

A comprehensive guide to programming & flashing the R-IoT wifi sensor module

The R-IoT module is based upon the CC3200 chip from Texas Instrument. Its core feature is to be compatible with Energia, a branch of the Arduino IDE that allows programming TI processors with the easiness of the Arduino look & feel.

<http://energia.nu/>

Just like the Arduino system for Atmel microcontrollers, Energia is essentially a code warper on top of the C API from Texas Instrument to program the core processor though R-IoT can still be programmed with TI tools chain & Code Composer.

The CC3200 is actually the combination of two ARM processors, one taking care of the WiFi modem and network stack, the second being the application processor running the user code.

The CC3200 embeds a bootloader triggered upon reset / power on by an external momentary switch. The firmware is uploaded to the chip using a simple UART available as a USB serial port.

The R-IoT exports only some of the available I/O's from the CC3200 for matters of size / form factor. While not exactly a hacker friendly board, it remains small enough to be installed anywhere, on the body, on an instrument or in an object. With the additional I/Os which include 2 analog inputs and 1 PWM output, it can be adapted to various contexts.

The i2c bus is also exported allowing a complete chain of accessories to be talked to, from LED controllers to additional sensors or I/O extenders.

The R-IoT also embeds a 9 axis digital sensor featuring a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer, allowing for instance the onboard computation of the absolute orientation of the module in space. The sensor is attached to the SPI port to grab the 16 bit motion data at high speed.

The sensors data are locally processed, analyzed and streamed over WiFi (UDP, Open Sound Control) by the module using its firmware, a program compiled using the Energia tool chain.

R-IoT code repository

Several code examples dedicated to the R-IoT platform are available on Ircam's public GIT repository
<https://github.com/ircam-rnd>

The repository contains the main firmware, which achieves several analysis on the sensor data and allow the configuration of the module (IP address, UDP port, module ID) via a web server or the USB serial port. Other simpler examples show how to write dedicated code for the R-IoT platform for specific motion analysis for instance.

Being a development platform, the R-IoT module role isn't bound to motion analysis or sensor data streaming. Using the additional I/Os and analog input and the WiFi modem, the unit can be turned into an efficient Internet Of Things (IoT) object, a miniature web server, a car alarm or a plant watering system.

Understanding the physical platform

We have developed the R-IoT platform using initially the Texas Instrument CC3200 launchpad, a development board for which Energia has standardized and named / numbered all the I/Os based on the physical header and pinhead connectors of the dev board.

http://energia.nu/pin-maps/guide_cc3200launchpad/

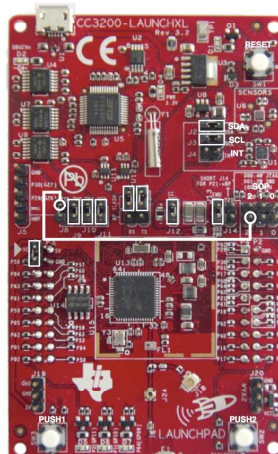


SRAM 256 KB
Flash 1 MB
Serial hardware
Analog input < 1.46V
Storage in Flash, execution in SRAM

			P1	P3		
		+3.3V	1	21		VBUS
			2	22		GROUND
RX			3	23		
TX	INT	PUSH1	4	24		TX (0)
			5	25		
			6	26		
			7	27		
SCK			8	28		
TX (1)	SCL	YELLOW_LED	9	29		RED_LED
RX (1)	SDA	GREEN_LED	10	30		
						GROUND
						GROUND
						VBUS

