CRICKIT Flippy Robot
Created by Dave Astels

https://learn.adafruit.com/crickit-flippy-robot

Last updated on 2021-11-15 07:19:59 PM EST
Table of Contents

Overview 3
• Additional Hardware 6

Design 7
• Operation Tips 9

Construction 10
• 3D Printing 10
• Files 10
• What If I Don't Have A 3D Printer? 11
• Motors 12
• Battery 14
• The CRICKIT Tray 16
• Legs and Final Assembly 18
• And we're done 20

Code 21
We found a cute little robot on thingiverse (https://adafru.it/CDG) that used a simple, yet clever locomotion system. Simple in that it used a single yellow (dual shaft) TT motor to control both sides, with the addition of only a battery and power switch.

The basic design is, as I said, clever. What could be done to expand on it? We could use two motors. A CRICKIT would provide finer, individual control of the motors. Using the Feather M0 Bluefruit with the Featherwing CRICKIT would let us drive it remotely. All this additional hardware will require the robot to be larger.

TT Motor All-Metal Gearbox - 1:90 Gear Ratio
These durable (but affordable!) gearbox motors (also known as 'TT' motors) are an easy, low-cost way to get your projects moving. This is a TT DC All-Metal Gearbox...
https://www.adafruit.com/product/3802
Adafruit Feather M0 Bluefruit LE
Feather is the new development board from Adafruit, and like its namesake, it is thin, light, and lets you fly! We designed Feather to be a new standard for portable microcontroller...
https://www.adafruit.com/product/2995

Adafruit CRICKIT FeatherWing for any Feather
Sometimes we wonder if robotics engineers ever watch movies. If they did, they'd know that making robots into slaves always ends up in a robot rebellion. Why even go down that...
https://www.adafruit.com/product/3343

3 x AA Battery Holder with 2.1mm Plug
Here's another addition to our growing family of AA battery holders. A holder for three (3) AA batteries!...
https://www.adafruit.com/product/3842

If you want to use a LiPo:
Lithium Ion Battery Pack - 3.7V 6600mAh

Need a massive battery for your project? This lithium-ion pack is made of 3 balanced 2200mAh cells for a total of 6600mA capacity! The cells are connected in parallel and spot-welded...


PowerBoost 1000 Charger - Rechargeable 5V Lipo USB Boost @ 1A

PowerBoost 1000C is the perfect power supply for your portable project! With a built-in load-sharing battery charger circuit, you'll be able to keep your power-hungry...

https://www.adafruit.com/product/2465

When working at your bench/desk, this wall powered supply will give you solid power and avoid issues with batteries running low.

5V 2A (2000mA) switching power supply - UL Listed

This is an FCC/CE certified and UL listed power supply. Need a lot of 5V power? This switching supply gives a clean regulated 5V output at up to 2000mA. 110 or 240 input, so it works...

https://www.adafruit.com/product/276

You'll probably want to add these to the legs and body for traction and noise dampening.
### Additional Hardware

