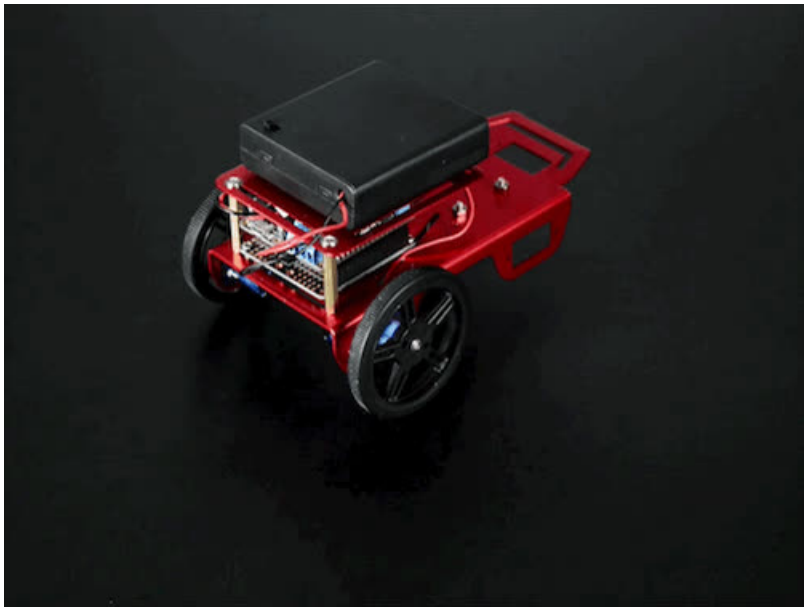
adafruit learning system

Bluefruit LE Feather Robot Rover

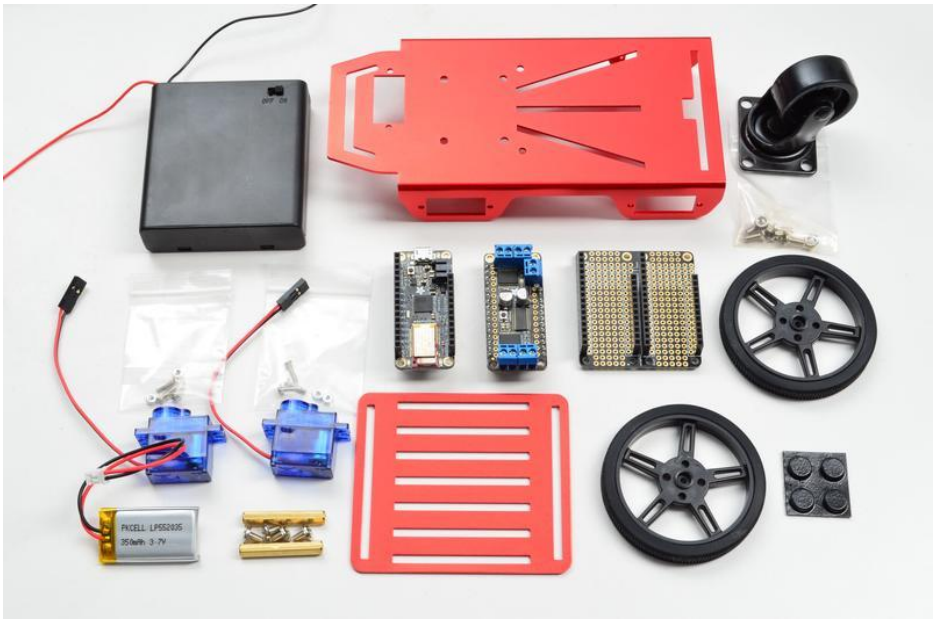
Created by James DeVito



Last updated on 2019-05-01 10:02:10 PM UTC

Overview

This little 3-wheeled robot can be put together in just an hour or two, and is a great place to start (or continue!) the adventure into robotics. You can control it with your iOS or Android phone over Bluetooth using Adafruit Bluefruit LE Connect!



Parts

Links to these products can also be found on the right sidebar

- Adafruit Feather 32u4 Bluefruit LE (https://adafru.it/keO)

- DC Motor + Stepper FeatherWing (http://adafru.it/2927)
- FeatherWing Doubler (https://adafru.it/kBi)
- 350mAh Lipo Battery (https://adafru.it/kBj)
- 4 X AA Holder with On/Off switch (http://adafru.it/830)
- Feather Stacking Headers (http://adafru.it/2830)
- Mini Robot Metal Frame
- Top Plate
- 2x DC Motors (in Micro Servo formfactor)
- 2x Wheels (https://adafru.it/kBk)
- 1x Support Wheel
- Rubber Bumper Feet (http://adafru.it/550)

The Mini Robot Rover Kit includes (http://adafru.it/2939):

- 2x Wheels (http://adafru.it/2744)
- 2x DC Motors (http://adafru.it/2941)
- 1x Support Wheel (http://adafru.it/2942)
- 1x Metal Chasis (http://adafru.it/2943)
- 1x Top Metal Plate (http://adafru.it/2944)

Also needed is the wonderful Adafruit Bluefruit LE Connect app (iOS (https://adafru.it/ddu) & Android (https://adafru.it/f4G))

NOTE: All components needed for this project are available in the store EXCEPT 4-40 screws & foam tape, needed for attaching the doubler & battery respectively.

Prerequisite Guides

Take a look over these guides and get familiar with these components!

