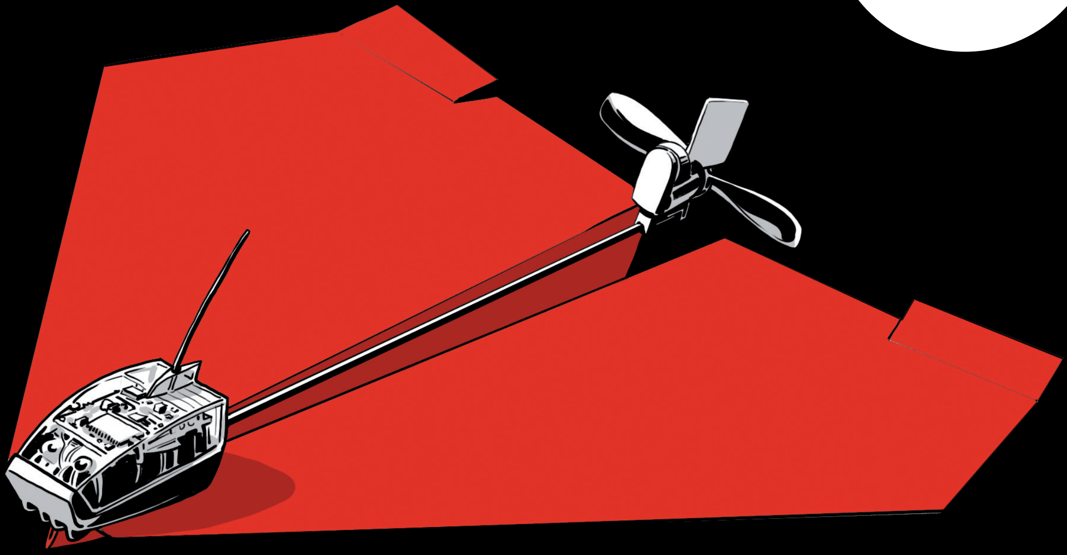


PowerUp® 3.0

Hacker package Documents

Ideas for projects

Welcome to the **PowerUp 3.0** hacker book! The best way to start hacking **PowerUp 3.0** is selecting a cool project to work on. Need some ideas to get started? Be inspired by the following pages.

The ideas are divided into three skill set levels. Choose the skill level that describes your abilities best. After finishing a project you can always try out a higher level of difficulty!

The skill levels

Beginners

You have some basic skills in building, soldering and electronics. No need to know any programming. For this skill level, you will use the **PowerUp** app without any modifications; you will just use it for different solutions.

Intermediate

You are familiar with soldering, remote control hobby projects and electronics. You also know how to program and modify an app. Go ahead, and create your own cool user interface for iOS, Android or Windows. Bluetooth Smart Generic Attribute Profile (GATT) protocols are something you can get into quite easily with support. We will provide you with some links and book references to help.

Advanced

Beginner and intermediate skills bore you? Dive deep and use the TI chip onboard our Bluetooth Smart PCB to do want you want! We will give you some Texas Instruments references and let you try boosting what is already at its limit! This is considered high level and is not for beginners. You should have some experience with embedded development and a lot of curiosity.

The ideas

Balsa Wood Glider

This is a basic, fun and quite simple project. A great choice to start with!

Beginners

Use one of your already existing balsa wood gliders, buy a 2-channel plane from your favorite hobby store or make one yourself. De-solder the existing printed circuit board (PCB) and solder the **PowerUp** Bluetooth PCB into the plane (Please refer to the connection diagram, further in this book).
A rudder and motor is enough for takeoff. Make sure the plane uses a magnetic actuator and no servo. Also try adding some wheels to the glider and build a launch ramp for a quick takeoff!

Intermediate

Use one of your already existing balsa gliders controlled by two motors, buy one from your favorite hobby store or make one yourself. You now have to establish a control for two motors. De-solder the actuator and solder another motor to its connection instead. Make sure the second motor is identical to the first (**PowerUp**) motor. Now it's time to change the app to allow convenient control of a plane using 2 motors (i.e., by using 2 thrust sliders, one for each motor).

