CJ STEALTH RC

VERSION 1.1

REFERENCE GUIDE

2/11/2014

1. CONTENTS

2. ACRONYMS AND TERMS

Arduino	Arduino is a single-board microcontroller. The hardware consists of an open-source hardware board designed around an 8-bit Atmel AVR microcontroller. The software consists of a standard programming language compiler and a boot loader that executes on the microcontroller.
CLI	Command Line Interface
DC	Direct Current
DOUT & DO	Digital Output – used for communicating to other devices
EEPROM	Electrically Erasable Programmable Read-Only Memory and is a type of non-volatile memory used in computers and other electronic devices to store small amounts of data that must be saved when power is removed.
FS	Fail Safe – upon failure detection, to return all outputs to known safe state
GHz	Gigahertz
GND	Ground
I2c	Inter-Integrated Circuit (multi-master serial bus)
Li-PO	Lithium-ion polymer batteries, polymer lithium ion or more commonly lithium polymer batteries (abbreviated Li-poly, Li-Pol, LiPo, LIP, PLI or LiP) are rechargeable batteries.
MHz	Megahertz
MP3	MP3 is an audio-specific format that was designed by the Moving Picture Experts Group (MPEG)
PPM	Pulse Position Modulation
PWM	Pulse Width Modulation
RC	Radio Control
TTL	Transistor-Transistor Logic – digital logic levels where a “1” is > 1.6 Volts and a “0” is below 0.8 Volts
Speed Controller	An electronic speed control or ESC is an electronic circuit with the purpose to vary an electric motor's speed, its direction and possibly also to act as a dynamic brake. ESCs are often used on electrically powered radio controlled models and robots.
USB	Universal Serial Bus (USB) is an industry standard that defines the cables, connectors and communications protocols used in a bus for connection, communication, and power supply between computers and electronic devices.
V	Voltage
XBEE	XBee is the brand name from Digi International for a family of form factor compatible radio modules.

3. TECHNICAL SUMMARY

3.1. OVERVIEW

CJ Stealth RC is a compact, fully integrated, advanced radio control with integrated sound system. It's designed for droid and robot builders to replace a conventional Radio Control (RC) setup. However, it can be used in conjunction with a conventional RC radio if desired.

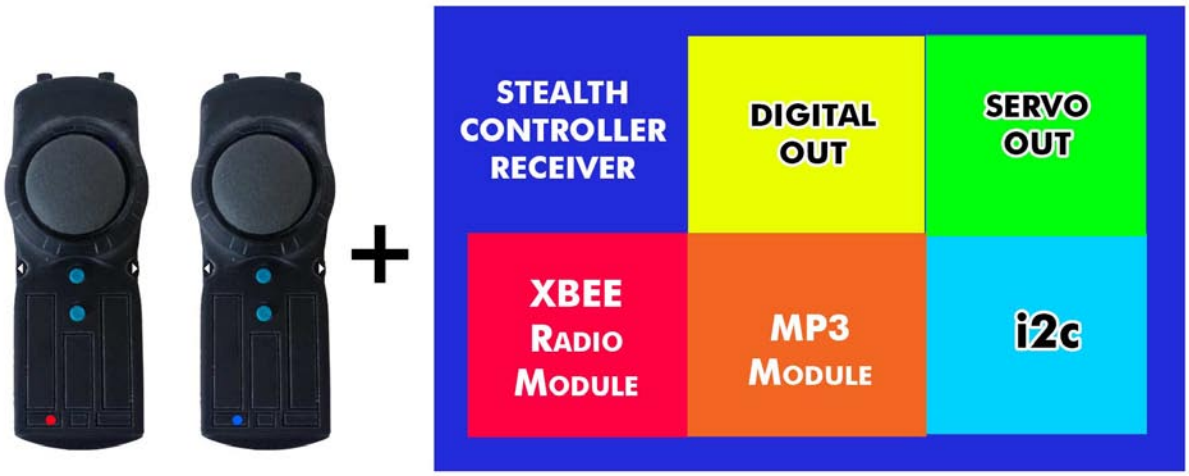
- **Super small** - fit in the palm of your hand - allowing the droid operator to be covert or stealth like
- **Light weight** - 2 ounces vs. typical 3 pounds RC Remote
- **Thumb Gestures** adds an additional way to trigger actions and sounds beyond normal buttons
- Integrated **MP3 Sound System**
- Radios operate at either 2.4GHz or 900MHz (later recommended for more range and less interference)
- **Advanced Digital Communications -** Digital Out & i2c Communication
- Highly configurable via a Command Line Interface (CLI) or USB Thumb Drive
- Control up to **12 Servos** directly, or dozens thru Servo Expansion Modules including advanced scripting
- As a bonus when using RC Mode you can trigger sounds and enter **Thumb Gestures** on your conventional RC remote

This guide is split into sections for each of the components, including technical specifications and setup for each, a physical and logical view, and how to configure. The guide also include a Troubleshooting/FAQ and Appendix.

3.2. COMPONENTS

In its simplest form the system is made up of the following components

- CJ Stealth Pocket Remotes (SPR)
- CJ Stealth Controller Receiver (SCR)



Optional components available as part of Stealth RC but not necessarily needed:

- CJ Stealth PWM/PPM Converter Module - use a conventional Radio Control as backup
- CJ Stealth Servo Expander Module - add more servos and outputs
- CJ Stealth DC/DC Converter - Reliable 6 Amp +5V power supply to drive the Stealth Controller Receiver and servos

Depending on your final setup, some additional components will be needed. For example:

- Sound Amplifier
- Ground Loop Isolator
- Speakers
- Speed Controller(s)
- Additional DC/DC Converter(s)
- Miscellaneous cables and connectors (servo extension cables, 3.5mm audio cable/adaptors, fuses etc.)

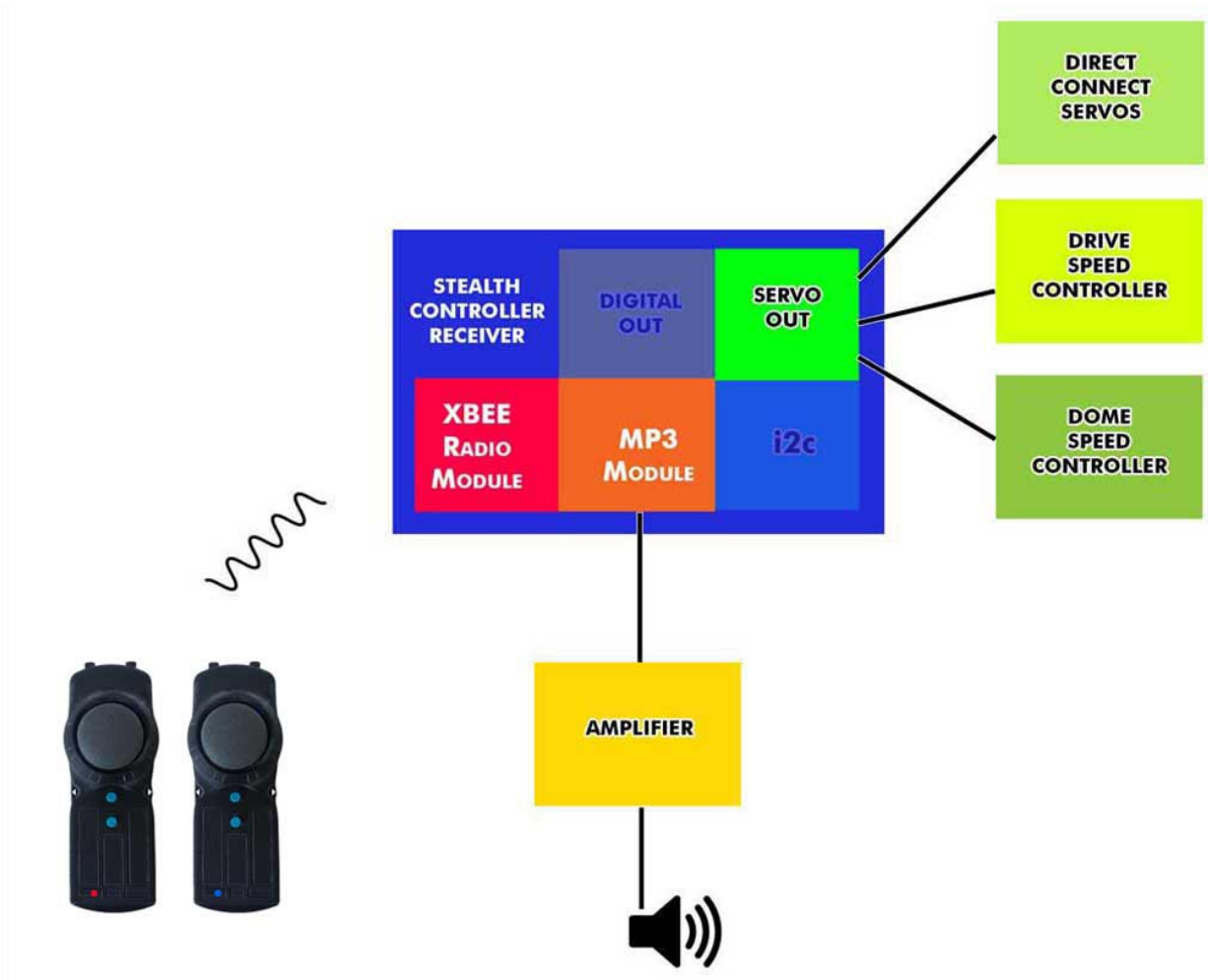
3.3. EXAMPLE CONFIGURATIONS

3.3.1. STANDARD/MINIMUM

For a user with no existing RC setup, this is what your typical Stealth RC setup may look like connected to other components in your system (speed controllers and amplifier.)

Grayed out boxes means the feature isn't used/needed in a minimal configuration.