LaunchPad with CC3200

Revision 3.2



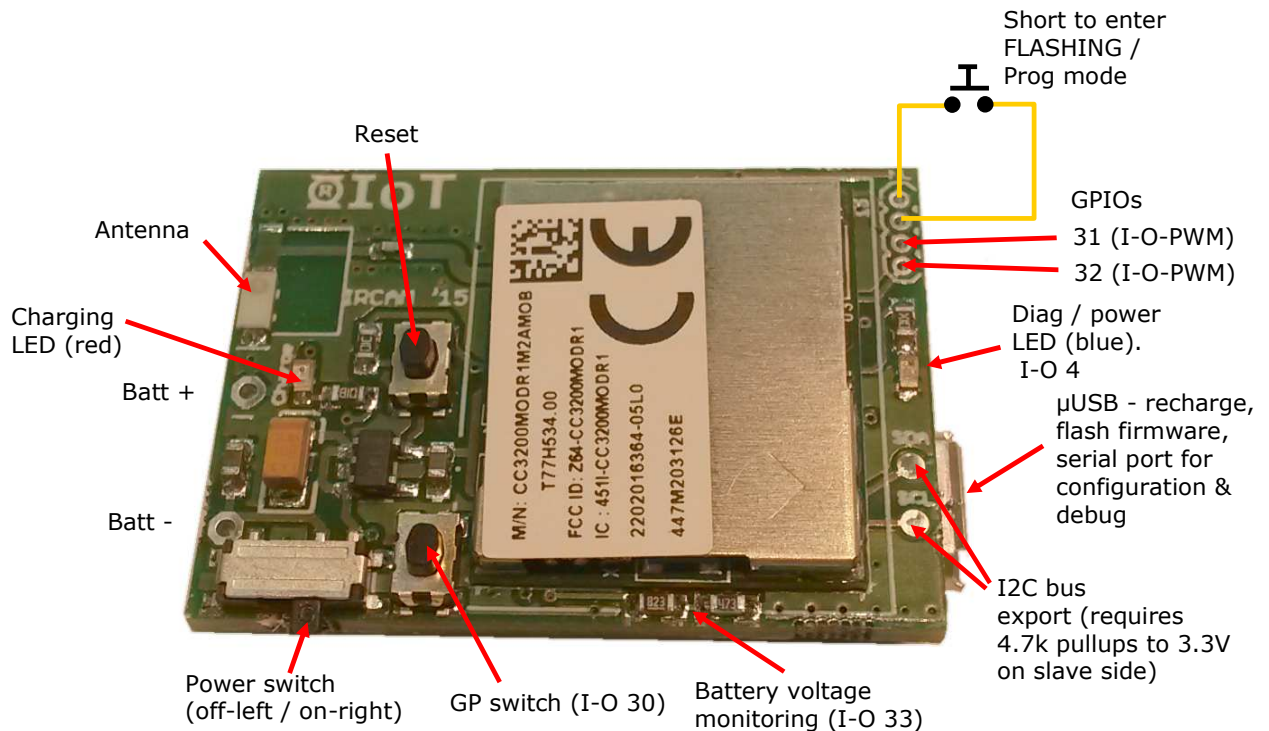
Hardware
Pin number
Other pin number
IC
Serial UART
SPI
analogRead()
digitalRead() and digitalWrite()
digitalRead(), digitalWrite() and analogWrite()

				P4	P2		
TX (1)	SCL	ANTSEL1		40	20		GROUND
TX (1)	SDA	ANTSEL2		39	19		
				38	18		
				37	17		CS
				36	16		RESET
				35	15		MOSI
				34	14		MISO
				33	13		SOP2
				32	12		
				31	11		PUSH2
							GROUND
							GROUND
							+3.3V

© 2014 Rel Vilo, 2012-2014
embeddedcomputing.weebly.com
version 1.2 2014-09-23

In order to remain 100% compatible with the former pin numbering, we kept the same numbering logic of the CC3200 I/Os in our code. The following picture details which I/Os are exported along with their #. That

number is the same used to access to the physical pin from the program written in Energia.



As an example, using the blue LED pin as an output and turning it on in Energia would be done with those 2 lines :

```
pinMode(4, OUTPUT);
digitalWrite(4, HIGH);
```

Programming the R-IoT

Install the Energia IDE

1. visit <http://energia.nu/> and look for the download section
2. download energia version 16
3. On windows, simply unzip the Energia folder on your main hard drive. On Mac, place the folder-program in the Application folder.

Install the USB serial port driver

The R-IoT unit uses a FTDI serial port. Install the drivers by visiting FTDI download section (look for VCP driver)

<http://www.ftdichip.com/Drivers/VCP.htm>

On windows, it will create a COM port. On Mac OS, the user must create the port by going in the network preferences (select standard NULL modem).