Hex head bolts are the easiest to work with.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 x <strong>M3 x .5mm x 25mm Machine Screws</strong></td>
<td>Metric Pan Head Screws for mounting the motors</td>
<td><a href="https://www.albanycountyfasteners.com/Phillips-Pan-Head-Machine-Screw-M3-x-5-p/1066-1008.htm">www.albanycountyfasteners.com/Phillips-Pan-Head-Machine-Screw-M3-x-5-p/1066-1008.htm</a></td>
</tr>
<tr>
<td>4 x <strong>M3 x .5mm Hex Nuts</strong></td>
<td>Metric Hex Jam Nuts for mounting the motors</td>
<td><a href="https://www.albanycountyfasteners.com/Metric-Hex-Jam-Nuts-A2-Stainless-Steel-p/5580000.htm">www.albanycountyfasteners.com/Metric-Hex-Jam-Nuts-A2-Stainless-Steel-p/5580000.htm</a></td>
</tr>
<tr>
<td>4 x <strong>M3 Flat Washer</strong></td>
<td>Metric Sized Round steel Washers for mounting the motors</td>
<td><a href="https://www.albanycountyfasteners.com/Flat-Washers-A2-Stainless-Steel-p/5620000.htm">www.albanycountyfasteners.com/Flat-Washers-A2-Stainless-Steel-p/5620000.htm</a></td>
</tr>
<tr>
<td>1 x <strong>M3 Threaded Inserts</strong></td>
<td>Brass Threaded Insert Nuts 100PCS for the CRICKIT tray</td>
<td><a href="http://a.co/363VPJ5">http://a.co/363VPJ5</a></td>
</tr>
<tr>
<td>8 x <strong>M3 x .5mm x 6mm Machine Screws</strong></td>
<td>Metric Pan Head Screws for mounting the CRICKIT to its tray</td>
<td><a href="https://www.albanycountyfasteners.com/Phillips-Pan-Head-Machine-Screw-M3-x-5-p/1066-1008.htm">www.albanycountyfasteners.com/Phillips-Pan-Head-Machine-Screw-M3-x-5-p/1066-1008.htm</a></td>
</tr>
<tr>
<td>8 x <strong>M3 x .5mm x 20mm Machine Screws</strong></td>
<td>Metric Pan Head Screws for mounting the CRICKIT and battery trays</td>
<td><a href="https://www.albanycountyfasteners.com/Phillips-Pan-Head-Machine-Screw-M3-x-5-p/1066-1008.htm">www.albanycountyfasteners.com/Phillips-Pan-Head-Machine-Screw-M3-x-5-p/1066-1008.htm</a></td>
</tr>
</tbody>
</table>

Little Rubber Bumper Feet - Pack of 4
Keep your electronics from going barefoot, give them little rubber feet!
These small sticky bumpers are our favorite accessory for any electronic kit or device. They are sticky, but...
Design

The design echoes that of the original project, though this version was done from scratch. The bot is roughly twice the size of the original. Each "leg" is completely independent, not joined as in the single motor design. This allows us to use a separate motor for each side. That in turn gives some control over turning.

The mechanical design is fairly clever, moves in a truly bizarre manner, makes a good amount of noise on a hard floor [the author's son's dog is weirded out by bots in general, but is especially bothered by this particular design], and is fairly resilient to tipping over.


This last point isn't as much of an issue with the version we'll be building due to it being wider in proportion to the original: while the height and length has been roughly doubled, the width has been substantially more than doubled to accommodate not only two motors, but also the CRICKIT and a larger battery pack.

Electrically, it is simply two DC motors connected to a CRICKIT. The FeatherWing CRICKIT was used so that we could use the Feather M0 Bluefruit. This lets us easily interact with the robot using the Adafruit Bluefruit app.

If you're not familiar with using DC motors with CRICKIT, see this guide (https://adafru.it/BNS).
You can use a 3xAA battery holder with Alkaline AAs or opt for a large LiPo battery and a boost converter to get the 3.7v to 5v. The former is cheap and easy, but has proven to not always work well with motor based projects due to voltage fluctuations caused by the motors. See the WobblyBot guide (https://adafruit.it/CDH) for a discussion of this.

This project does use two motors as opposed to the five that the WobblyBot uses, so this isn’t as much of a problem.

Maybe a bigger (at least more annoying) problem is that motors can drain Alkaline AAs pretty quickly and they aren’t rechargeable. An alternative is a 4xAA holder with NiMH AAs. They’re rechargeable, but do have to be removed for charging. The large LiPo+boost is a far better approach [in the opinion of this maker] as it provides plenty of power and charging is a matter of simply plugging in a USB cable. The boost converter gives you the ability to add a handy power switch as well.

Regardless of how you are powering the CRICKIT, if you are getting weird behavior from the Feather (e.g. crashing/resetting, especially if the bot gets stuck and the
motors stall) you can try adding a separate, smaller LiPo (1200mAh is a good choice) directly to the Feather. Recharging this one is simply a matter connecting a USB cable to the Feather.

