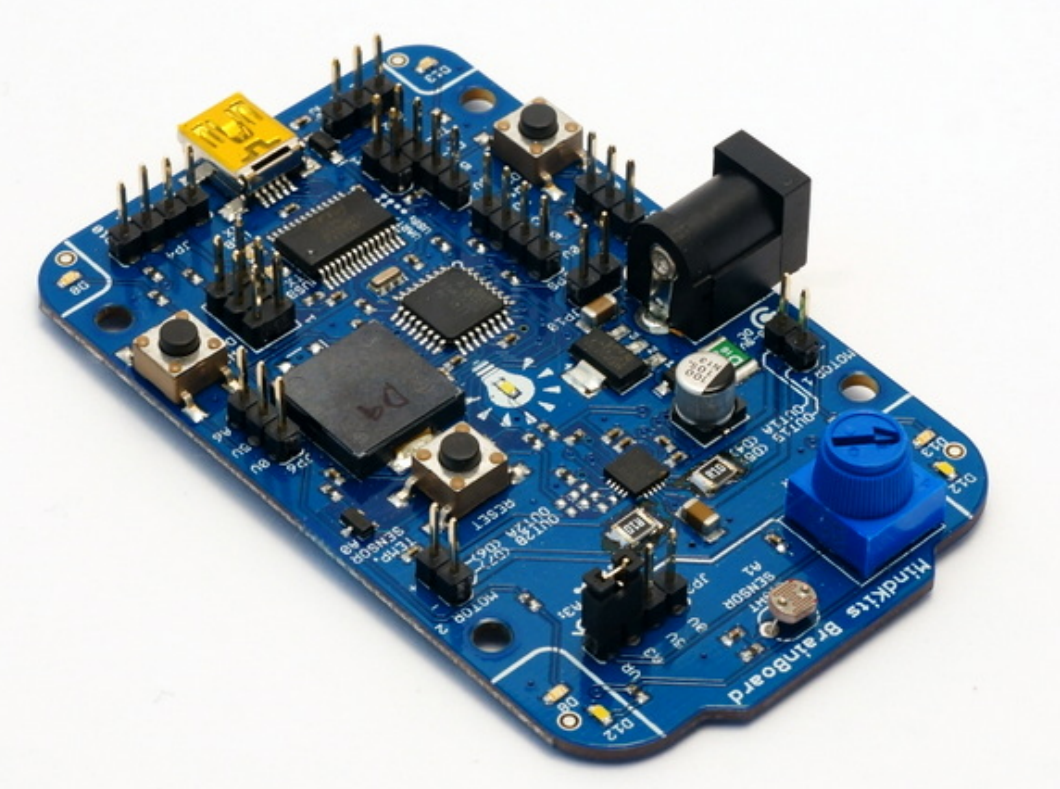


MindKits BrainBoard V2

Congratulations!

Congratulations on purchasing your MindKits BrainBoard — a fun way to get started with electronics, programming, and robotics.

Contents

Introduction

The MindKits BrainBoard is the little robot controller that could — it's Arduino-compatible and packed with sensors, inputs, and outputs — a motor driver that can handle two DC motors or a stepper motor, light and temperature sensors, a handy variable resistor with a knob, two programmable buttons, three sets of LEDs, a buzzer (…but where is it?), USB, digital and analog I/O and optional plug-in expansion modules with Bluetooth, micro-SD card, motion sensing, and many more in the works. And it's all open-source so you can study how it works and hack it!

BrainBoard ships with the Arduino Nano bootloader installed so just plug in a mini-USB cable to start coding — no programmer necessary.

You can supply power from USB, DC jack, or both at once.

All that makes for an excellent general-purpose microcontroller board for home automation, science experiments, wireless sensing, motion control or just learning about electronics and programming without any mind-numbing or pocket-hurting mega-kits; all the basics are there in one place to get started.

Setup

Installing your BrainBoard should be no more trouble than setting up a normal Arduino. If you do have issues, email our support guru before you become frustrated. Support at MindkItsinvent dot com are waiting to hear from you.

Download the Arduino IDE (the program you write code in) from arduino.cc
http://arduino.cc/en/main/software

At the time of writing 1.0.5 was the most recent but this will change with time.

If you're on a Mac or Linux, you're now ready to plug your brainboard in and start interacting with your world, if you're on Windows, there's a couple of easy steps before we're ready.

Windows

Once you install the Arduino IDE there is a directory called 'Drivers'. In here you have the drivers needed to enable your BrainBoard to talk to your windows computer. Install them or have your administrator install them and your brainboard will be set up as a com port and ready to go.
Go to device manager to see if your com port has been set up yet or if there is a yellow exclamation mark beside it, this means you still need to install the drivers.
Any problems, email support and we'll help.

Watch Out! This bit's important.

History belongs in the past

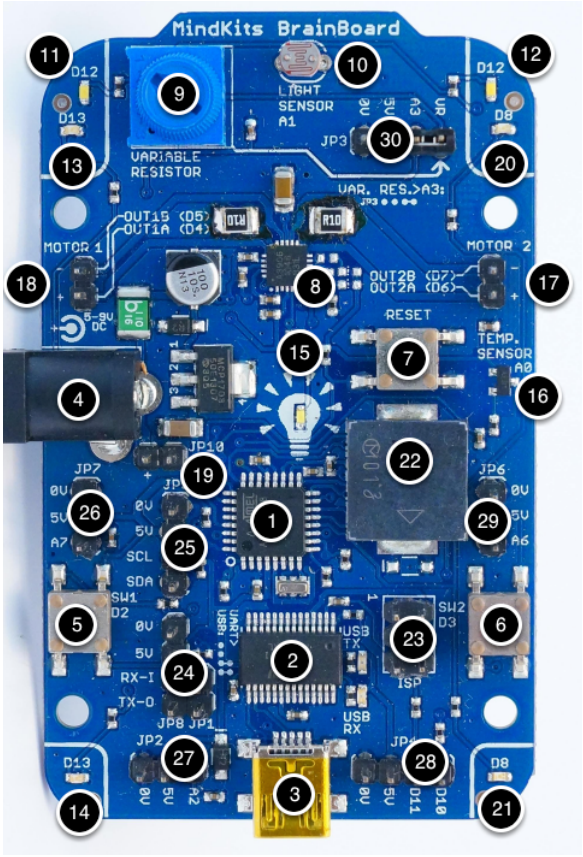
The original BrainBoard (black, with the plugin LeoStick module) was a very different beast. The new BrainBoard is blue, better, and brainier, with lots of changes and improvements – an updated motor driver, onboard processor and USB, more I/O pins, easier to follow pinouts and labelling, a temperature sensor, and the push buttons are now active high instead of active low. So that's great, but don't assume anything! If you're porting code from the old (black) board you'll need to refer to the new pinouts printed on the board.

9V Max….No Really!!!

Be careful with the power supply too — we had to make a tradeoff between a dinosaur of a motor driver that would run at 12V but waste a lot of battery, or a modern chip that fixes that with better silicon but doesn't like more than 9V. We went with the modern chip which just means you'll need to keep your DC input below 9V or risk letting the smoke out. Don't do it, it's really hard to put back in! For battery-powered robots this shouldn't be a problem anyway — you can still run BrainBoard on 4-6 AA or AAA cells, a 9V battery, or a 7.2V LiPo. You can also run BrainBoard from USB power, just not heavy loads like motors. More info under *Power Supply Requirements*.

