DRAWING IS PUTTING A LINE ROUND AN IDEA

HENRI MATISSE
MAKING IS GIVING FORM TO AN IDEA
main() {
    printf ("Hello World!
");
}
WHY?
Make to Create
Make to Learn
Make to Communicate
Make to Inspire
Make to Inspire
Make to Inspire Innovate
Make to Inspire Innovate Disrupt
Let’s Be Inspired
MAKERS KIT

LED (LIGHT)

BATTERY TOOL

BATTERY

JUMPER WIRE
POWER UP
I AM A MAKER
Maker Culture
FEARLESSNESS
COLLABORATION
INDEPENDENCE
VERSATILITY
Democratization + Making
Democratized Technology
Democratized Technology

Open source hardware/software, Arduino, Raspberry Pi, Sparkfun, Adafruit, Tindie, Seeed, Shenzen
Democratized Manufacturing
Democratized Manufacturing

3D printing, small batch PCB services, laser cutting, CNC, Makerspaces, OSHPark
Democratized Funding
Democratized Funding

Kickstarter, Crowd Supply, crowd funded equity
Democratized Markets
Democratized Markets

eBay, Etsy, Tindie, Amazon
Democratized Learning
Democratized Learning

MOOCs, MIT open courseware, Khan Academy, Stack Overflow, Coursera, Skillshare, Udacity, Instructables, Google
Makers in Society
Makers in Society Business
Makers in Society
Business
Education
HOW MANY MAKERS ARE THERE IN THE WORLD?
HOW MANY MAKERS ARE THERE IN THE WORLD?

7.3 BILLION
ARE YOU A MAKER?
A microcontroller runs firmware, and senses and controls voltages.
RGB LED controlled by the Bean
TEMPERATURE + ACCELEROMETER

Temperature sensor and 3-axis accelerometer
BLE / Bluetooth Smart module for connecting with other devices
Connection points for power, sensing and control
PROTOTYPING AREA

Convenient for adding additional components
LIGHTBLUE® BEAN

REFERENCE

BLUE TOOTH
TEMPERATURE SENSOR
ACCELEROMETER
COLOR LED
MICRO-CONTROLLER (ARDUINO COMPATIBLE)
EXTERNAL CONNECTIONS
RESET

VERY USEFUL TO KNOW
PROGRAMMING YOUR BEAN

WORKFLOW

1. Launch Bean Loader
2. Pair with your Bean
3. Upload a sketch
4. Edit
LAUNCH BEAN LOADER

ANDROID

IOS
MAKING INNOVATION AND DISRUPTION

MAKER KIT

1 x LIGHTBLUE® BEAN
1 x LED (LIGHT EMITTING DIODE)
1 x JUMPER WIRE
1 x SPARE BATTERY
1 x BATTERY TOOL

BEAN NAME

awesomeBean
PAIR YOUR BEAN (ANDROID)

1. SELECT BLUETOOTH
2. SELECT YOUR BEAN
   DONE
PAIR YOUR BEAN (iOS)

SELECT A SKETCH

SELECT BLUETOOTH

SELECT YOUR BEAN

DONE
UPLOAD TO BEAN (ANDROID)

1. CHOOSE SKETCH...
2. NAVIGATE TO SKETCH
3. UPLOAD

COMPILES AND UPLOADS
UPLOAD TO BEAN (iOS)

SELECT A SKETCH

SELECT UPLOAD

COMPILLES AND UPLOADS
// Blink Bean LED
void setup() {
}

void loop() {
    // Turn LED ON
    Bean.setLed(150, 0, 0);
    Bean.sleep(300);

    // Turn LED OFF
    Bean.setLed(0, 0, 0);
    Bean.sleep(300);
}
// Blink Bean LED
void setup() {

}

void loop() {

}

Statements (C/C++)
Bean.setLED (r, g, b);
Bean.sleep (time);
EDIT SKETCH (ANDROID)

1. LAUNCH DROID EDIT
2. NAVIGATE TO SKETCH
3. EDIT SKETCH
4. SAVE + CONTINUE
EDIT SKETCH (iOS)

EDIT SKETCH IN BEAN LOADER
void setup() {
}

void loop() {

    // Blink ON Color #1
    Bean.setLed(150, 0, 0);
    Bean.sleep(1000);

    // Blink ON Color #2
    Bean.setLed(0, 150, 0);
    Bean.sleep(1000);

    // Blink ON Color #3
    Bean.setLed(0, 0, 150);
    Bean.sleep(1000);

    // Blink OFF
    Bean.setLed(0, 0, 0);
    Bean.sleep(1000);

}
int temperature;
int blinkDelay;

void setup() {
}

void loop() {

  temperature = Bean.getTemperature();

  // Blink delay depends on temperature
  if (temperature < 26) {
    blinkDelay = 600;
  } else {
    blinkDelay = 100;
  }

  Bean.setLed (0, 150, 0);
  Bean.sleep (blinkDelay);

  Bean.setLed (0, 0, 0);
  Bean.sleep (blinkDelay);
}

C_BEANTEMPERATURE
Variables store values for use in a sketch. Variable type is required. Variable scope is determined by the location of declaration.
Bean.getTemperature();

Returns the current temperature in degrees Celsius.