As always, when you're working on the bot at your bench/desk you can use a wall wart power source to avoid any battery issues.

---

5V 2A (2000mA) switching power supply - UL Listed
This is an FCC/CE certified and UL listed power supply. Need a lot of 5V power? This switching supply gives a clean regulated 5V output at up to 2000mA. 110 or 240 input, so it works...
https://www.adafruit.com/product/276

---

Operation Tips

The Feather M0 Bluefruit was selected to make remote control of the bot easy. Simply install the Adafruit Bluefruit app and connect to the bot. Select the controller and control pad and use the arrows to drive the bot around. We have a guide on the app (https://adafru.it/iCi) that walks through its features and operation.

The bot works best if both legs are in sync. On the original design this was a direct result of using a single motor. In this version the legs are independent, so jostling back and forth a bit to get them aligned before charging forward will result in much better locomotion.

This uses the high torque, all-metal-gear TT motors. Even so, the motors are called on to generate significant leverage. One result is that moving forward works well, but the motors have a harder time with reverse. If it's having trouble getting going in reverse, try swinging the legs to the other extreme before trying to move. This will allow some momentum to be built up before having to lift the body.
Construction

3D Printing

An couple attempts were made to use corrugated cardboard as the construction material, but it proved to be not rigid or strong enough. In the end 3D printed components were used. STL files are below. You should use sparse infill (~10%) on all parts to minimize the mass of the finished bot. It probably doesn't do much to offset the large battery, but it doesn't hurt.

You will need to print 2 legs, 2 motor arms, 4 plain arms, 10 joint caps, and one each of the body and tray pieces.

The CRICKIT and battery trays also work as structural elements to attach the sides together, and are based on the mount in this guide (https://adafru.it/BXU). As such it can accept threaded brass M3 inserts. If you use inserts, only put them in the CRICKIT tray. Note that the hole spacing on the two trays isn't identical so you will need to print the two files and keep them sorted out.

Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint_Cap.stl</td>
<td><a href="https://adafru.it/CHG">https://adafru.it/CHG</a></td>
</tr>
<tr>
<td>Leg.stl</td>
<td><a href="https://adafru.it/CEm">https://adafru.it/CEm</a></td>
</tr>
<tr>
<td>Motor_Arm.stl</td>
<td><a href="https://adafru.it/CHH">https://adafru.it/CHH</a></td>
</tr>
<tr>
<td>Plain_Arm.stl</td>
<td><a href="https://adafru.it/CHI">https://adafru.it/CHI</a></td>
</tr>
<tr>
<td>Body_1.stl</td>
<td><a href="https://adafru.it/CHJ">https://adafru.it/CHJ</a></td>
</tr>
<tr>
<td>Body_2.stl</td>
<td></td>
</tr>
</tbody>
</table>
What If I Don't Have A 3D Printer?

Not to worry! You can use a 3D printing service such as 3DHubs (https://adafruit.it/jNb) or MakeXYZ (https://adafruit.it/veh) to have a local 3D printer operator 3D print and ship you parts to you. This is a great way to get your parts 3D printed by local makers. You could also try checking out your local Library or search for a Maker Space.

Monoprice Inventor II 3D Printer with Touchscreen and WiFi
The Monoprice Inventor II 3D Printer Touchscreen with WiFi is a perfect entry-level 3D printer with small footprint and reliable performance. It comes equipped with...
https://www.adafruit.com/product/3897
Motors

Solder an approximate 15 cm (6") wire to each motor connection if there isn't any yet. Strip and tin the other ends.

Snap a motor into each side piece and secure with the 25mm M3 bolts. Side the bolts through the motor mounting holes from the outside of the body pieces. Add a washer and hex-nut to each and tighten the bolts being careful not to overtighten.

