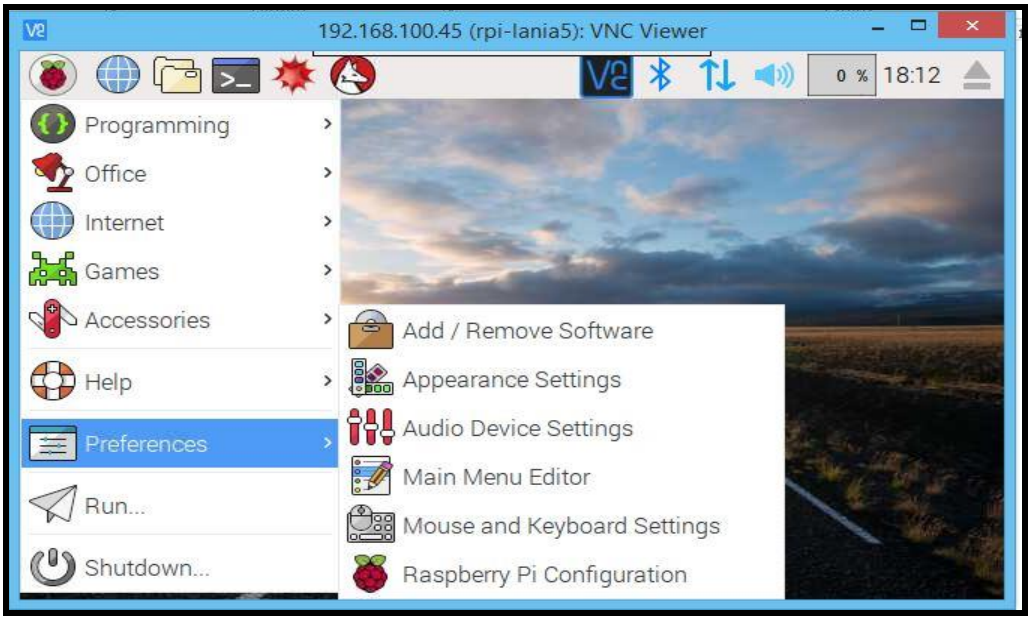
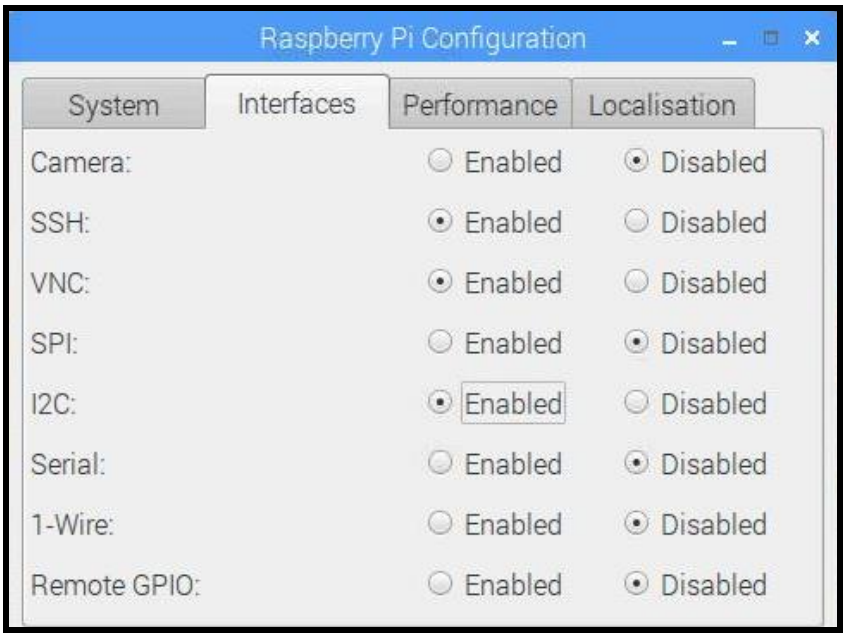
Obtener lectura del sensor BH1750 utilizando RaspBerry.

Para poder hacer uso del sensor BH1750 hay que realizar configuraciones que no se tiene en la RaspBerry.