Customize the IDE

In order to compile the "full" firmware (currently named R-IoT 1.4), the default linker file used by the Energia tool chain must be modified as the reserved heap size / stack is too high when compiling big programs.

On windows:

- open the location of your Energia folder, such as C:\Program Files (x86)\energia-0101E0016\
 - keep going into hardware\cc3200\cores\cc3200
 - edit the file cc3200.ld

On Mac OS:

- open the location of your Energia folder, usually in the Application folder
- OSX applications are actually a folder. Right click on the app icon and select "show package contents"
- browse to Contents/Resources/Java/hardware/cc3200/cores/cc3200
- edit the file cc3200.ld

In the .ld file, simply edit the first line and change the heap size to 0x0008000 and save the file:

```
HEAP_SIZE = 0x0008000;
```

Use Energia IDE

Launch Energia. A blank sketch appears. A sketch is composed of two essential functions, setup() and loop() which is equivalent to the main function in traditional C language. Energia, just like Arduino uses C and C++, along with a basic API and set of classes to access the hardware in what became the Arduino standard.

<https://www.arduino.cc/en/Reference/HomePage>

The setup function, called prior the main loop, is used to configure the module hardware, default behaviors and everything that requires some initialization before executing the main program.

Once returning from the setup function, the loop function is called repetitively and is equivalent to any traditional endless program loop used on embedded platforms (a while(1) statement within the main function).