Having the motor nestle into the side panel like this spreads out the rotational force of moving the large legs. These are high torque motors and work pretty hard. By having the motors suck into the side plates, the bolts aren't subjected to all the force, in fact they mostly serve to hold the motors in place.
Mount the battery you are using on the battery tray.

A couple heavy rubber bands can be used to secure the battery in place.

If you use a large LiPo (e.g. the 4400 or 6600 mAh versions) and the boost converter, the converter can be secured under the battery mounting tray. You can add a screw terminal connector to the 5v output. Then you can use one of the products below to connect to the CRICKIT.

**Terminal Block - 2-pin 3.5mm - pack of 5!**

Nothing makes a project harder to maintain than a lot of loose wiring. That's why we like to use terminal blocks whenever making PCB-to-Wire connections. These particular 3.5mm...

[https://www.adafruit.com/product/724](https://www.adafruit.com/product/724)
2.1mm DC Barrel Plug to Alligator Clips
Turn any battery pack into a power supply with this incredibly useful adapter. A high quality molded 2.1mm DC barrel plug is brought out to red (center positive) & black (ring...  
https://www.adafruit.com/product/1329

Male DC Power adapter - 2.1mm plug to screw terminal block
If you need to connect a battery pack or wired power supply to a board that has a DC jack - this adapter will come in very handy! There is a 2.1mm DC plug on one end, and a screw...  
https://www.adafruit.com/product/369

5.5 / 2.1mm Barrel Connector - DC Power Plug
DIY? How about D-I-Wire! This barrel jack plug is great for adding a common power connector to the end of your wires. The jack is compatible with 5.5mm barrel jacks that have a...  
https://www.adafruit.com/product/3310
The CRICKIT Tray

Add threaded inserts to the CRICKIT mounting tray.

Mount your controller board onto the CRICKIT now and mount the CRICKIT on the tray.
Now you can start putting the bot together. First you can secure the CRICKIT mount to one of the side pieces with a couple of the 20mm bolts.

Next, wire the motors to the CRICKIT. Do a quick test (using the actual software is a good approach) to verify that the motors are wired the right way around. Fix it if the connections are backward.

Finally, secure the other side to the CRICKIT tray with two more of the 20mm bolts.
Legs and Final Assembly

Place the motor arms over the motor drive shafts. The holes will likely need to be cleaned up somewhat; it should be a tight fit but shouldn’t require excessive force.

The other legs simply have an end peg inserted into one of the other holes of each side piece. They should rotate freely and shouldn't need any cleanup.

Secure each non-motor leg with a joint cap and a M3x6mm bolt.

Once all legs are secured, the battery tray can be mounted in the same way the CRICKIT tray was. Mount it so that the wiring to the CRICKIT is the most direct.

Slip a legs onto the pegs of the three arms on one side. Try not to move the motor arms much when doing this. Secure all three leg joints with a joint cap. Do the same for the other side.
And we're done

Now the bot is assembled, and it's just a matter of making sure the batteries are fresh (or charged if using a LiPo) and uploading the software if you haven't already.
For this project we use the Feather M0 Bluefruit and write the code in C/C++ using the Arduino framework and tools. This board was selected for its Bluetooth capabilities, which will let us drive the robot using the Adafruit Bluefruit app controller interface, as well as run pre-written sequences using the controller's 1-4 buttons.

You'll need to have the Arduino IDE installed as well as the appropriate board packages and libraries. The Feather M0 Bluefruit LE guide (https://adafruit.it/CBf) covers this in detail. It's a good idea to read through that guide (https://adafruit.it/CBf) if you haven't yet; it will show you all the tricks of this board.

This code has a couple support files with it that were copied from the controller.ino example: BluefruitConfig.h and packetParser.cpp. Either start a new sketch called FlippyBot and copy them into the directory along with the FlippyBot.ino file below, or clone the repo from GitHub; it has everything in place that's needed. In either case, load FlippyBot into the Arduino IDE, set your board and port (see the linked guide for this Feather board), and compile/upload the code.