Instead of a 2-motor plane, you can connect 2 actuators. Now it's a real glider! Go soaring!

Advanced

Follow the intermediate level, and then try modifying the firmware to allow easier support for a 2-motor plane. **You should change the firmware so that it will receive just a thrust level and a direction and will adjust the motors respectively**. (this is a bit awkward…just to clarify: by adjusting the firmware, the plane will only receive thrust level, and the direction "instructions" will automatically adjust the motors?)

Car

Everything can be controlled with your smartphone, now you will be able to drive around a small car with your **PowerUp** module!

Beginners

Build a 3-wheeled car controlled by **PowerUp**. One wheel in the front, two wheels in the back and a rudder to control the turns. Leonardo da Vinci already used this wheel formation in his 3-wheeler in 1478, the Italian car manufacturer "Piaggio" called it "Ape" – now it's your turn!
Use the Bluetooth PCB for this project. You will need to solder the motor and actuator. The motor will be used for powering the car

wheels, and the actuator will control the front wheel. Try experimenting with different gears on the wheels to get to the right speed and power for you.

Intermediate

Follow the beginner level and then modify your app to easily control your car. Change the interface to something with a steering wheel and acceleration slider on it and make it easy, and more realistic to control the car.

Advanced

Try building a car controlled by 2 motors. Make a relevant app and change the firmware to allow control of speed and direction.

Boat

By Air, by land and by sea! It's time for a smartphone-controlled boat!

Beginners

Just use a plate of styrofoam and place a motor and propeller on top. The actuator will be used as a rudder in the water. You can also use a water propeller. In this case, you have to build a water-proof solution for the motor and the propeller. It is easier to just use the **PowerUp** propeller and use a hovercraft or airboat as your inspiration.

Intermediate

Follow the beginner level and then modify your app to easily control your boat with a creative sea navigation interface. Perhaps change the artificial horizon to an Everglades boat with some alligators in the background to set the mood.

Advanced

Try building a boat controlled by 2 motors. Make a relevant app and change the firmware to allow control of speed and direction.

Space Shuttle

The Space Shuttle is the advancement of an airplane with a rocket motor. This can get a little dangerous, so we suggest a minimum of an intermediate skill level for this idea. Always use caution when handling rocket motors!

Intermediate

Get a model rocket motor from one of the various internet shops or your local hobby store. Attach the rocket motor to your balsa glider from our first idea (see page XX). This time, your take-off is vertical. Launch the rocket and use your smartphone later for soaring in the sky. Re-design your balsa glider to look like a space shuttle! Try modifying the app to look like an authentic space shuttle cockpit.

For advanced

Follow the intermediate level, but this time use the motor! Put the motor at the top of your shuttle to give it some extra flying and thrust power. Try creating a landing sequence where you slowly decrease your thrust for a smooth landing.

Rail Zeppelin

Create your own propeller-controlled train!

Beginners

Create or buy a small model of a train car. Put the **PowerUp 3.0** unit on top of it, and you are ready to go. Build a cool-looking rail system for the train to drive through.

Intermediate

Follow the beginner level. Replace the actuator with a motor and create a braking mechanism to allow for smooth stops. Design your very own rail zeppelin app. Use the throttle to control the motor as well as the speed. Add some LED (lights?) to the brake motor output so that braking will simulate a real train.

Advanced

Follow the intermediate level, but instead of a brake motor, try putting it in the other end of the train car to allow for braking and driving in the opposite direction. Modify the firmware to support this function, and create your own good-looking app interface.

More ideas?

What about an airship (did you mean jet engine?) A construction crane? A sailboat? The sky is the limit, enjoy!