- Adafruit Feather 32u4 Bluefruit LE (https://adafru.it/kcc)
- Adafruit Motor Shield (https://adafru.it/doc) (Motor FeatherWing is functionally identical)
- Bluefruit LE Connect (https://adafru.it/iCi)

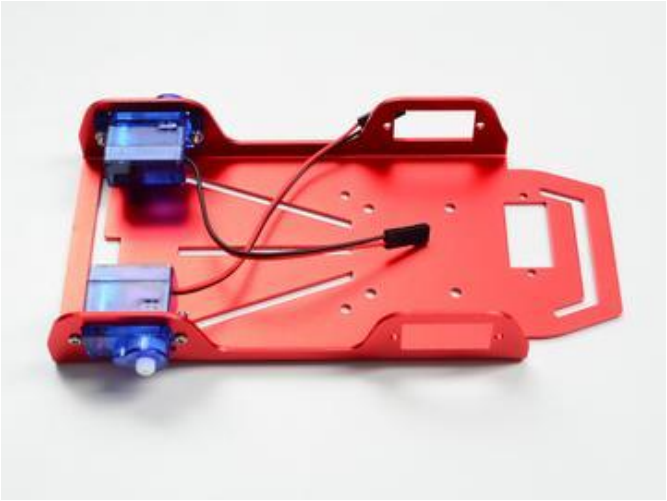
Wiring & Assembly

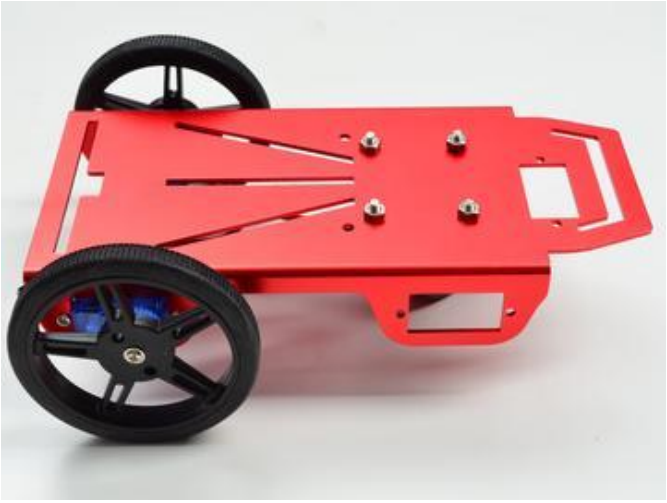
The wiring is pretty straight forward since the BLE Feather & the doubler do most of the connections for us :). The 4 x AA batteries will be used to power the Motor Featherwing & the compact 350mAh lipo will keep the BLE 32u4 Feather going.

- Connect **AA pack** + to **Motor Wing Motor** +
- Connect **AA pack** + to **Motor Wing Motor -**
- The **Right side Motor** goes into **M3**
- The **Left side Motor** goes into **M3**

Solder female headers to the doubler, male headers & terminal blocks to the Motor Featherwing, and stacking female headers to the BLE 32u4 Feather.

You don't **need** to use the Feather Doubler, you can also just use stacking headers for the BLE Feather and Motor Wing... we just think its very convenient and gives you some prototyping area for sensors or LEDs.

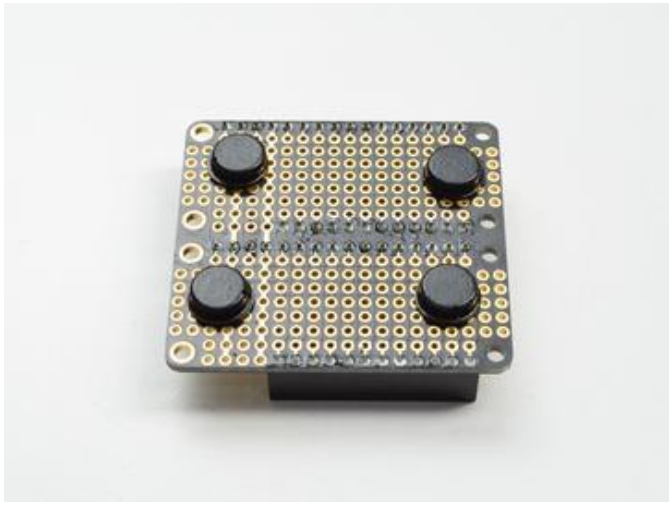




Flip over the chassis and attach the DC motors with the supplied screws. Make sure the wires are facing towards the center of the body (not critical, but my preference to not have the wheels stick out far behind).

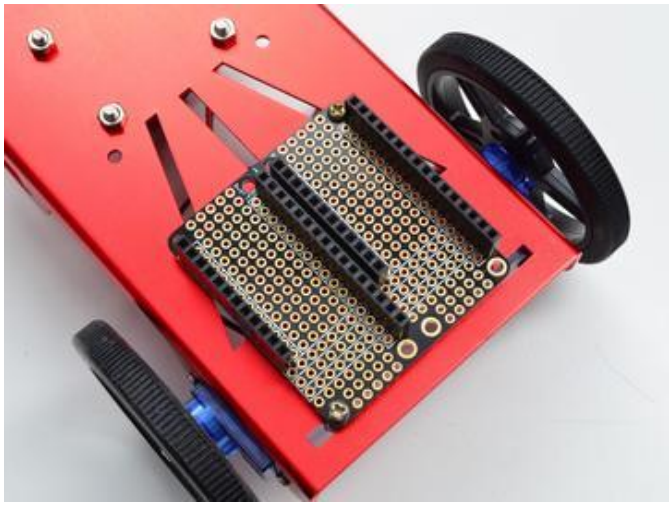
Attach the swivel caster wheel to the front of the chassis.