We won't go into detail on the boilerplate BLE setup and use. That's covered in the linked guide (https://adafruit.it/CBf) and the comments from the example code that was used have been kept intact.

We start by creating motor objects, and placing them in an array. We define a constant index for each leg. Additionally there is an array of strings used in debug output for leg names.

```c
Adafruit_Crickit crickit;
seesaw_Motor right_leg(&amp;crickit);
seesaw_Motor left_leg(&amp;crickit);

seesaw_Motor *legs[2] = {&right_leg, &left_leg};
const __FlashStringHelper *leg_names[] = {F("right"), F("left")};

const int RIGHT = 0;
const int LEFT = 1;
```
Speaking of debugging, there are two small functions for outputting errors and information. If you want to output to the serial console simply uncomment the `#define DEBUG` line near the start of the file. Remember to recompile/upload with it commented out before running the robot untethered.

```c
void error(const __FlashStringHelper *err)
{
    digitalWrite(13, HIGH);
#ifdef DEBUG
    Serial.println(err);
#endif
    while (1);
}

void log(const __FlashStringHelper *msg)
{
#ifdef DEBUG
    Serial.println(msg);
#endif
}
```

To encapsulate everything needed to set the velocity of a leg motor, we have the `set_leg` function. It checks the validity of the arguments, prints debugging information if `DEBUG` is defined, and sets the velocity of the leg.

```c
void set_leg(int leg, float velocity)
{
    if (leg != RIGHT && leg != LEFT) {
        error(F("Bad leg specifier"));
    }
    if (velocity < -1.0 || velocity > 1.0) {
        error(F("Velocity out of -1.0//1.0 range"));
    }
#ifdef DEBUG
    Serial.print(F("Setting "));
    Serial.print(leg_names[leg]);
    Serial.print(F(" to "));
    Serial.println(velocity);
#endif
    legs[leg]->throttle(velocity);
}
```

Note that the parameter here is called `velocity`, whereas in the next set of functions, the corresponding parameter is `speed`. Speed is just how fast something is going. In our case it's always positive and specifies how fast to spin the drive shaft (0.0 being stopped, and 1.0 being full speed). Velocity also contains a direction. With respect to a motor that's simply forward or reverse and is indicated by the sign of the value: positive for forward and negative for reverse.
These next functions control motion of the bot, manipulating the speed and direction of the two motors. Remember that if a motor is spinning in the opposite direction than it should, simply switch its connections to the CRICKIT motor driver.

```c
void stop()
{
    set_leg(RIGHT, 0.0);
    set_leg(LEFT, 0.0);
}

void forward(float speed)
{
    set_leg(RIGHT, speed);
    set_leg(LEFT, speed);
}

void reverse(float speed)
{
    set_leg(RIGHT, speed * -1);
    set_leg(LEFT, speed * -1);
}

void rotate_clockwise(float speed)
{
    set_leg(RIGHT, speed * -1);
    set_leg(LEFT, speed);
}

void rotate_counterclockwise(float speed)
{
    set_leg(RIGHT, speed);
    set_leg(LEFT, speed * -1);
}
```

The setup function is primarily boilerplate configuration and initialization of the bluetooth stack. The last bit is relevant to the bot, though: initializing the CRICKIT and connecting the motor objects to the motor drivers.

```c
if (!crickit.begin()) {
    error(F("Error initializing CRICKIT!")));  
} log(F("Crickit started"));

right_leg.attach(CRICKIT_MOTOR_A1, CRICKIT_MOTOR_A2);
left_leg.attach(CRICKIT_MOTOR_B1, CRICKIT_MOTOR_B2);
```