Other LightBlue® Bean peripherals, such as the LED and accelerometer, are available through this same mechanism.
If statements are used for branching. The usual logical operators are supported: ==, <, >, <=, >=, !=

```java
if (temperature < 26) {
    blinkDelay = 600;
} else {
    blinkDelay = 100;
}
```
electronics
Most of us are familiar with the idea of **voltage**.
VOLTAGE DRIVES CURRENT
BUT, NOT WITHOUT A CIRCUIT
COMPONENTS

BATTERY

LED
This is a Schematic
INPUTS & OUTPUTS

MICROCONTROLLER

INPUT

OUTPUT
PINS A0, A1

ANALOG

BOOLEAN

INTEGER

FCC ID: 2AAV5-LBM313-2540
IC: 11371A-LBM3132540

Model: LBM313-2540-256
SN: EPC1404001583
SIMPLE LED CIRCUIT
void setup() {
    // Initialize pin #5 as an output pin
    pinMode(5, OUTPUT);
}

void loop() {
    // Set external pin voltage to HIGH, then LOW to cause blinking
    digitalWrite(5, HIGH);
    Bean.sleep(500);

digitalWrite(5, LOW);
    Bean.sleep(500);
}
pinMode (5, OUTPUT);

Sets pin #5 to be an output pin. That allows the microcontroller to set the pin voltage to high or low. Pin number can range from 0-5. Other modes are INPUT, and INPUT_PULLUP.
digitalWrite (5, HIGH);

Sets pin #5 to either “HIGH”, or “LOW”. Pin number can range from 0-5. The pinMode() must be set to OUTPUT for this pin.
SIMPLE BUTTON CIRCUIT
PIN VOLTAGE WITH BUTTON

HIGH

0V
void setup() {
    // Initialize pin #0 as an
    // input pin (with default
    // HIGH voltage)
    pinMode (0, INPUT_PULLUP);
}

void loop() {
    // Turn the Bean's LED on.
    // Color depends on the state
    // of pin #0
    if (digitalRead (0) == HIGH) {
        Bean.setLed (200, 0, 0);
    } else {
        Bean.setLed (0, 200, 0);
    }
    Bean.sleep(1000);
    // Turn the LED off
    Bean.setLed(0, 0, 0);
    Bean.sleep(1000);
}
pinMode (0, INPUT_PULLUP);

Sets pin #0 to be an input pin. By default, the pin shows a high voltage. In order to create a low voltage, the pin must be connected to ground by some method. Pin number can range from 0-5. Other pin modes are INPUT, and OUTPUT.
digitalRead (0);

Returns a value of HIGH or LOW depending on the circuit connected to pin #0. In our case, the value will be HIGH by default, then change to LOW when the wire switch is connected.
// Variables
const byte MOTION_RESET_PIN = 0;
const byte EXTERNAL_LED_PIN = 5;

// Accelerometer monitoring
const int MOTION_THRESHOLD = 100;

AccelerationReading currentReading;
AccelerationReading previousReading;
int motion;

// Temperature (celsius)
const int LOW_TEMPERATURE = 24;
const int HIGH_TEMPERATURE = 32;
const float TEMPERATURE_RANGE = (HIGH_TEMPERATURE - LOW_TEMPERATURE);

int temperature;

// Sample and blink intervals
int sampleCount;
int sampleInterval;
int blinkCount;

// LED State
boolean lightOn;

// Interaction states
boolean motionDetected;
boolean wasReset;

// LED Color
const byte MAX_COLOR = 200;
byte red = 0;
byte green = 0;
byte blue = 0;

void setup()
{
    motionDetected = false;
    lightOn = true;
    sampleCount = 0;
    sampleInterval = 250;
    blinkCount = 2;
    wasReset = false;
    previousReading = Bean.getAcceleration();
    pinMode(MOTION_RESET_PIN, INPUT_PULLUP);
    pinMode(EXTERNAL_LED_PIN, OUTPUT);
    digitalWrite(EXTERNAL_LED_PIN, LOW);
}

void loop()
{
    // Check the temperature, update LED color
    temperature = Bean.getTemperature();
    if (temperature < LOW_TEMPERATURE) temperature = LOW_TEMPERATURE;
    if (temperature > HIGH_TEMPERATURE) temperature = HIGH_TEMPERATURE;
    // Calculate color values from temperature
    red = MAX_COLOR * (float(temperature - LOW_TEMPERATURE) / TEMPERATURE_RANGE);
    blue = MAX_COLOR * (float(HIGH_TEMPERATURE - temperature) / TEMPERATURE_RANGE);