Hardware

Physical Specs

- Dimensions: 24,2x11,3x2,0mm
- Weight: 0.9 Gram ± 5%
- Input voltage: 2.0 - 3.6 V
- ¼ wavelength Monopole RF Antenna
- Link Budget: -92 dBm (approx. 50 meter)
- 3 outputs, max 880mA, voltage same as power supply, **PWM**
- 1x **unidirectional** 8bit resolution,
- 1x **bidirectional** output pair (two outputs, polarity reversible) 7 bit resolution in each direction
- LiPo cell charger (1S), 3.7 V, charge up to 4.2 V. The Charger will charge with 200 mAh.
- On/Off switch
- Status LED

Status LED patterns

Long blinks: waiting for connection
Two short blinks: connected
Continuous fast blinks: battery charging

Power Supply

The PCB can be powered though a battery (with 2.0 - 3.6 V) or using a DC Power Supply Just connect B+ to the supply Voltage and B-to Ground (or – from the battery)

Outputs

The High Power Outputs (A1L, A2R, Motor Output) can supply 1.13A each in Steady state (up to 1.28A peak for 5 seconds) or drain 0.88A in Steady state (up to 1A peak for 5 seconds).

Using Motors and/or Coil actuators

A2L and A2R is designed as H-Bridge so the Output pair can be reversed (for using an coil-actuator or reversing a motor). Those Outputs have a 7 bit resolution PWM signal running at 50 Hz, in each direction and are controlled by the PowerUp app by tilting the phone left/right.

The motor output is just unidirectional which means that just have a half-H-bridge which just hasone Polarity and can not be reversed. This means, if you hook up a motor, you can only regulate thespeed of the motor using the Throttle Slider in the app, but not the direction. The output generated is PWM with 8-bit resolution running at 500 Hz.

Using LEDs

Since there are 3 Physical Outputs/Channels on the PCB (M- is connected to GND) you can easily connect 3 different LEDS using the Outputs as Power Source and the Battery- (or M-) as Ground.

Firmware

Introduction

The firmware is the program running on the little micro-controller on-board the PowerUp. It controls all the functions of the module. By modifying the firmware, the PCB can be made to perform any function, including things not at all related to flying a paper plane! This is suitable only for advanced level hackers.

WARNING: If you don't take care, it is easy to damage your PowerUp by modifying the firmware!

What you need

To be able to customise the firmware, you need knowledge of several domains:
- Good knowledge of the C programming language
- Good knowledge of Bluetooth Smart (see the bluetooth section for more information)
- Basic knowledge of micro-controller programming
- Basic knowledge of electronics

Once you've acquired this knowledge, or are ready to experiment, gather the following materials:

CC Debugger

This is a small black box that lets you connect your PowerUp to a computer through USB for programming the firmware. Without this, it's not possible to program the CC2541 chip. You will need to install the drivers before this works.

IAR Embedded Workbench for 8051

This is a set of tools (compiler, IDE, header files) to compile your firmware programs written in C into 8051 machine code, and debug them. This is a commercial product, and currently there are no free alternatives available. However, IAR offers a 30-day trial version which will work just as well.

Texas Instruments BLE-STACK

This is software provided by Texas Instruments which takes care of most of the Bluetooth communication, so that you can concentrate on the actual functionality. It comes with a lot of examples as well. This is required software and your firmware will not be doing much without it.

MORE INFO: More information about all the required tools is available on the product page of the CC2541 Development Kit: **http://www.ti.com/tool/cc2541dk-mini#Technical%20Documents** under "User Guides."

The firmware image

The electronic module of PowerUp 3.0 is based on the Texas Instruments CC2541 chip. This chip includes an 8051 micro-controller and Bluetooth Smart radio. On the micro-controller, TI's **BLE STACK** is running, which controls the Bluetooth communication. On top of the BLE stack, the PowerUp main application is built which includes code to control the motor and rudder, battery status, provide device information and communicate with the app on the phone using a well-define protocol.

The BLE stack and the main application are together formed into one executable binary image. This image is approximately 128 kB.