Features

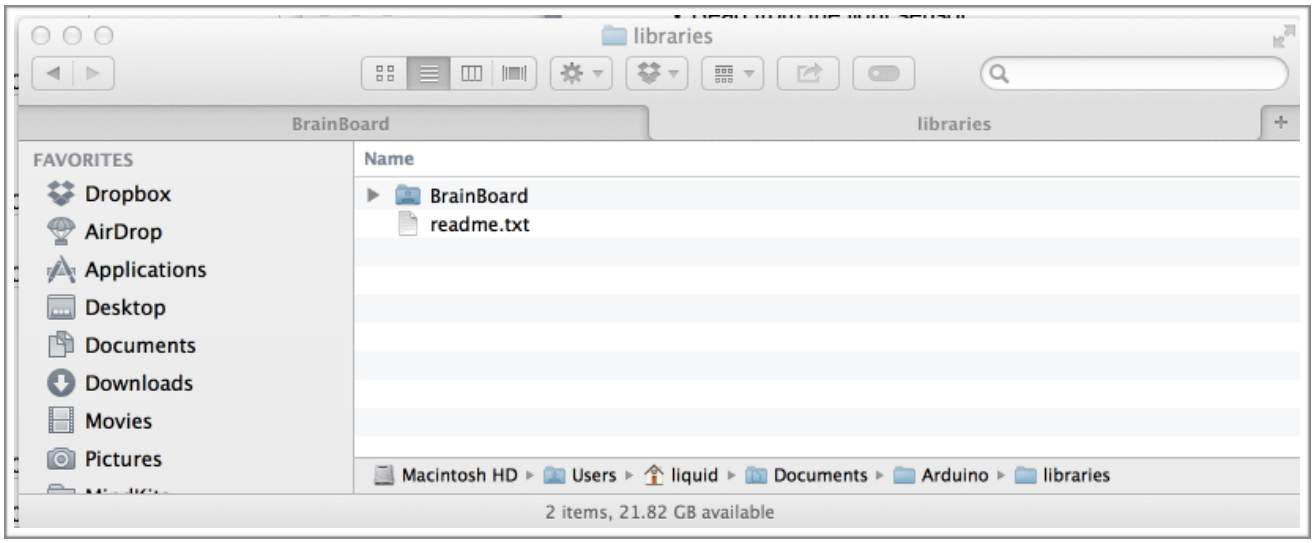
BrainBoard has 13 available I/O pins with 11 capable of digital I/O and 6 capable of analog input (some do both), three serial ports (UART, I2C, SPI), a two-channel motor driver, triple power connectors, plus a good share of LEDs, buttons, and sensors.



1	ATMega328 CPU
2	FTDI USB chip
3	Mini-USB socket
4	DC input jack, 5-9VDC
5	Left push-button (D2)
6	Right push-button (D3)
8	A3906 1 A, 2-channel motor driver
9	Variable resistor, jumper to A3
10	LDR light sensor (A1)
11, 12	Front LEDs (D12)
13, 14	Left LEDs (D13)
15	Power LED
16	MCP9700 temperature sensor (A0)
17	Left motor connector (D4, D5)
18	Right motor connector (D6, D7)
19	Auxiliary DC input JP10, 5-9VDC
20, 21	Right LEDs (D8)
22	Buzzer (D9)
23	ISP (external prog.)/SPI serial header
24	UART serial port header. Fit jumpers when programming.
25	I2C / TWI / A4 / A5 serial header
26	A7 header
27	A2 header
28	D10 & D11 / servo header
29	A6 header
30	A3 / variable resistor header

Pin Assignment

BrainBoard has some pins assigned to its built in sensors which are already wired up and ready to go. There are more pins ready for you to use for add-on sensors, extra motor drivers and other devious plans. Here's a handy list of how everything is plugged together now and what's free for you to use in your programs.



Primary Signal	Secondary Signal	Pin
Temp sens		A0
LDR		A1
Rear sensor		A2
Front sens/pot		A3
SDA		A4
SCL		A5
Rx-I		D0
Tx-O		D1
PB sw L/SDA		D2/INT0
PB sw R/SCL		D3/INT1
Motor IN1		D4
Motor IN2		D5/0C0B

Motor IN3		D6/0C0A
Motor IN4		D7
Right indicator		D8
Buzzer	0C1A	D9
D10	0C1B	D10
D11	MOSI	D11
→	MISO	D12
Left ind	SCK	D13
Right sensor		A6
Left sensor		A7

Power Supply Requirements

BrainBoard accepts between 5V and 9V DC and runs on 5V internally, meaning any device that connects to it must also run on 5V.

The onboard regulator has very low dropout to get the most out of batteries, but be careful not to pull too much current from it.

Power can be supplied from USB (5V) or the external DC jack (or auxiliary header connected to it, JP10), or both at once. Be careful with polarity — the external DC inputs have reverse-polarity protection but don't rely on this. If you connect power and the power LED (the MindKits bulb) doesn't light up immediately then unplug it and check your connections.

The external DC input also has a self-resetting thermal fuse that will offer some short-circuit or over-current protection, but again don't rely on it. Check your wiring and make sure any heavy loads you're driving (like motors) are in spec before plugging in the power.

Take extra care to avoid short circuiting anything or driving heavy loads like motors when only connected to USB power. If you're driving motors then it's a good idea to program one of the push buttons to start your program once you've unplugged the USB cable. Something like this in your startup code will do:

```
#define SW1    2
void setup()
{
  while(digitalRead(SW1) == 0); //wait while button input is low (not
 active)
}
```

BrainBoard Library - The Easy Way

Using the BrainBoard functions can be done in two ways. You'll see in the quick start guides below that you can control the BrainBoard inputs and outputs directly in the program, but we have another way….the easy way - the BrainBoard Library.

With the library you can call a function and the rest is done for you.

With the library you can:

Control the motors including speed and direction
bb.motorWrite(Motor Channel, speed);
eg
bb.motorWrite(1, 255); // full forward
bb.motorWrite(1, -255); // full back
bb.motorWrite(1, 0); // full stop
bb.motorWrite(2, 255); // full forward on channel two

Turn the left, right and front LEDs on and off
bb.leftLeds(ON);
bb.leftLeds(OFF);
bb.rightLeds(ON);
bb.frontLeds(ON);

Read from the light sensor
Serial.print("Light sensor = ");
Serial.println(bb.lightRead());

Read from the temperature sensor
Serial.print("Temperature sensor counts = ");
Serial.println(bb.tempRead());

Read analogue values from the potentiometer
Serial.print("Variable resistor (A3) = ");
Serial.println(bb.vrRead());

Read button presses
if(bb.sw1Read()){ //read push button 1
Do this stuff when the button is pressed;
}

if(bb.sw2Read()){ //read push button 2

Setup
Your code will need a line at the top to tell your sketch to use the library and this is done each time you use it in a sketch.

```
#include <BrainBoard.h>
```

```
BrainBoard bb; // create BrainBoard object. this automatically sets up
 all the pin modes (input/output).

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600); //
}
```

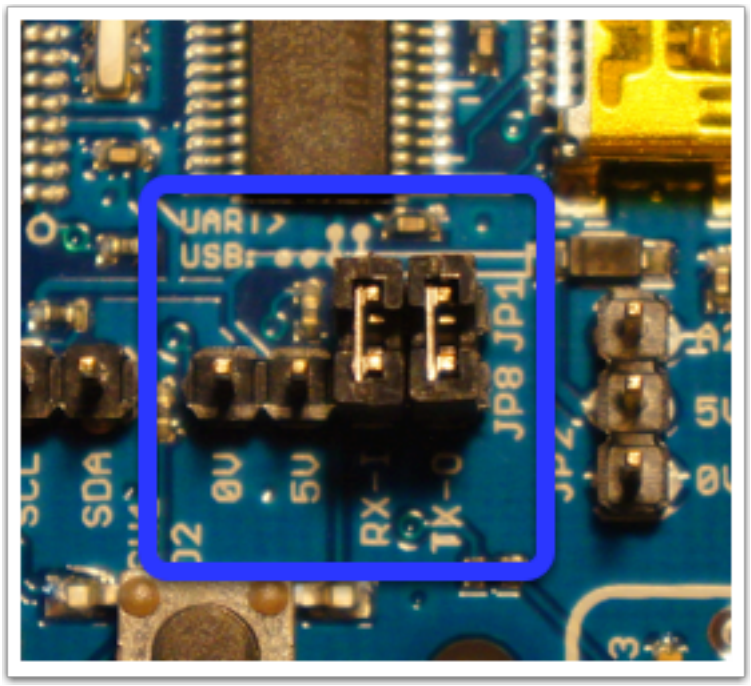
The first time you use the library, you'll need to copy the whole BrainBoard library directory into your libraries folder under /Users/liquid/Documents/Arduino/libraries on a mac.
You can check if your Arduino install is using the same location by opening the Arduino IDE, and opening the preferences. Your sketchbook folder should be displayed. Add a folder called libraries under here and copy the BrainBoard library directory into it.