3.3.1.1. LOGICAL VIEW



3.3.1.2. PICTORIAL VIEW OF MINIMAL STEALTH RC DEVICES

3.3.2. ADVANCED

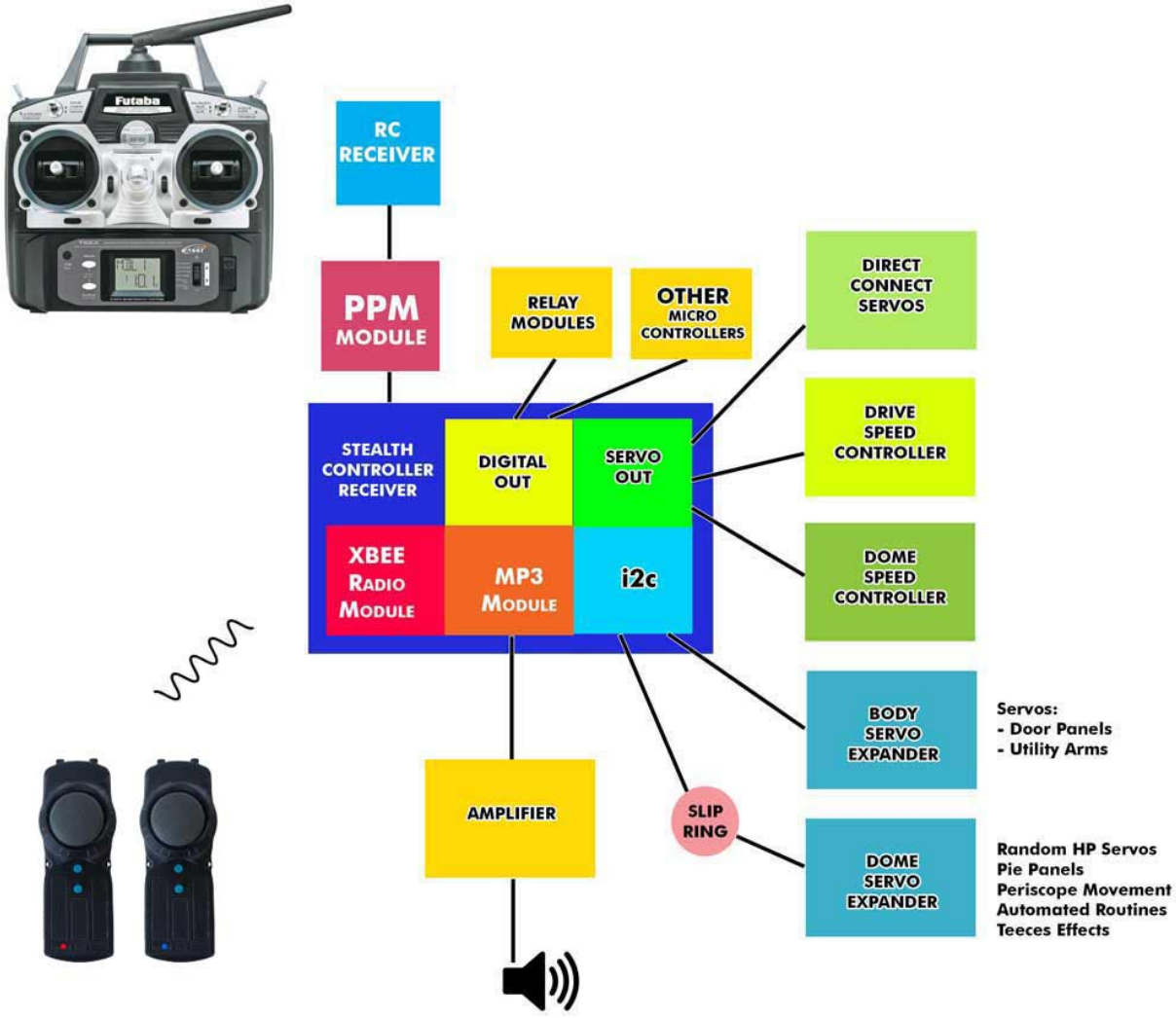
This is a much more complex setup, with:

- ***PWM/PPM Converter Module*** to allow switching over to an existing RC radio as backup.
- Two ***Servo Expander Modules***, one in the dome and another in the body. An example use of the modules maybe to control random servo movement for the Holo (HP) in the dome, or to open panels and doors, open utility arms, trigger periscope, etc. See example code in section 6.6. Example Code
- You can also drive external devices via the **Digital Out** ports, e.g. to turn a relay on or off, turn on lights, or to talk with another micro-controller which does not support i2c

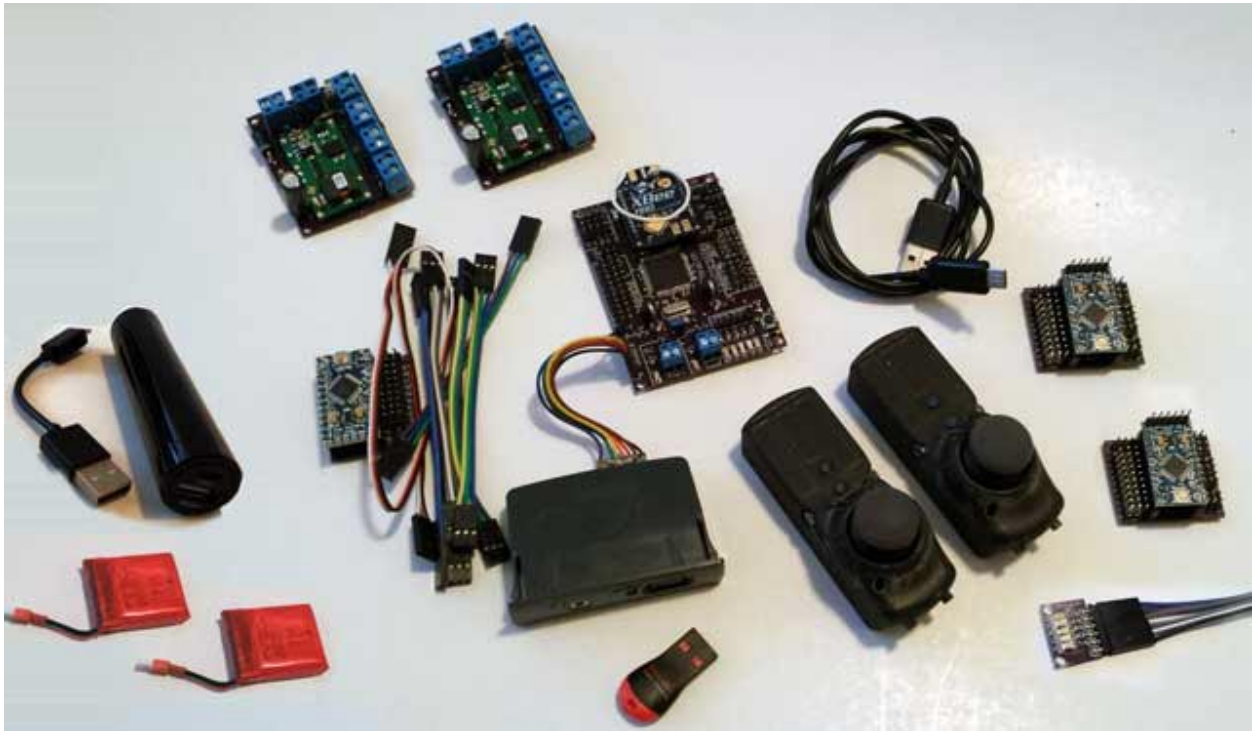
Advanced coordination of routines can be orchestrated by timing i2c events among the main Stealth Controller and various Expander Modules.

With some small modifications to a Teeces lights setup, you could trigger effects or scrolling text on the display.

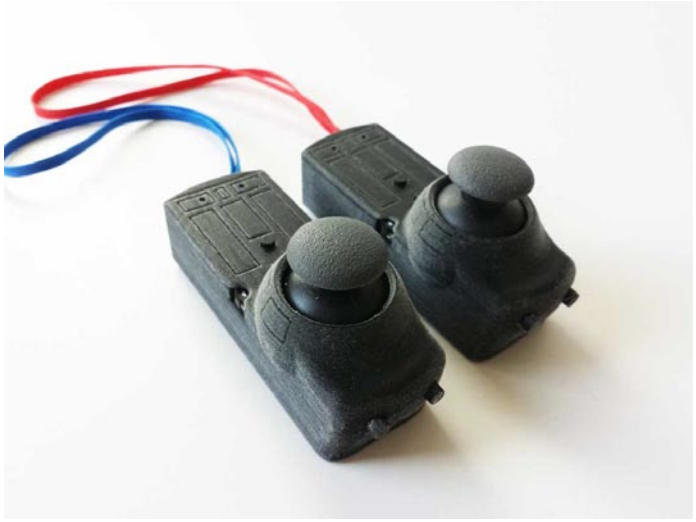
3.3.2.1. LOGICAL VIEW



3.3.2.2. PICTORIAL VIEW OF STEALTH RC DEVICES



4. CJ STEALTH POCKET REMOTES



There are two remotes in a setup, and in their simplest configuration they are designed to mimic a traditional 6 channel radio setup. Remote 1 (**R**ed) maps to the right side and Remote 2 (b**L**ue) to the left side of a conventional RC transmitter.

4.1. FEATURES

- Each Stealth Pocket Remote (SPR) has an analog joystick, 5 buttons and 2 analog thumb wheels.
- An XBEE Radio Module inside
- Powered by a rechargeable Li-Po battery
- Integrated Li-Po charger
- Power Status Indicator LED
- On/Off Switch
- Charging Port and Status LED

Except for the Power LED color, each SPR is physically identical, but logically they differ slightly. Remote 2 (blue/left) is used for Thumb Gesture input and sound/volume control.

Both joysticks control servos (1,2 and 3,4 respectively) and the two thumb wheels on the Remote 1 (red/right) controls servos 5 and 6.