Image types

The firmware images are of two types:
A and **B.** Only one image is running at any given time. During startup, the bootloader. The CC2541 chip has 256 kB of flash storage, and each firmware image takes half of this. In this way, the other half of the chip's storage is used to download a firmware update through Bluetooth Smart. This means, if image A is running, only image B can be uploaded to the chip, and vice versa. This also provides safety in case the firmware upload is interrupted - the original running image will keep running.

MORE INFO: More information about the firmware image types, and the firmware upload Bluetooth protocol, is available in the following PDF document:
http://processors.wiki.ti.com/images/8/82/OAD_for_CC254x.pdf

Connecting the CC Debugger to the PCB

The PCB has 4 ports exclusively for programming. These are marked on the PCB connection diagram. You need to do two things before the PCB is detectable and ready to accept firmware upload over USB:

1. Ensure that the **B+** and **B-** pads have 3V power supply (maximum 3.6V).
2. Connect the remaining 4 pins plus the RESET pin to respective wires on the CC Debugger flat cable.

Refer to the CC Debugger documentation to find out which wires should be connected to whichpad.

Programming the PCB

Once you have successfully connected the PCB to the CC Debugger, and pressing the RESET button on it turns the LED green, you are ready to program. From this point onwards, you will need to dive into the Texas Instruments BLE-STACK and code your firmware in C using the IAR 8051 compiler.

MORE INFO: The Texas Instruments e2e forums are a great resource for firmware programmers. It is available here:
http://e2e.ti.com/support/wireless_connectivity/f/538.aspx

MORE INFO: Documentation for the CC2541 chip can be found here:
http://ti.com/cc2541

Bluetooth Smart: A Primer

The following pages describe the various BluetoothSmart services and characteristics that the SmartLink Lite PCB supports. A basic knowledge of how BluetoothSmart works is helpful to understand the description. A short primer is given below.

Data structure

Every device (a "**peripheral**") exposes its functionality via several small pieces of data called "**characteristics**" which are typically a single number, string etc. Related characteristics are grouped together into a "**service**." A few related services are grouped together into a "**profile**" which is purely a convention and is not enforced by the BluetoothSmart protocol.

The BluetoothSmart protocol includes functionality to query a peripheral for the various services it supports, which characteristics are included in each service, its data type, read or write permissions and so on. Access to this database is managed by the Generic Attribute Profile, abbreviated as GATT. Whenever you see reference to a GATT service or a GATT characteristic, it simply means a BluetoothSmart service or characteristic. Every device includes a GATT.

UUIDs

Each service, and each characteristic within any service has a unique identifier: its UUID. These UUIDs are 128-bit long and can essentially be random, but are usually generated using a UUID generation algorithm. Standardised Bluetooth services and characteristics have a 16-bit "UUID," which are actually shortcuts for 128-bit UUIDs constructed from a base "Bluetooth SIG.

Advertising and connection

Every peripheral advertises its presence periodically (the "**advertising interval**") along with basic information (e.g. its name and the primary service it supports). A smartphone (the "central") can listen for these advertisements and connect to the desired peripheral. It can then query the services and characteristics that the peripheral exposes. Once it has all this information, the smartphone can read or write data.

Characteristics

All data is exposed via **characteristics**. A characteristic can be read from, written to, or both. The peripheral decides what it permits. The peripheral may respond to the read or write request acknowledging it (regular read or write), or it may simply act upon it without confirming (read or write without response). When a value of any characteristic changes, the peripheral may notify the connected smartphone (i.e. the "central") about it automatically, without the central periodically checking for it. This is a "**notification**". If the central acknowledges that the notification was received successfully, it is called an "**indication.**"

To receive any notifications or indications, the central must request it to be enabled for the characteristics of interest.
All data exchange happens in periodic bursts, and the interval between two data exchange events is called the "**connection interval**" which is generally chosen by the peripheral depending mainly on how quickly the data needs to be exchanged. Between two data exchanges, the peripheral is typically in power-saving sleep mode to save battery.

