

How to Reuse the IoT Badge

Created by Jared Bellows, last modified by James Brown on May 17, 2018

Introduction

The badge was designed with reuse as a top priority. As such we have two methods of reusing the badge. First, the badge is designed to emulate the LilyPad USB. As such, you can use the Arduino IDE to program the badge. Second, the current firmware has a simple scripting language implemented (Bitlash). Functions have been exposed that allow basic control of the components of the badge.

This page is intended to be a launching point for your creative ideas. Please add your ideas below.

The Tech Summit presentation on [How to Upcycle Your IoT Badge After Tech Summit](#) is available as a connect recording. Skip ahead to 37:25 for the start of the badge portion of the session.

Power

The badge can be powered from 2x AA batteries as well as through the USB connector. It is safe for both power sources to be attached at the same time. When both are attached, the badge will pull power from the batteries. Consequently, if you leave batteries in the badge while it is plugged into USB, the batteries will still be depleted over time.

NOTE: The badge operates at 3.3V. Be careful when connecting other components to use 3.3V.

Using the Arduino IDE

Download the IDE

The IDE for Arduino includes the compiler and uploading tools necessary to program the board. You can download from the link below.

<https://www.arduino.cc/en/Main/Software>

Unlock the Badge

The badge as delivered during Tech Summit has a locked bootloader, preventing code upload via the USB port. In order to unlock the bootloader a command must be issued to the board.

1. Open the Arduino IDE.
2. Plug the IoT Badge into the computer using a USB cable.
3. Select the serial port of the badge by opening the Tools -> Port menu and finding the port that indicates it is the LilyPad.
4. Open the Serial Monitor by opening Tools -> Serial Monitor in the IDE.
5. Check that the the baud rate set in the lower left is 9600 and "Newline" is selected as the line ending.
6. In the top text input box enter **p_wrt** and press Enter.
7. You should get back a single ">" prompt in response. If an error occurs, simply issue the p_wrt command again.

The badge is now unlocked to allow for code upload via the IDE.

Configure the IDE to program the badge

The badge emulates the LilyPad USB board. In order to have the code compiled correctly for that platform, you need to select the "LilyPad Arduino USB" board under Tools -> Board.

Badge Parts Pin Definitions

A [schematic](#) of the board is available. Information about the board is [available in git](#). The badge has the following components onboard:

Component	Pins	Notes
OLED Display	SDA SCL	I2C Address: 0x3c This is a 128x64 monochrome OLED display with the SSD1306 controller. The following libraries can be used: https://github.com/adafruit/Adafruit_SSD1306 https://github.com/greiman/SSD1306Ascii https://github.com/Defragster/ssd1306xled.git https://github.com/olikraus/U8glib_Arduino
LPD8806 RGB LED Controller	Data Pin: 4 Clock Pin: 5	https://github.com/adafruit/LPD8806.git
IR Receiver	Receive Pin: 7	The included IR library with the IDE will work with this device (Vishay TSOP2438 Datasheet).
Front IR LED	Anode: 13 Cathode: 9	Pin 13 is used by the IR library to send IR signals. In order to allow multiple LED options, the cathode of each LED is tied to a pin. In order to use this LED, Pin 13 needs to be set as OUTPUT and HIGH, pin 9 needs to be set as OUTPUT and LOW, and pin 11 needs to be as INPUT
Back IR LED	Anode: 13 Cathode: 11	Pin 13 is used by the IR library to send IR signals. In order to allow multiple LED options, the cathode of each LED is tied to a pin. In order to use this LED, Pin 13 needs to be set as OUTPUT and HIGH, pin 11 needs to be set as OUTPUT and LOW, and pin 9 needs to be as INPUT
HM-10 Bluetooth Low Energy Radio (BLE) Radio	RX: 0 TX: 1	Using the standard Serial1 object will allow communication with the BLE radio. The default baud rate is 9600 and there are many configuration options that are controlled via AT commands. You can find datasheets online that describe the commands. Make sure you check the radio firmware version vs. the docs you are using. Commands and responses change between versions. The radio is made by JNHumao Technology Company but their website is generally blocked due to security concerns. You can find versions of the datasheet elsewhere on the web. Here are some examples: https://seeedoc.github.io/BLE_Bee/res/Bluetooth40_en.pdf https://wiki.microduino.cc/images/f/fc/Bluetooth40_en.pdf