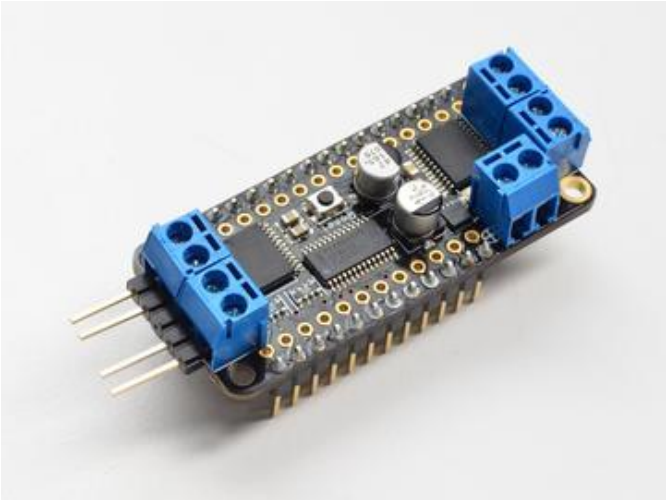
Flip it over and attach the servo wheels with the supplied screws.



Stick 4 bumper feet near the corners of the FeatherWing Doubler, but don't cover up any of the mounting holes. These bumpers act as spacers to keep the PCB from shorting against the chassis.

Attach the doubler to the rear of the chassis using two 4-40 screws and nuts. Getting the holes to line up requires just a bit of sliding around to find the sweet spot.

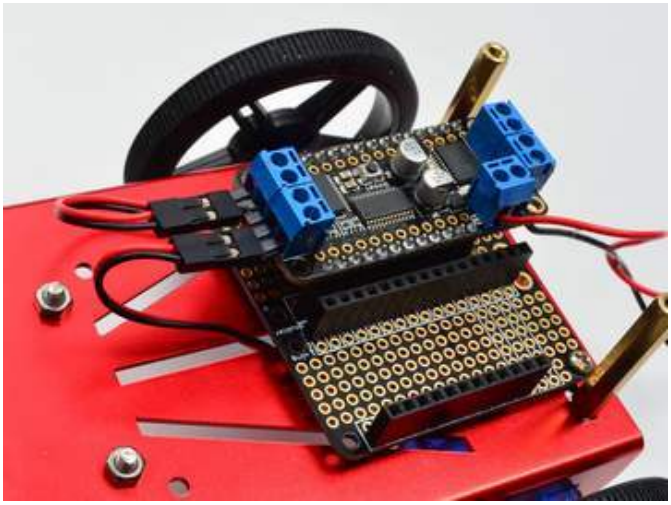


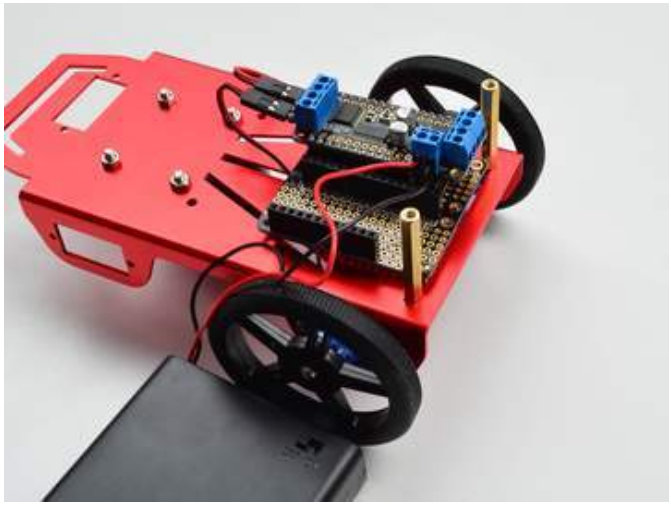


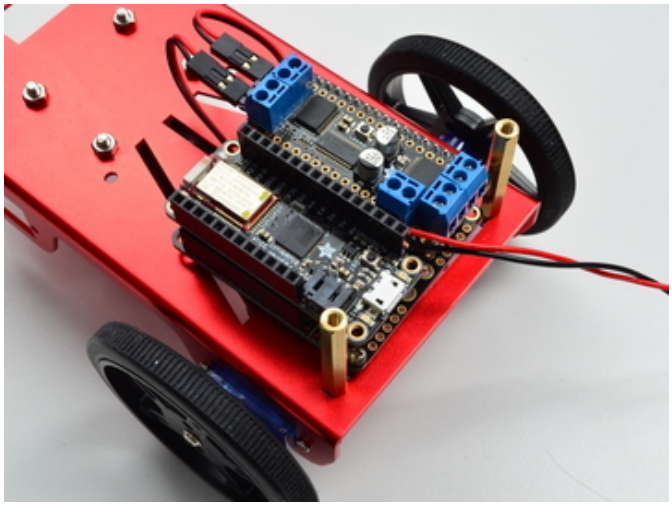
Take a stick of extra long male header and break off 5 pieces, carefully pulling (or cutting) out the middle one.

Insert it into M3 & M4. This will make it easy to connect and disconnect the DC motors.

Feed the DC motor wires up through the slots in the body and connect to the motor featherwing. For the code to work without modification, the negative lead of the DC motor should be towards the inside of the chassis.