Quick Start 1: Hello, World! Blinking an LED

You're probably itching to get something to happen on your new BrainBoard.
And you're in luck, because it comes pre-loaded with a demo app. Power up BrainBoard and the LEDs will start flashing.

OK. Demo done — now to load your own code.

First things first — make sure two jumpers are fitted between JP8 and JP1 exactly like this:

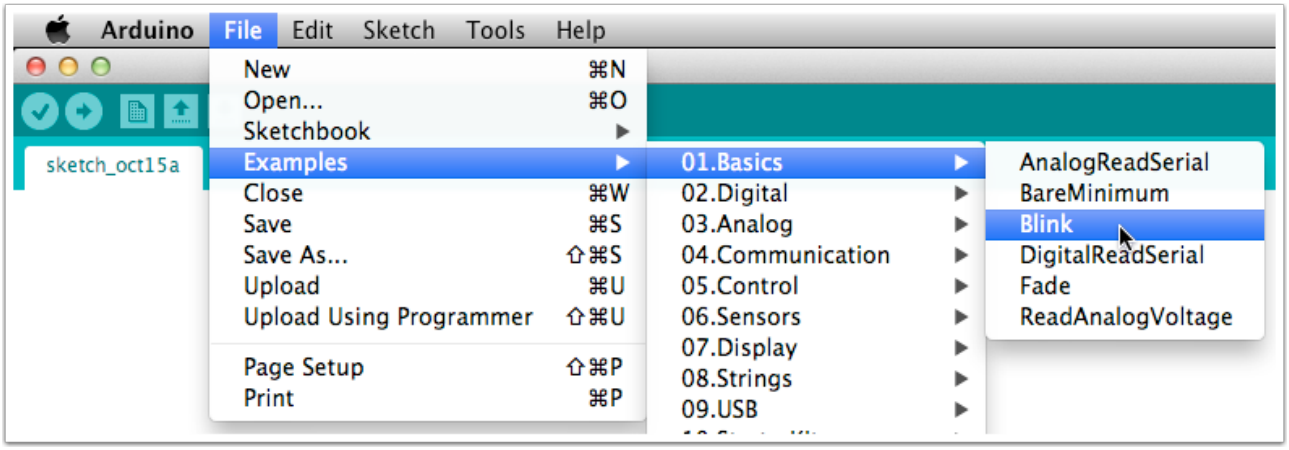


These jumpers connect the CPU serial port (JP8) to the USB side (JP1). If you want to connect the CPU to another serial device just remove the jumpers and connect your device to JP8.

Now let's get our "Hello World" running to make sure everything is set up and working properly.

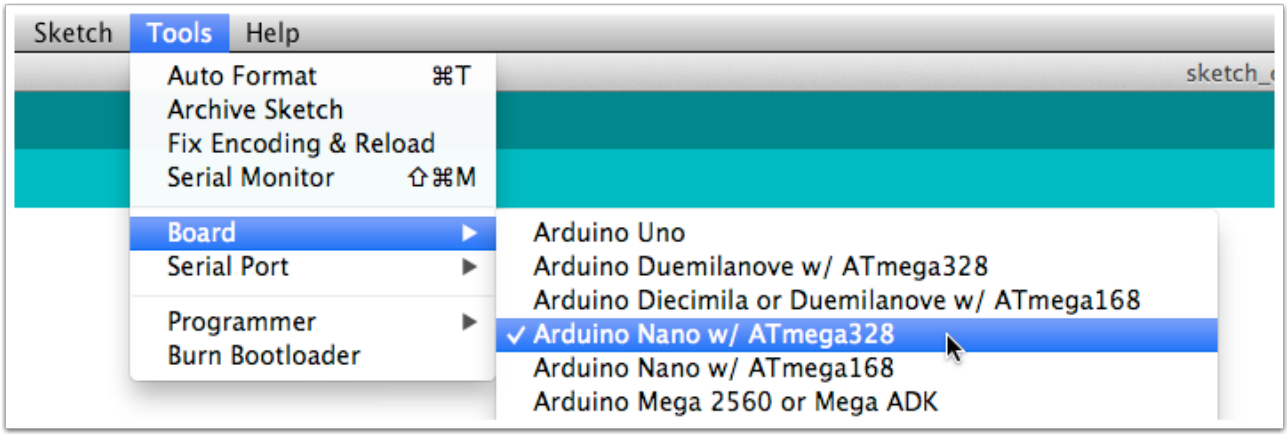
Install the Arduino IDE if you haven't already. You can download it from http://www.arduino.cc/download. It's important to note that installing the IDE also installs some drivers which tell your computer how to talk to the BrainBoard (well the FTDI USB chip, to be exact). If you're on a Mac or Linux system then these drivers are probably already installed but you'll still need the IDE.

Fire up the Arduino IDE and open the '*Blink*' example sketch (sketch = project in Arduino-speak):