Component	Pins	Notes
CdS Light Sensor	Power: 30 Analog Pin: 7	In order to conserve battery on the badge, the light sensor is not powered except when taking readings. In order to read the value, you will need to power it by setting pin 30 HIGH. Below is example code for reading the light sensor. <pre>pinMode(30,OUTPUT); digitalWrite(30, HIGH); uint16_t retVal = analogRead(7); digitalWrite(30, LOW);</pre>
Battery Voltage	Analog Pin: A0	It is possible to get the battery voltage using this pin and an <i>analogRead</i> call. The ADC used is a 10-bit ADC (1024 values) with a voltage reference of 3.3V.
Buttons	Button 1: A5 Button 2: A4 Button 3: A3 Button 4: A2 Button 5: A1	While these are on analog pins, it is recommended to read them as digital pins. In order to use them, you will also need to enable the pull-up resistors (INPUT_PULLUP). This means that when reading the pins, the unpressed value will be HIGH and pressed value will be LOW.
Tilt Switch	Pin: 10	HIGH indicates that the badge is hanging normally (with the buttons towards the top of the badge). LOW indicates that the badge is flipped 'up' meaning that the buttons are at the bottom of the badge and the lettering is 'right side up' to the observer.
SPI Flash	CS: 14	The external flash chip (S25FL1) is connected to the standard SPI pins. It is recommended to use a library when accessing this chip. https://github.com/BleepLabs/S25FLx A special note about flash storage: an address can only be written to once between erases. The smallest page that can be erased on these chips is 4kB.

Using the Existing Firmware

The firmware shipped with the badge contains a slimmed down version of Bitlash. This allows for control of most of the components via commands sent on the USB serial port. It is possible to write your own functions that would run without needing to be attached to a computer, or to send the commands over the serial connection from a program running on a computer. You can find all the original code in [these git repos](https://git.corp.adobe.com/techsummit2017) [https://git.corp.adobe.com/techsummit2017].

Configure Serial Port

It is possible to communicate with the badge via a serial connection. If you have a serial terminal you prefer already, you may use it, otherwise you can download the Arduino IDE which includes a serial terminal. Information about getting and installing the IDE are available above. The badge communicates over serial at 9600 baud. Below are instructions on how to open and use the "Serial Monitor" that is part of the IDE.

1. Open the Arduino IDE.
2. Plug the IoT Badge into the computer using a USB cable.
3. Select the serial port of the badge by opening the Tools -> Port menu and finding the port that indicates it is the LilyPad.
4. Open the Serial Monitor by opening Tools -> Serial Monitor in the IDE.
5. Check that the the baud rate set in the lower left is 9600 and "Newline" is selected as the line ending.

At this point you may type **help** for information about available commands or try some of the commands below.

Command Reference

In addition to standard control commands, we have commands to control the components on the board. Those are described below.

Command	Description
rgb(led, red, green, blue)	Possible "led" values are -1, 0, 1. "red", "green", "blue" accept values from 0 to 127
o_clear	Clears the OLED display
o_println	Outputs the string with a newline following using the currently selected font (see o_font)
o_print	Outputs the string using the currently selected font (see o_font)
o_2x	Set the font size to be 2x designed font size
o_1x	Set the font size to be the designed font size
o_cursor(x, y)	Place the cursor of the display to the specified x,y coordinates, with 0, 0 being the upper left corner
o_orient(direction)	Direction of 0 is when the badge is oriented with the buttons on top. Direction of 1 is when the badge is oriented with the buttons on the bottom.
o_font("name")	Use the following names (in quotes) to select the font. For example: <code>o_font("sys5x7");</code> There are 4 fonts installed, but only one of them is a typical font. sys5x7 : A typical system font with all printable characters in 128 character ASCII table. The size of each character at 1x is 5px wide by 7px tall small : Only has numbers 0 - 9 and the character is the full height of the display, placing the number in the middle of the display horizontally. large : Only has numbers 0 - 9 and the character is the full height of the display, placing the number in the middle of the display horizontally. images : The glyphs for this font are full screen (128x64). There are 51 glyphs beginning with the "A" character in the ASCII table.
tilt	Returns a 0 or 1 depending on the orientation of the badge.
light	Returns a value between 0 and 1024.

