

GitHub Gist | Search... All gists GitHub New gist

firread / hidmouse_diff.ino Last active 2 minutes ago

Code Revisions 3 Embed <script src="https://gist.github.com/firread/1132000">Edit Delete Star 0 Download ZIP

Revisions

firread revised this gist 2 minutes ago.

1 changed file with 110 additions and 54 deletions.

Unified Split

```

164  hidmouse_diff.ino View
...
1  @@ -1,17 +1,17 @@
2  - This is an example for our nRF51822 based Bluefruit LE modules
3
4  - Pick one up today in the adafruit shop!
5
6  - Adafruit invests time and resources providing this open source code,
7  - please support Adafruit and open-source hardware by purchasing
8  - products from Adafruit!
9
10 - MIT license, check LICENSE for more information
11 - All text above, and the splash screen below must be included in
12 - any redistribution
13 ****
14 */
15 This example shows how to send HID (keyboard/mouse/etc) data via BLE
16 Note that not all devices support BLE Mouse!
17 - OSX, Windows 10 both work
...
18 @@ -20,50 +20,57 @@
19 */
20
21 #include <Arduino.h>
22
23 #include <SPI.h>
24 #if defined (_VARIANT_ARDUINO_DUE_X_)
25 - #include <SoftwareSerial.h>
26 #endif
27
28 #include "Adafruit_BLE.h"
29 // #include "Adafruit_BluefruitLE_SPI.h"
30 #include "Adafruit_BluefruitLE_UART.h"
31 -
32 #include "BluefruitConfig.h"
33
34 ****
35 APPLICATION SETTINGS
36
37 - FACTORYRESET_ENABLE Perform a factory reset when running this sketch
38 -
39 - Enabling this will put your Bluefruit LE module
40 - in a 'known good' state and clear any config
41 - data set in previous sketches or projects, so
42 - running this at least once is a good idea.
43 -
44 - When deploying your project, however, you will
45 - want to disable factory reset by setting this
46 - value to 0. If you are making changes to your
47 - Bluefruit LE device via AT commands, and those
48 - changes aren't persisting across resets, this
49 - is the reason why. Factory reset will erase
50 - the non-volatile memory where config data is
51 - stored, setting it back to factory default
52 - values.
53 -
54 - Some sketches that require you to bond to a
55 - central device (HID mouse, keyboard, etc.)
56 - won't work at all with this feature enabled
57 - since the factory reset will clear all of the
58 - bonding data stored on the chip, meaning the
59 - central device won't be able to reconnect.
60 MINIMUM_FIRMWARE_VERSION Minimum firmware version to have some new features
61 ****
62 - #define FACTORYRESET_ENABLE 0
63 - #define MINIMUM_FIRMWARE_VERSION "0.6.6"
64 ****
65
66 - Adafruit_BluefruitLE_UART ble(BLUEFRUIT_HWSERIAL_NAME, BLUEFRUIT_UART_MODE_PIN);
67
68 // A small helper
69 void error(const __FlashStringHelper*err) {
...
69 - @ -76,14 +83,21 @ void error(const __FlashStringHelper*err) {
70 - @brief Sets up the HW on the BLE module (this function is called
71 - automatically on startup)
72
73 - ****
74 void setup(void)
75 {
76
77 - while (!Serial); // required for Flora & Micro
78 - delay(500);
79
80 void setup(void)
81 {
82
83 - Serial.begin(115200);
84 - Serial.println(F("Adafruit Bluefruit HID Mouse Example"));
85
86 - Serial.println(F("-----"));
87
88 /* Initialise the module */
89
...
89 - @ -99,7 +113,7 @ void setup(void)
90 {
91
92 - /* Perform a factory reset to make sure everything is in a known state */
93 - Serial.println(F("Performing a factory reset: "));
94
95 - if ( ! ble.factoryReset() ) {
96 -     error(F("Couldn't factory reset"));
97
98 - Serial.begin(115200);
99 - Serial.println(F("Adafruit Bluefruit HID Mouse Example"));
100
101 - Serial.println(F("-----"));
102
103 /* Initialise the module */
104
105 {
106
107 - /* Perform a factory reset to make sure everything is in a known state */
108 - Serial.println(F("Performing a factory reset: "));
109
110 - if ( ! ble.factoryReset() ) {
111 -     error(F("Couldn't factory reset"));
112
113
114
115
116
117