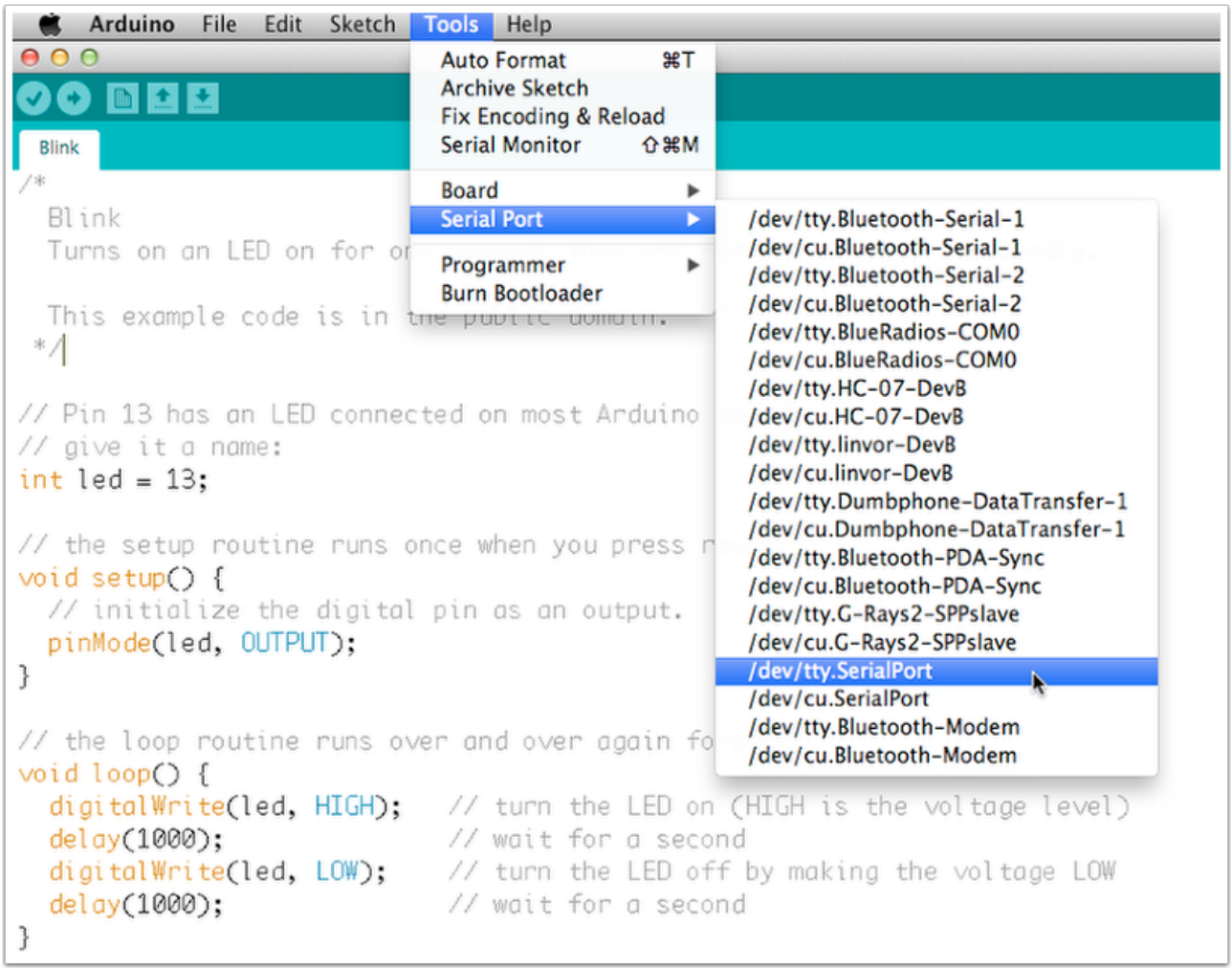


This sketch will flash the left blue LEDs (both connected to D13) once the board is programmed. But first...

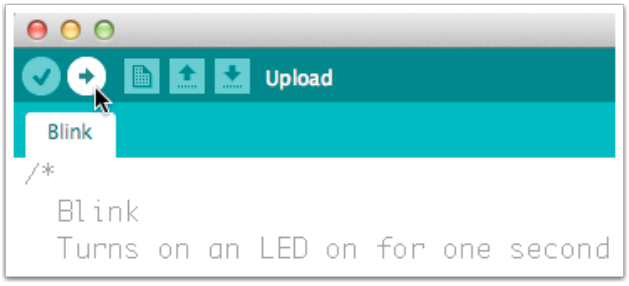
Select Arduino Nano as the Board:



Select the COM/serial port the board is connected to. Hint: It's likely the highest numbered one. Hopefully you don't have this many to choose from! They're named a bit differently depending on your operating system. Mac/Linux serial ports are named tty.xxxx and COMxx in Windows. If the first one doesn't work, try another.



Next hit Upload to compile and send the code to the microcontroller:



After a few seconds you should see green USB activity LEDs and the left blue LEDs flashing on BrainBoard indicating that your code is being programmed. Once this is finished the left side blue leds (marked "D13") should start flashing every two seconds indicating success. If you don't see what you're expecting then go back and check the board type, serial port, and jumpers are all set correctly. If that doesn't work get in touch with us ASAP — see contact details at the end of this document.

Quick Start 2: BrainBoard I/O Sample Code

Here's some sample code that gives the pins more descriptive names and enables the outputs. This is a great place to start with your own code.

Copy and paste the code below into a new Arduino sketch, or if the text doesn't copy properly you can download bb_demo.ino from the project page. A header file with the same definitions is also available.

```
/*
BrainBoard Notes
----------------------
1.  Maximum voltage on any pin is 5V.
2.  Power BrainBoard with 5-9V DC (centre +ve) using the barrel socket
 or JP10. Note the '+' on JP10. Connecting reversed power will probably
let out all the smoke and void your warranty.
3.  Make sure you have external power (from barrel socket or JP10)
 before turning on motors in your code so you don't overload your USB
 port.
4.  There's a handy Arduino Nano pin reference at http://
 forum.arduino.cc/index.php?topic=147582.0 -- the board layout is
 different but the pin functions (and chip) are common.
5.  Some useful pin definitions follow.
*/

// Digital inputs
#define SW1 2             // left push button, active high
#define SW2 3             // right push button, active high

// Digital outputs
#define FRONT_LEDS  12    // white leds
#define LEFT_LEDS   13    // left blue leds
#define RIGHT_LEDS  8     // right blue leds
#define BUZZER      9     // use tone() to drive the BUZZER.
#define MOTOR1_A    4     // motor 1 (left) drive pin 1
#define MOTOR1_B    5     // motor 1 (left) drive pin 2, PWM capable
 with analogWrite()
#define MOTOR2_A    6     // motor 2 (right) drive pin 1, PWM capable
 with analogWrite()
#define MOTOR2_B    7     // motor 2 (right) drive pin 2

// Digital input/output
#define D10 10            // digital 10
#define D11 11            // digital 11

// Analog inputs & sensors
#define TEMP_SENSOR   A0  // mcp9700 temperature sensor
#define LDR           A1  // light-dependent resistor / light sensor
 (analog)
#define ANALOG2       A2  // analog input 2
#define ANALOG3       A3  // analog input 3, use jumper to connect to
 variable resistor
```

```
#define ANALOG4       A4  // analog input 4, also I2C SDA
#define ANALOG5       A4  // analog input 5. also I2C SCL
#define ANALOG6       A6  // analog input 6
#define ANALOG7       A7  // analog input 7

void setup() {
  // put your setup code here, to run once:

  // configure output pins
  pinMode(FRONT_LEDS, OUTPUT);
  pinMode(LEFT_LEDS, OUTPUT);
  pinMode(RIGHT_LEDS, OUTPUT);
  pinMode(BUZZER, OUTPUT);
  pinMode(MOTOR1_A, OUTPUT);
  pinMode(MOTOR1_B, OUTPUT);
  pinMode(MOTOR2_A, OUTPUT);
  pinMode(MOTOR2_B, OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Quick Start 3: Driving LEDs, Buzzer, & Motors

With the code in Quick Start 2 loaded, add this code to the loop function:

```
// generate a 1 kHz tone on the buzzer:
tone(BUZZER, 1000);

// turn on all LEDs:
digitalWrite(RIGHT_LEDS,HIGH);
digitalWrite(LEFT_LEDS,HIGH);
digitalWrite(FRONT_LEDS,HIGH);

// drive a DC motor on Motor 1 at 50% power:
analogWrite(MOTOR1_B,127); // connect the motor to the Motor 1 pins,
OUT1A and OUT1B

// and full power on Motor 2:
analogWrite(MOTOR2_A,255);
```

Quick Start 4: Reading Analog Inputs

We read analog inputs with the analogRead() function.

Light to tone: read the light sensor voltage (0 - 1023) and send it to the buzzer:

```
int analogValue = analogRead(LDR);
tone(BUZZER, analogValue);
```

Temperature to tone: read the temperature sensor voltage and send it to the buzzer:

```
int analogValue = analogRead(TEMP_SENSOR);
tone(BUZZER, analogValue);
```

Mixed light and temperature:

```
int analogValue = analogRead(LDR) + (analogRead(TEMP_SENSOR)-154)*10;
tone(BUZZER, analogValue);
```

154 is the approximate temperature sensor reading at 25°C.
Converting to °C: The default analog reference is 5V, and analogRead() returns a value from 0 - 1023, with 1023 = 5V.

The sensor outputs 500mV + 10mV/°C, so let's say we read a value of 150. We scale the reading by the full scale value (1023) then multiply by the reference voltage to get the measured voltage:

150 / 1023 x 5V = 733mV

Then subtract the offset and divide by the sensitivity:

(733mV - 500mV) / 10mV = 23.3°C

Read the variable resistor (potentiometer) and buttons and the mixed output to the buzzer:

```
  int analogValue = analogRead(ANALOG3) + digitalRead(SW1)*100 +
digitalRead(SW2)*1000;
  tone(BUZZER,analogValue);
```

Make sure a jumper is connected between VR and A3 on JP3.

Quick Start 5: Reading Buttons

BrainBoard has two momentary push buttons (SW1 and SW2) respectively connected to digital pins 2 and 3. These are active high, meaning they read '1' or 'HIGH' when pushed. Don't try to drive these pins as outputs.

Read each button and turn on left, right, or all LEDs:

```
  digitalWrite(LEFT_LEDS, digitalRead(SW1));
  digitalWrite(RIGHT_LEDS, digitalRead(SW2));
  digitalWrite(FRONT_LEDS, digitalRead(SW1) && digitalRead(SW2));
```

Debouncing buttons:

To check if a button has been pushed once the button must be debounced (filtered).

In the Arduino IDE, upload Examples > General > 02. Digital > Debounce and try pushing SW1.

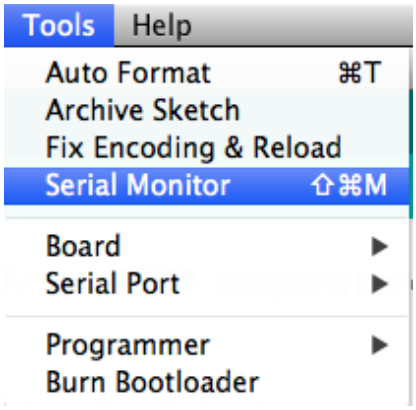
Quick Start 6: Serial Data

You can read analog or digital data from BrainBoard and use your computer as a display to debug your code or record data.

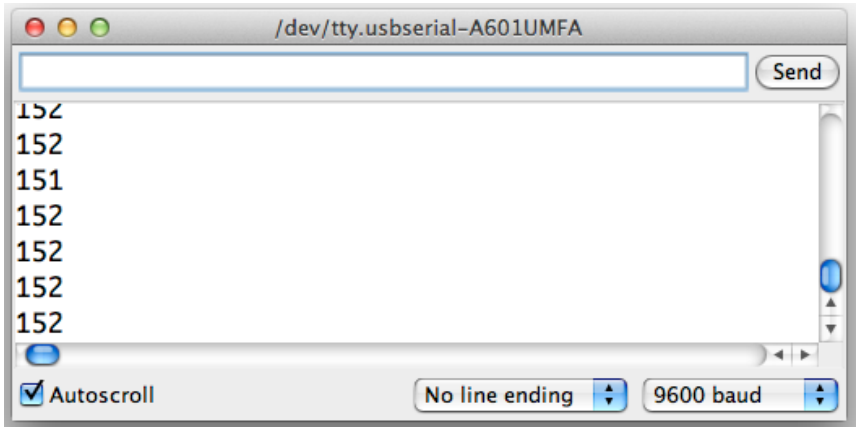
This code sets up the serial port at 9,600 bits per second (the 'baud' rate), reads the raw temperature sensor value and transmits it through the USB serial port.

```
#define TEMP_SENSOR   A0  // mcp9700 temperature sensor

void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println(analogRead(TEMP_SENSOR));
}
```

Program the code, and once it's running open the serial monitor and set the baud rate to 9600:



You'll see data streaming in with each sample.



You can use RealTerm to capture data and graph it in Excel.

BrainBoard Code Examples

Now it's time for some full code examples so you know how to use each of the BrainBoards functions well. We use the BrainBoard library for this so if you haven't installed it already, go ahead and do that now (instructions earlier in the document).

Educator's Question: Can you describe what each of the following are to a microcontroller? Input/Output/Analogue/Digital

Outputs

LEDs

```
// BrainBoard LED demo.
// Blinks all LEDs.
// Version 2014-03-31

#include <BrainBoard.h>

BrainBoard bb; // create BrainBoard object. this automatically sets up
 the pin modes (input/output).

void setup() {
    // put your setup code here, to run once
}

void loop() { // the code below runs forever

    // flash the LEDs
    bb.leftLeds(ON);
    bb.rightLeds(ON);
    bb.frontLeds(ON);
    delay(5); //wait 5 milliseconds

    bb.leftLeds(OFF);
    bb.rightLeds(OFF);
    bb.frontLeds(OFF);
    delay(200); // wait 500 milliseconds
}
```

Challenge

Can you make your BrainBoard flash the LEDs three times each starting with the front LEDs and moving around to the right LEDs and finally the left LEDs?

Buzzer

Educators question: What is a piezo buzzer? How does it work?

```
// BrainBoard buzzer demo.
// ta-daaa!
// Version 2014-03-31

#include <BrainBoard.h>

BrainBoard brainBoard; // Create BrainBoard object. this automatically
 //sets up the pin modes (input/output).

void setup() {

    // put your setup code here, to run once
}

void loop() { // the code below runs forever

    // The beep function has two inputs: the first is frequency in Hz,
 //the second is the tone duration in milliseconds
  brainBoard.beep(2500, 40); // 2.5 kHz for 40 millisec

    // Silence for 40 ms
    delay(40);

    // Another beep
    brainBoard.beep(2500, 500); // 2.5 kHz for 500 millisec

    // Stop the program -- do nothing forever.
    while(1);
}
```

Challenge:
Can you make your brainboard play a tune? You'll need to figure out some rough pitches of the tune and string them together.

Motors

Educators Question: How does a DC motor work? How do we make a motor go faster or slower using a micro controller? What limits how fast we can make the motor run?

```
// BrainBoard motor driver demo.
// Plug a small 6-12V motor into one or both of the motor driver ports
 "Motor 1" and "Motor 2" and see it rotate.
// ** Caution! ** Make sure you have an external batter or supply
 connect because the USB port might not have enough power.
// Version 2014-03-31

#include <BrainBoard.h>

BrainBoard brainBoard; // Create BrainBoard object. this automatically
 sets up the pin modes (input/output).

void setup() {

    // Put your setup code here, to run once
}

void loop() { // the code below runs forever

    // The motorWrite() function takes two inputs, the channel (1 or 2)
 and the power (includes direction), a number from -255 (full reverse)
 to 255 (full forward). 0 = off.
    // Now we're going to test the motor drivers by turning on both
 channels to 128/255 = 50% power
    brainBoard.motorWrite(1, 128); // drive motor 1 forward
    brainBoard.motorWrite(2, -128); // drive motor 2 backward

    // Now wait half a second (500 ms)
    delay(500);

    // Now reverse the motors
    brainBoard.motorWrite(1, -128); // Drive motor 1 forward
    brainBoard.motorWrite(2, 128); // Drive motor 2 backward

    // Wait another half second
    delay(500);

    // Now turn off the motors
    brainBoard.motorWrite(1, 0); // Turn off motor 1
    brainBoard.motorWrite(2, 0); // Turn off motor 2

    // Stop the program -- do nothing forever.
    while(1);
}
```

Challenge
Create a gauntlet using blocks, objects you have laying around and make an obstacle course for your BrainBot. Now, can you write a program to drive your robot around the gauntlet without crashing?

Inputs

Light Sensor

Educators Question: What does LDR stand for? How does an LDR work? Is it digital or analogue?

```
/*

 Module: BrainBoard light sensor example.

 Background: BrainBoard has a light sensor which outputs a voltage
which increases as light increases. We can read the analogue voltage
and convert this
 to a digital value (see the analogue example for more details) and
transmit the values over USB to your computer where you could save or
graph them.
 The range of values is from 0 (total darkness) to 1023 (bright
lights).

 Learning outcomes: After following this example you will be able to
read the analogue light sensor values using the
 BrainBoard library lightRead() function.

 Open Tools > Serial Monitor and set the baud rate to 9600 to view the
light sensor values.

 Author: Brody Radford
 Version: 2014-04-05
 License: GPLv3, http://gnu.org/licenses/gpl.html

 */

#include <BrainBoard.h>

BrainBoard brainBoard; // Create BrainBoard object. this automatically
 sets up the pin modes (input/output).

void setup() {
    // Put your setup code here, to run once
    Serial.begin(9600); // Open the serial port at 9,600 bps to
 transmit data values.
}

void loop() { // the code below runs forever

    Serial.print("Light sensor = ");         // Send some text to the
 coputer describing what the numbers mean. We have to send the text
 separate from values.
```

```
    Serial.println( brainBoard.lightRead() );  // Now we read the value
using brainBoard.lightRead() and send the result to the computer.