Command	Description
buttons	Returns a bitmap of the buttons pressed, thereby allowing detection of multiple button presses.
fl_er_4(high_addr, low_addr)	This will erase a 4k page from the external flash at the specified address. The external flash takes a 32-bit address, but our version of bitlash only supports 16-bit integers, therefore the address is broken into the high 16 bits and the low 16 bits.
p_wrt	Unlock the bootloader so that the badge can be programmed via USB.

Hello World using Node.js

One option for controlling your badge is to create an application that executes bitlash commands remotely. An example of a simple wall clock is available as a node.js application. This will simply read the time from your computer and display it on the badge. This is intended as a simple application to demonstrate remote execution on the badge, not the extent of what could be done.

1. Download node.js for your platform from <https://nodejs.org/en/>
2. Install the required serial package via `npm install serialport`
3. Install the required time formatting package via `npm install strftime`
4. Download the node program from <https://git.corp.adobe.com/loTBadge/sampleNodeApp>
5. Attach your badge to the computer via USB
6. Run the program via `node badgecontrol.js` to get a list of ports and find your badge
7. Run the program via `node badgecontrol.js <portname>` with the port you found in the previous step

Your badge should now be displaying a basic clock with hours, minutes and seconds.

Fun with Python

[@Marc Abramowitz](#) wrote some Python code for interacting with the badge. See the [GitHub repo](#).

How to wipe your badge data

We went to some effort to prevent your badge from holding any personally identifiable information. However, if you are still concerned about what data might be on your badge, you can wipe the flash memory with the following bitlash commands:

```
p=2;i=0;function wipe{fl_er_4(p,i*4096);i++;if(i==16){i=0;print"page",p;p++;if(p==9)p=0xE;if(p==0x40){print"finished";stop;rm wipe;}}run wipe, 1
```

You can do this via any serial terminal program (putty, the Arduino IDE, realterm, etc.) Follow these steps:

1. Plug the IoT Badge into the computer using a USB cable.
2. Select the correct serial port for the badge in your terminal program.
3. Check that the the baud rate set to 9600 and the line ending is a newline.
4. Paste the above line into the terminal and press Enter.
5. You should see a set of "page n" (where n is a number) lines slowly printed (about 1 per second)
6. When the process completes it will print "finished"

Here are the specific steps if you are using the Arduino IDE


1. Open the Arduino IDE.
2. Plug the IoT Badge into the computer using a USB cable.
3. Select the serial port of the badge by opening the Tools -> Port menu and finding the port that indicates it is the LilyPad.
4. Open the Serial Monitor by opening Tools -> Serial Monitor in the IDE.
5. Check that the the baud rate set in the lower left is 9600 and "Newline" is selected as the line ending.
6. In the top text input box paste in the above command and press Enter.
7. You should see a set of "page n" (where n is a number) lines slowly printed (about 1 per second)
8. When the process completes it will print "finished"


Community Ideas


Please add your ideas here for things to do with the badge or success you've had repurposing the badge


arduino


28 Comments

- 

Benoit Ambry
There's no reference to the tilt detector in the component list.
- 

Jared Bellows
I added the reference. Thanks for catching that omission.
- 

Jason Boyer
Will this board be open sourced?
- 

Jared Bellows
What do you mean by open sourced? Do you mean publicly or within Adobe?
- 

Adrian Sandu
A public version of the design would be nice but within I guess it would be interesting for us first. Also, the current firmware would be interesting for finding bugs 😊

**Jared Bellows**

Unlikely to be public, but the current code and hardware design can be found here, git.corp.adobe.com/techsummit2017.

**Laurie Byrum**

@Jared Bellows can you put the badge wipe instructions on here?

**Jared Bellows**

@James Brown, do you think you can put together a NodeJS script that would wipe the external flash? At least from the game data to the end.

