## TODO: Infineon DPS310

The Infineon DPS310 Pressure Sensor kit is a small, battery power combination pressure sensor, temperature sensor and altimeter mounted onto the IFX\_NANOHUB Bluetooth-enabled carrier board.

The basic sensor can be supported with a generic Rfcomm Bluetooth interface, but it’s much easier to use the DPS310 specialization. The specialization takes in up to three functions, one each for the altitude, pressure and temperature readings. The altitude is in nominal meters above sea level, the pressure is in mBar and the temperature is in degrees Celsius.

In the example, a raw Rfcomm (serial port) device is picked. The DPS310 is mounted on an IFX\_NANOHUB and shows up in Bluetooth with that name. Then the dps310 is created as a specialization of the raw Rfcomm device. Note that there’s no good way for Best Calculator to know which device has which specialization for Rfcomm devices. You just have to write the code correctly.

When the specialization is made, three function names are passed in, one each for the altitude, pressure and temperature data. You can also just pass in a blank string (“”) for any function that you don’t need a callback for.

CLS BLUE

PRINT “Altitude from the DPS310”

bt = Bluetooth.PickDevicesRfcommName ↲

(“IFX\_NANOHUB”)

dps310 = bt.As (“DPS310”, ↲  
 “Dps310Altitude”, “Dps310Pressure”, ↲  
 “Dps310Temp”)

FOREVER

FUNCTION Dps310Altitude (dps310, value)

Screen.ClearLine (4)

PRINT “Altitude”, value

END

FUNCTION Dps310Pressure (dps310, value)

Screen.ClearLine (5)

PRINT “Pressure”, value

END

FUNCTION Dps310Temp(dps310, value)

Screen.ClearLine (6)

PRINT “Temp”, value

END

## C:\Users\toomr\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Littlebot.pngTODO: Slant Robotics LittleBot

The LittleBot from Slant Robotics is a small, cute, Arduino-based robot kit released in 2017. The brains of the robot are a small Arduino-based controller that can be controlled with a built-in Bluetooth serial port.

Before you can use the LittleBot you need to download the Walteros ([walteros\_v1.ino](http://www.littlearmrobot.com/uploads/3/8/4/8/38484491/walteros_0_1.ino)) sketch for the robot from their [downloads](http://www.littlearmrobot.com/downloads.html) page. Install it using an Arduino IDE (Integrated Development Environment). When I programmed mine, I had to remove the Bluetooth board for the USB serial connection to work correctly. I also had to manually set the Arduino type (it’s an “Arduino Nano”)

The Bluetooth connection is “HC-06”

## Rfcomm (Serial-port) Bluetooth

Some Bluetooth devices (like the Infineon IFX\_NANOHU  
B and the Slant Robotics devices) work entirely using the Bluetooth Rfcomm mechanism. This is also called the Serial Port Profile (SPP).

The Rfcomm fundamental communication is that you can send data (often a string) to the device at any time, and it can send you information at any time. Often the data looks like it’s individual packets – for example, you might send a command as one string and get a string in return. However, the underlying protocol is not one-for-one.

The little example sends a command to an Rfcomm device and prints the results.

bt = Bluetooth.PickDevicesRfcommName (“\*”)  
bt.ReceiveString (“BtRecv”)  
bt.Send (“$hello\n”)  
FOREVER  
  
FUNCTION BtRecv (device, data)  
 PRINT “RECV:”, data  
END

There are device.As (“name”) specializations for selected Rfcomm devices like the IFX\_NANOHUB and the Slant Robitics devices.

## Release Notes (3.10, September 2017)

The Rectangle object (from g.Rectangle()) now includes a Data member. This is for your use; it will never be examined or changed by BC BASIC. It’s for when you need to save a little bit of information about a graphics object.

The Bluetooth device list methods (Devices, DevicesName and PickDevicesName) all have an Rfcomm version (DevicesRfcommm, DevicesRfcommName, PickDevicesRfcommName). The Rfcomm version will list Bluetooth devices that support the RF COMM (Serial port) protocols. Many Arduino devices, for example, use serial-port Bluetooth chips for communicating with computers.

TODO: camera = Sensor.Camera()  
camera.Viewport (x, y, h, w) sets the portion of the camera to display; default is all of it  
camera.AnalyzePort(x, y, h, w) sets the portion of the camera to analyze. This should generally be small (50px by 50px or less)  
camera.Analyze(“<type>”, “<callback>”) where type can be any of these:  
RGB: will calculate the average color of the area. The callback will get the camera and a color value from which you can pull out the r, g and b values. You can also specify just the R, G or B values and get the individual colors. You can also ask the color object for the HLS value. The color value can be passed into a graphics command.   
  
g = Screen.Graphics (100, 100, 100, 100)  
camera = Sensor.Camera()  
view = g.Camera (camera, 0, 0, 100, 100)   
REM these are x1, y1, x2, y2 coordinates  
colordup = Screen.Graphics (100, 300, 100, 100)  
camera.Analyze (“RGB”, “SetColor”)  
  
FOREVER  
  
FUNCTION SetColor (camera, color)  
 GLOBAL colordup  
 colordup.Fill (color)  
 Screen.ClearLine (1)  
 PRINT color.R, color.G, color.B  
END   
graphics.SetCamera (camera) will display the camera in the graphics area

Manual Fixes: section 38.2.5 automatic update. The update also happens during the FOREVER command.