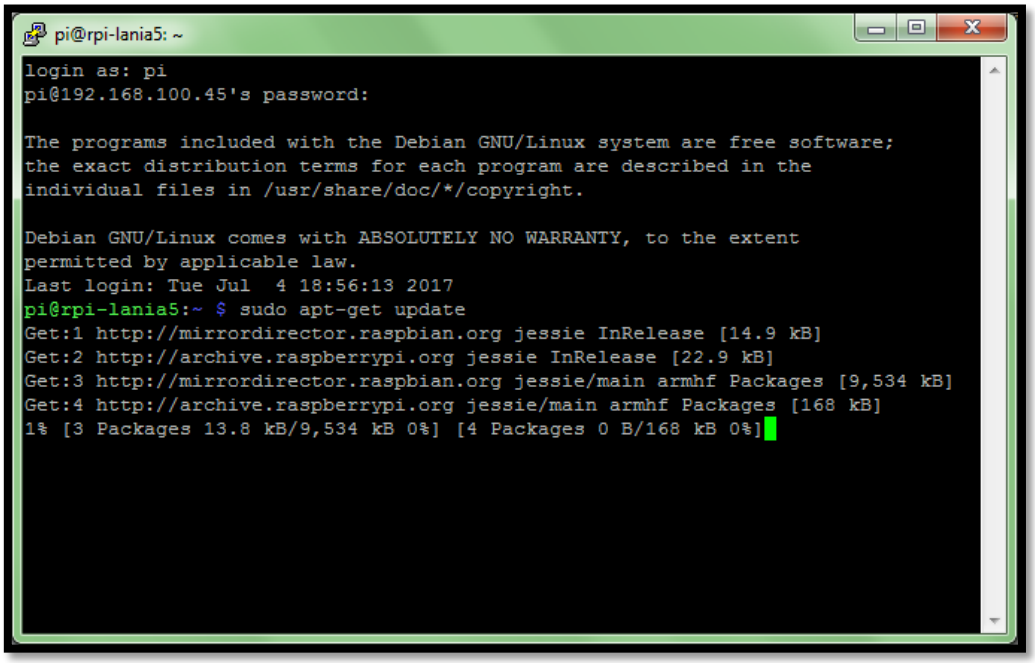
- Hay que configurar el módulo de I2C, lo cual se realizó de manera gráfica.
- Selecciona la opcion de ***Preferences*** y despúes la opcion de ***Raspberry Pi Configuracion***; como se muestra en la siguiente imagen.

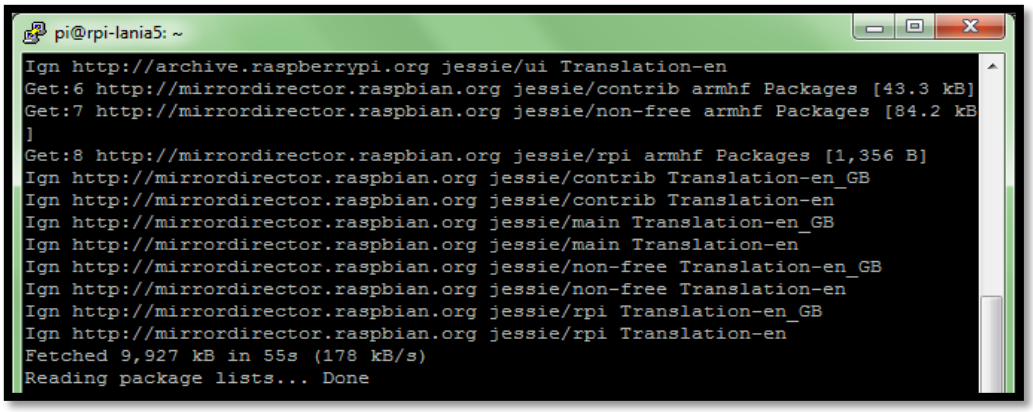


- Luego se habilita la opción de ***Enabled*** de ***I2C***; como se muestra en la imagen.



- Luego se realiza la instalación de utilidades.
- El primer comando que se utiliza es sudo ***apt-get update*** para una actualización y hay que esperar unos minutos mientras se realiza el proceso; como se muestra en las siguientes dos imágenes.