```

```

100   /* Perform a factory reset to make sure everything is in a known state */
101  Serial.println(F("Performing a factory reset: "));
102 - if ( ! ble.factoryReset() ){
103 -   error(F("Couldn't factory reset"));
104 }
105 ...
106 @ @ -111,6 +125,12 @ void setup(void)
107   /* Print Bluefruit information */
108   ble.info();
109
110
111 // This demo only available for firmware from 0.6.6
112 if ( !ble.isVersionAtLeast(MINIMUM_FIRMWARE_VERSION) )
113 {
114 ...
115 @@ -147,41 +167,76 @ void setup(void)
116   @brief Constantly poll for new command or response data
117   /*
118   /*****
119   -void loop(void)
120   -{
121 -  Serial.println(F("x,y = "));
122
123
124 // Check for user input and echo it back if anything was found
125 - char input[BUFSIZE+1];
126 - getUserInput(input, BUFSIZE);
127
128 Serial.println(input);
129
130
131 // Press (and hold) the Left mouse's button
132 if ( ble.sendCommandCheckOK(F("AT+BleHidMouseButton=L,press")) )
133 {
134   // delay a bit
135   delay(250);
136
137   // Mouse moves according to the user's input
138 - ble.print(F("AT+BleHidMouseMove="));
139 - ble.println(input);
140
141 - if( ble.waitForOK() )
142 - {
143
144     Serial.println(F("OK!"));
145   } else
146   {
147     Serial.println(F("FAILED!"));
148   }
149
150   // Way for user to release left button
151 - Serial.println(F("Enter anything to release Left Button"));
152 - getUserInput(input, BUFSIZE);
153
154   // Release the Left mouse's button
155   ble.sendCommandCheckOK(F("AT+BleHidMouseButton=0"));
156
157 - }else
158
159   // Failed, probably pairing is not complete yet
160   Serial.println(F("Please make sure Bluefruit is paired and try again"));
161 ...
162 @ @ -193,18 +248,19 @ void loop(void)
163   @brief Checks for user input (via the Serial Monitor)
164   /*
165   /*****
166 - void getUserInput(char buffer[], uint8_t maxSize)
167 -{
168 -  memset(buffer, 0, maxSize);
169 -  while( Serial.available() == 0 ) {
170 -    delay(1);
171 -  }
172 -
173 -  uint8_t count=0;
174 -
175 -  do
176 -  {
177 -    count += Serial.readBytes(buffer+count, maxSize);
178 -    delay(2);
179 -  } while( (count < maxSize) && !(Serial.available() == 0) );
180 -}@@
181
182
183 // Change the device name to make it easier to find */
184 + Serial.println(F("Setting device name to 'Bluefruit clickKick'"));
185 + if ( ! ble.sendCommandCheckOK(F("AT+GAPDEVNAME=Bluefruit clickKick" )) ) {
186 +   error(F("Could not set device name"));
187 + }
188 +
189
190 // This demo only available for firmware from 0.6.6
191 if ( !ble.isVersionAtLeast(MINIMUM_FIRMWARE_VERSION) )
192 {
193 ...
194   @brief Constantly poll for new command or response data
195   /*
196   /*****
197   +void loop(void) {
198   +
199   + //-----Notification Input from Sensor
200   + Wire.beginTransmission(MPU);
201   + Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
202   + Wire.endTransmission(false);
203
204   + Wire.requestFrom(MPU, 14, true); // request a total of 14 registers
205   + Ax = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
206   + AcY = Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
207   + AcZ = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
208   + Tmp = Wire.read() << 8 | Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
209   + GyX = Wire.read() << 8 | Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
210   + GyY = Wire.read() << 8 | Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
211   + GyZ = Wire.read() << 8 | Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
212
213   + Serial.print(" | Ax="); Serial.print(Ax);
214   + Serial.print(" | AcY="); Serial.print(AcY);
215   + Serial.print(" | AcZ="); Serial.print(AcZ);
216   + Serial.print(" | Tmp="); Serial.print(Tmp / 340.00 + 36.53); //equation for temperature in
217   + //degrees C from datasheet
218   + Serial.print(" | GyX="); Serial.print(GyX);
219   + Serial.print(" | GyY="); Serial.print(GyY);
220   + Serial.print(" | GyZ="); Serial.println(GyZ);
221
222   + //Serial.println(F("x,y = "));
223
224 // Check for user input and echo it back if anything was found
225 + char input[BUFSIZE+1];
226 + getUserInput(input, BUFSIZE);
227
228 Serial.println(input);
229
230
231 //-----Mouse Action
232 + // BLE Services for Command mode
233 + // https://learn.adafruit.com/introducing-adafruit-ble-bluetooth-low-energy-friend/ble-
234 + //services#at-plus-blehidmousemove
235 +
236 // Press (and hold) the Left mouse's button
237 if ( ble.sendCommandCheckOK(F("AT+BleHidMouseButton=L,press")) )
238 {
239   // delay a bit
240   delay(250);
241
242   // Mouse moves according to the user's input
243   // Parameter: X Ticks (+/-), Y Ticks (+/-), Scroll Wheel (+/-)
244
245
246 //Right
247 + if (AcY > 0) {
248   ble.print(F("AT+BleHidMouseMove= 2,,,"));
249   ble.println(input);
250 }
251
252 //Up
253 + if(input){
254   ble.print(F("AT+BleHidMouseMove= ,-2,,,"));
255   ble.println(input);
256 }
257
258
259 + if ( ble.waitForOK() ){
260   Serial.println(F("OK!"));
261 } else
262 {
263   Serial.println(F("FAILED!"));
264 }
265
266 // Way for user to release left button
267 + Serial.println(F("Enter anything to release Left Button"));
268 + // getUserInput(input, BUFSIZE);
269
270
271 // Release the Left mouse's button
272 ble.sendCommandCheckOK(F("AT+BleHidMouseButton=0"));
273
274 + }else
275
276 {
277   // Failed, probably pairing is not complete yet
278   Serial.println(F("Please make sure Bluefruit is paired and try again"));
279 ...
280
281
282 @brief Checks for user input (via the Serial Monitor)
283 */
284
285 //*****
286 +// void getUserInput(char buffer[], uint8_t maxSize)
287 +{
288 + // Sets the first num bytes of the block of memory pointed
289 + // to
290 + // memset(buffer, 0, maxSize);
291 + // While ( Serial.available() == 0 ) {
292 + //   delay(1);
293 + // }
294 + // uint8_t count = 0;
295 + // do
296 + // {
297 + //   count += Serial.readBytes(buffer + count, maxSize);
298 + //   delay(2);
299 + // } while ( (count < maxSize) && !(Serial.available() == 0) );
299 + // }