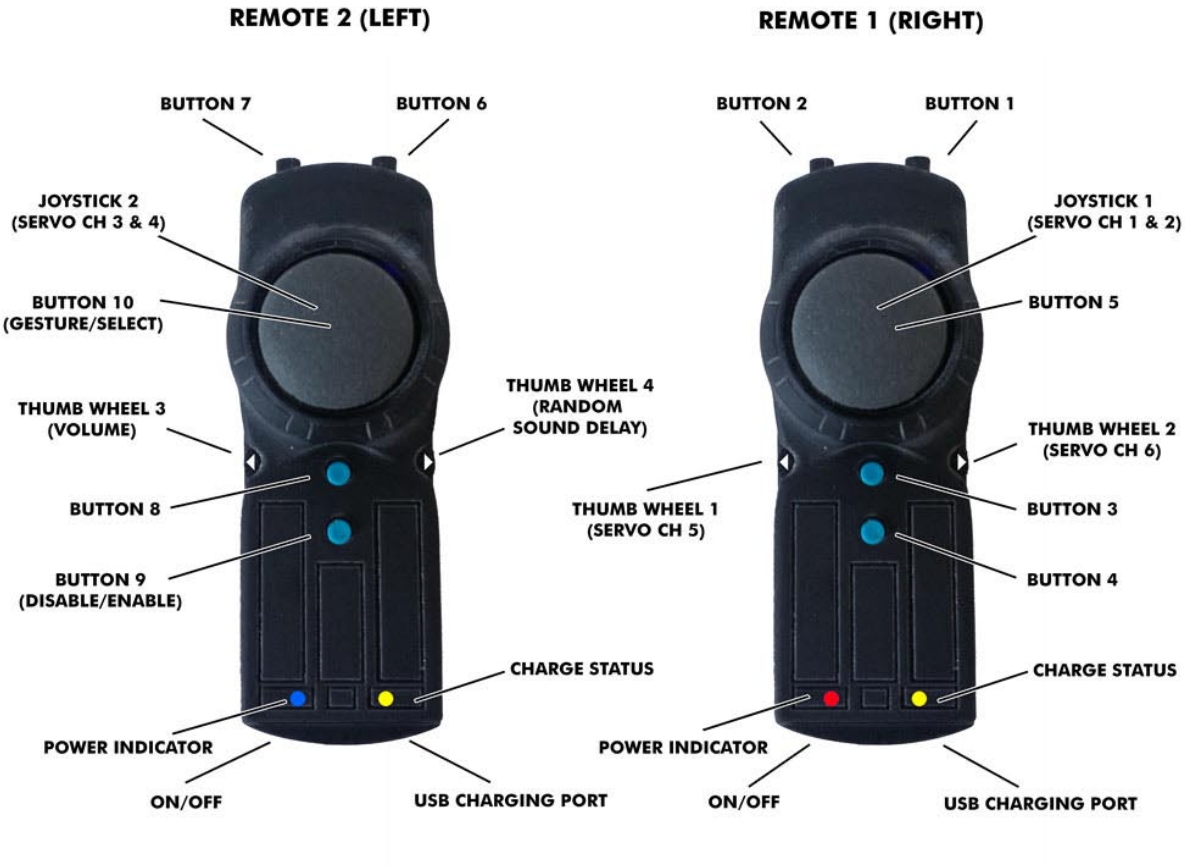
But there's much more you can control and reconfigure. There are 4 input methods: Joysticks, Buttons, Thumb Wheels and Thumb Gestures, and 4 output action types: Servos, Sounds, Digital Outs, and i2c Commands.

The remotes uses XBEE Radios for communication (Series S2B 2.4GHz and original 900MHz XBEE radios are supported.) Range will vary depending on model used, surrounding environment, build material, and antenna placement and frequency used.

Like a conventional RC setup, your remotes can only communicate with its paired receiver. The XBEE radios modules in the Stealth RC setup come preconfigured with a unique ID and encryption key, and shouldn't need changing. Multiple Stealth RC setups should be able to operate in the same vicinity without problem.

4.2. STEALTH REMOTE LAYOUT

Even though the remotes are physically two separate units, logically they work together as one.



4.3. REMOTE INPUT METHODS

There are four input methods - Analog Joysticks, Buttons, Analog Thumb Wheels and Thumb Gestures.

These are either mapped permanently or can be configured to perform actions if noted.

4.3.1. JOYSTICKS

Joystick	Remote	Description
Joystick 1	1	Servo Channels 1 and 2
Joystick 2	2	Servo Channels 3 and 4

4.3.2. BUTTONS

Button	Remote	Description
1-8	1 and 2	Can be mapped to an individual sound, a sound bank, servo port (S7- S11), digital out (D1-D8), or send an i2c command to an external device
9	2	Disable or Enable both joysticks including the buttons, except the Button 9. This button cannot be reassigned
10	2	Start / End Gesture - this is the large joystick button on Remote 2 This button cannot be reassigned

4.3.3. THUMB WHEELS

Wheel	Remote	Description
Thumb Wheel 1	1	Servo Channel 5
Thumb Wheel 2	1	Servo Channel 6
Thumb Wheel 3	2	Control sound volume
Thumb Wheel 4	2	Control frequency/delay between random sounds

4.3.4. THUMB GESTURES

A Thumb Gestures is a different way to signal that you want the receiver to perform a specific action (Play a sound, move a servo, send an i2c Command, Trigger a Digital Output.)

The reason for Thumb Gestures is that we often have more things we want t to trigger than the number of buttons we have available. Basically we now have a "one too many" mapping through one joystick.

Gestures are entered thru Joystick 2 when it's in a special mode - it is a sequence or pattern you "spell out" with joystick movements (up/down, left/right and the diagonal corners). Thumb Gesture patterns are entirely configurable to which action they trigger.

For example, to trigger a sound (but could be any of our 4 action types)

1. press *Button 10 (Gesture Select)* - Stealth Controller starts capturing gesture
2. wiggle *Joystick 2* to input the predefined gesture pattern
3. press *Button 10 (Gesture Select)* - Stealth Controller stops capturing gesture
4. Stealth Controller translates your gesture to the predefined action. In this case play a sound

4.3.4.1. HOW THIS WORKS INTERNALLY

Each Thumb Gesture pattern is represented internally by a unique string of numbers. Those numbers map to one of 8 directions our joystick moves (up/down, left/right and the diagonal corners). Imagine the possible joystick positions as numbers on a grid, with 5 being the center position.

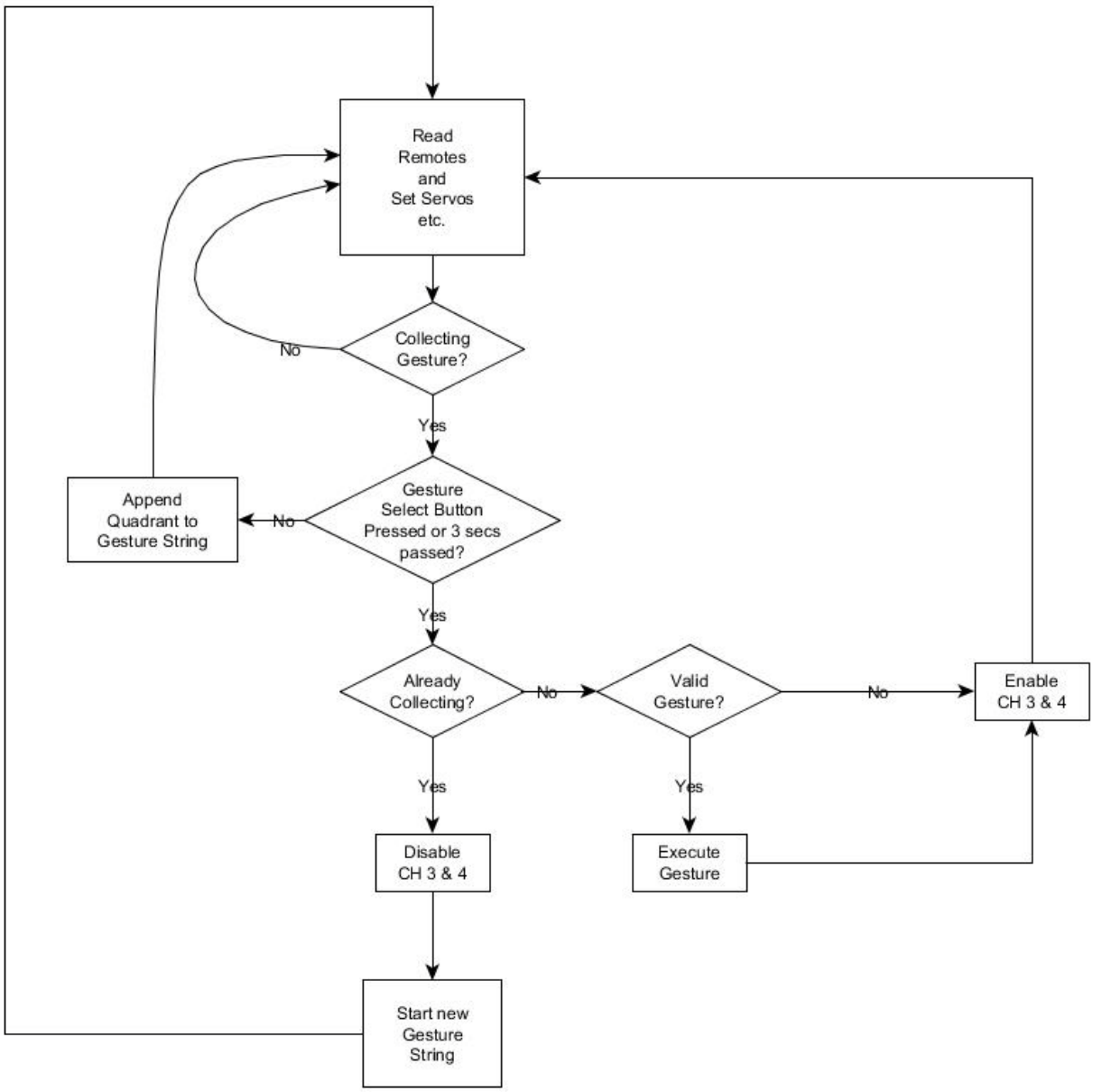
1	**2** (UP)	**3**
4 (LEFT)	**5** (CENTER)	**6** (RIGHT)
7	**8** (DOWN)	**9**

As your joystick moves from quadrant to quadrant the receiver adds the corresponding number to the pattern.

For example, moving the joystick left once, is "4", or moving the joystick twice left is "454".

4.3.4.2. INTERNAL LOGIC OF CAPTURING A GESTURE

Simplified logic when starting, capturing and stopping a Thumb Gesture



To initiate a gesture press ***Button 10***. Then you move the stick to input the gesture. Either press ***Button 10*** again to end/close the gesture, or wait 3 seconds and it will time out and perform the gesture action. If no gesture is made i.e. you didn't move the stick the action assigned to Gesture "5" is performed. It's recommended this is assigned to a random sound bank. If no valid gesture is found then nothing happens.

If you configure Gesture 5 as a sound, a simple double tap of the button will let you know if you're in a valid state and also act like an extra sound button.

You can also turn on *Gesture Start Acknowledgment* (a sound) by setting "***ackon=y***" in the configuration file, or performing the *ackon* gesture (if defined correctly in the configuration file.)

During gesture entry, servo channels 3 and 4 are returned to neutral, and restored when done.

4.3.4.3. THUMB GESTURES PATTERNS

There are two constraints when defining a Thumb Gesture pattern:

- **Physically** - the resolution of the joystick is small and if you're keeping the remote in your pocket it sometimes be hard to be accurate. Best to use simple patterns.
- **Logically** - internally we have an upper limit of 9 digits in our pattern string.

See section 5.7.4

Thumb Gestures on how to assign Thumb Gestures to specific actions.