    // Now check if the value is greater than 700 (about 68% of full
scale) and turn on the front LEDs if it is, otherwise turn them off.
    if( brainBoard.lightRead() > 700 ){
        brainBoard.frontLeds(HIGH);
    }else{
        brainBoard.frontLeds(LOW);
    }
    delay(250); // Wait a little so the data is slow enough to read.
}
```

Challenge

Using the buzzer code from earlier as a guide, can you make your BrainBoards buzzer change with input from the light sensor?

Potentiometer

Educators Question: How would we use a potentiometer as input? What is it used for in our ever day lives now? How does a potentiometer work? What other sensor on the BrainBoard is this similar to?

```
/*

 Module: BrainBoard variable resistor example.

 Background: BrainBoard has a variable resistor (potentiometer) with an
easy to turn knob which makes it very easy to adjust the behaviour of
your program when it's running.
 The variable resistor is an analogue sensor (see analogue input
example for more info), so it outputs values increasing from 0 at the
counter-clockwise limit to 1023 when turned all
 the way to the right. Using the BrainBoard library function we can
easily get the digital value and transmit the values over USB to your
computer where you could save or graph them.

 ** Important note ** : The variable resistor shares analogue input 3
with JP3, so the VR and A3 pins must be connected with a jumper!
Otherwise the variable resistor will be disconnected.

 Learning outcomes: After following this example you will be able to
read the variable resistor v alues using the BrainBoard library
vrRead() function.

 Open Tools > Serial Monitor and set the baud rate to 9600 to view the
variable resistor values.

 Author: Brody Radford
 Version: 2014-04-05
 License: GPLv3, http://gnu.org/licenses/gpl.html

 */

#include <BrainBoard.h>

BrainBoard brainBoard; // Create BrainBoard object. This automatically
 sets up the pin modes (input/output).

void setup() {
    // Put your setup code here, to run once
    Serial.begin(9600); // Open the serial port at 9,600 bps to
 transmit data values.
}

void loop() { // the code below runs forever
```

```
    Serial.print("Variable resistor = ");  // Send some text to the
coputer describing what the numbers mean. We have to send the text
separate from values.
    Serial.println( brainBoard.vrRead() ); // Now we read the value and
send the result to the computer.

    // Now check if the value is greater than 512 (knob in the middle)
and turn on the front LEDs if it is, otherwise turn them off.
    if( brainBoard.vrRead() > 512 ){
        brainBoard.frontLeds(HIGH);
    }else{
        brainBoard.frontLeds(LOW);
    }
    delay(250); // Wait a little so the data is slow enough to read.
}
```

Challenge

Using the potentiometer as input, can you make your LEDs flash faster and slower as you wind the potentiometer knob? If you can't program it to work, describe how you would use the potentiometer to adjust the led frequency.

Button

Educators Question: Buttons are everywhere, how do they work? What is the symbol in electronics for a button in a circuit? How does this help us explain how a button works? What is 'bounce' when we're talking about a button being pressed in electronics?

```
/*

 Module Name:
 BrainBoard button example.

 Background:
 BrainBoard has two user programmable buttons named SW1 and SW2. These
 can be programmed to control BrainBoard.

 Learning outcomes:
 After following this example you will be able to read BrainBoard's
 buttons with the sw1Read() and sw2Read() functions and use their
 states to control your program.

 Author: Brody Radford
 Version: 2014-04-05
 License: GPLv3, http://gnu.org/licenses/gpl.html

 */

#include <BrainBoard.h>

// Create BrainBoard object. This automatically sets up the pin modes
 (input/output) for the buzzer, LEDs, and motor driver.

BrainBoard brainBoard;

void setup() {  // The following code in setup() {...} just runs once
 at power up.

    // From here the program continues to loop() below and stays there
 forever.
}

void loop() { // The code below runs forever, looping back to the start
 when it gets to the end (unless we have stopped the program).

    // Read button 1 and turn on the left LEDs if it is pushed, or turn
 them off if not. We use the "if" statement to read the buttons and
 make a decision:
    if( brainBoard.sw1Read() == HIGH ){
        brainBoard.leftLeds(HIGH);
    }
    else{
        brainBoard.leftLeds(LOW);
```

```
    }

    // Read button 2 and turn on the right LEDs:
    if( brainBoard.sw2Read() == HIGH ){
        brainBoard.rightLeds(HIGH);
    }
    else{
        brainBoard.rightLeds(LOW);
    }

    // If both buttons are pushed, also turn on the front LEDs:
    if( (brainBoard.sw1Read() == HIGH) && (brainBoard.sw2Read() ==
HIGH) ){
        brainBoard.frontLeds(HIGH);
    }
    else{
        brainBoard.frontLeds(LOW);
    }

    // Now the program goes back to the start of the loop and repeats
the cycle of checking the pin value and updating the LED.
}
```

Challenge

Can you write some code to make the buzzer sound when you press a button? Look up the sequence of dots and dashes needed to spell your name in morse code. Can you use the button to make your BrainBoard sound out your name in morse code?

Temperature Sensor

Educators Question: Can you find the temperature sensor on the board? What is happening when you put your finger on the sensor? Does that make this sensor a digital or analogue sensor?

```
/*

 Module: BrainBoard temperature sensor example.

 Background: BrainBoard has a temperature sensor which outputs a
voltage that increases with temperature. The BrainBoard library has a
function to read the analogue voltage and convert it to
 temperature in degrees celcius: tempRead(). We can transmit the
temperatures over USB to a computer where you could save or graph
them.
 The temperature sensor is next to the reset button. Try pressing it
with your finger and watching the temperature measurements rise in the
serial monitor.

 Learning outcomes: After following this example you will be able to
read the temperature using the BrainBoard library tempRead() function.

 Open Tools > Serial Monitor and set the baud rate to 9600 to view the
light sensor values.

 Author: Brody Radford
 Version: 2014-04-05
 License: GPLv3, http://gnu.org/licenses/gpl.html

 */

#include <BrainBoard.h>

BrainBoard brainBoard; // Create BrainBoard object. this automatically
 sets up the pin modes (input/output).

void setup() {
    // Put your setup code here, to run once
    Serial.begin(9600); // Open the serial port at 9,600 bps to
 transmit data values.
}

void loop() { // the code below runs forever

    Serial.print("Temperature = ");         // Send some text to the
 coputer describing what the numbers mean. We have to send the text
 separate from values.
    Serial.println(brainBoard.tempRead());  // Now we read the value
 using brainBoard.tempRead() and send the result to the computer.
```

```
    // Now check if the value is less greater than 28 degrees and turn
 on the front LEDs if it is, otherwise turn them off.
    if(brainBoard.tempRead() > 28){
        brainBoard.frontLeds(HIGH);
    }else{
        brainBoard.frontLeds(LOW);
    }
    delay(250); // Wait a little so the data is slow enough to read.
}
```

Challenge

Use the temperature sensor on the BrainBoard to send out a temperature reading every minute via serial. Can you capture these inputs and put them into a spreadsheet for graphing? Hint: Google docs is another way to do this if you don't have a spreadsheet on your computer.

Digital

Along with the built in sensors on the BranBoard there are pins available to add extra sensors. Sensors can be either digital or analogue. Digital sensors are 0v or 5v and switch between these voltages quickly to convey their readings.

```
/*

Module Name:
BrainBoard digital I/O (input/output) example.

Background:
Digital signals take one of two values: high or low, also called 1 or
 0, true or false, and on or off. The point is there are only two
 possible values,
although for electrical signals these states also correspond to
 voltages. BrainBoard signals are 5V high, and 0V low.
This example demonstrates how to read and write digital values to
 BrainBoard's digital pins.