```

 firmread revised this gist 4 minutes ago.

 1 changed file with 6 additions and 25 deletions.

31 hidmouse_diff.ino

[View](#)

```

...
@@ -10,8 +10,7 @@
10   MIT license, check LICENSE for more information
11   All text above, and the splash screen below must be included in
12   any redistribution
13

```

```

10   MIT license, check LICENSE for more information
11   All text above, and the splash screen below must be included in
12   any redistribution
13

```

```

10  MIT license, check LICENSE for more information
11  All text above, and the splash screen below must be included in
12  any redistribution
13  ****
14  -
15  /*
16   * This example shows how to send HID (keyboard/mouse/etc) data via BLE
17   * Note that not all devices support BLE Mouse!
18  */
19  @@
20
21  #include <Arduino.h>
22  #include <SPI.h>
23  #if not defined (_VARIANT_ARDUINO_DUE_X_) && not defined(ARDUINO_ARCH_SAMD)
24  #include <SoftwareSerial.h>
25  #endif
26
27  #include "Adafruit_BLE.h"
28  #include "Adafruit_BluefruitLE_SPI.h"
29  #include "Adafruit_BluefruitLE_UART.h"
30
31  #include "BluefruitConfig.h"
32
33  @@
34  #define MINIMUM_FIRMWARE_VERSION  "0.6.6"
35
36  /**
37  * Create the bluefruit object, either software serial...uncomment these lines
38  */
39  SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_TXD_PIN,
40  BLUEFRUIT_SWUART_RXD_PIN);
41
42  Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,
43  BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);
44
45
46  /* ...or hardware serial, which does not need the RTS/CTS pins. Uncomment this line */
47  // Adafruit_BluefruitLE_UART ble(BLUEFRUIT_HWSERIAL_NAME, BLUEFRUIT_UART_MODE_PIN);
48
49  /* ...hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user selected CS/IRQ/RST */
50  //Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI IRQ, BLUEFRUIT_SPI_RST);
51
52  /* ...software SPI, using SCK/MOSI/MISO user-defined SPI pins and then user selected CS/IRQ/RST */
53  //Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_SCK, BLUEFRUIT_SPI_MISO,
54  //BLUEFRUIT_SPI_MOSI, BLUEFRUIT_SPI_CS,
55  //BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
56
57  // A small helper
58  void error(const __FlashStringHelper*err) {
59  @@
60  #ifdef _-64,25 +63,7 @@
61  /*!
62   * @brief Sets up the HW an the BLE module (this function is called
63   * automatically on startup)
64  */
65
66  /**
67  * A small helper
68  void error(const __FlashStringHelper*err) {
69
70  /*!
71   * @brief Sets up the HW an the BLE module (this function is called
72   * automatically on startup)
73
74  /**
75  * ****
76  * While (!Serial); // required for Flora & Micro
77  * delay(500);
78
79  Serial.begin(115200);
80  Serial.println(F("Adafruit Bluefruit HID Mouse Example"));
81
82  Serial.println(F("-----"));
83
84  /* Initialise the module */
85
86
87
88
89

```

 firmread created this gist 5 minutes ago.

```

229 hidmouse_diff.ino
...
@@ -0,0 +1,229 @@
1  /**
2  * This is an example for our nRF51822 based Bluefruit LE modules
3  *
4  * Pick one up today in the adafruit shop!
5  *
6  * Adafruit invests time and resources providing this open source code,
7  * please support Adafruit and open-source hardware by purchasing
8  * products from Adafruit!
9  *
10 * MIT license, check LICENSE for more information
11 * All text above, and the splash screen below must be included in
12 * any redistribution
13  ****
14
15  /**
16   * This example shows how to send HID (keyboard/mouse/etc) data via BLE
17   * Note that not all devices support BLE Mouse!
18   * - OSX, Windows 10 both work
19   * - Android has limited support
20   * - iOS completely ignores mouse
21  */
22
23  #include <Arduino.h>
24  #include <SPI.h>
25  #if not defined (_VARIANT_ARDUINO_DUE_X_) && not defined(ARDUINO_ARCH_SAMD)
26  #include <SoftwareSerial.h>
27  #endif
28
29  #include "Adafruit_BLE.h"
30  #include "Adafruit_BluefruitLE_SPI.h"
31  #include "Adafruit_BluefruitLE_UART.h"
32
33  #include "BluefruitConfig.h"
34
35  /**
36  * APPLICATION SETTINGS
37
38  * FACTORYRESET_ENABLE Perform a factory reset when running this sketch
39
40  * Enabling this will put your Bluefruit LE module
41  * in a 'known good' state and clear any config
42  * data set in previous sketches or projects, so
43  * running this at least once is a good idea.
44
45  * When deploying your project, however, you will
46  * want to disable factory reset by setting this
47  * value to 0. If you are making changes to your
48  * Bluefruit LE device via AT commands, and those
49  * changes aren't persisting across resets, this
50  * is the reason why. Factory reset will erase
51  * the non-volatile memory where config data is
52  * stored, setting it back to factory default
53  * values.
54
55  * Some sketches that require you to bond to a
56  * central device (HID mouse, keyboard, etc.)
57  * won't work at all with this feature enabled
58  * since the factory reset will clear all of the
59