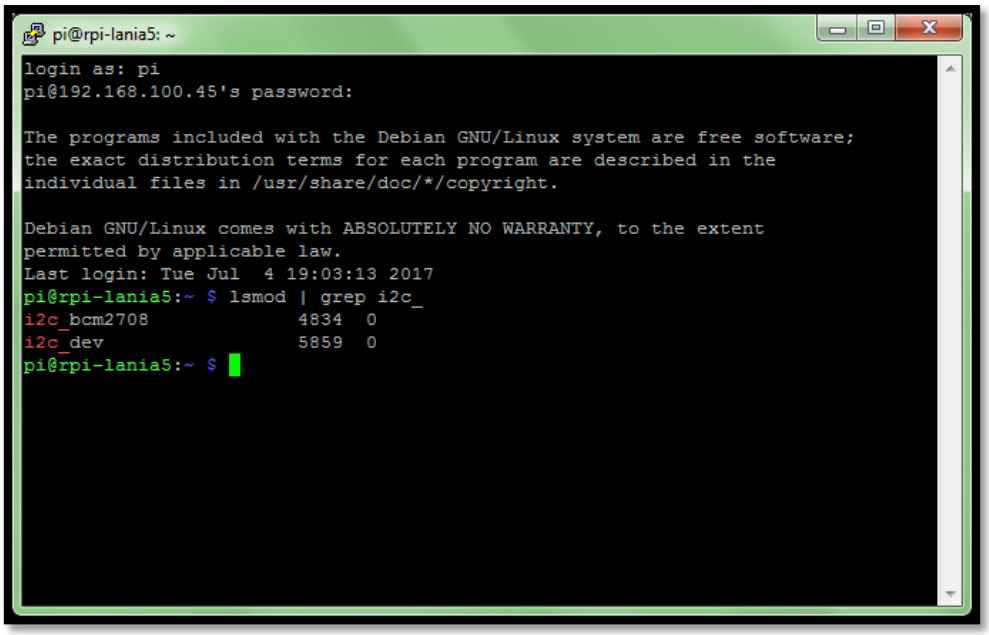




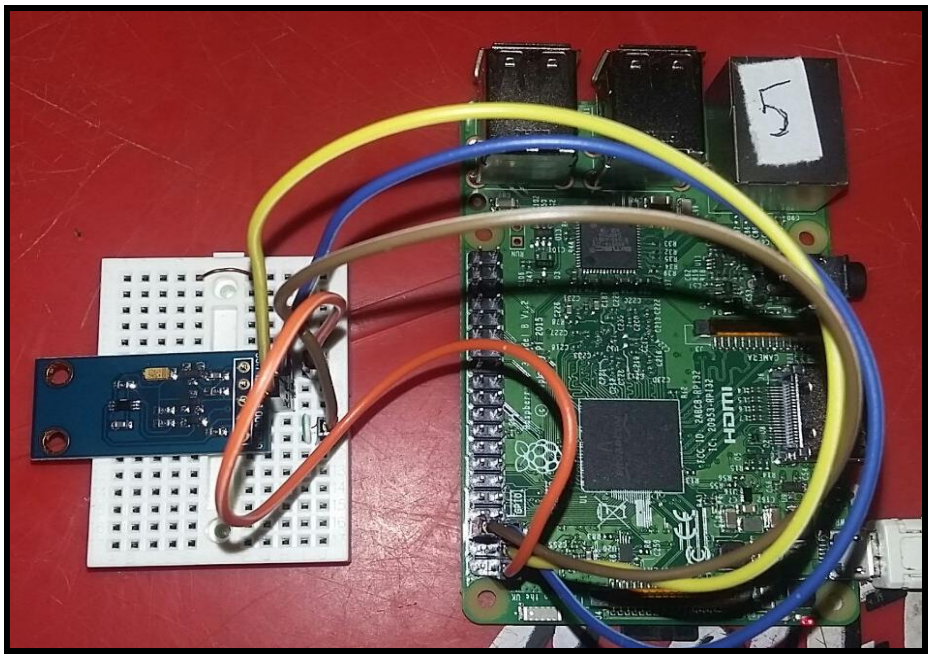
- Luego se instala las herramientas necesarias de I2C con el siguiente comando ***sudo apt-get install –y Python-smbus i2c-tools.***
- Una vez realizado los pasos anteriores se procede a reiniciar la RaspBerry.

```
pi@rpi-lania5:~ $ sudo apt-get install -y python-smbus i2c-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
i2c-tools is already the newest version.
i2c-tools set to manually installed.
python-smbus is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 103 not upgraded.
pi@rpi-lania5:~ $ sudo halt
```

- Una vez reinaciada la RaspBerry se utiliza el siguiente comando ***lsmod | grep i2c_*** para verificar que se instalaron los módulos; como se muestra en la siguiente imagen.



- Imagen del sensor conectado a la RaspBerry, se recomienda soldar el sensor a sus pines ya que si no esta puede que se pierda la comunicación porque se puede mover.



- Código utilizado para obtener dato del sensor, el código esta realizado en Python.

```
import smbus

import time

# Define some constants from the datasheet

DEVICE     = 0x23 # Default device I2C address

POWER_DOWN = 0x00 # No active state

POWER_ON   = 0x01 # Power on

RESET      = 0x07 # Reset data register value

# Start measurement at 4lx resolution. Time typically 16ms.

CONTINUOUS_LOW_RES_MODE = 0x13

# Start measurement at 1lx resolution. Time typically 120ms
```

```
CONTINUOUS_HIGH_RES_MODE_1 = 0x10

# Start measurement at 0.5lx resolution. Time typically 120ms

CONTINUOUS_HIGH_RES_MODE_2 = 0x11

# Start measurement at 1lx resolution. Time typically 120ms

# Device is automatically set to Power Down after measurement.

ONE_TIME_HIGH_RES_MODE_1 = 0x20

# Start measurement at 0.5lx resolution. Time typically 120ms

# Device is automatically set to Power Down after measurement.

ONE_TIME_HIGH_RES_MODE_2 = 0x21

# Start measurement at 1lx resolution. Time typically 120ms

# Device is automatically set to Power Down after measurement.

ONE_TIME_LOW_RES_MODE = 0x23

#bus = smbus.SMBus(0) # Rev 1 Pi uses 0

bus = smbus.SMBus(1)  # Rev 2 Pi uses 1

def convertToNumber(data):

  # Simple function to convert 2 bytes of data

  # into a decimal number

  return ((data[1] + (256 * data[0])) / 1.2)

def readLight(addr=DEVICE):

  data = bus.read_i2c_block_data(addr,ONE_TIME_HIGH_RES_MODE_1)

  return convertToNumber(data)

def main():

  while True:
```

```
    print "Light Level : " + str(readLight()) + " lx"
    time.sleep(0.5)
main()
```

- Los resultado que dio al momento de correr el código se muestra en la siguiente imagen, para correr el código se ejecutó el siguiente comando ***Python nombre_archivo.py***