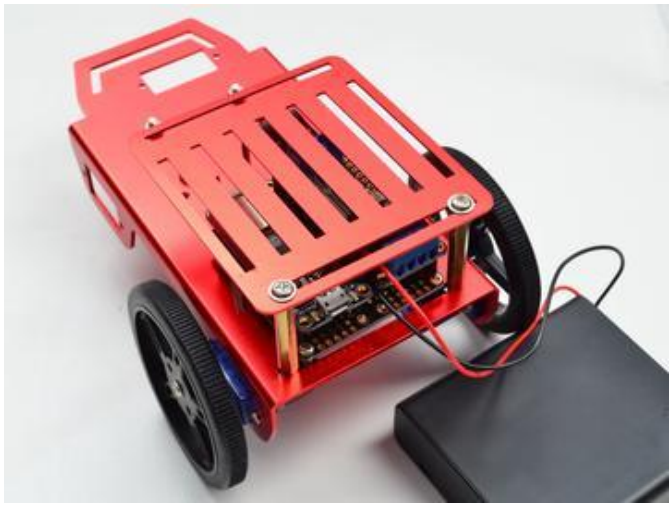


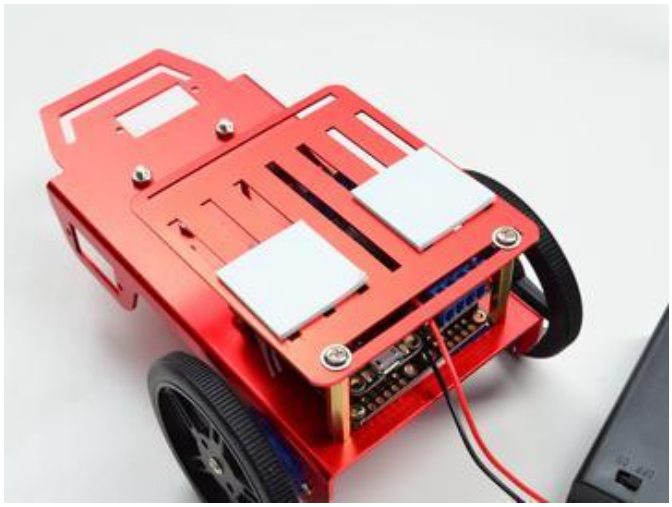
Connect the + and - lines from the AA battery holder to the Motor Wing terminals.

Place the BLE feather next to it, and attach the standoffs for the top metal plate.

Attach the top plate and apply foam tape or velcro to keep the AA battery pack stuck to the bot.

Plug the 350 mAh lipo into the feather and it should tuck in nicely underneath the top plate.





Code

The code is located on GitHub here (https://adafru.it/EGU).

Note: there are three files containing this project's code: BluefruitConfig.h, packetParser.cpp, and Ada_BLE_RC.ino that should all be placed in an Arduino project directory named Ada_BLE_RC