```

```

34 +
35 + Some sketches that require you to bond to a
36 + central device (HID mouse, keyboard, etc.)
37 + won't work at all with this feature enabled
38 + since the factory reset will clear all of the
39 + bonding data stored on the chip, meaning the
40 + central device won't be able to reconnect,
41 + MINIMUM_FIRMWARE_VERSION Minimum firmware version to have some new features
42 + //=====================================================================
43 + #define FACTORYRESET_ENABLE 0
44 + #define MINIMUM_FIRMWARE_VERSION "0.6.6"
45 +//=====================================================================
46 +
47 +
48 // Create the bluefruit object, either software serial...uncomment these lines
49 +/*
50 +SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_TXD_PIN,
51 +BLUEFRUIT_SWUART_RXO_PIN);
52 +
53 +Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,
54 +BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);
55 +*/
56 +
57 /* ...or hardware serial, which does not need the RTS/CTS pins. Uncomment this line */
58 +//Adafruit_BluefruitLE_UART ble(BLUEFRUIT_HWSERIAL_NAME, BLUEFRUIT_UART_MODE_PIN);
59 +
60 /*...hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user selected CS/IRQ/RST */
61 +//Adafruit_Bluefruit_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
62 +
63 /*...software SPI, using SCK/MOSI/MISO user-defined SPI pins and then user selected CS/IRQ/RST */
64 +//Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_SCK, BLUEFRUIT_SPI_MISO,
65 +BLUEFRUIT_SPI_MOSI, BLUEFRUIT_SPI_CS,
66 +BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
67 +
68 +// A small helper
69 +void error(const _FlashStringHelper*err) {
70 +    Serial.println(err);
71 +    while (1);
72 +}
73 +//*****************************************************************************
74 +/*! @brief Sets up the HW on the BLE module (this function is called
75 + automatically on startup)
76 +*/
77 +//*****************************************************************************
78 +void setup(void)
79 +{
80 +    // While (!Serial); // required for Flora & Micro
81 +    delay(500);
82 +
83 +    Serial.begin(115200);
84 +    Serial.println(F("Adafruit Bluefruit HID Mouse Example"));
85 +    Serial.println(F("-----"));
86 +
87 +    /* Initialise the module */
88 +    Serial.print(F("Initialising the Bluefruit LE module: "));
89 +
90 +    if ( !ble.begin(VERBOSE_MODE) )
91 +    {
92 +        error(F("Couldn't find Bluefruit, make sure it's in CoMmmand mode & check wiring?"));
93 +    }
94 +    Serial.println(F("OK!"));
95 +
96 +    if ( FACTORYRESET_ENABLE )
97 +    {
98 +        /* Perform a factory reset to make sure everything is in a known state */
99 +        Serial.println(F("Performing a factory reset: "));
100 +        if ( !ble.factoryReset() )
101 +            error(F("Couldn't factory reset"));
102 +    }
103 +
104 +    /* Disable command echo from Bluefruit */
105 +    ble.echo(false);
106 +
107 +    Serial.println("Requesting Bluefruit info:");
108 +    /* Print Bluefruit information */
109 +    ble.info();
110 +
111 +    // This demo only available for firmware from 0.6.6
112 +    if ( !ble.isVersionAtLeast(MINIMUM_FIRMWARE_VERSION) )
113 +    {
114 +        error(F("This sketch requires firmware version " MINIMUM_FIRMWARE_VERSION " or higher!"));