Below an example of blinking endlessly the blue LED of the module (I/O #4):



```
sketch_oct07a | Energia 0101E0016
File Edit Sketch Tools Help
sketch_oct07a $
|
int ledState;

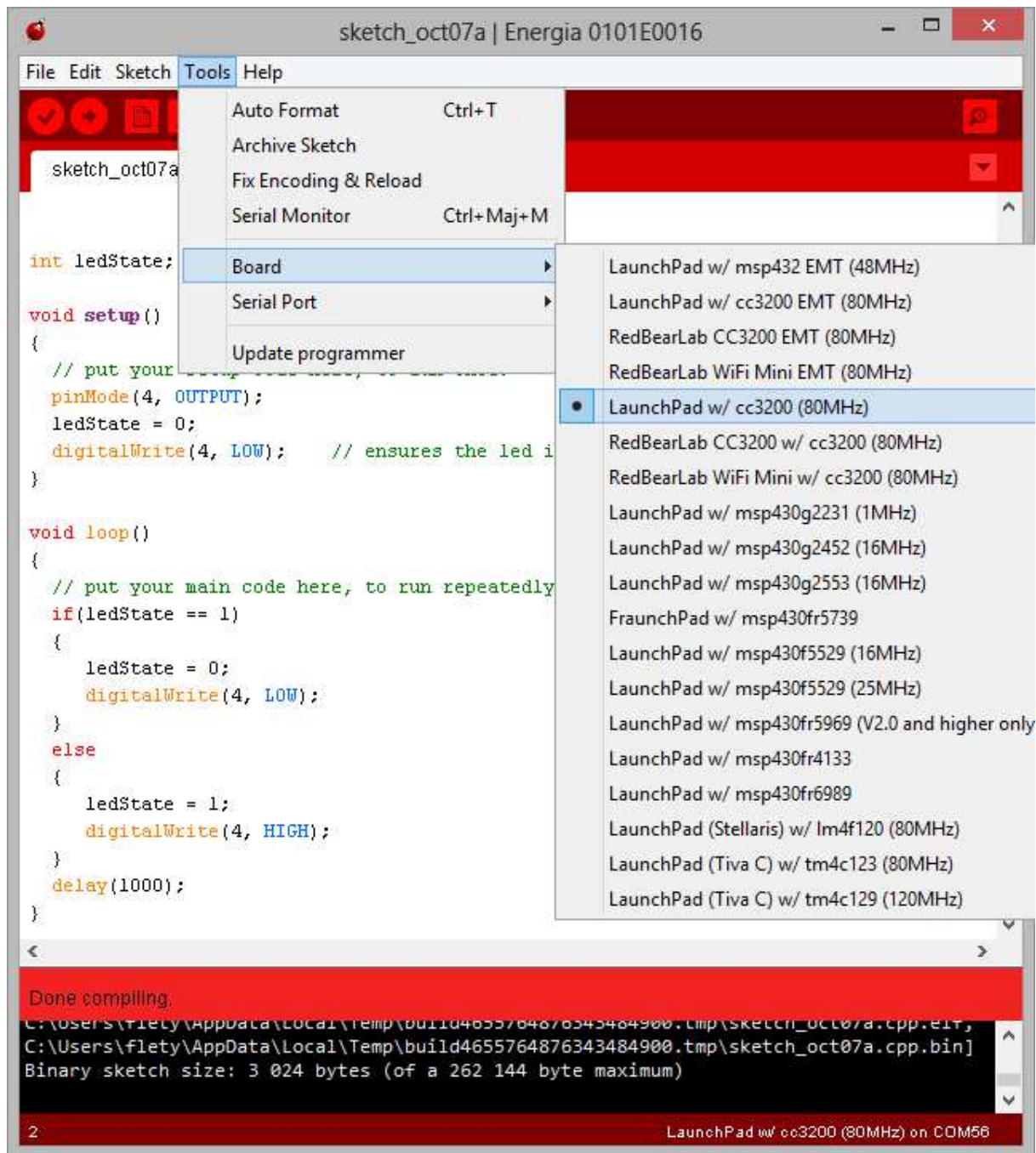
void setup()
{
  // put your setup code here, to run once:
  pinMode(4, OUTPUT);
  ledState = 0;
  digitalWrite(4, LOW); // ensures the led is off
}

void loop()
{
  // put your main code here, to run repeatedly:
  if(ledState == 1)
  {
    ledState = 0;
    digitalWrite(4, LOW);
  }
  else
  {
    ledState = 1;
    digitalWrite(4, HIGH);
  }
  delay(1000);
}

Done compiling.
C:\Users\flety\AppData\Local\Temp\build4655764876343484900.tmp\sketch_oct07a.cpp.e17,
C:\Users\flety\AppData\Local\Temp\build4655764876343484900.tmp\sketch_oct07a.cpp.bin]
Binary sketch size: 3 024 bytes (of a 262 144 byte maximum)

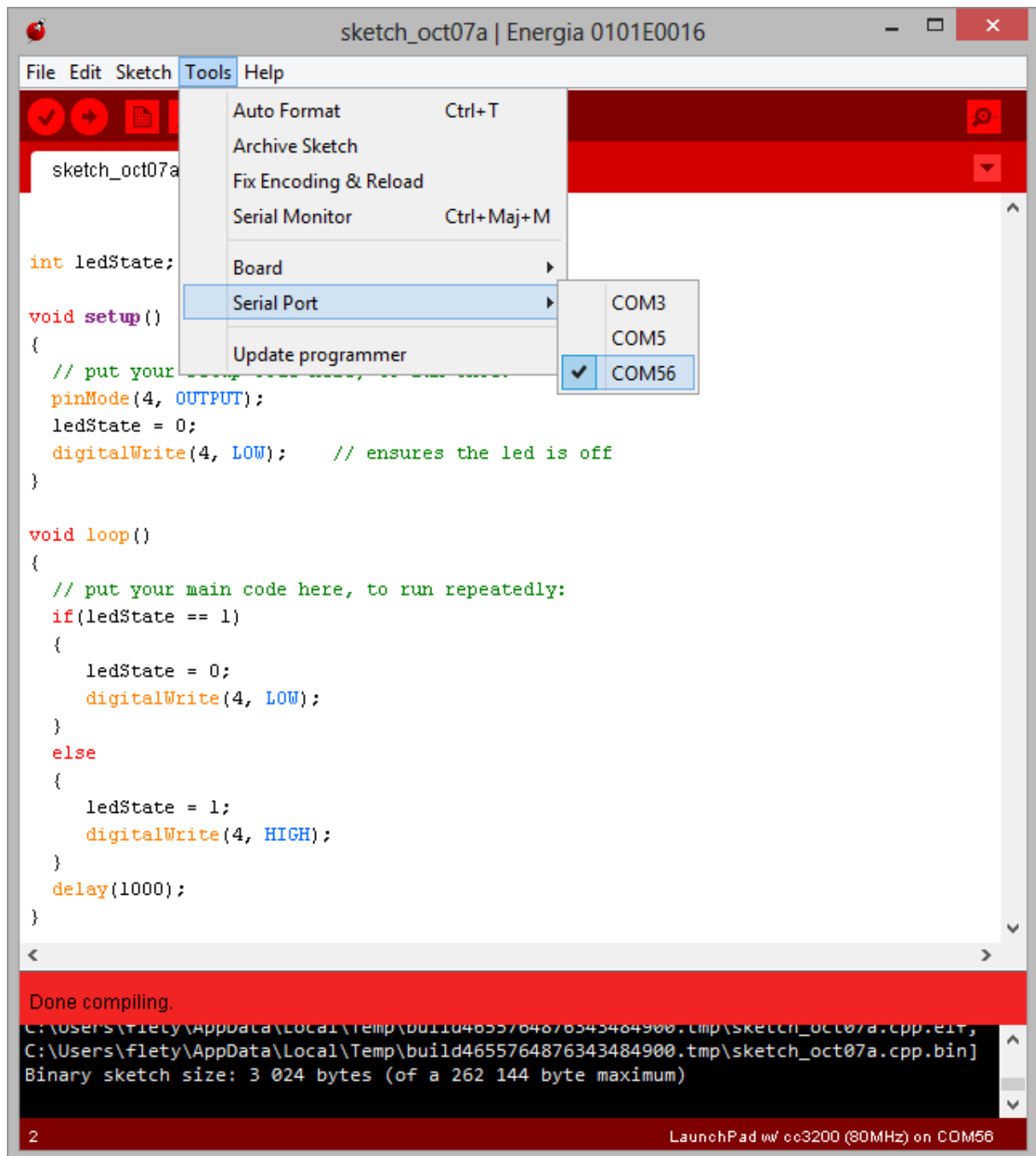
2 LaunchPad w/ cc3200 (80MHz) on COM56
```