Since the design goal is to control the bot from the Bluefruit phone app, there are four slots for predefined scripts that get triggered by the 1-4 buttons. The first one has been filled in as an example.

```c
void demo1()
{
    forward(1.0);
    delay(5000);
    rotate_clockwise(1.0);
    delay(2000);
    forward(0.75);
}```
void loop()
{
    // Wait for new data to arrive
    uint8_t len = readPacket(&ble, BLE_READPACKET_TIMEOUT);
    if (len == 0) return;

    // Got a packet!
    // printHex(packetbuffer, len);

    // Buttons
    if (packetbuffer[1] == 'B') {
        uint8_t buttnum = packetbuffer[2] - '0';
        boolean pressed = packetbuffer[3] - '0';

        #ifdef DEBUG
        Serial.print ("Button "); Serial.print(buttnum);
        if (pressed) {
            Serial.println(" pressed");
        } else {
            Serial.println(" released");
        }
        #endif

        switch(buttnum) {
            case 1:
                if (pressed) {
                    demo1();
                }
                break;
            case 2:
                if (pressed) {
                    demo2();
                }
                break;
            case 3:
                if (pressed) {
                    demo3();
                }
                break;
            case 4:
                if (pressed) {
                    demo4();
                }
                break;
            case 5:
                if (pressed) {

The main loop checks for a command from the controller via Bluetooth and moves the bot as requested.
In case you haven't run into the `switch` statement before, it chooses a `case` block based on the value given to it, `buttnum` in this case. The `case` block with the corresponding value is executed. Notice that each `case` block ends with a `break` statement. This exits the `switch`. If they weren't there, the next `case` block would be executed, and so on until the end of the `switch` was reached or a `break` statement was encountered.

The entire FilppyBot.ino source file is below.

```c
#include <stdarg.h>
#include <string.h>
#include <Arduino.h>
#include <SPI.h>
#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"

forward(1.0);
} else {
    stop();
} break;

case 6:
    if (pressed) {
        reverse(1.0);
    } else {
        stop();
    } break;

case 7:
    if (pressed) {
        rotate_counterclockwise(1.0);
    } else {
        stop();
    } break;

