Prototyping Functions

– Optional Energia

Joe George, Northeast Digital Field Applications

Texas Instruments

Americas Sales and Marketing

Prototyping Functions

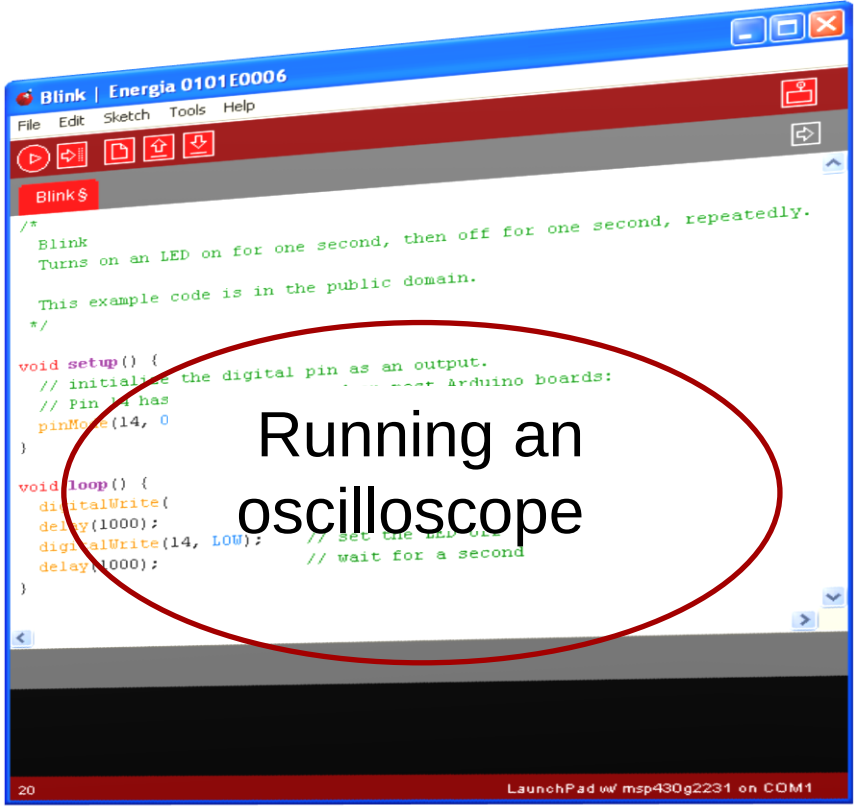
- Step-by-step Functionality (Demos)
  - GP Output (GPIO – General Purpose I/O)
  - Read A/D
  - I2C/SMBus (Wire)
    - *Analog EVM/Boosterpack's*
    - *PinMap Housekeeping*
- Optional WiFi
  - STA (station)
  - AP (access point)
- Optional Energia
- UI
  - Button (GP Input - GPIO, add debounce)
  - LCD Display ("Hello")
  - Music
- UI - Serial Interface (i.e. Putty for echo "Hello World")

And Now Energīa

A simplified programming environment developed by the community (GitHub)

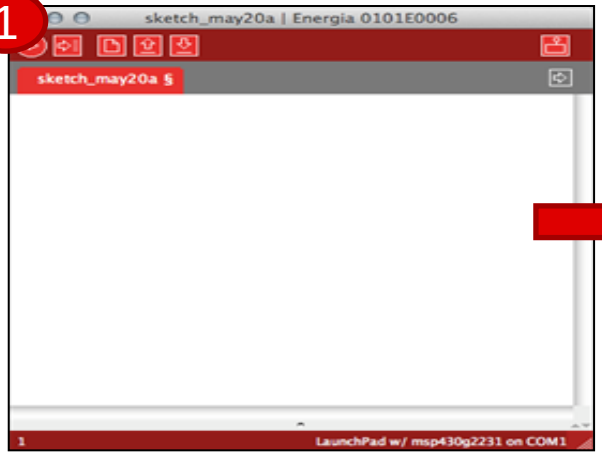
- Open-source electronics prototyping platform for the Texas Instruments LaunchPad (and Boosterpacks)
- Simplify coding to high level functions (Layer over C++)
- Based on the Wiring language
- Brief Energia History and Install http://www.ti.com/tool/ENERGIA
- Getting Started
- Now featuring debugging with integration in CCS