Click on "Download: Project Zip" below to download all three files containing the code of this project.

```
/*********************************************************************
 This is an example for our nRF51822 based Bluefruit LE modules

 Modified to drive a 3-wheeled BLE Robot Rover! by http://james.devi.to
 Pick one up today in the Adafruit shop!
 Adafruit invests time and resources providing this open source code,
 please support Adafruit and open-source hardware by purchasing
 products from Adafruit!
 MIT license, check LICENSE for more information
 All text above, and the splash screen below must be included in
 any redistribution
*********************************************************************/

#include <string.h>
#include <Arduino.h>
#include <SPI.h>
#if not defined (_VARIANT_ARDUINO_DUE_X_)
  #include <SoftwareSerial.h>
#endif

#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"

#include "BluefruitConfig.h"

#include <Wire.h>
#include <Adafruit_MotorShield.h>
// #include "utility/Adafruit_PWMServoDriver.h"
// #include <Servo.h>

// Create the motor shield object with the default I2C address
Adafruit_MotorShield AFMS = Adafruit_MotorShield();

// And connect 2 DC motors to port M3 & M4 !
Adafruit_DCMotor *L_MOTOR = AFMS.getMotor(4);
Adafruit_DCMotor *R_MOTOR = AFMS.getMotor(3);

//not used, testing acceleration
// int accelTime = 200;

//Name your RC here
String BROADCAST_NAME = "adafruit red robot rover";

String BROADCAST_CMD = String("AT+GAPDEVNAME=" + BROADCAST_NAME);

Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
```

```
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);


// A small helper
void error(const __FlashStringHelper*err) {
  Serial.println(err);
  while (1);
}

// function prototypes over in packetparser.cpp
uint8_t readPacket(Adafruit_BLE *ble, uint16_t timeout);
float parsefloat(uint8_t *buffer);
void printHex(const uint8_t * data, const uint32_t numBytes);

// the packet buffer
extern uint8_t packetbuffer[];

char buf[60];

/**************************************************************************/
/*!
    @brief  Sets up the HW an the BLE module (this function is called
            automatically on startup)
*/
/**************************************************************************/
void setup(void)
{
  Serial.begin(9600);

  AFMS.begin();  // create with the default frequency 1.6KHz

  // turn on motors
  L_MOTOR->setSpeed(0);
  L_MOTOR->run(RELEASE);

  R_MOTOR->setSpeed(0);
  R_MOTOR->run(RELEASE);

  Serial.begin(115200);
  Serial.println(F("Adafruit Bluefruit Robot Controller Example"));
  Serial.println(F("-----------------------------------------"));

  /* Initialize the module */
  BLEsetup();


}

int velocity = 0;

float x, y;

int L_restrict = 0;
int R_restrict = 0;

unsigned long lastAccelPacket = 0;

bool modeToggle = false;

void loop(void)
```

```
{
    // read new packet data
    uint8_t len = readPacket(&ble, BLE_READPACKET_TIMEOUT);
    // if (len == 0) return;

  // Read from Accelerometer input
    if( accelMode() ) {
      lastAccelPacket = millis();
      modeToggle = true;
      return;
    }

  // Stop motors if accelerometer data is turned off (100ms timeout)
    if( millis() - lastAccelPacket > 100 & modeToggle) {
      L_MOTOR->run(RELEASE);
      R_MOTOR->run(RELEASE);
      modeToggle = false;
      return;
    }

    //if no accelerometer, use control pad
    if( !modeToggle ) buttonMode();

}


bool accelMode(){
  if (packetbuffer[1] == 'A') {
          x = parsefloat( packetbuffer + 2 );
          y = parsefloat( packetbuffer + 6 );

        if( x <= -0.55 ){
          x += 0.55;
          x *= -100.0;
          L_MOTOR->run( BACKWARD );
          R_MOTOR->run( BACKWARD );
          if( x >= 45 ) x = 45;
          if( x <= 0 ) x = 0;
          velocity = map( x, 0, 45, 0 ,255 );
        }
        else if( x >= -0.25 ){
          x+= 0.25;
          x *= 100;
          L_MOTOR->run( FORWARD );
          R_MOTOR->run( FORWARD );
          if( x>= 45 ) x = 45;
          if( x<= 0 ) x = 0;
          velocity = map( x, 0, 45, 0, 255 );
        }
        else{
          L_MOTOR->run( RELEASE );
          R_MOTOR->run( RELEASE );
          velocity = 0;
        }

        //account for L / R accel

        if( y >= 0.1 ){
            y -= 0.1;
            y *= 100;
```

```
            y    100;
            if( y >= 50 ) y = 50;
            if( y <= 0 ) y = 0;

            L_restrict = fscale( y, 0.0, 50.0, 0.0, 100.0, -4.0 );
        }
        else if( y <= -0.1 ){
            y += 0.1;
            y *= -100;
            if( y>= 50 ) y = 50;
            if( y<= 0 ) y = 0;

            R_restrict = fscale( y, 0.0, 50.0, 0.0, 100.0, -4.0 );
        }
        else{
            L_restrict = 0;
            R_restrict = 0;
        }

          float Lpercent = ( 100.0 - L_restrict ) / 100.00 ;
          float Rpercent = ( 100.0 - R_restrict ) / 100.00 ;

          // Serial.print( x );
          // Serial.print( "\t" );
          // Serial.print( Lpercent );
          // Serial.print( "\t" );
          // Serial.print( velocity );
          // Serial.print( "\t" );
          // Serial.println( Rpercent );

          L_MOTOR->setSpeed( velocity * Lpercent );
          R_MOTOR->setSpeed( velocity * Rpercent );

          return true;
    }
    return false;
}

bool isMoving = false;

bool buttonMode(){

  static unsigned long lastPress = 0;


 // Buttons
  if (packetbuffer[1] == 'B') {

    uint8_t buttnum = packetbuffer[2] - '0';
    boolean pressed = packetbuffer[3] - '0';

    // Serial.println(buttnum);

  Serial.println(isMoving);
    if (pressed) {
      isMoving = true;
      if(buttnum == 5){
        L_MOTOR->run(FORWARD);
        R_MOTOR->run(FORWARD);
      }
```

```
      if(buttnum == 6){
        L_MOTOR->run(BACKWARD);
        R_MOTOR->run(BACKWARD);
      }
      if(buttnum == 7){
        L_MOTOR->run(RELEASE);
        R_MOTOR->run(FORWARD);
      }
      if(buttnum == 8){
        L_MOTOR->run(FORWARD);
        R_MOTOR->run(RELEASE);
      }

      lastPress = millis();

        L_MOTOR->setSpeed(255);
        R_MOTOR->setSpeed(255);
    }

    else {
      isMoving = false;
      L_MOTOR->run(RELEASE);
      R_MOTOR->run(RELEASE);
    }
     return true;
  }
    // if(isMoving){

      // unsigned long timeSincePress = millis() - lastPress;

      // if(timeSincePress <= accelTime){

      //   Serial.println( timeSincePress ) ;

      //   int motorSpeed = map( timeSincePress, 0, accelTime, 0, 255 );

      //   L_MOTOR->setSpeed(motorSpeed);
      //   R_MOTOR->setSpeed(motorSpeed);
      // }

      // else{
       // // full speed ahead!
        // L_MOTOR->setSpeed(255);
        // R_MOTOR->setSpeed(255);
      // }
   // }

  return false;

}

void BLEsetup(){
  Serial.print(F("Initialising the Bluefruit LE module: "));

  if ( !ble.begin(VERBOSE_MODE) )
  {
    error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode & check wiring?"));
  }
  Serial.println( F("OK!") );
```

```
  /* Perform a factory reset to make sure everything is in a known state */
  Serial.println(F("Performing a factory reset: "));
  if (! ble.factoryReset() ){
       error(F("Couldn't factory reset"));
  }

  //Convert the name change command to a char array
  BROADCAST_CMD.toCharArray(buf, 60);

  //Change the broadcast device name here!
  if(ble.sendCommandCheckOK(buf)){
    Serial.println("name changed");
  }
  delay(250);

  //reset to take effect
  if(ble.sendCommandCheckOK("ATZ")){
    Serial.println("resetting");
  }
  delay(250);

  //Confirm name change
  ble.sendCommandCheckOK("AT+GAPDEVNAME");

  /* Disable command echo from Bluefruit */
  ble.echo(false);

  Serial.println("Requesting Bluefruit info:");
  /* Print Bluefruit information */
  ble.info();

  Serial.println(F("Please use Adafruit Bluefruit LE app to connect in Controller mode"));
  Serial.println(F("Then activate/use the sensors, color picker, game controller, etc!"));
  Serial.println();

  ble.verbose(false);  // debug info is a little annoying after this point!

  /* Wait for connection */
  while (! ble.isConnected()) {
      delay(500);
  }

  Serial.println(F("******************"));

  // Set Bluefruit to DATA mode
  Serial.println( F("Switching to DATA mode!") );
  ble.setMode(BLUEFRUIT_MODE_DATA);

  Serial.println(F("******************"));
}

//Logarithmic mapping function from http://playground.arduino.cc/Main/Fscale
float fscale( float inputValue,  float originalMin, float originalMax, float newBegin, float newEnd, floa

  float OriginalRange = 0;
  float NewRange = 0;
  float zeroRefCurVal = 0;
  float normalizedCurVal = 0;
  float rangedValue = 0;
```

```
boolean invFlag = 0;


// condition curve parameter
// limit range

if (curve > 10) curve = 10;
if (curve < -10) curve = -10;

curve = (curve * -.1) ; // - invert and scale - this seems more intuitive - postive numbers give more w
curve = pow(10, curve); // convert linear scale into lograthimic exponent for other pow function

/*
 Serial.println(curve * 100, DEC);   // multply by 100 to preserve resolution
 Serial.println();
 */

// Check for out of range inputValues
if (inputValue < originalMin) {
  inputValue = originalMin;
}
if (inputValue > originalMax) {
  inputValue = originalMax;
}

// Zero Refference the values
OriginalRange = originalMax - originalMin;

if (newEnd > newBegin){
  NewRange = newEnd - newBegin;
}
else
{
  NewRange = newBegin - newEnd;
  invFlag = 1;
}

zeroRefCurVal = inputValue - originalMin;
normalizedCurVal  =  zeroRefCurVal / OriginalRange;   // normalize to 0 - 1 float

/*
Serial.print(OriginalRange, DEC);
 Serial.print("   ");
 Serial.print(NewRange, DEC);
 Serial.print("   ");
 Serial.println(zeroRefCurVal, DEC);
 Serial.println();
 */

// Check for originalMin > originalMax  - the math for all other cases i.e. negative numbers seems to w
if (originalMin > originalMax ) {
  return 0;
}

if (invFlag == 0){
  rangedValue =  (pow(normalizedCurVal, curve) * NewRange) + newBegin;

}
else     // invert the ranges
{
```

```
  `
    rangedValue =  newBegin - (pow(normalizedCurVal, curve) * NewRange);
  }

  return rangedValue;
}
```

```
// COMMON SETTINGS
// ------------------------------------------------------------------------------------------
// These settings are used in both SW UART, HW UART and SPI mode
// ------------------------------------------------------------------------------------------
#define BUFSIZE                        128   // Size of the read buffer for incoming data
#define VERBOSE_MODE                   true  // If set to 'true' enables debug output
#define BLE_READPACKET_TIMEOUT         500   // Timeout in ms waiting to read a response


// SOFTWARE UART SETTINGS
// ------------------------------------------------------------------------------------------
// The following macros declare the pins that will be used for 'SW' serial.
// You should use this option if you are connecting the UART Friend to an UNO
// ------------------------------------------------------------------------------------------
#define BLUEFRUIT_SWUART_RXD_PIN       9    // Required for software serial!
#define BLUEFRUIT_SWUART_TXD_PIN       10   // Required for software serial!
#define BLUEFRUIT_UART_CTS_PIN         11   // Required for software serial!
#define BLUEFRUIT_UART_RTS_PIN         -1   // Optional, set to -1 if unused


// HARDWARE UART SETTINGS
// ------------------------------------------------------------------------------------------
// The following macros declare the HW serial port you are using. Uncomment
// this line if you are connecting the BLE to Leonardo/Micro or Flora
// ------------------------------------------------------------------------------------------
#ifdef Serial1    // this makes it not complain on compilation if there's no Serial1
  #define BLUEFRUIT_HWSERIAL_NAME      Serial1
#endif


// SHARED UART SETTINGS
// ------------------------------------------------------------------------------------------
// The following sets the optional Mode pin, its recommended but not required
// ------------------------------------------------------------------------------------------
#define BLUEFRUIT_UART_MODE_PIN        12    // Set to -1 if unused


// SHARED SPI SETTINGS
// ------------------------------------------------------------------------------------------
// The following macros declare the pins to use for HW and SW SPI communication.
// SCK, MISO and MOSI should be connected to the HW SPI pins on the Uno when
// using HW SPI.  This should be used with nRF51822 based Bluefruit LE modules
// that use SPI (Bluefruit LE SPI Friend).
// ------------------------------------------------------------------------------------------
#define BLUEFRUIT_SPI_CS               8
#define BLUEFRUIT_SPI_IRQ              7
#define BLUEFRUIT_SPI_RST              6    // Optional but recommended, set to -1 if unused

// SOFTWARE SPI SETTINGS
// ------------------------------------------------------------------------------------------
// The following macros declare the pins to use for SW SPI communication.
// This should be used with nRF51822 based Bluefruit LE modules that use SPI
// (Bluefruit LE SPI Friend).
// ------------------------------------------------------------------------------------------
#define BLUEFRUIT_SPI_SCK              13
#define BLUEFRUIT_SPI_MISO             12
#define BLUEFRUIT_SPI_MOSI             11
```