4.3.4.4. AUTO CORRECT GESTURES

The ***auto*** parameter turns on auto correction of some common gestures, like the hard to hit corners. This is useful when learning gestures. But it does limit the number of gestures available. For example if you try and do a "LEFT TOP DIAGONAL" gesture which equals to "1", you can get there through several paths which may not be a direct line to the corner. You may actual do a UP LEFT ("21") instead of diagonal. Auto correct will convert this to a "1".

The system isn't perfect and it may fail to auto correct some patterns, especially performing double taps to the corners but it's should prove useful.

This feature is enabled by default.

4.3.4.5. EXAMPLE GESTURES

Pattern String	Thumb Gesture / Joystick Movement
2	UP
4	LEFT
5	NOTHING / DEFAULT
6	RIGHT
8	DOWN
1	LEFT TOP DIAGONAL
3	RIGHT TOP DIAGONOL
7	LEFT BOTTOM DIAGONAL
9	RIGHT BOTTOM DIAGONAL
258	UP, CENTER, DOWN
852	DOWN, CENTER, UP
456	LEFT, CENTER, RIGHT
654	RIGHT, CENTER, LEFT
252	UP, CENTER, UP
454	LEFT, CENTER, LEFT
656	RIGHT, CENTER, RIGHT
858	DOWN, CENTER, DOWN
25252 or 45454 or 65656 or 85858	Easy Triples
2585258 or 8525852	Double Up/Downs
4565456 or 6545654	Double Left Rights

And you could have more complex combinations, but I'm sure you'll soon forget what's set to what.

4.4. MAINTENANCE AND SAFETY FEATURES

4.4.1. FAILSAFES

The Stealth Remotes can be used independently, and they both do not need to be turned on simultaneously for the system to function.

When you first turn on the remotes, you may see that Stealth Controllers status LEDs J1 and J2 blink off and on a few times. This is the XBEEs establishing a network. On 900MHz radios the network is established almost immediately, but can take 5-10 seconds on 2.4GHz radios.

Not critical but you could turn on the radios first before powering on your droid, that way the network is established in parallel with the Stealth Controller Receiver booting up. And you can turn the remotes on and off anytime during normal operation - it may just take a few seconds to re-establish the network.

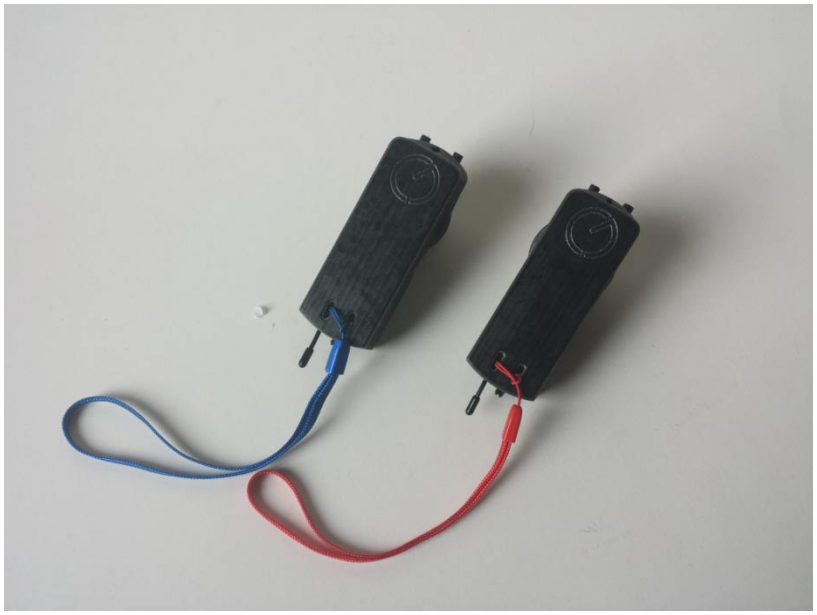
How Stealth RC handles failsafes is the opposite to how most conventional RC setups do it, especially aircraft RC models - where failsafes can be a major issue when using them in robotics applications. Most will NOT returned servos to neutral but continue repeating the last instruction/action/channel they received. Result being a run-away droid. Stealth RC returns servos 1-6 back to neutral in a failsafe situation.

If the Stealth Controller does not received a signal from a remote for any reason within 500 milliseconds, then its assigned servos are returned to a predefined neutral position and disengaged, this includes speed controllers.

In normal operation, the Stealth Remotes send data to the Stealth Controller every 100 milliseconds. This is adjustable by using an XBEE programmer. Increasing the send rate may improve joystick sensitivity, but at the expense of battery life. The default value will work fine for robotics applications.

4.4.2. SAFETY STRAP

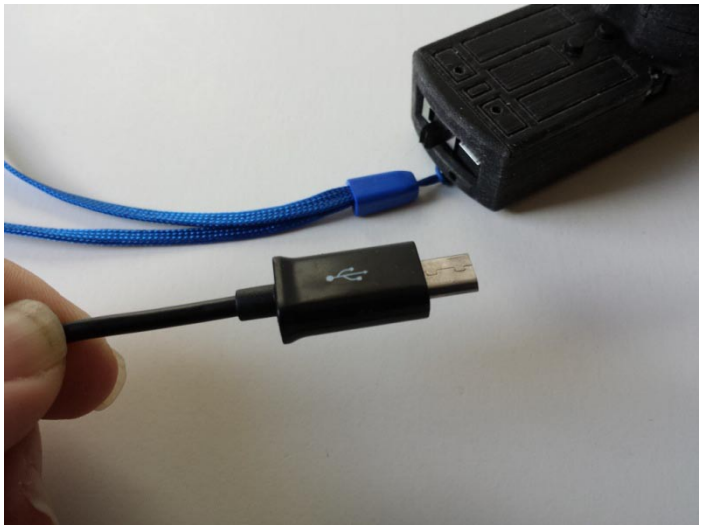
Each remote has a slot on the bottom where you can insert a safety strap to secure the remote to your wrist.



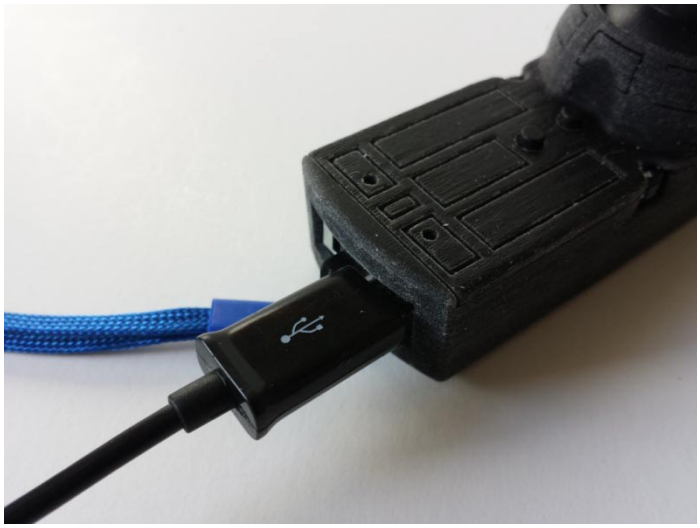
4.4.3. CHARGING

Each remote is powered by a small 3.7V 400MAh rechargeable Li-Po battery. Depending on the XBEE radios installed and condition/age of the battery you should get around 4 hours of use on 900MHz radios and 6 hours on 2.4GHz radios.

Smaller Li-Po batteries will not have an under voltage protection circuit and can be damaged if left turned on and drained.



Device is charged through a micro USB port on back using the cable provided. This can be connected to an AC Adapter (not provided), computer USB port or External Booster Battery. Charge time will vary depending use, but should be about 2-3hrs from "empty".



When charging the yellow charging LED at the rear will come on, and turn off when done.

It's tricky to replace the internal battery on the fly.

If you you're worried about being stuck in the field with a low battery, you may want to purchase an External USB Booster Battery.

They come in varying sizes and capacities.

4.4.4. REPLACING INTERNAL BATTERIES

The remotes use small, 3.7V 400mAh Li-Po rechargeable batteries. These can come in varying sizes and capacities, but the one's included will either measure 9mm x 25mm x 31mm or 8mm x 22mm x 37mm. The critical dimension is the width which can't exceed 25mm.

When replacing please pay close attention to the orientation/polarity of the plug.

To replace the battery:

1. Turn off power
2. Remove front screw
3. Open the case
4. Carefully lift out the circuit board
5. Disconnect old battery
6. Reconnect battery (pay close attention to polarity)
7. Insert circuit board
8. Don't forget buttons
9. Close case and replace screw (do not over tighten otherwise you may crack the case)

5. CJ STEALTH CONTROLLER RECEIVER

At the heart of the system is the CJ Stealth Controller Receiver (SCR). It's configured via a series of jumpers, a configuration file stored on the USB Thumb Drive (CONFIG.TXT) or stored configuration in memory (EEPROM.)

Your servos and speed controllers plug in directly as you would with a conventional RC Receiver.

5.1. COMPONENTS

The Stealth Controller Receiver is made up of 4 physical parts:

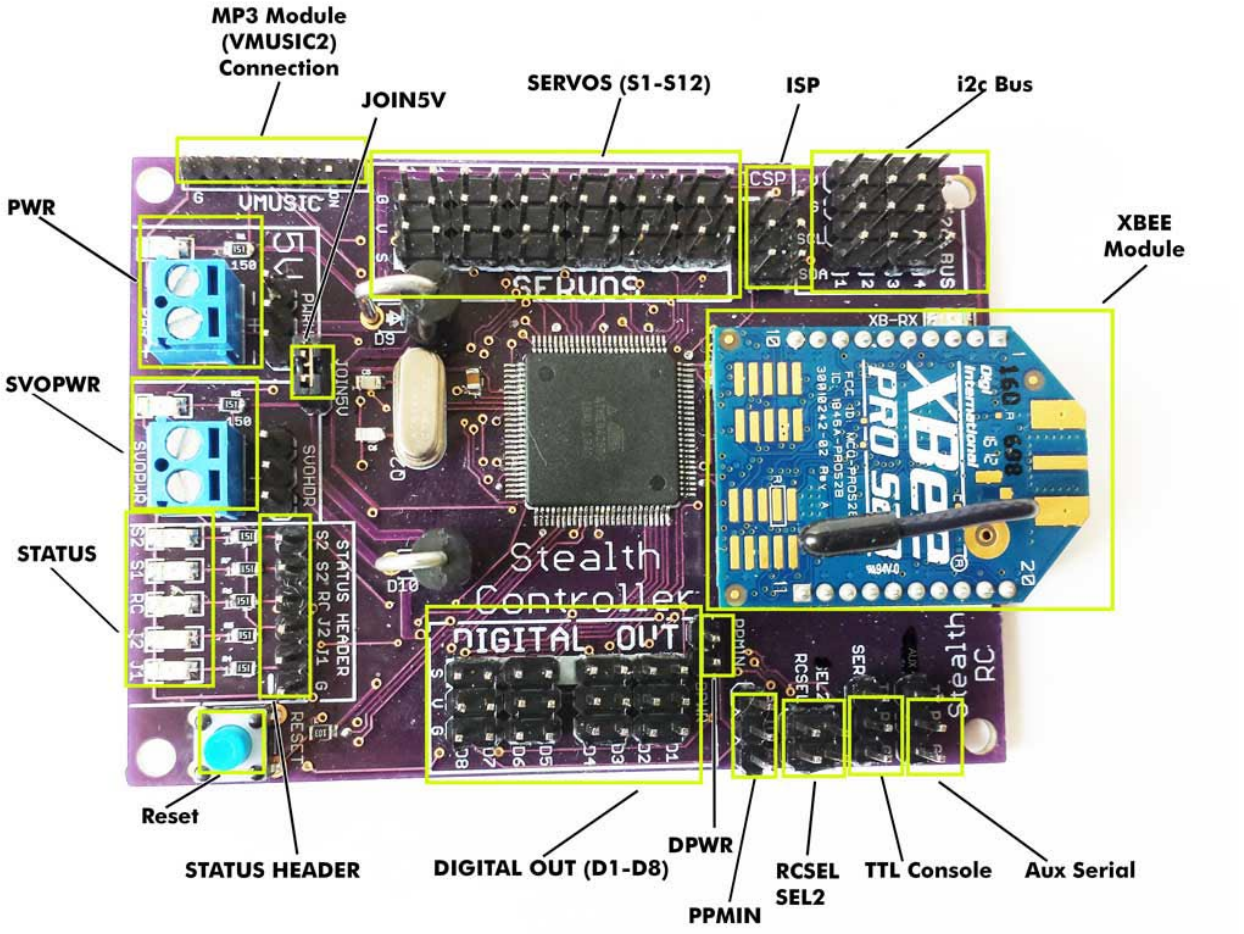
- Controller Board
- MP3 Sound Module (VMUSIC2)
- XBEE Module
- USB Thumb Drive

Optional parts:

- PPM/ PWM RC Converter
- Servo I/O Expander Module
- DC/DC Converter

5.2. STEALTH CONTROLLER RECEIVER CONNECTIONS

5.2.1. OVERVIEW



In a basic configuration many of the connections are not used. At a minimum the following would be connected in your droid:

- Power (PWR)
- SERVOS: Drive Speed Controller (S1 and S2) + Dome Speed Controller (S4)
- XBEE Module
- MP3 VMUSIC2 Module + USB Thumb Drive

Please be careful on the orientation of all cables.

Connections are label on the board and detailed below

5.2.2. POWER

Before connecting power to the Stealth Controller please read section 5.5. Power Considerations.

DO NOT CONNECT +12V directly to the Stealth Controller. This is a +5V ONLY device.

5.2.3. PWR - EXTERNAL POWER FOR CONTROLLER

The **PWR** connector should be used to supply +5V to the main controller. You can use jumper **JOIN5V** to connect this supply to the internal servo +5V supply (but use with caution.)

Terminal	Description
-	Ground
+	+5V

5.2.4. PWRHDR

Pin	Description
1	Ground
2	+5V
3	Ground

PWR and **PWRHDR** are the same connected input, one is a screw terminal the other a convenient 3 pin jumper header.

5.2.5. SVOPWR

Terminal	Description
-	Ground
+	+5V or +6V

SVOPWR supplies power to any servos connected, including the PPM Module and the +V pins on the Digital Out connections (if **DPWR jumper** is used/enabled). Typically SVOPWR is +5V but could be +6V to when driving larger servos. Please be careful not to use the JOIN5V jumper if you plan on using +6V as this will damage the Stealth Controller.

5.2.6. SVOHDR- SERVO POWER SUPPLY

Pin	Description
1	Ground
2	+5V
3	Ground

SVOPWR and **SVOHDR** are the same input, one is a screw terminal the other a convenient 3 pin jumper header.

5.2.7. JOIN5V - COMBINE CONTROLLER AND SERVO +5V SUPPLY

Pin	Description
1	Controller +5V Supply
2	Servo +5 Supply

This jumper allows you to connect the Stealth Controllers +5V Supply with the Servos +5V Supply, mitigating the need for an extra +5V supply in some instances.

5.2.8. DPWR - ENABLE +5V SUPPLY TO DIGITAL OUT +5V PINS

Pin	Description
1	Servo +5 Supply
2	Digital Out +5V Bus

This jumper allows you to connect the Servo +5V Supply to the Digital Out +5V pins.

5.2.9. SERVOS - SERVO OUT CONNECTIONS (S1 TO S12)

Pin	Description
S	Servo PPM Signal (usually white or yellow wire)
V	+5V (usually red wire)
G	Ground (usually black or brown wire)

There are 12 servo connections, they provide PWM signals to drive your servos or speed controllers.

Servo PWM Output Channels are mapped to the Stealth Remotes as follows:

Servo	Description	Function
1	Remote 1 Joystick (Left/Right)	Drive Motor Speed Controller
2	Remote 1 Joystick (Up/Down)	Drive Motor Speed Controller
3	Remote 2 Joystick (Up/Down)	Spare
4	Remote 2 Joystick (Left/Right)	Dome Motor Speed Controller
5	Joystick 1 Thumbwheel 1 (Left)	Spare
6	Joystick 1 Thumbwheel 2 (Right)	Spare
7	Programmable *	
8	Programmable *	
9	Programmable *	
10	Programmable *	
11	Programmable *	
12	Programmable *	

* Servos S7 to S12 can be mapped to buttons or Thumb Gestures through setting parameters in the configuration file.

With the initial release, Servo Channels 1 and 2 are not mixed (Tank Mode.) Mixing should be done in the speed controller. See section 9.1 Speed Controllers for recommended and supported manufacturers.

5.2.10. DIGITAL OUT - DISCRETE DIGITAL OUTPUT CONNECTORS (D1 TO D8)

Pin	Description
S	Signal / Digital Output
V	Not connected (+5V if DPWR jumper installed)
G	Ground

These outputs provide 8 +5V digital discrete outputs to connect external devices, e.g. lights, LEDs, small relays or solenoid. These outputs can't drive large devices but should drive a circuit to drive power hungry relays etc.

They can be configured/mapped to buttons, thumb gestures through setting parameters in the configuration file.

5.2.11. I2C BUS

Pin	Description
V	+5V
G	Ground
SCL	i2c Clock Signal
SDA	I2c Data Signal

Used to connect Servo Expander Modules and other i2c devices.

5.2.12. PPM IN - PPM MODULE CONNECTOR

Pin	Description
1	Signal
2	+5V (from Servo Bus)
3	Ground

This is the connection for the optional PPM/PWM Converter Module.

5.2.13. XBEE - XBEE RADIO MODULE TTL SERIAL CONNECTION

Pin	Description
1	RX
2	TX
3	+5V
4	Ground

The XBEE Radio can either be plugged in directly to the XBEE socket, or to an external XBEE adapter via the 4 pin header. See section 5.3 XBEE Radio Module for more details.

5.2.14. SER - TTL SERIAL CONNECTION

Pin	Description
1	TX
2	RX
3	Ground

TTL serial connection for the debug console and Command Line Interface (CLI). TTL Levels.

5.2.15. AUX - AUXILIARY TTL SERIAL CONNECTION

Pin	Description
1	TX
2	RX
3	Ground

This connector is used to connect 3rd party devices. Not used at this time.

5.2.16. VMUSIC2 - MP3 MODULE CONNECTOR

Pin	Description
1	Ground
2	CTS - not connected
3	+5V
4	VMUSIC2 TX
5	VMUSIC 2 RX
6	RTS - connected to Ground
7	No Pin / Key
8	RI - not connected

5.2.17. RCSEL- RC MODE SELECT JUMPER/HEADER

Pin	Description
1	Ground
2	RC mode select

When enabled, control input is switched to run the system from your conventional RC remotes connected to the PPM Converter Module (**PPMIN**.)

It's recommended you use header/jumper wire with a panel mount toggle switch on one end and mount it somewhere convenient and easily accessible.

5.2.18. SEL2 - HEADER

Not used at this time

Pin	Description
1	Ground
2	N/A

5.2.19. STATUS - STATUS HEADER BREAKOUT

Pin	Description
1	Ground
2	J1
3	J2
4	RC
5	S1
6	S2

These pass thru the status LED signals to an optional LED module to mount elsewhere in your droid. See section 5.4 Status LEDs.

5.2.20. ISP- ICSP PROGRAMMING HEADER

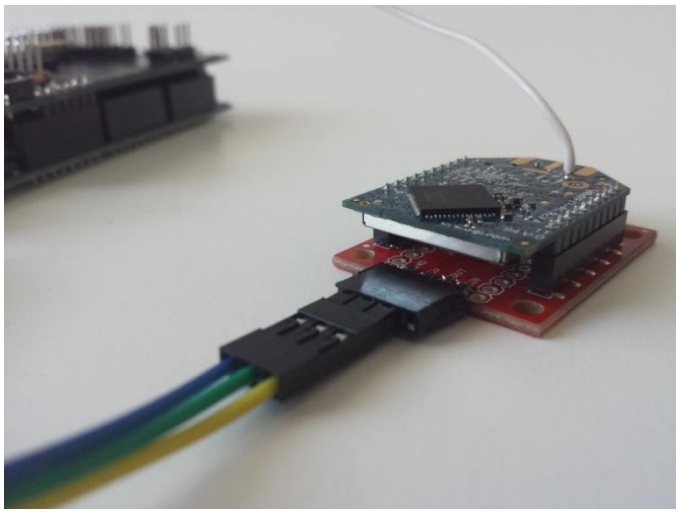
Pin	Description
1	MISO
2	+5V/VCC
3	SCK
4	MOSI
5	RESET
6	GND

AVR/ATMEL ICSP Programming Header. Pin one is marked on the board with a bar (top left.)

5.3. XBEE RADIO MODULE

5.3.1. CONNECTING

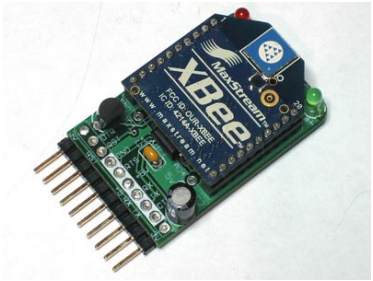
The *XBEE Radio* is normally plugged directly in to the Stealth Controller Receiver board, but it can be used in a 3rd party XBEE Adapter board connect via a cable allow placement of the antenna in a different location.