**James Brown**

Yes, I'll put something together. I think we could do it via only bitlash as well.

**James Brown**

@Laurie Byrum - I put instructions at the bottom of the wiki to wipe the badge data.

**Raffaele Sena**

I found out that with the battery connected, I couldn't "talk" to the board via the IDE or a serial terminal. So, if you don't see the board show up in the IDE even if it's connected to your computer, remove the battery!

**Jared Bellows**

You should still be able to communicate over serial. I have done it many times.

**James Brown**

I've seen this occasionally, especially with old / low batteries. Generally it is best to power the badge from USB if you are talking to it over USB unless you have special circumstances that require both batteries and USB. For example, if you just plug it in briefly to update a bitlash function or execute a command and you plan to unplug it almost immediately.

**Raffaele Sena**

How much space is available for program with the current bitlash firmware ?

**Jared Bellows**

We have allocated 16k of external flash for Bitlash functions to be saved.

**James Brown**

Note that bitlash was designed to work with a small memory space (on the order of 1-2K). The algorithm for finding functions by name is inherently inefficient. Also, because of the nature of flash memory, we can't easily erase and rewrite the space in function-sized chunks. Consequently, the more functions you create / replace, the slower and slower your badge will run.

Also note that in the default firmware there is a watchdog task running that will reset your badge if you spend 8 seconds or more executing a single bitlash command. So, if you create a function to display all the numbers from 0 to 32000 for example, you might cause your badge to reboot.

**Mike Ashcraft**

Please add the link mentioned above for the AT command reference for the BLE

**James Brown**

I added info and links to the table

**John Peterson**

A comment on the `o_cursor(x,row)` function: The *x* coordinates are in *pixels* (0..127), but the *row* coordinate is in *lines* (0..7).

To position characters, multiply $x * 6$ to set a character position when using the `sys5x7` font, and multiple $x * 12$ (and $rows * 2$) when using the `o_x2` double font.

Any clues on how to use the "small" or "large" fonts? I've found it difficult to get these to show up (yes, I know it's numbers only)

**John Peterson**

Turn your badge into a [stock ticker](#).

**Marc Abramowitz**

Thank you for creating the badge also for publishing this doc on how to connect to it!

I was able to connect to my badge using the Serial Monitor in the Arduino IDE and I can set the colors of the LED successfully (which was exciting for my 7 year old!).

I've been trying to use `o_print` to print stuff on the OLED screen, but I'm not seeing anything. I'm wondering if I'm doing something wrong or maybe did my OLED get damaged? How could I tell?

**Mike Ashcraft**

If you have not replaced the code, a quick way to test the OLED display is by pressing buttons 1 and 5 simultaneously. It should display the badge ID, firmware version and a few other details.

**Marc Abramowitz**

The two LEDs blink green (probably from pressing button 1?) but nothing on the OLED, so maybe it's dead?

I wonder if someone has a badge they're not using? I was going to use this so that my 7 year old can toy around with Arduino at home.

**James Brown**

It sounds like your display may be dead. I'll contact you directly to work out what's wrong.

**Nipun Poddar**

I am not getting any response for the HM10 BLE commands. I have tried to wake the BLE radio, but I am not getting OK+WAKE result from that.

I have used Serial1, as well as a serial software object with 0,1 ports.

PS: I have wiped out the firmware.

**James Brown**

Unless the radio is actually asleep (put to sleep using the sleep commands which only work in the correct mode) you will simply get an OK response to an AT command. Try that first.

In fact, the easiest thing is to load a "serial repeater" sketch (https://git.corp.adobe.com/techsummit2017/Serial_Repeater) which simply maps everything coming in over USB to the radio and vice versa. That way you can easily issue commands via your serial terminal and see how the radio responds.

You will need to carefully review the entire HM10 documentation (linked above) in order to understand how this radio works. It is a somewhat complicated subsystem that requires significant effort to put to use. I suggest studying the code in ble.h (from the source code repo linked above) to understand how we configured and used the radio.

Drop another question here if you continue to have problems.

**Manish Gupta**

I am experiencing a similar issue at my end.

**James Brown**

See my reply above to Nipun
