Micro:bit Serial Communication for Tethered Applications (Preliminary)
(Version 0.2, 10-Jan-2019)
John Maloney

Overview

The immediate goal of this project is to allow communication between a BBC micro:bit and Code.org's Maker application over USB-serial cable to support the Code.org physical computing curriculum. The implementation consists of firmware written in C++ and based on the micro:bit DAL (data abstraction layer) library running on the micro:bit and a client library written in JavaScript running on Node.js.

The design takes into consideration the potential for potential future possibilities such as porting the firmware to run on other microcontroller boards (e.g. AdaFruit Circuit Playground Express) or using the firmware in other applications (e.g. a Scratch extension).

Serial Protocol

The serial protocol is based on Firmata. Although this initial implementation will use only a subset of Firmata, it will follow Firmata command formats and conventions, allowing for potential future extension into a compete Firmata implementation.

Firmata is based on the robust and byte-efficient MIDI protocol. Framing is accomplished by using the high bit of each byte to distinguish command bytes from data bytes. Since the high bit is used for that, data bytes have only seven bits of data.

MIDI supports fixed-length messages of one to three bytes and variable-length "system exclusive" messages. System exclusive messages provide a way to extend the protocol with application-specific messages. In this project, they will be used to add commands for micro:bit specific features such as the 5x5 LED display and to report events.

Outputs

Preliminary set of protocol-level output commands:

```
clearDisplay
setDisplay <4 data bytes>          // sets the on/off state of all LEDS at once
setDisplayPixel x y brightness     // brightness range 0-127 (also used for on/off)

displayNumber num                  // scroll number across display
displayString string               // scroll string across display

setPinMode pin mode                // pin: edge connector pin number
                                   // mode: input, input-pullup, output
setDigitalPin pin onOff            // turn digital pin on or off
setPWMPin pin dutyCycle            // duty cycle is 0-100%
```

Inputs

The system supports streaming of data from input pins and built-in sensors. Streaming is off by default. The client can request streaming of one or more analog data channels or digital ports. (A digital port represents a group of eight input pins as a single packed byte, where each bit represents the state of one pin.) The client can specify the streaming update frequency. The update frequency applies to all channels and ports being streamed.

There are 16 analog channels. The first 8 are reserved for analog input pins. (Although the micro:bit has only six analog input pins, the AdaFruit CPX has 8.) Here is a list:

chan	what
0-7	analog input pins (edge connector pins 0-4 and 10; 6-7 not used)
8	accelerometer x
9	accelerometer y
10	accelerometer z
11	light sensor
12	temperature sensor
13	magnetometer x (cpx: sound level)
14	magnetometer y (cpx: IR distance sensor)
15	magnetometer z (cpx: not used)

The micro:bit supports three 8-bit digital ports:

port	edge connector pins
0	0-7
1	8-15
2	16-20

Note: Pins 17-18 of the edge connector are 3.3 volts, not I/O pins.

Updates require three bytes for each analog channel or digital port. Because the USB-serial channel has limited throughput, there is a tradeoff between the number of channels streamed and update frequency. For example, streaming three axes of accelerometer data at 100 updates/second requires 900 bytes/sec. While the theoretical capacity of a USB-serial connection running at 115200 baud is about 11.5 kilobytes/sec, actual performance can be 5 kilobytes/sec or less.

Preliminary set of protocol-level streaming control commands:

```
streamAnalog channel onOff      // turn streaming on or off for given channel
streamDigital port onOff        // turn streaming on or off for given port
```

Events

The micro:bit sends messages to report events such as the A or B buttons being pressed/released or the micro:bit being shaken. To keep things simple, and because such events are expected to be infrequent (i.e. at most a few events per second), all the following events are sent whenever they occur, and any desired filtering and event dispatching is handled by Javascript.

```
button A/B down
button A/B up
button A/B long press
shake
free-fall
```

Event messages specify the DAL device and event ID's represented as 14-bit unsigned integers (i.e. two Firmata data bytes each).

System Commands

Preliminary list of system commands:

```
reset               // turn off streaming, clear the display, put pins in input mode
firmataVersion      // return Firmata major and minor version numbers
firmwareVersion     // return the name and major and minor version of firmware
microbitInfo        // return info about the board and software (e.g. board
                    // revision, DAL version, SoftDevice version, etc.)
```