Typically XBEE Adapters have RX, TX, +5V and Ground pins at a minimum. In our setup only TX, +5V and Ground is used, so a simple 3-wire jumper cable could be used if you can't find a 4-pin jumper cable. Please be careful of orientation of the cable as to not connect the +5V to an incorrect pin.

5.3.2. EXAMPLE XBEE ADAPTERS

Adafruit



www.adafruit.com/products/126

Sparkfun

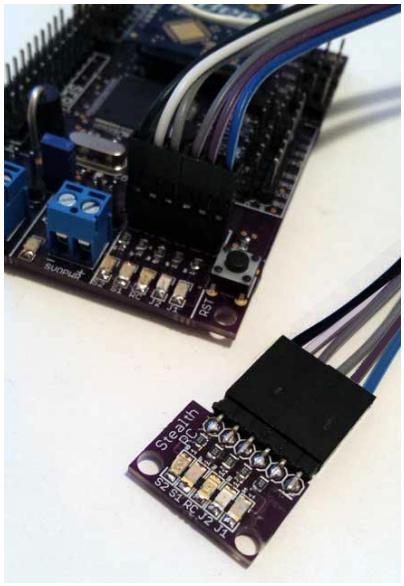
www.sparkfun.com/products/11373

5.4. STATUS LEDS

There are 5 on board status LED, 2 power LEDs and an XBEE TX LED.

Label	LED Color	Description
PWR	Red	Receiver Power - On when power applied to barrel jack or EXTPWR
SVOPWR	Red	Servo Power - On when power applied to SVOPWR
XB-RX	Red	XBEE Radio Module Data Receive Activity (RX)

Label	LED Color	Description
J1	Red	On when signal is being received from Remote 1
J2	Blue	On when signal is being received from Remote 2
RC	Yellow	RC Radio mode enabled (*RCSEL*): • On constantly when signal being received from PPM Converter Module. • Blinking means RC mode selected but no PPM signal detected/connected. Note: Failsafes for PPM Module are handled in your RC Receiver.
S1	Green	Main Receiver Status: • On constantly is normal after initialization/bootup. • Slow blinking means running okay, but didn't read configuration correctly from VMUISC2 (Running on parameters from EEPROM.) • Rapid blinking means we failed to load either CONFIG.XT or the EEPROM parameters. We're operating on a very basic configuration and at this point waiting for input from the Command Line Interface. Joysticks will not work in this mode unless you manually enter the XBEE addresses.
S2	Amber	Joysticks Enabled Status: • On when joysticks disabled (Dead Man's Switch) • Off when joysticks enabled



The 5 status LED can also be broken out on to an optional status board which you can located elsewhere in your robot for easy viewing.

If on startup you see LEDs *J1, J2, RC, S1* and *S2* briefly blinking in unison, then this means the Stealth Controller Receiver could not read CONFIG.TXT. Some reasons that may be the cause of the problem:

- VMUSIC2 Module not connected
- Missing or corrupt USB Thumb Drive
- CONFIG.TXT file missing, unreadable or corrupt
- +5V "brown outs" / power issues not allowing the VMUSIC2 to start up correctly
- Incorrectly configured multiple +5V supplies causing feedback loop/short

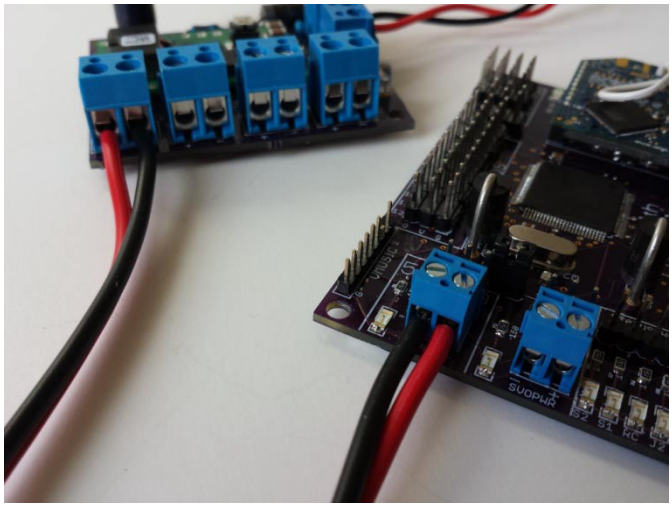
If the controller fails to read the CONFIG.TXT or the backup EEPROM parameters it will load a very minimal default configuration. Until XBEE addresses are configured the controller will not do much in this state. You can interact with it through the Command Line Interface on the Serial port (**SER**).

See section 5.6. Command Line Interface (CLI) on how to setup default parameters in EEPROM.

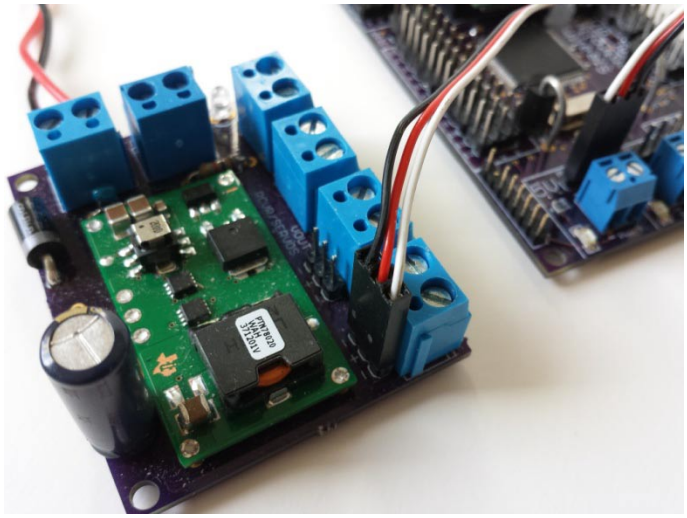
5.5. POWER CONSIDERATIONS

There are two power buses on the Stealth Controller Receiver:

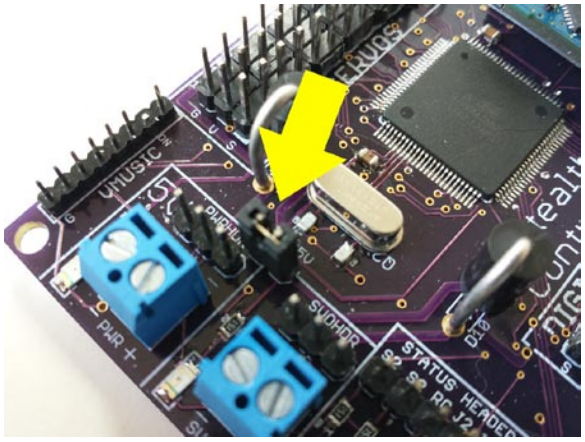
- The primary one, **PWR** (+5V), for the micro-controller, MP3 and XBEE Modules.
- A secondary bus, **SVOPWR**, to power Servos, Digital Outputs and the PPM Module/RC Receiver. This can either be +5V or +6V, and the connection is optional.



You can either connect your +5V supply using regular wire to the ***PWR*** terminal block (up to 16 AWG) - - a +5V CJ DC Converter is recommended (pictured.)



of using a 2 or 3 pin jumper wire to the ***PWRHDR*** header behind the ***PWR*** terminal block



If you plan on connecting servos (not speed controllers) and/or the PPM Module, then a +5V supply should be connected to **SVOPWR** (see section 8.)

OR this can be shared with **PWR** by using **JOIN5V** in most instances.

Most, if not all, Speed Controllers do not require power on the servo cables center wire to operate (normally colored red). In fact most Speed Controllers include a built in Battery Elimination Circuits (BEC) to power a conventional RC Receiver thru the red wire center wire.

You should **not** connect this +5V (red) wire to the servo bus. It could damage the Stealth Controller, any connected DC Converter or other Speed Controllers.

The PPM Module (and tethered RC receiver) is powered from the Servo +5V/+6V line (center pin). Please check if your RC Receiver can operate at +6V if you plan on powering this bus with a +6V supply.

It's tempting to try and use the Speed Controller's BEC to power servos and/or the PPM Module/RC Receiver, but it will probably not provide enough amperage and will could cause brown out issues.

Don't try and power the entire setup from a cheap power suppy, Speed Controller BEC or a simple "7805" regulator. The BEC or 7805 will probably overheat and cause brown out issues.

5.6. COMMAND LINE INTERFACE (CLI)

The Stealth Controller Receiver has a Console or Command Line Interface (CLI) via a TTL Serial port **SER** (TX, RX, GND) . It operate at 57,600 Baud, 8N1 and you will need a TTL level serial adapter/cable to access it.

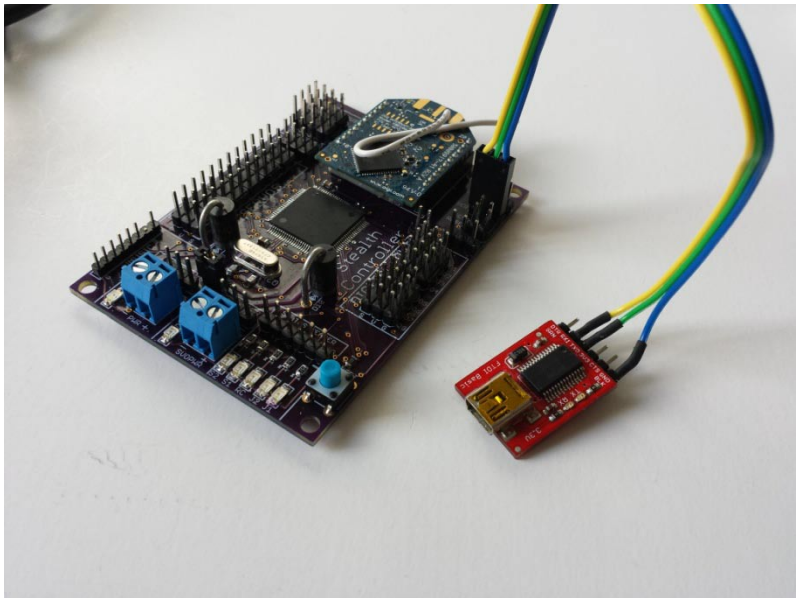
The CLI is used to interact with the controller to:

- View/set parameters which are used as backup when a CONFIG.TXT file can't be found/read
- View live servo digital outputs values
- Perform debug/troubleshooting