Before compiling, select the proper target in the Tools -> Board menu and select the launchPad w/ CC3200 (80 MHz).



To compile to program, simply click on the left-most icon (tick).

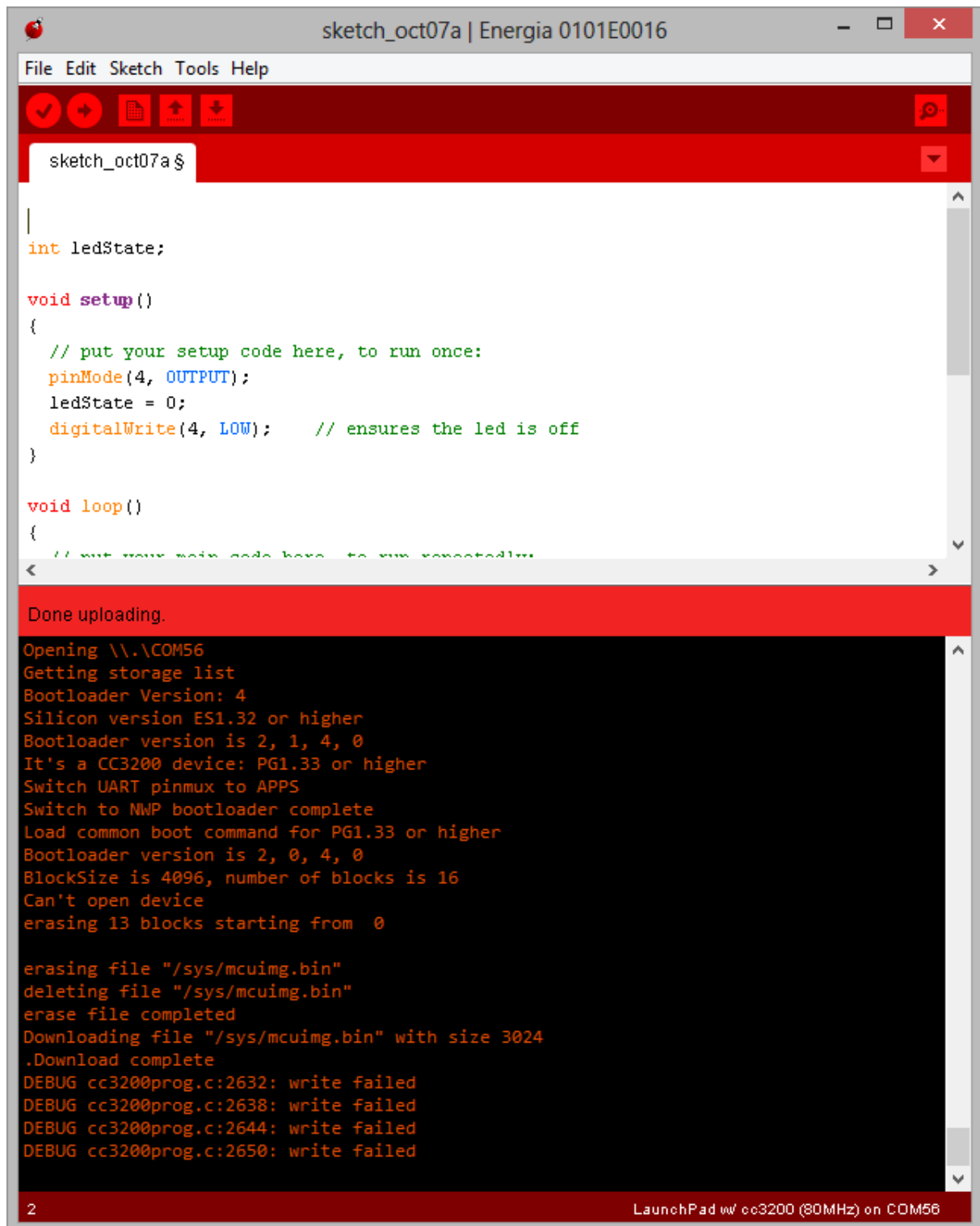
To flash the code on the platform first plug the unit on a USB port then select the matching COM port of the R-IoT