PowerUp 3.0 connection parameters

PowerUp 3.0 uses an advertising interval of **500** ms. The connection interval is **20** ms (lowest value allowed by iOS). Both these values may change in the future!
128 bit UUIDs are used everywhere except for the Battery Service which is standardized and thus uses 16-bit UUIDs.
The TailorToys standard 128-bit base UUID (in hexadecimal) is:
86C3810E-xxxx-40D9-A117-26B300768CD6
where **xxxx** is replaced by the 16-bit UUID shorthand mentioned in the following documentation.

PowerUp 3.0 Bluetooth protocol

The SmartLink Lite PCB includes several services:
- Device information service
- Battery level service
- PowerUp service
- Factory test service

Of these, the **PowerUp** service is the most important. It has fields to control the engine and rudder.

NOTE: Not all services may be present on your module. Out of the box, modules have the Factory test service but not the PowerUp service. After the "first flight setup," the factory test service is removed and PowerUp service is installed.
A detailed description of every service follows.

0xF171 - PowerUp Service

This service currently has three characteristics: one for the motor speed, one for the rudder turning angle, and one for the battery charger on board the PowerUp. The motor speed and rudder angle can be read or written. The charging status can only be read (or notified). When the motor is turned off (value 0) and the rudder is centered (value 0), the chip can go into power saving mode. Otherwise, the chip is continuously running, drawing approximately 10 mA from the battery, even if the motor and rudder are set to very low values. So to save power, write 0 to motor and rudder when they are not actually being used. Furthermore, write a value only if it has actually changed, i.e. avoid writing the same value over and over again. This will help reduce data transmission.

0x0010 - Motor Speed

Allowed	Read/Write
Type	unsigned 8-bit integer
Range	0 to 254

This characteristic controls the motor speed. A value of 0 turns the motor completely off, a value up to 254 increases the motor speed proportionally from 0% to 100%. A value of 255 is out of range and will turn the motor off again. On the hardware, the motor speed is controlled by applying a PWM signal of varying duty cycle at a frequency of ~500 Hz (subject to change). The pulse width is controlled by this characteristic.

0x0021 - Rudder (actuator) angle

Allowed	Read/Write
Type	signed 8-bit integer
Range	-128 to 127

This characteristic controls the rear rudder's turn angle. A value of 0 centers the rudder and uses negligible power. A value of -128 turns the rudder fully to the left, while a value of 127 turns the rudder fully to the right. Intermediate values cause the rudder to turn proportionally either left (negative values) or to the right (positive values) from the center position.
On the hardware, the rudder is attached to a magnetic coil actuator, to which a PWM signal is applied at approximately ~50 Hz (subject to change). The pulse width as well as the phase inversion is controlled by this characteristic. Negative values invert the phase of the PWM, which causes the actuator to turn in the opposite direction.

0x0040 - Battery charger status

Allowed	Read/Notify
Type	unsigned 8-bit integer
Range	0 or 1

This characteristic provides information about the battery charger. When 0, the battery is not being charged. When 1, the battery is being charged. You can enable notifications on this characteristic to avoid having to read it periodically - the chip will send a notification when the value changes. On the hardware, this is connected to the STAT pin of the LiPo charger IC. Normally the battery is "in-use" and this characteristic will read 0. When a microUSB cable is plugged in, supplying 5V, and the charger IC has started charging the battery, this characteristic will change to 1.
It may take a few seconds for the PCB to recognise that charging has started or stopped.

0xF181 - Factory test service

This service is used only during production. It includes functionality to test proper functioning of the electronics and that the various connections are correctly made.
It also has commands to turn on special

RF transmission and reception modes for regulatory (FCC/CE) testing.

WARNING: This service is not intended to be accessed by end-users, and is therefore not documented.

NOTE: This service is only present on a brand new PowerUp. It is erased and replaced with the PowerUp service during the first flight setup.

0x180F - Battery Service

This is a standardized Bluetooth service to report the battery level in percentage. It has only one characteristic, the battery level. Please refer to the Bluetooth specification documentation for more detailed information.