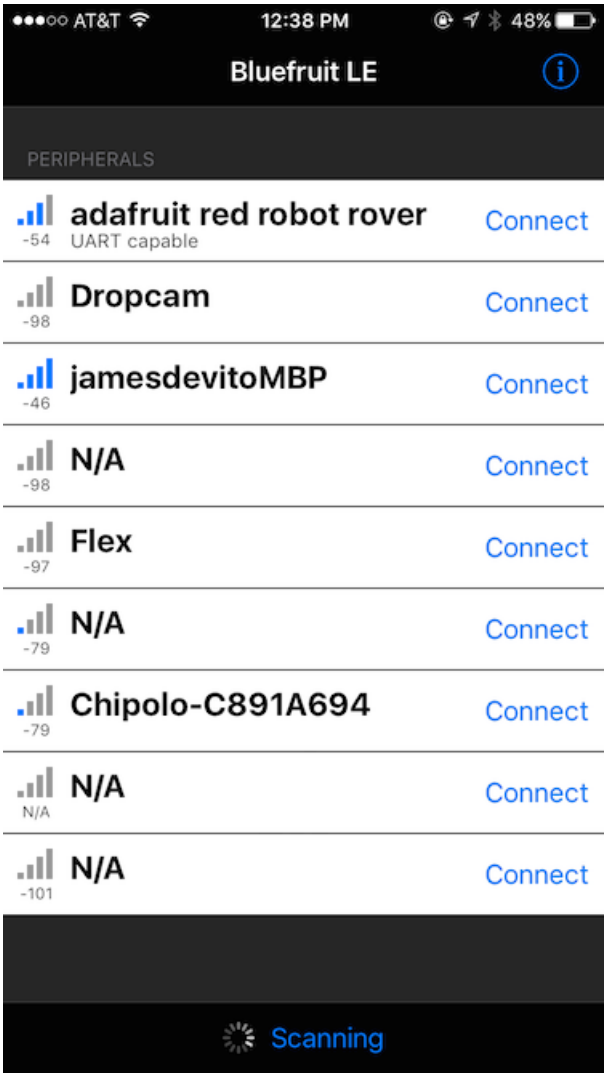
```
#include <string.h>
#include <Arduino.h>
#include <SPI.h>
#include <SoftwareSerial.h>

#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"


#define PACKET_ACC_LEN                  (15)
#define PACKET_GYRO_LEN                 (15)
#define PACKET_MAG_LEN                  (15)
#define PACKET_QUAT_LEN                 (19)
#define PACKET_BUTTON_LEN               (5)
#define PACKET_COLOR_LEN                (6)
#define PACKET_LOCATION_LEN             (15)

//    READ_BUFSIZE            Size of the read buffer for incoming packets
#define READ_BUFSIZE                    (20)


/* Buffer to hold incoming characters */
uint8_t packetbuffer[READ_BUFSIZE+1];

/**************************************************************************/
/*!
    @brief  Casts the four bytes at the specified address to a float
*/
/**************************************************************************/
float parsefloat(uint8_t *buffer)
{
  float f = ((float *)buffer)[0];
  return f;
}

/**************************************************************************/
/*!
    @brief  Prints a hexadecimal value in plain characters
    @param  data      Pointer to the byte data
    @param  numBytes  Data length in bytes
*/
/**************************************************************************/
void printHex(const uint8_t * data, const uint32_t numBytes)
{
  uint32_t szPos;
  for (szPos=0; szPos < numBytes; szPos++)
  {
    Serial.print(F("0x"));
    // Append leading 0 for small values
    if (data[szPos] <= 0xF)
    {
      Serial.print(F("0"));
      Serial.print(data[szPos] & 0xf, HEX);
    }
    else
    {
      Serial.print(data[szPos] & 0xff, HEX);
    }
```

```
    // Add a trailing space if appropriate
    if ((numBytes > 1) && (szPos != numBytes - 1))
    {
      Serial.print(F(" "));
    }
  }
  Serial.println();
}

/**************************************************************************/
/*!
    @brief  Waits for incoming data and parses it
*/
/**************************************************************************/
uint8_t readPacket(Adafruit_BLE *ble, uint16_t timeout)
{
  uint16_t origtimeout = timeout, replyidx = 0;

  memset(packetbuffer, 0, READ_BUFSIZE);

  while (timeout--) {
    if (replyidx >= 20) break;
    if ((packetbuffer[1] == 'A') && (replyidx == PACKET_ACC_LEN))
      break;
    if ((packetbuffer[1] == 'G') && (replyidx == PACKET_GYRO_LEN))
      break;
    if ((packetbuffer[1] == 'M') && (replyidx == PACKET_MAG_LEN))
      break;
    if ((packetbuffer[1] == 'Q') && (replyidx == PACKET_QUAT_LEN))
      break;
    if ((packetbuffer[1] == 'B') && (replyidx == PACKET_BUTTON_LEN))
      break;
    if ((packetbuffer[1] == 'C') && (replyidx == PACKET_COLOR_LEN))
      break;
    if ((packetbuffer[1] == 'L') && (replyidx == PACKET_LOCATION_LEN))
      break;

    while (ble->available()) {
      char c =  ble->read();
      if (c == '!') {
        replyidx = 0;
      }
      packetbuffer[replyidx] = c;
      replyidx++;
      timeout = origtimeout;
    }

    if (timeout == 0) break;
    delay(1);
  }

  packetbuffer[replyidx] = 0;  // null term

  if (!replyidx)  // no data or timeout
    return 0;
  if (packetbuffer[0] != '!')  // doesn't start with '!' packet beginning
    return 0;

  // check checksum!
  uint8_t xsum = 0;
```