With the module powered on, even already executing the code present in the chip, keep the flashing switch pressed and click on the upload icon (arrow). It will first compile the code then self reset the chip into bootloading mode. After a while, you'll get a done uploading message with the following log in the console of the IDE.

Sometimes, if the bootloader cannot be triggered properly, first click on the upload after pressing on both the flashing switch and the reset. Keep the flashing switch pressed, and release only the reset switch once Energia moves from compiling to uploading stage in the log console.

The DEBUG write fail messages are normal, as the bootloading program tries also to set the chip in debug mode while our platform doesn't have any JTAG debugger / port exported.



The screenshot shows the Arduino IDE window titled "sketch_oct07a | Energia 0101E0016". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for saving, running, and uploading. The sketch editor displays the following code:

```
sketch_oct07a $  
  
|  
int ledState;  
  
void setup()  
{  
  // put your setup code here, to run once:  
  pinMode(4, OUTPUT);  
  ledState = 0;  
  digitalWrite(4, LOW); // ensures the led is off  
}  
  
void loop()  
{  
  // put your main code here, to run repeatedly
```

The output window shows the following messages:

```
Done uploading.  
Opening \\.\COM56  
Getting storage list  
Bootloader Version: 4  
Silicon version ES1.32 or higher  
Bootloader version is 2, 1, 4, 0  
It's a CC3200 device: PG1.33 or higher  
Switch UART pinmux to APPS  
Switch to NWP bootloader complete  
Load common boot command for PG1.33 or higher  
Bootloader version is 2, 0, 4, 0  
BlockSize is 4096, number of blocks is 16  
Can't open device  
erasing 13 blocks starting from 0  
  
erasing file "/sys/mcuimg.bin"  
deleting file "/sys/mcuimg.bin"  
erase file completed  
Downloading file "/sys/mcuimg.bin" with size 3024  
.Download complete  
DEBUG cc3200prog.c:2632: write failed  
DEBUG cc3200prog.c:2638: write failed  
DEBUG cc3200prog.c:2644: write failed  
DEBUG cc3200prog.c:2650: write failed
```

The status bar at the bottom indicates "LaunchPad w/ cc3200 (80MHz) on COM56".

Once there, the module can be reset (cycle the on-off switch, or press the reset onboard momentary switch) to leave flashing mode and execute the

freshly uploaded code. With the above code, the blue LED should flash once per second.