    // Check the accelerometer and calculate motion
    currentReading = Bean.getAcceleration();
    int deltaX = abs(currentReading.xAxis - previousReading.xAxis);
    int deltaY = abs(currentReading.yAxis - previousReading.yAxis);
    int deltaZ = abs(currentReading.zAxis - previousReading.zAxis);
    motion = deltaX + deltaY + deltaZ;
    previousReading = currentReading;

    // Update state based on motion
    if (motion > MOTION_THRESHOLD) motionDetected = true;

    // Reset the blink state if RESET_MOTION_PIN is shorted to ground
    if (digitalRead(MOTION_RESET_PIN) == LOW) {
        motionDetected = false;
        lightOn = true;
        wasReset = true;
    }

    // Update blink state
    --sampleCount;
    if (sampleCount <= 0) {
        sampleCount = blinkCount;
        lightOn = !lightOn;
    }

    // Update the motion detector LED
    if (!motionDetected || (lightOn && motionDetected)) Bean.setLed(red, green, blue);
    else Bean.setLed(0, 0, 0);

    // Update the reset indicator
    if (lightOn && wasReset) digitalWrite(EXTERNAL_LED_PIN, HIGH);
    else digitalWrite(EXTERNAL_LED_PIN, LOW);

    // Put the Bean to sleep until the next loop
    Bean.sleep(sampleInterval);
}
Wax On. Wax Off.
This is a Schematic
The Bean supports easy connection to a universe of sensors and other devices like displays, motors, GPS modules, and buzzers.
SMART PHONE HUB
A mobile device connected with a Bean can be a hub for collecting data, providing a user interface, connecting to the cloud and controlling the Bean. Punch Through has published iOS and Android libraries.
A BLE-compatible laptop can also be a hub communicating with the Bean. It can connect to the internet and provides a more robust development environment.
IoT + NODE-RED

Open source graphical programming environment built on node.js. Connecting modules defines behavior and actions. Bean nodes are available. See the Bean website for more information.
**LightBlue® | Bean+**

- Rechargeable LiPo Battery
- Standard Headers for Add-on Boards
- Grove Connectors for Easy Prototyping
- 3.3V / 5V Operation

[www.punchthrough.com](http://www.punchthrough.com)
PROJECTS?

Wearable
Bicycle Navigator
Hack your Coffee Pot or Kegorator
Phone-Controlled Mood Light

hackster.io
instructables.com
ARE YOU A MAKER?
Thank You!
USEFUL APPS

LIGHTBLUE EXPLORER (iTunes)
General tool for working with BLE. Can control the Bean LED and shows accelerometer and external I/O


BEAN CONSOLE (iTunes)
Allows you to monitor Serial I/O with a Bean. Very handy for prototyping and debugging

# ONLINE RESOURCES

<table>
<thead>
<tr>
<th>Resource</th>
<th>Website</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUNCH THROUGH</td>
<td>punchthrough.com</td>
<td>all things LightBlue Bean, guides, forums</td>
</tr>
<tr>
<td>ARDUINO</td>
<td>arduino.cc</td>
<td>reference, tutorials</td>
</tr>
<tr>
<td>INSTRUCTABLES</td>
<td>instructables.com</td>
<td>projects</td>
</tr>
<tr>
<td>HACKSTER.IO</td>
<td>hackster.io</td>
<td>projects</td>
</tr>
<tr>
<td>MAKE MAGAZINE</td>
<td>makezine.com</td>
<td>articles, tutorials, reviews</td>
</tr>
<tr>
<td>MAKER SHED</td>
<td>makershed.com</td>
<td>components, kits, stuff</td>
</tr>
<tr>
<td>SPARKFUN</td>
<td>sparkfun.com</td>
<td>components, devices, tutorials</td>
</tr>
<tr>
<td>ADAFRUIT</td>
<td>adafruit.com</td>
<td>components, devices, tutorials</td>
</tr>
<tr>
<td>AMAZON</td>
<td>amazon.com</td>
<td>books, components, devices</td>
</tr>
<tr>
<td>O’REILLY</td>
<td>oreilly.com</td>
<td>reference, books</td>
</tr>
<tr>
<td>GOOGLE</td>
<td>google.com</td>
<td>everything</td>
</tr>
</tbody>
</table>
BEAN PINOUT - COMPLETE