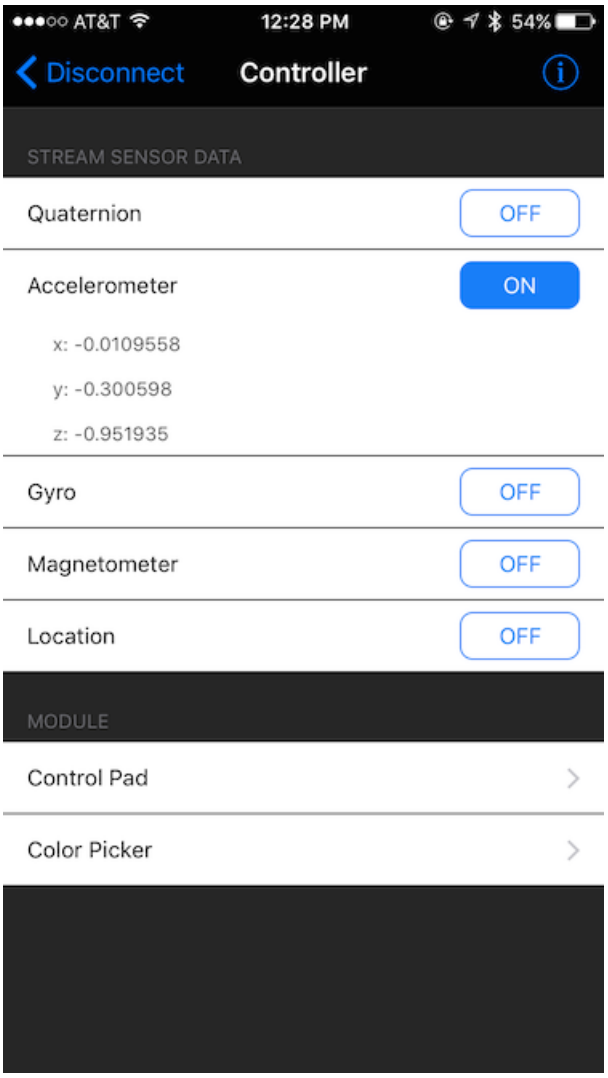
```
  uint8_t checksum = packetbuffer[replyidx-1];

  for (uint8_t i=0; i<replyidx-1; i++) {
    xsum += packetbuffer[i];
  }
  xsum = ~xsum;

  // Throw an error message if the checksum's don't match
  if (xsum != checksum)
  {
    Serial.print("Checksum mismatch in packet : ");
    printHex(packetbuffer, replyidx+1);
    return 0;
  }

  // checksum passed!
  return replyidx;
}
```