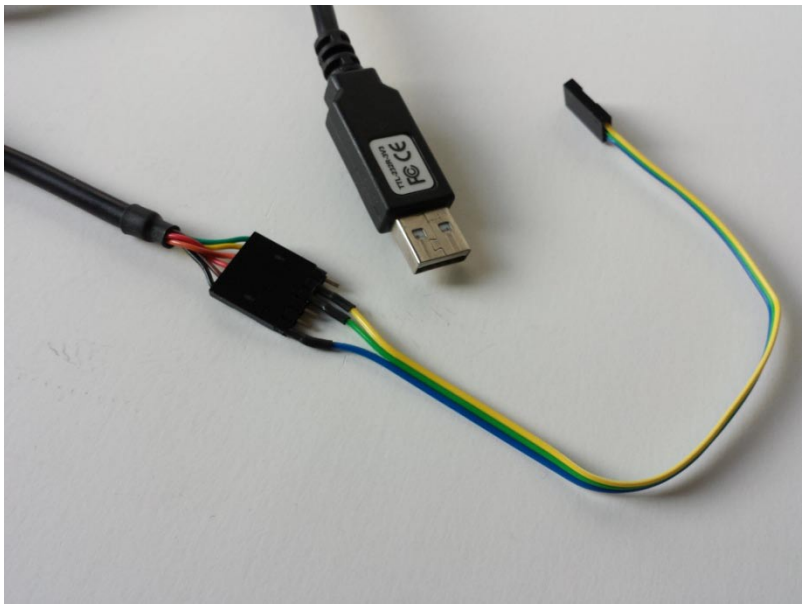
5.6.1. CONNECTING TO THE CONSOLE

Depending on your serial port/device, you may need to fashion a cable to break out the pins in the correct order.

If you plan on using the Servo I/O Expander Modules you will need one of these anyway to program the onboard Ardunio Pro Mini.



Here's an example using the Sparkfun USB FTDI Serial Adapter/Breakout (DEV-09716).



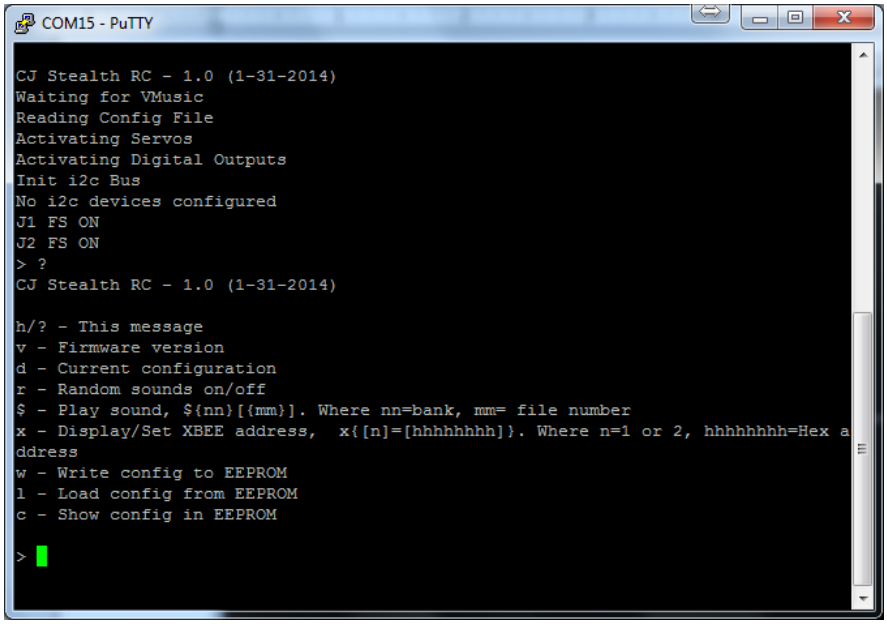
Or using the more traditional FTDI Cable

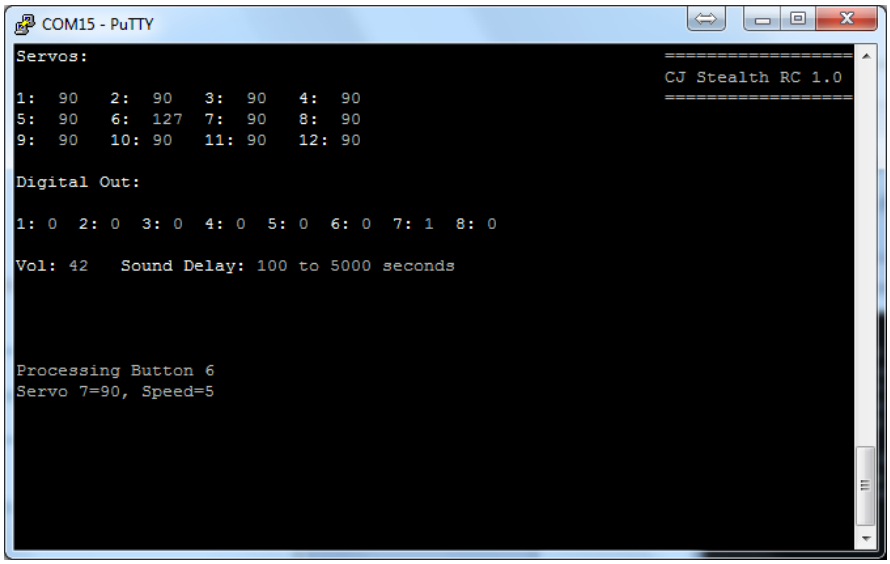
5.6.2. SERIAL TERMINAL SOFTWARE

There are many free terminal software application out there, and it's really a personal choice and will also depend on if you're on a PC or Mac. Putty (www.putty.org) is one example, it's a very old application but works well.

Some functions in the CLI use VT100 escape sequences to paint the screen, for example the live Servo/Output monitor.

5.6.3. EXAMPLE OUTPUT





5.6.4. COMMANDS

You should get a ">" prompt if you hit enter.

Command	Description
h/?	Display help message
v	Display version information
d	Display current running configuration
r	Turn random sounds on/off
$*nn*	Play Sound from Sound Bank *nn*
$*nnmm*	Play Sound *mm* from Sound Bank *nn*
x	Display XBEE radio addresses
x{[n]=[hhhhhhhh]}	Display/Set XBEE address. Where n=1 or 2, hhhhhhhh=Hex address
c	Show configuration information stored in EEPROM (don't load)
l	Load configuration setting from EEPROM (XBEE addresses and RC/Servo parameters only)
w	Write configuration setting to EEPROM (XBEE addresses and RC/Servo parameters only)
s	Display live Servo and Digital Out values - requires VT100 compatible terminal. Entering s again will turn the monitor screen off.
*****	Enter a configuration command as if it came from CONFIG.TXT Useful for temporarily flipping servo channel parameters like direction without changing CONFIG.TXT. Or configuring the Stealth Controller without a VMUSIC2 Module where parameters are saved/loaded from EEPROM. Not all CONFIG.TXT entries are appropriate. Valid entries are: "s", "maxpulse", "minpulse", "xbr", "xbl", "mindelay", "maxdelay", "rcchn", "rndon" For Examples: *s=1,0,180,89,3,-3,100,1 *xbr=409AB010 *maxdelay=240

5.7. CONFIGURATION FILE (CONFIG.TXT)

In normal mode the Stealth Controller Receiver is configured through the CONFIG.TXT file located in the top directory of the USB Thumb Drive.

It's made up of 5 sections, General Parameters, Sound Banks, Buttons, Thumb Gestures and Servo Channels

Due to memory constraints minimal validation is done while reading the configuration file. Parameters are case sensitive and are **all lower case**. Extra spaces are not removed or recognized and will most likely make the line invalid (basically don't use spaces.)

Comments are supported (line starting with #) in the file, but the more text you have the more time the controller will take to start up.

Max length of a line is 60 character.

5.7.1. GENERAL PARAMETERS

General for of the parameter is *parameter=value*

Parameter	Range/Valid Values	Example/ Typical	Description
volume	0-100	50	Initial volume. In most instances only the startup sound will play at this volume
startup	y/n	y	Play startup.mp3 when the system boots
rndon	y/n	y	Play random sounds, can be disabled/enabled after boot by using the *rnd* gesture
rnd	Gesture	3	Gesture assigned to turn random sounds on or off
ackon	y/n	n	Play acknowledgment sound (ack.mp3 in root directory) when triggering start of gesture. Useful when learning to use Thumb Gestures
ack	Gesture	222	Gesture assigned to turn acknowledge gesture sound on or off
mindelay	0-1000	60	Min delay before playing next random sound (seconds)
maxdelay	0-1000	120	Max delay before playing new random sound (seconds)
xbr	HEX	419998E4	Right XBEE's unique serial number (lower address in hex)
xbl	HEX	41995973	Left XBEE's unique serial number (lower address in hex)
mix12	y/n	n	Mix/Tank Mode. Mix servo channels 1 and 2 (not supported yet.)
minpulse		1000	Minimum pulse width for all servo outs (internal default 1000)
maxpulse		2000	Maximum pulse width for all servo outs (internal default 2000)
rcchn	6/8	6	How many channels does the RC radio have
rcd	1-50	30	RC Radio deadband (all channels) - this is in milliseconds
rcj	1-40	5	RC Radio jitter adjust - this is in milliseconds. Also reduces joystick accuracy/sensitivity.
myi2c	0-100	1	Stealth Controller Receiver's unique i2c address
auxbaud		9600	Baud Rate of the Auxiliary Serial Connection , typically 9600, 57600 etc.
mem	y/n	n	Write current thumb drive config to EEPROM on bootup.
auto	y/n	n	Auto correct common gesture. e.g. corners are hard to hit, so we autocorrect miss-gestures. Downside limits number of available gestures. Default=yes.

5.7.2. SOUND BANKS

You can configure up to 20 sound banks. Each sound bank is stored in its own(named) directory, and can have up to 100 files. If there's multiple files in the directory, then a random one is used. If there's a single file in the directory that file is used.

Playback is stopped when you trigger another sound or the file ends.

If random sounds are enabled (***rndon=y***), files from sound banks 1,2, and 3 are randomly used (favoring banks 1 and 2, typically groups of chatty and generic sounds.)

Delay between random sounds is governed by *mindelay* and *maxdelay* (in seconds). You can dynamically adjust the delay with Remote 2's right Thumb Wheel. This increases or decreases the maximum delay.

If a specific sound is triggered while random sounds are enabled, random sounds will be suspended until the specific sound is completed. Random sounds will then resume.

Sound banks have a unique number (1-20) and are sequentially assigned based on the order read in the configuration file. In the example below, "gen" is sound bank 1, "whistle" is sound bank 4.