Learning outcomes:
After following this example you will be able to read and write digital
 signals on BrainBoard's I/O pins using the digitalRead() and
 digitalWrite() functions.

Author: Brody Radford
Version: 2014-04-05
License: GPLv3, http://gnu.org/licenses/gpl.html

*/

#include <BrainBoard.h>

// Create BrainBoard object. This automatically sets up the pin modes
 (input/output) for the buzzer, LEDs, and motor driver.

BrainBoard brainBoard;

void setup() {  // The following code in setup() {...} just runs once
 at power up.

    // We must first set the pin mode (signal direction) to input or
 output. Digital pins have three possible states:
    // Input, and two Output states: high, and low. If we set the pin
 mode to output we can program the pin to be high or low
    // (say to turn something on or off).

    // All pins are set to input by default so that they don't conflict
 with anything they're connected to (think of
    // two people trying to steer one car -- bad things happen!).
```

```
    // To set a pin as output we use the pinMode() function, giving it
the pin number with or without the 'D' and the mode:
    pinMode(10, OUTPUT); // Sets pin D10 to output. The pin value is
now logical 0 (low, 0V) by default.

    // Now we're going to set the pin to logical 1 (high, 5V). If you
have a multimeter you can probe the voltage between D10 and 0V
    // and confirm it's 5V for 5 seconds after connecting the power,
then it will go to digital 0 (0V).
    digitalWrite(10, HIGH);
    brainBoard.leftLeds(HIGH); // Turn on the left LEDs to mirror D10,
so we know which part of the program we're in.
    delay(5000); // Wait 5 seconds.
    digitalWrite(10, LOW); // Now set the output to low.
    brainBoard.leftLeds(LOW); // Turn off the left LEDs to mirror D10.

    // From here the program continues to loop() below and stays there
forever.

}

void loop() { // The code below runs forever, looping back to the start
when it gets to the end (unless we have stopped the program).

    // Now we're going to combine the digitalWrite() function we
learned about in setup() with the digitalRead() function to check the
input on
    // pin D11 and mirror it to the front front (D12) LEDs.

    // To connect pin 11 to a signal we'll use a ballpoint pen and
carefully use the conductive metal tip to connect D11 to the pins
either side.

    // Connecting D11 to the 5V pin will input a digital high signal
and the front LEDs will turn on. Connecting D11 to D10 will turn off
the LEDs,
    // because we previously set D10 to low. Note that the LEDs might
flash on and off unpredictably when nothing is connected to D11
because the input state
    // isn't defined as high or low.

    // We use the "if" statement to make decisions:
    if( digitalRead(11) == HIGH){
        brainBoard.frontLeds(HIGH);
    }else{
        brainBoard.frontLeds(LOW);
    }

    // In plain English this says: IF digital pin D10's value equals
high (digital 1, 5V) THEN turn on the front LEDs, ELSE turn them off.
```

```
    // Don't forget to use double equals (==) to check for equality!
This is very important because a single equal sign is for assignment,
meaning
    // "set the variable on the left of the equals to the value on the
right".

    // A simpler but less obvious way to do this feeds the input signal
straigh to the output:
    brainBoard.frontLeds(digitalRead(11));

    // Now the program goes back to the start of the loop and repeats
the cycle of checking the pin value and updating the LED.
}
```

Analogue

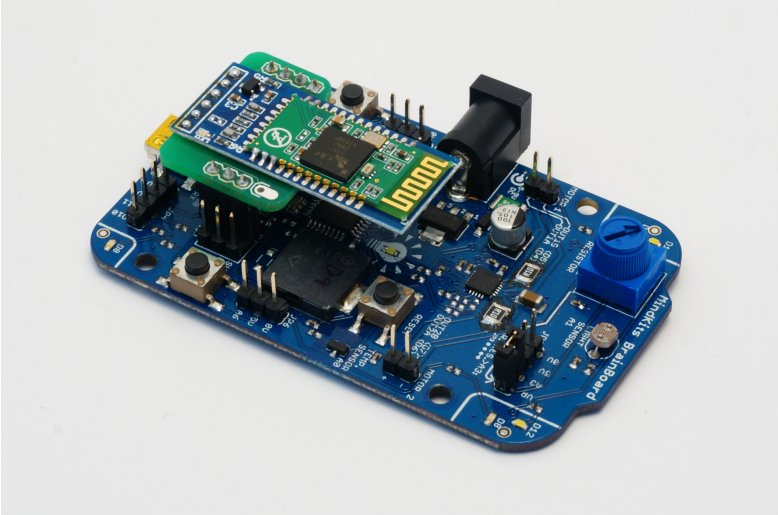
Along with the built in sensors on the BranBoard there are pins available to add extra sensors. Sensors can be either digital or analogue. Analogue sensors convey their readings by giving the BrainBoard a reading between 0v and 5v. The higher the voltage, the higher the reading.

```
/*

 Module: BrainBoard analogue input example.

 Background: BrainBoard has is capable of measuring voltages between 0V
and 5V, known as analogue (or analog, US spelling) signals. BrainBoard
can convert analogue signals
 in this range to a digital value between 0 and 1023, meaning that 0V =
0 digital, 5V = 1023 digital, and 2.5V would equal about 512. This
process is called analogue to digital
 conversion, or digitisation (because we're converting real-world
continuous values to discrete digital values) and we use the
analogRead() function.

 Learning outcomes: After following this example you will be able to
read analogue signals on BrainBoard's analogue input pins using the
analogRead() function.

 Author: Brody Radford
 Version: 2014-04-05
 License: GPLv3, http://gnu.org/licenses/gpl.html

 */

#include <BrainBoard.h>

// Create BrainBoard object. This automatically sets up the pin modes
 (input/output) for the buzzer, LEDs, and motor driver.
BrainBoard brainBoard;

void setup() {  // The following code in setup() {...} just runs once
 at power up.
    // From here the program continues to loop() below and stays there
 forever.
}

void loop() { // The code below runs forever, looping back to the start
 when it gets to the end (unless we have stopped the program).

    // Here we'll read the analogue value on the variable resistor
 (analogue pin 3 -- make sure there is a jumper connecting pin A3 to VR
 on JP3!) and use it to generate a tone
    // and turn on a LED:
    tone(BUZZER, analogRead(3));
```

```
    if(analogRead(3) > 512){
        brainBoard.frontLeds(HIGH);
    }else{
        brainBoard.frontLeds(LOW);
    }

    // Now the program goes back to the start of the loop.
}
```

Wireless Bluetooth Expansion Module

The Bluetooth expansion module (sold separately) is a quick and easy way to give BrainBoard wireless communication. Remove the jumpers on JP8/JP1 and line up the Bluetooth module's sockets. Pair the module with your computer or smartphone (the password is 1234) and you're away — get remote sensor measurements or remote control your robot using your phone's tilt-sensor (for example).



See the module's user guide or visit http://mindkits.co.nz for more info.

Pro Tips

Running out of digital pins? Most 'analog' pins are also digital pins, so:

```
pinMode(A0, OUTPUT);
digitalWrite(A0, HIGH);
```

sets pin A0 to output and and pulls it high. See http://arduino.cc/en/Tutorial/AnalogInputPins Regular digital pins don't works as analog inputs, and A6 and A7 are analog-only.

Fast direct access to port registers with DDRx, PORTx, and PINx is done the usual way; see http://www.arduino.cc/en/Reference/PortManipulation

Analog pins have a small (10nF) smoothing capacitor to GND to reduce noise. If you have very fast changing signals you can carefully desolder them.

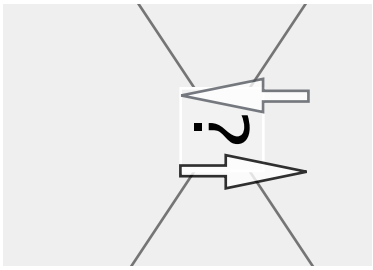
Digital pins RX-I/D0, D10, and D11 have 100Ω buffer resistors for protection.

ISP programming is available (through the ISP header) to burn code directly rather than using the bootlader, say if you need another 2 kB of code space. You can download and save the flash ROM contents to a hex file in case you change your mind.

This is an Arduino-compatible board, but you can still develop using regular AVR-GCC, Atmel Studio, and so on. Refer to the Arduino Nano pinout guide for the chip's actual port pins and registers (DDRA etc).

You can power BrainBoard from USB and external DC at the same time.

BrainBoard has a built-in FTDI USB-to-UART bridge connected to JP1, which you can use for other projects like programming an Arduino Micro or Cheapduino, which don't have the chip. Connect TX and RX to JP1, GND and 5V pins. TX-O (transmit, output) on JP1 is the signal from your device, and RX-I (receive, input) the signal to your device. Arrows show the signal direction from FTDI (JP1) to AVR CPU (JP8), so connect as appropriate. Careful with these pins, hooking them up wrong might blow up the FTDI chip or you device.



The Arduino IDE isn't perfect, but you can always use another editor. The cross-platform (OS X, Linux, Windows) **Sublime Text 2 + Arduino-like IDE + SublimeClang** make a really nice development environment. Unfortunately Sublime Text 2 isn't free, but it might be worth it for someone doing a lot of coding. There are lots of free + good editors out there too, like Notepad++ and TextWrangler.

bb_template.ino | brainboard.h

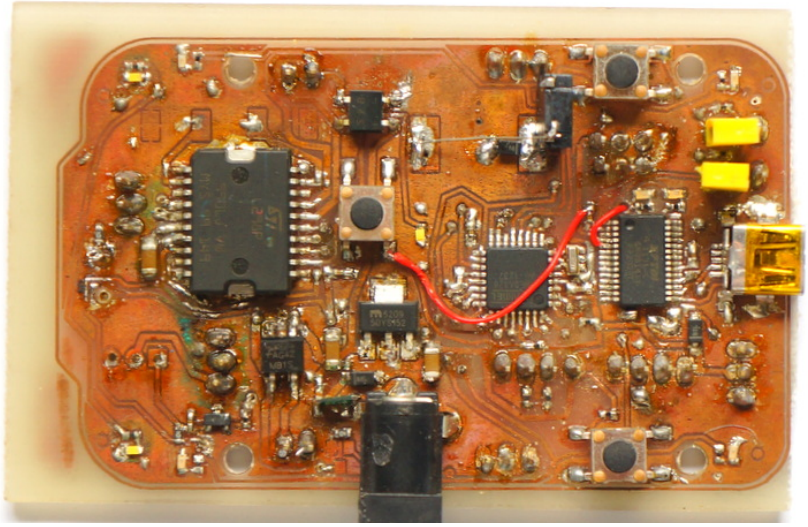
```
1  /*
2  BrainBoard Notes
3  ------------------------
4  1.  Maximum voltage on any pin is 5V.
5  2.  Power BrainBoard with 5-9V DC (centre +ve) using the barr
6  let out all the smoke and void your warranty.
7  3.  Make sure you have external power (from barrel socket or
8  4.  There's a handy Arduino Nano pin reference at http://foru
9  5.  Some useful pin definitions follow.
10 */
11
12 // Digital inputs
13 #define SW1 2            // left push button, active high
14 #define SW2 3            // right push button, active high
15
16 // Digital outputs
17 #define FRONT_LEDS  12   // white leds
18 #define LEFT_LEDS   13   // left blue leds
19 #define RIGHT_LEDS  8    // right blue leds
20 #define BUZZER      9    // use tone() to drive the BUZZER.
21 #define MOTOR1_A    4    // motor 1 (left) drive pin 1
22 #define MOTOR1_B    5    // motor 1 (left) drive pin 2, PWM
23 #define MOTOR2_A    6    // motor 2 (right) drive pin 1, PWM
24 #define MOTOR2_B    7    // motor 2 (right) drive pin 2
```

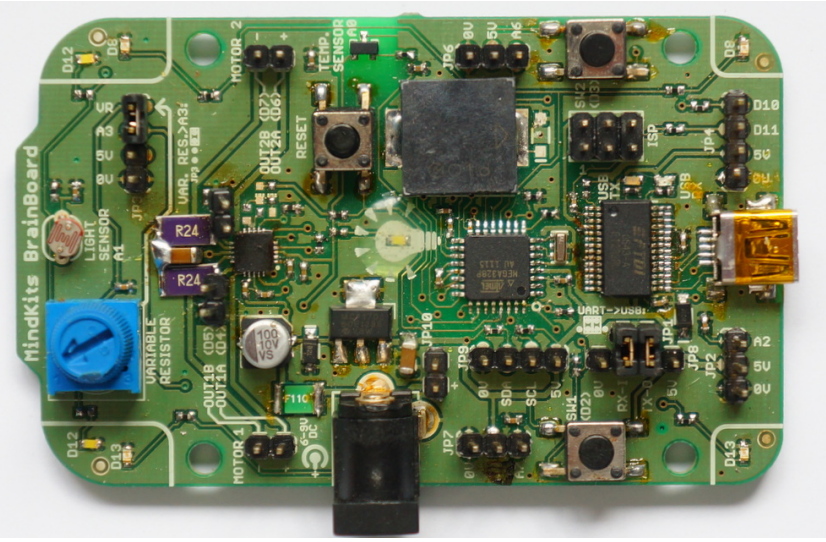
The Birth of BrainBoard

Ever wonder how electronic products are born? Here's a quick pictorial of the prototyping process we used on BrainBoard.

The Prototype
...to prove the concept.

- Press 'n Peel toner-transfer to PCB
- PCB etched by hand
- Hand-drilled vias, plugged with wire
- Hand assembled and soldered



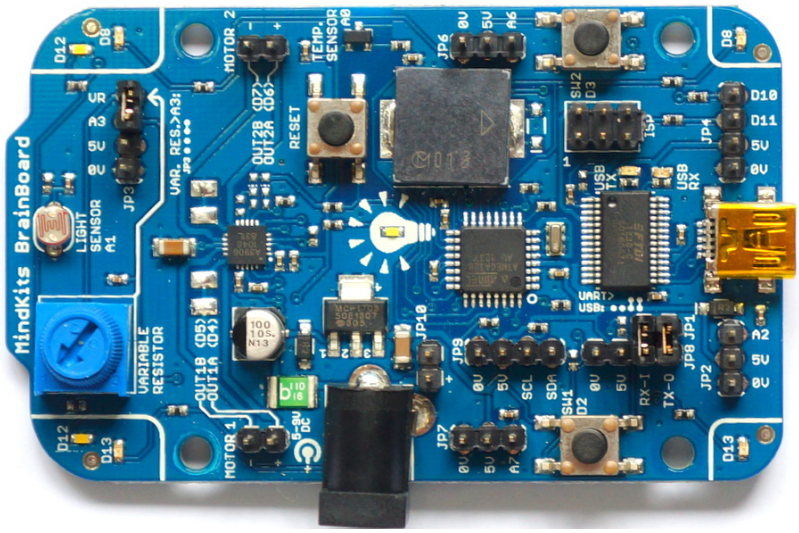


Design Verification Prototype
...to test and tweak the design and confirm manufacturability.

- Commercially-produced PCB
- Hand assembled and soldered

Factory Finished Board

And finally, a robot controller...
assembled and tested by robots!



Show Us Yer Projects!

We want to hear about all the awesome projects you're doing with BrainBoard! Drop us an e-mail or hit us up on twitter — photos are great too. We won't use your project for marketing or anything like that (unless we ask and get your permission of course), we're just keen to see where in the world our creations end up.

Getting Help

If something doesn't make sense, doesn't work as expected, or you have any comments or suggestions then get in touch with us and we'll sort you out.

Contact support@mindkitsinvent.com

twitter @MindKits

Facebook www.facebook.com/MindKits