Learn more @ www.energia.nu

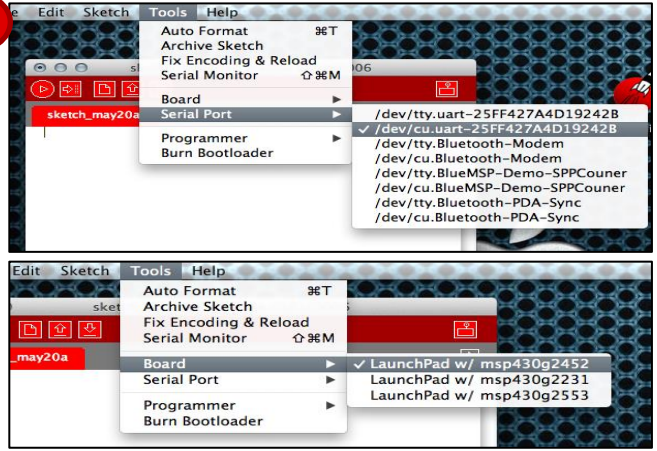


Blinky Demo

1

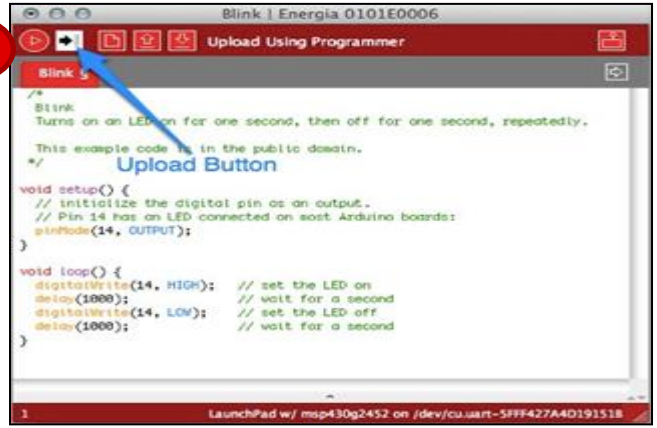


2



3

4



NOTE:

- On **Windows**, ports will be listed as COMXXX port
- On **Linux**, ports will be listed as ttyACM port
- On **Mac OS X** make sure to select the /dev/cu.uart-XXXX NOT the /dev/tty.uart-XXX

Is the green LED blinking? Congratulations! You just successfully uploaded your first sketch.

Look at the function (digitalwrite) in Code

http://energia.nu/reference/digitalwrite/

Example

```
int ledPin = 14;                    // LED connected to digital pi

void setup()
{
  pinMode(ledPin, OUTPUT);          // sets the digital pin as outp
}

void loop()
{
  digitalWrite(ledPin, HIGH);       // sets the LED on
  delay(1000);                      // waits for a second
  digitalWrite(ledPin, LOW);        // sets the LED off
  delay(1000);                      // waits for a second
}
```

Sets pin 14 to HIGH, makes a one-second-long delay, and sets the pin back to LOW.

Look at the Library Reference

Language Reference

Energia programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

Structure

Program Structure

- setup()
- loop()

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Variables

Constants

- HIGH | LOW
- INPUT | OUTPUT
- INPUT_PULLUP
- INPUT_PULLDOWN
- true | false
- integer constants
- floating point constants

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int

Functions

Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - *PWM*

Advanced I/O

- tone()
- noTone()
- shiftOut()
- shiftIn()
- pulseIn()

Look at the Library Reference

Language Reference

Energia programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

Structure

Program Structure

- setup()
- loop()

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Variables

Constants

- HIGH | LOW
- INPUT | OUTPUT
- INPUT_PULLUP
- INPUT_PULLDOWN
- true | false
- integer constants
- floating point constants

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int

Functions

Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - *PWM*

Advanced I/O

- tone()
- noTone()
- shiftOut()
- shiftIn()
- pulseIn()