardamos con Ctrl+O y luego Enter.
salimos con Ctrl+X

reiniciar:

```
sudo reboot
```

al reiniciar, podemos verificar si quedó el driver asociado al periférico abriendo un terminal:

ir a:	*/proc/device-tree/ocp/td3_i2c@4819c000*		
ejecutar:	*cat compatible*	se obtiene:	*td3_i2c_fcc*
ejecutar:	*cat name*	se obtiene:	*td3_i2c*

Instalar el driver:

copiar todo el contenido de 04_driver a la bbb en:

```
home/debian/tp-td3-fcc
```

Entrar a tp-td3-fcc/platform-driver, y para loguear en tiempo real la instalación del módulo y su posterior ejecución de los métodos del driver, en un nueva terminal se deja ejecutando:

```
make tail
```

Desde otra terminal, ejecutar el siguiente make de manera tal que se va a compilar y re-instalar nuestro módulo del driver en el kernel:

```
make all
```

Para verificar si la instalación fue correcta, buscar a ¨td3_i2c_fcc" en "/dev/" y sino ejecutar:

```
make cat
```

Para desinstalar este módulo se hace

```
make rmmod
```

O para instalar el módulo a partir del .ko generado anteriormente con make all:

```
make insmod
```

Para hacer un test de open() del driver utilizar el test:

```
make test
```

Instalar el server:

copiar todo el contenido de **02_cuat** a la bbb en la ruta:

```
home/debian/tp-td3-fcc
```

Para que el server utilice como productor de datos el I2C2 verificar que config.ini contiene:

```
producer_mode = 1
```

Entrar a tp-td3-fcc desde una terminal y ejecutar el server con:

```
make do_server
```

Para que el server utilice como productor de datos el random generator verificar que config.ini contiene:

```
producer_mode = 0
```

Para observar en tiempo real un administrador de tareas en terminal ejecutar:

```
htop
```

Para ver los ipcs en uso hacer

```
ipc
```

Ejecutar clientes que se conecten al server:

Se pueden ejecutar clientes directamente en una terminal de la BBB con:

```
make do_client
```

Esto hace que por default se use a localhost como IP

Si se quiere conectar clientes desde la PC al server corriendo en la BBB, en la BBB ejecutar:

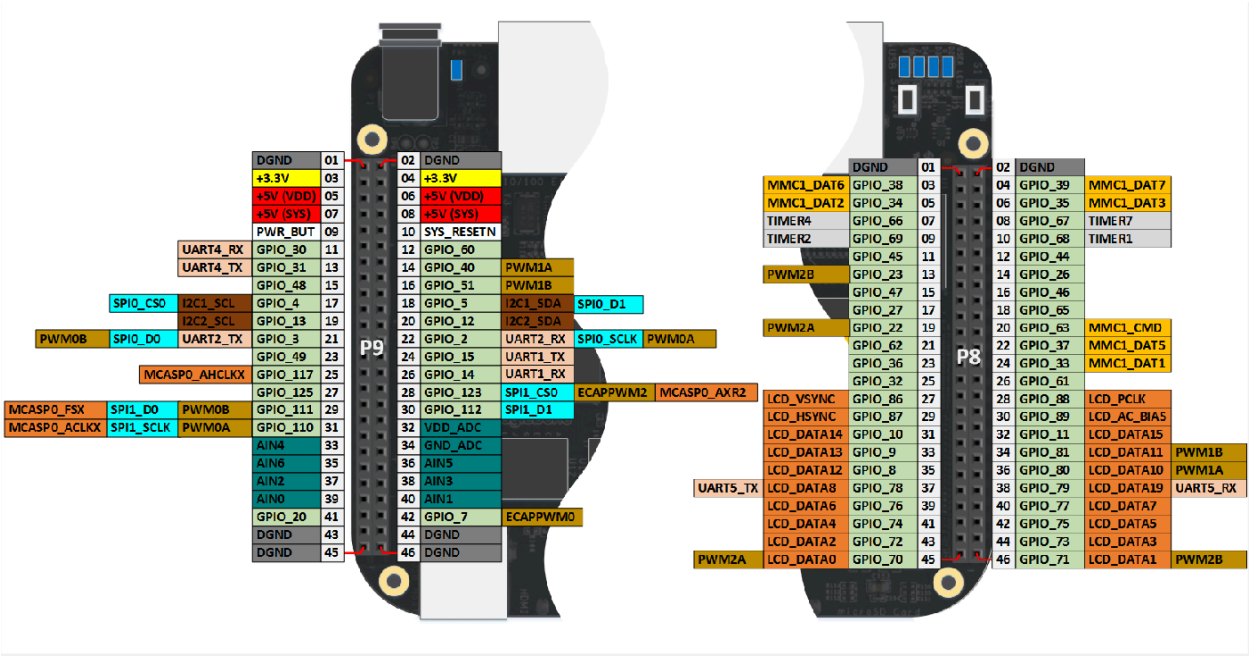
```
ifconfig
```

Esto nos va a listar dispositivos, y uno de estos va a ser un USB que dice algo como "inet 192.168.7.2"

Entonces ahora desde la Notebook ejecutamos los clientes con:

```
make IP=192.168.7.2 do_client
```

Conexión de módulo MPU6050 a BBB:



BBB	MPU6050
P9 → Pin 02 = **DGND**	**GND**
P9 → Pin 06 = **+5V (SYS)**	**VCC**
P9 → Pin 19 = **I2C2_SCL**	**SCL**
P9 → Pin 20 = **I2C2_SDA**	**SDA**
others	**non connected**

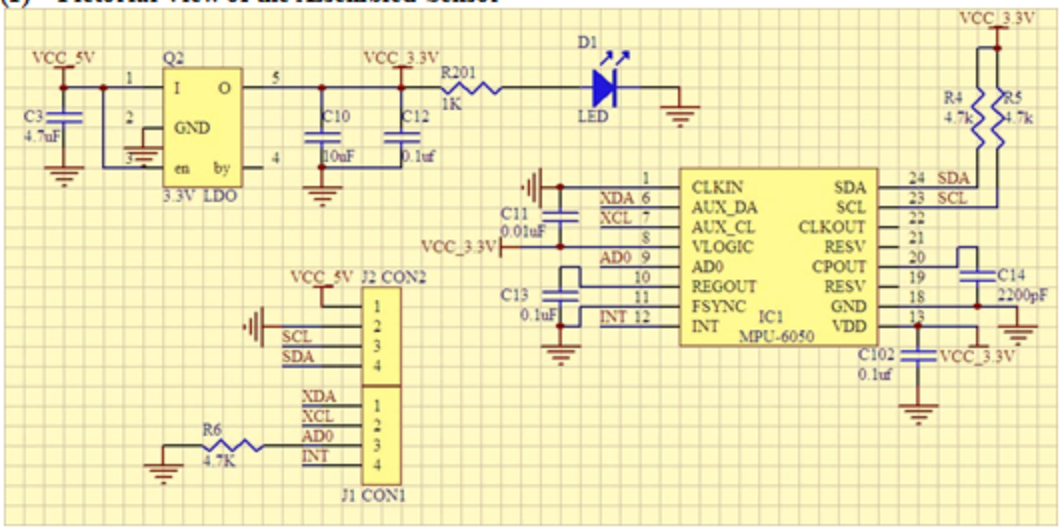
Schematic y Pinout del módulo MPU6050:

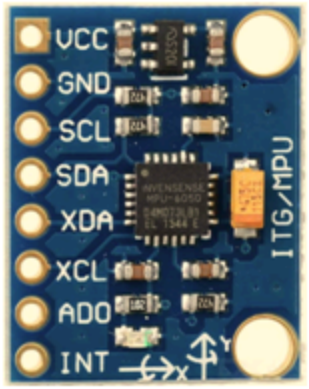
Experiment – 13

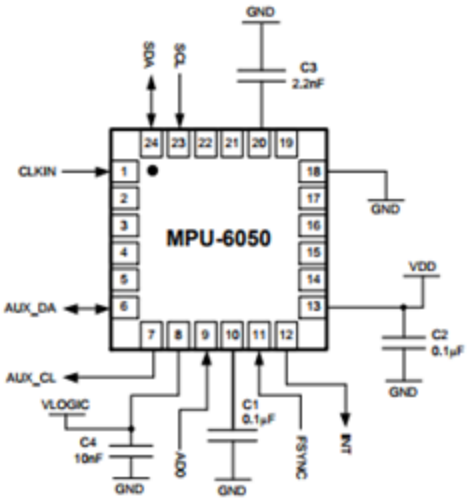
TWI Bus Interfacing between UNO and MPU6050 Sensor
Acquisition of Temperature Signal, Accelerometer Signal and Gyro Signal

Task-13.1 Introduction

(1) Pictorial View of the Assembled Sensor







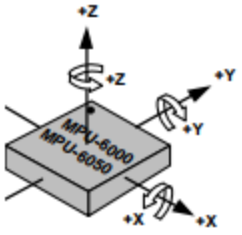


Figure-13.1: Pictorial view of the assembled sensor

In Arduino, the deviceAddress has been keyed as 0x68 in the **MPU6050.h** and **MPU6050.cpp** files. Therefore, the user must connect the AD0-signal at GND.

Dificultades adicionales:

- Se me generó el dtb como formato texto en vez de binario y no me estaba dando cuenta
- Cambiaba uEnv y sin embargo me seguía levantando otro device-tree
- Tuve que mover de lugar todos los dtbs de /boot/dtbs/4.14.71-ti-r80/ para que levante el mio
- El device-tree lo armé siguiendo el video de youtube para el canal 1 del I2C y resulta que el driver lo estaba armando para el canal 2 del I2C y entonces no me levantaba el driver
- Error de null pointer exception al querer hacer un test (que ejecutaba FCCopen) -> Despues logré determinar que era porque si bien parecia que se instalaba correctamente, no se estaba ejecutando el probe durante la instalacion asi que el test fallaba y la instalacion tambien pero no lo estaba capturando correctamente.
- Para armar el server al principio encaré utilizando un patron de diseño de threadpool que fue la sugerencia de un amigo que la curso el año pasado, pero no solo no me estaba funcionando sino que tambien me estaba complicando al pedo asi que tuve que rehacer el server una vez resuelto el driver
- No Space left on device: No me corria el driver ni me compilaba por este error, asi que tuve que revisar con ">>df -h" y tenia la memoria de 4GB al 98%, asi que vaciando logs llegue a 94%, luego borre caches y datos temporales de .vscode server, y caches de instalaciones de var/cache y ahi llegue al 82%.