115 +    }
116 +
117 +    /* Enable HID Service (including Mouse) */
118 +    Serial.println(F("Enabling HID Service (including Mouse): "));
119 +    if ( !ble.sendCommandCheckOK(F("AT+BleHIDEn=On" )) )
120 +        error(F("Failed to enable HID (firmware >=0.6.6?)"));
121 +
122 +
123 +    /* Add or remove service requires a reset */
124 +    Serial.println(F("Performing a SW reset (service changes require a reset): "));
125 +    if ( ! ble.reset() ) {
126 +        error(F("Could not reset??"));
127 +    }
128 +
129 +    Serial.println();
130 +    Serial.println(F("Go to your phone's Bluetooth settings to pair your device"));
131 +    Serial.println(F("then open an application that accepts mouse input"));
132 +    Serial.println();
133 +
134 +    Serial.println(F("The example will try to draw a rectangle using the left mouse button with"));
135 +    Serial.println(F("your input"));
136 +    Serial.println(F("Parameters are a pair of 8-bit signed numbers (x,y) e.g:"));
137 +    Serial.println(F(" 100,100 : draw toward bottom right corner"));
138 +    Serial.println(F(" -100,-100: draw toward top left corner"));
139 +
140 +    Serial.println();
141 +
142 +    // Check for user input and echo it back if anything was found
143 +    char input[BUFSIZE+1];
144 +    getUserInput(input, BUFSIZE);
145 +
146 +    Serial.println(input);
147 +
148 +    // Press (and hold) the Left mouse's button
149 +    if ( ble.sendCommandCheckOK(F("AT+BleHidMouseButton=L,press")) )
150 +    {
151 +        delay(200);
152 +
153 +        // delay a bit
154 +        delay(200);
155 +
156 +        // Check for user input and echo it back if anything was found
157 +        char input[BUFSIZE+1];
158 +        getUserInput(input, BUFSIZE);
159 +
160 +        Serial.println(input);
161 +
162 +    }
163 +
164 +//*****************************************************************************
165 +/*! @brief Constantly poll for new command or response data
166 +*/
167 +//*****************************************************************************
168 +//*****************************************************************************
169 +void loop(void)
170 +{
171 +    Serial.println(F("x,y = "));
172 +
173 +    // Check for user input and echo it back if anything was found
174 +    char input[BUFSIZE+1];
175 +    getUserInput(input, BUFSIZE);
176 +
177 +    Serial.println(input);
178 +
179 +    // Press (and hold) the Left mouse's button
180 +    if ( ble.sendCommandCheckOK(F("AT+BleHidMouseButton=L,press")) )
181 +    {
182 +        // delay a bit
183 +        delay(200);
184 +
185 +        // Check for user input and echo it back if anything was found
186 +        char input[BUFSIZE+1];
187 +        getUserInput(input, BUFSIZE);
188 +
189 +        Serial.println(input);
190 +
191 +        // Release the Left mouse's button
192 +        if ( ble.sendCommandCheckOK(F("AT+BleHidMouseButton=L,release")) )
193 +        {
194 +            delay(200);
195 +
196 +            // Check for user input and echo it back if anything was found
197 +            char input[BUFSIZE+1];
198 +            getUserInput(input, BUFSIZE);
199 +
200 +            Serial.println(input);
201 +
202 +        }
203 +
204 +    }
205 +
206 +}