IMPORTANT: The base UUID for this service and its characteristics is the Bluetooth SIG base, not the Tailor Toys base.

MORE INFO: You can download the official specification here: **https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=245138**

0x2A19 - Battery level

Allowed	Read-only
Type	unsigned 8-bit integer
Range	0 to 100
Unit	Percent

This characteristic reports the battery level in percentage. Value 0 represents an almost depleted battery while value 100 represents a fully charged battery. On the hardware, the battery voltage is measured once per 6 seconds, and converted to a percentage value for voltages between 3V - 3.75V. Note that this measurement is very approximate, and may fluctuate often.

0x180A - Device Information service

This is another standardised Bluetooth GATT service which provides information about the device, such as the manufacturer's name, model number, serial number and so on. Refer to the official specification for more information.

MORE INFO: You can find the official specification here: **https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=244369**

Below we'll provide a short explanation of what each field means in the **PowerUp**:

Manufacturer name, model number

These fields are as they are named, to provide information about the manufacturer and model number. It will be **Tailor Toys LTD** and **PowerUp 3.0** respectively.

Serial number

We use this field in a slightly different way. It provides you the production batch number and not individual serial number. This is a string. Example: **TTPU/rev1.5/001** means:

- **TTPU**: internal code for Tailor Toys PowerUp
- **rev1.5**: internal code for hardware revision (here, rev 1.5)
- **001**: first production batch

The exact format can differ. For example, the beta units say **TTPU/r1.6/BETA-200** . It is recommended not to interpret this string. Use it only for informational/display purposes.

System ID

This field is a 64 bit number which uniquely identifies a particular device. It is built from a combination of the internal MAC address and other values. This value should be used as an identifier only, the actual value has no pattern to it and is not a "serial" number.

Firmware version

A string that gives information about the currently running firmware image. This is a quick way to check which firmware version your PowerUp is running. It is a string, typically containing a 3 digit number. Example: **207**.
If your PowerUp is still fresh off the factory, the factory firmware version will be displayed (example: **factory-160**). Higher numbers mean newer firmware versions.
Factory firmwares generally have versions in the 100 range. Full firmwares have version in the 200 range.

NOTE: his field is for informational purposes only. The version information provided by the "Firmware service" is the accurate value. The two might mismatch.

Hardware version

String containing internal hardware revision number. Typically, it is **1.5** or **1.6** or similar.

Firmware Upload

The firmware upload service is used to wirelessly upload newer firmware versions to the PowerUp.
This follows the **OAD protocol** defined by Texas Instruments.

MORE INFO: You can find the official documentation about the protocol on Texas Instrument's Wiki page: **http://processors.wiki.ti.com/index.php/OAD**

The 128 bit UUID for this service is: **F000FFC0-0451-4000-B000-000000000000**

It has two characteristics:
- Image identification: UUID **F000FFC1-0451-4000-B000-000000000000**
- Block upload: UUID **F000FFC2-0451-4000-B000-000000000000**

Please refer to the official TI OAD guide (linked above) for detailed information. See also the Firmware Programming section of this guide.

How to explore the PowerUp via Bluetooth Smart, without programming?

Want to access the internals of your PowerUp over Bluetooth Smart but don't want to write your own code? No problem! There are several ready-made "Bluetooth explorer" apps available. For iOS, we recommend you to try out the free "Light Blue" app made by the fine folks at Punch Through Design. A non-free alternative is the "BLExplr" app made by Michael Kroll.

Android users can try out Texas Instruments' BLE Device Monitor app (free), or BLE Scanner by MacDom Apps Studio.