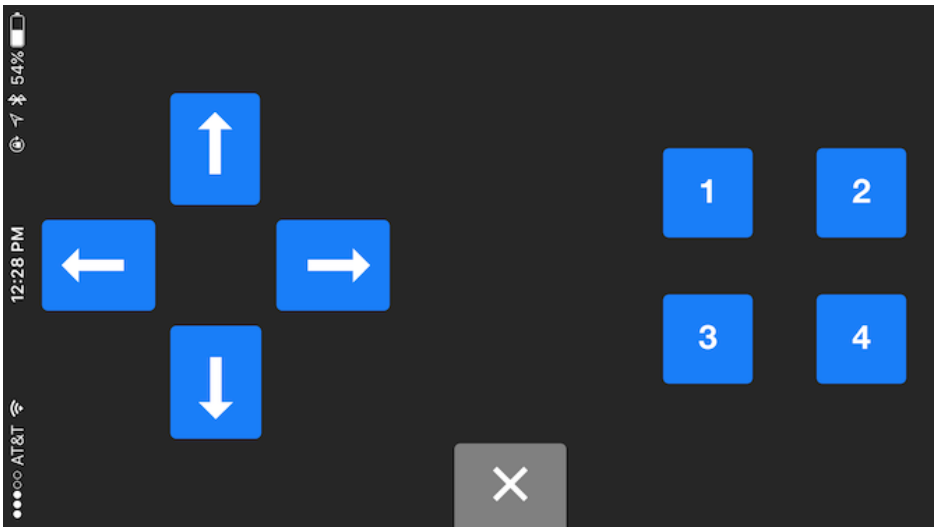
Use it!

Open the Adafruit Bluefruit LE Connect app, and you should see 'adafruit red robot rover'. Connect to it and select controller!



Use your phone's accelerometer to steer, or pull up the on screen control pad with the Adafruit Bluefruit LE connect app.


●●●○○ AT&T
12:28 PM
54%
Disconnect
Controller
STREAM SENSOR DATA
Quaternion
OFF
Accelerometer
ON
x: -0.0109558
y: -0.300598
z: -0.951935
Gyro
OFF
Magnetometer
OFF
Location
OFF
MODULE
Control Pad
Color Picker


54%
12:28 PM
●●●○○ AT&T
1
2
3
4