case 8:
    if (pressed) {
        rotate_clockwise(1.0);
    } else {
        stop();
    } break;
}
```

// Triangular leg robot.

// Bluetooth code is from Feather M0 Bluefruit controller example.
// Explanatory comments kept intact.

// Adafruit invests time and resources providing this open source code.
// Please support Adafruit and open source hardware by purchasing
// products from Adafruit!

// Written by Dave Astels for Adafruit Industries
// Copyright (c) 2018 Adafruit Industries
// Licensed under the MIT license.

// All text above must be included in any redistribution.

#include <stdarg.h>
#include <string.h>
#include <Arduino.h>
#include <SPI.h>
#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_SPI.h"
#include "Adafruit_BluefruitLE_UART.h"

©Adafruit Industries
#include "BluefruitConfig.h"
#include "Adafruit_Crickit.h"
#include "seesaw_motor.h"

#define FACTORYRESET_ENABLE         1
#define MINIMUM_FIRMWARE_VERSION    "0.6.6"
#define MODE_LED_BEHAVIOUR          "MODE"

// 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[];

//#define DEBUG 1

Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);

// setup crickit & motors
Adafruit_Crickit crickit;
seesaw_Motor right_leg(&crickit);
seesaw_Motor left_leg(&crickit);
seesaw_Motor *legs[2] = {&right_leg, &left_leg};
const __FlashStringHelper *leg_names[] = {F("right"), F("left")};

const int RIGHT = 0;
const int LEFT = 1;

// conditional output routines

void error(const __FlashStringHelper *err)
{
    digitalWrite(13, HIGH);
    #ifdef DEBUG
        Serial.println(err);
    #endif
    while (1);
}

void log(const __FlashStringHelper *msg)
{
    #ifdef DEBUG
        Serial.println(msg);
    #endif
}

void set_leg(int leg, float velocity)
{
    if (leg != RIGHT && leg != LEFT) {
        error(F("Bad leg specifier"));
    }
    if (velocity < -1.0 || velocity > 1.0) {
        error(F("Velocity out of -1.0/1.0 range"));
    }
    #ifdef DEBUG
Serial.print(F("Setting ") );
Serial.print(leg_names[leg]);
Serial.print(F(" to ") );
Serial.println(velocity);
#endif

legs[leg]->throttle(velocity);
}

void stop()
{
    set_leg(RIGHT, 0.0);
    set_leg(LEFT, 0.0);
}

void forward(float speed)
{
    set_leg(RIGHT, speed);
    set_leg(LEFT, speed);
}

void reverse(float speed)
{
    set_leg(RIGHT, speed * -1);
    set_leg(LEFT, speed * -1);
}

void rotate_clockwise(float speed)
{
    set_leg(RIGHT, speed * -1);
    set_leg(LEFT, speed);
}

void rotate_counterclockwise(float speed)
{
    set_leg(RIGHT, speed);
    set_leg(LEFT, speed * -1);
}

void initialize()
{
    stop();
}

// Start things up
void setup()
{
    pinMode(13, OUTPUT);
digitalWrite(13, LOW);
#endif
while (!Serial);  // required for Flora & Micro
delay(500);
Serial.begin(115200);
#endif

log(F("FlippyBot"));
log(F("-----------------------------------------"));

// Initialise the module
log(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?"));
}

log(F("OK!"));

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

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

log(F("Requesting Bluefruit info:"));
// Print Bluefruit information
ble.info();

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

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

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

log(F("******************************"));

// LED Activity command is only supported from 0.6.6
if ( ble.isVersionAtLeast(MINIMUM_FIRMWARE_VERSION) )
{
  // Change Mode LED Activity
  log(F("Change LED activity to " MODE_LED_BEHAVIOUR));
  ble.sendCommandCheckOK("AT+HWModeLED=" MODE_LED_BEHAVIOUR);
}

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

log(F("******************************"));

if (!crickit.begin()) {
  error(F("Error initializing CRICKIT!"));
}

log(F("Crickit started"));

right_leg.attach(CRICKIT_MOTOR_A1, CRICKIT_MOTOR_A2);
left_leg.attach(CRICKIT_MOTOR_B1, CRICKIT_MOTOR_B2);

// Fill these functions in with the movement scripts you want attached to
// the controller's 1-4 buttons

void demo1()
{
  forward(1.0);
  delay(5000);
rotate_clockwise(1.0);
delay(2000);
forward(0.75);
delay(4000);
rotate_counterclockwise(1.0);
delay(3000);
stop();
}

void demo2()
{
}

void demo3()
{
}

void demo4()
{
}

// Main loop
void loop()
{

// Wait for new data to arrive
uint8_t len = readPacket(&ble, BLE_READPACKET_TIMEOUT);
if (len == 0) return;

// Got a packet!
// printHex(packetbuffer, len);

// Buttons
if (packetbuffer[1] == 'B') {
    uint8_t buttnum = packetbuffer[2] - '0';
    boolean pressed = packetbuffer[3] - '0';
    #ifdef DEBUG
    Serial.print("Button "); Serial.print(buttnum);
    if (pressed) {
        Serial.println(" pressed");
    } else {
        Serial.println(" released");
    }
    #endif
    switch(buttnum) {
    case 1:
        if (pressed) {
            demo1();
        }
    break;
    case 2:
        if (pressed) {
            demo2();
        }
    break;
    case 3:
        if (pressed) {
            demo3();
        }
    break;
    case 4:
        if (pressed) {
            demo4();
        }
    }
break;
case 5:
    if (pressed) {
        forward(1.0);
    } else {
        stop();
    }
    break;
case 6:
    if (pressed) {
        reverse(1.0);
    } else {
        stop();
    }
    break;
case 7:
    if (pressed) {
        rotate_counterclockwise(1.0);
    } else {
        stop();
    }
    break;
case 8:
    if (pressed) {
        rotate_clockwise(1.0);
    } else {
        stop();
    }
    break;