MORE INFO: Light Blue app can be downloaded from **https://itunes.apple.com/us/app/lightblue-bluetooth-low-energy/id557428110**

MORE INFO: BLExplr app can be downloaded from **https://itunes.apple.com/us/app/blexplr/id524018027**

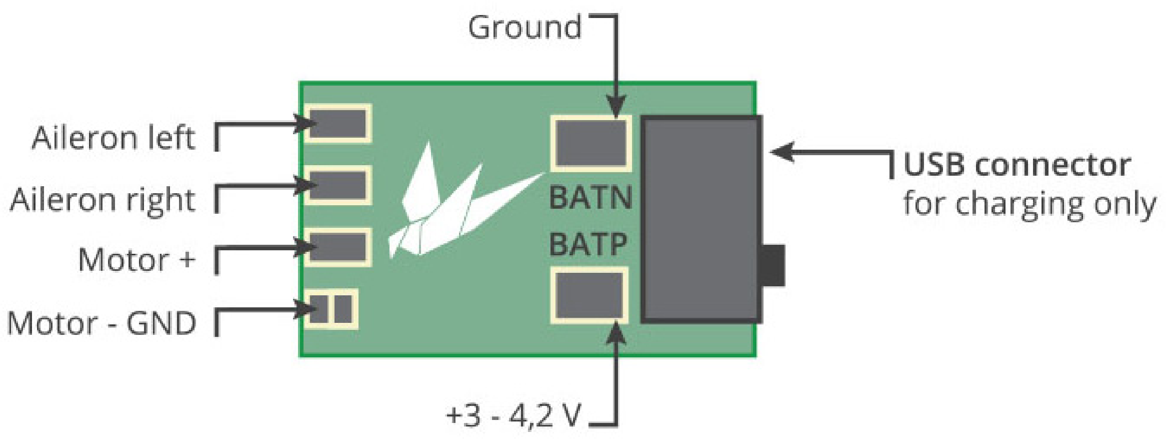
MORE INFO: BLE Device Monitor app can be downloaded from **https://play.google.com/store/apps/details?id=ti.android.ble.devicemonitor**

MORE INFO: BLE Scanner app can be downloaded from **https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner**

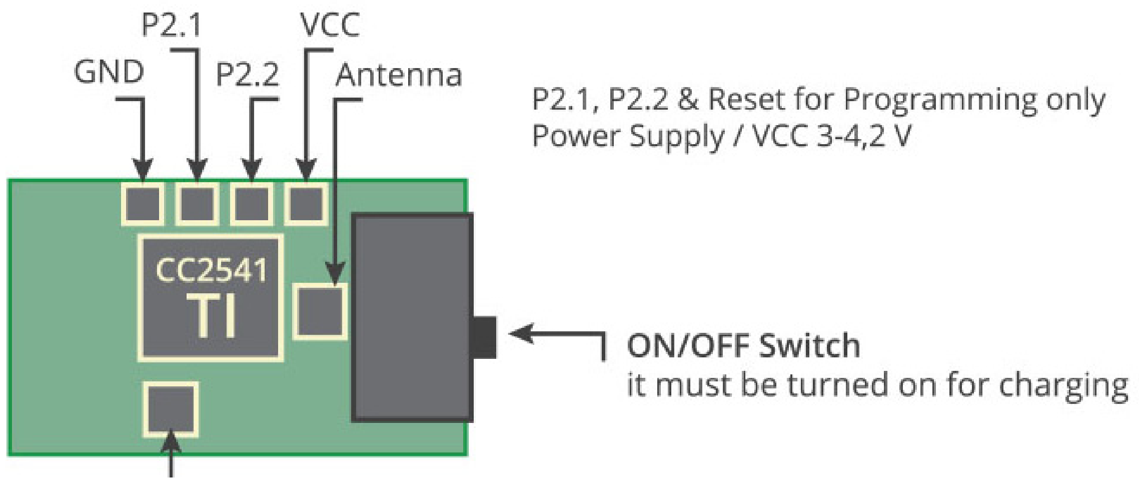
Smartlink technology Diagram

Back

Aileron direction can be reversed in the app.



Front


P2.1
VCC
GND
P2.2
Antenna
P2.1, P2.2 & Reset for Programming only
Power Supply / VCC 3-4,2 V
CC2541
TI
ON/OFF Switch
it must be turned on for charging