```

```
179 + // Press (and hold) the Left mouse's button
180 + if ( ble.sendCommandCheckOK(F("AT+BleHidMouseButton=L,press")) )
181 + {
182 +   // delay a bit
183 +   delay(250);
184 +
185 +   // Mouse moves according to the user's input
186 +   ble.println(F("AT+BleHidMouseMove"));
187 +   ble.println(input);
188 +
189 +   if( ble.waitForOK() )
190 +   {
191 +     Serial.println( F("OK!") );
192 +   }else
193 +   {
194 +     Serial.println( F("FAILED!") );
195 +   }
196 +
197 +   // Way for user to release left button
198 +   Serial.println( F("Enter anything to release Left Button") );
199 +   getUserInput(input, BUFSIZE);
200 +
201 +   // Release the Left mouse's button
202 +   ble.sendCommandCheckOK(F("AT+BleHidMouseButton=0"));
203 + }else
204 + {
205 +   // Failed, probably pairing is not complete yet
206 +   Serial.println( F("Please make sure Bluefruit is paired and try again") );
207 + }
208 +
209 +
210 +//*********************************************************************
211 +/*!*
212 + * @brief Checks for user input (via the Serial Monitor)
213 + */
214 +//*********************************************************************
215 +void getUserInput(char buffer[], uint8_t maxSize)
216 +{
217 +  memset(buffer, 0, maxSize);
218 +  while( Serial.available() == 0 ) {
219 +    delay(1);
220 +  }
221 +
222 +  uint8_t count=0;
223 +
224 +  do
225 +  {
226 +    count += Serial.readBytes(buffer+count, maxSize);
227 +    delay(2);
228 +  } while( (count < maxSize) && !(Serial.available() == 0) );
229 +}
```