Parameter	Option/Value Format	Examples
sb	[directory name],[number of files]	sb=gen,46 sb=vader,1 sb=scream,1 sb=whistle,3 sb=leia,1

5.7.3. BUTTONS

There are a total of 8 user definable buttons (1-8) that can be assigned to perform actions. Each action is of one of 4 *Types* and is the second parameter passed in the configuration line:

Type	Action	Format	Examples	Description
1	Sound	b=[button #],1, [sound bank#]	b=1,1,1	Button 1, Sound Action, Sound Bank 1
2	Servo	b=[button #],2,[servo #],[On Position]	b=2,2,7,180	Button 2, Servo Action, Servo #7, On Position of 180
3	Digital Out	b=[button #],3,[digital out #], [out-type]	b=6,3,1,0	Button 6, Digital Out Action, Digital Out #1, Normally Open
4	i2c Command	b=[button #],4,[target i2c address],[command]	b=8,4,99,1	Button 8, i2c Action, i2c target=99, command=1

5.7.4. THUMB GESTURES

Thumb Gesture configuration is very similar to buttons. Each one is allocated an action, and is of one four *Types*:

Type	Action	Format	Examples	Description
1	Sound	g=[gesture code],1,[Sound Bank #]	g=66,1,1	Gesture "66", Sound Action, Sound Bank 1
2	Servo	g=[gesture code],2,[servo #],[On Position]	g=2,2,7,180	Gesture "2", Servo Action, Servo 7, On Position of 180
3	Digital Out	g=[gesture code],3,[digital out #],[out-type]	g=4,3,1,2	Gesture "4", Digital Out Action, Digital Out 1, Momentary
4	i2c Command	g=[gesture code],4,[target i2c address],[command]	g=88,4,99,1	Gesture "88", i2c Action, i2c target=99, command=1

See section 4.3.4.3. Thumb Gestures Patterns.

5.7.5. COMMON NOTES ON GESTURES AND BUTTONS

Servos are always in the neutral/center position on startup (defined in the Servo section of the configuration file.)

Digital Outputs can be one of 3 types:

Output Type	Value
Normally Open	0
Normally Closed	1
Momentary	2

Digital Outs are always set to OFF initially on power on. It's the nature of the Micro-Controller and how it handles the output pins. We also have to first read the configuration file to determine what we want the outputs to do. If any are set to Normally Closed (NC) in the configuration, then they will be set to ON after reading the file. It's probably not a good idea to use the Normally Closed output type unless it's for something non-critical like a status LED.

Digital Outs and Servos objects can be shared between Buttons and Gestures. It's not a good idea to have different meanings or values for the same object. e.g. A digital out being defined as normally open for one buttons, but the gesture redefines it as a momentary output. Whichever one is defined last in the configuration file will override the value of the former, and the button will also trigger a momentary action.

Maximum addressable i2c devices is 5. Each device has to have a unique ID.

5.7.6. SERVOS AND CHANNELS

Parameters only apply to servos/channels when using Stealth Pocket Remotes. For RC Mode, radio channels 1, 2, 3 and 4 values are passed straight thru to the corresponding Stealth Receiver Servo Channels (S1-S4.)

In this section of the configuration file we can set the characteristics of our servos/channels

Structure	Examples	Description
s=[servo #],[min],[max],[neutral], [deadband],[trim],[speed],[reversed]	s=1,0,180,89,3,-3,100,0	Servo 1, min=0, max=180, Center/Neutral=89, Deadband=3 around center, Trim = -3, speed=fastest, not reversed
	s=2,10,170,90,30,10,1	Servo 2, min=10, max=170, Center/Neutral=90, Deadband=30 around center, Trim =0, speed=slow, direction reversed

Typically servos rotation is defined in degrees around a center point. From 0 to 180 degrees, with 89 or 90 being the center position, 0 being one extreme and 180 being the other. Physically your servo may only turn in either direction less than 90 degrees, but internally we still define the extreme positions (min and max) as 0 and 180.

Speed Controller behave in a similar way, with 0 to 180 defining the "speed" of the motor rather than position.

When using a mixed setup that includes a traditional RC Remote- Match your Servo Reverse and Trim Parameters to your RC radio settings.

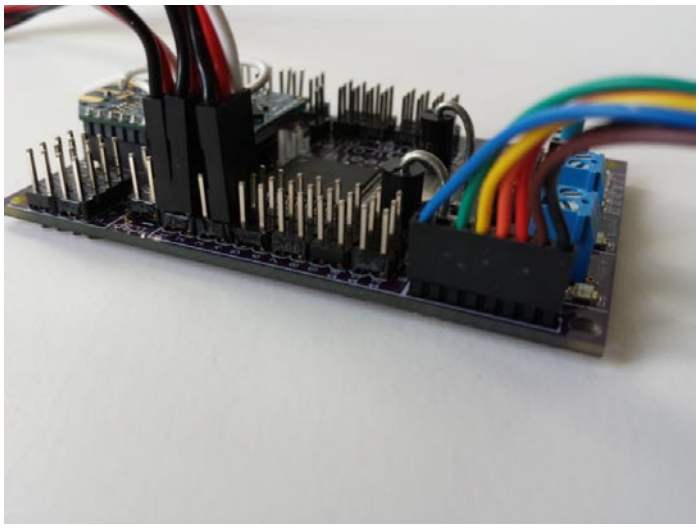
You can use these parameters to limit or change the characteristics of the servos and speed controller.

What	Range or Values	Description
min	0 to 88	Min position you want the servo to reach. Always has to be less than servo max.
max	90 to180	Max position you want the servo to reach. Always has to be greater than servo min. You can flip min and max, use reverse parameter for that.
neutral	0 to 180	Center or Neutral position of the servo
deadband	0-89	Number of degrees around center that will still register as center. Basically override the current joystick or thumbwheel value with the neutral position if it falls within our deadband. This can be used to reduce the sensitivity of the joysticks. Another use is to set a deadband on the side thumbwheels so you don't have to return it to dead center to return the servo to center.
trim	-90 to 90	Logically shift the center position of the servo/joystick either left or right. Typically this is only set to a few degrees in one direction. e.g. -1, -2, 1 or 2. None of the joysticks are built 100% perfect and what it considers center may not map to the logical "neutral" position.
speed	1 to 100	How fast the servo will move or accelerate, 1 = Slowest, 100=Fastest This parameter is useful if you're speed controller doesn't offer adjustable acceleration/breaking curves. A good value to start with for the drive speed controller would be 30-40
reversed	0,1	Flip the direction of the servo, 0=normal, 1=reversed

5.7.7. EXAMPLE CONFIG.TXT

```
#START
xbr=408C27B9
xbl=408C27B8
s=1,0,180,90,1,-2,30,0
s=2,0,180,90,1,-2,30,0
s=3,0,180,90,1,-1,40,0
s=4,0,180,90,1,0,40,0
s=5,0,180,90,10,0,50,0
s=6,0,180,90,10,0,50,0
s=7,0,180,89,0,0,100,0
s=8,0,180,89,0,0,100,0
volume=50
startup=y
rndon=y
ackon=n
mindelay=10
maxdelay=120
rcchn=6
myi2c=0
sb=gen,16
sb=chat,22
sb=sad,11
sb=rasb,1
sb=whis,12
sb=scream,4
sb=warn,2
sb=short,1
sb=leia,1
sb=vader,1
sb=sw,1
sb=dance,2
sb=cant,1
b=1,1,3
b=2,1,4
b=3,1,7
b=4,1,8
b=5,1,2
b=6,1,5
b=7,1,6
b=8,1,2
rnd=3
ack=252
g=5,1,1
g=6,1,10
g=4,1,9
#END
```

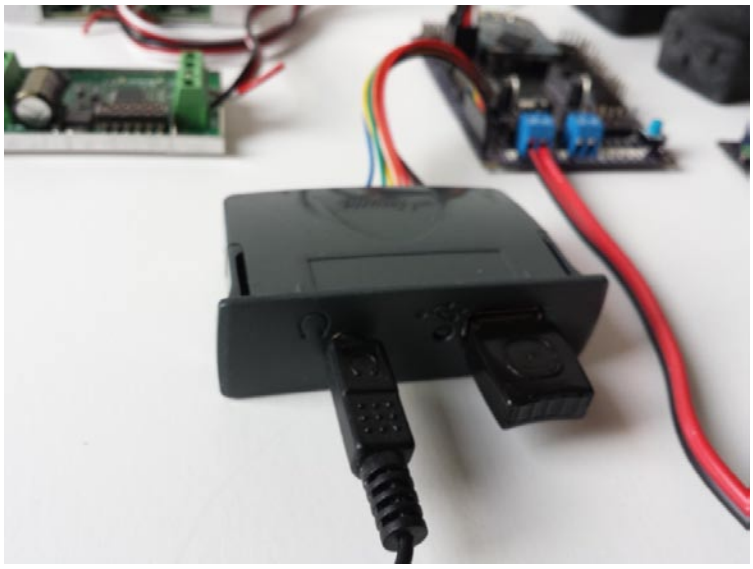
5.8. MP3 MODULE (VMUSIC2)



Connect the VMUSIC2 module using the 7 pin multi-colored jumper wire. The plug is keyed and will only go one way.

On power-up the green LED on the VMUSIC2 module will blink as it reads the configuration file, and when it plays sounds.

5.8.1. LINE OUT / AUDIO SETUP



Connect your amplifier to the VMUSIC2 Module via the 3.5mm jack. A Ground Loop Isolator is highly recommended. For testing you could also use earphones.

Note: Depending on the amplifier you use, it's recommended that a **Ground Loop Isolator** is installed between the VMUSIC2 module and the amplifier to eliminate any feedback or possible damage to the VMUSIC2 module.

5.8.2. USB THUMB DRIVE

A standard USB thumb drive can be used. Format as FAT32.

5.8.3. AUDIO FILES AND FORMAT

All sound files should be encoded as MP3.

See VMUSIC2 Reference Guide for details

http://www.ftdichip.com/Products/Modules/ApplicationModules.htm

5.9. INCOMING I2C SOUND COMMANDS

The receiver recognizes incoming sounds commands on the i2c bus. They take the form of 3 bytes.

You can assign/change the receivers unique i2c address with the *myi2c* parameter in the configuration file. See 5.7.1. General Parameters.

Byte	Description	Example
1	Play Command - P (80 Decimal)	P
2	Sound Bank	1 (sound bank 1)
3	Sound Number	3 (sound 3)

If *Sound Number* is zero, then a random sound is played from the Sound Bank.

6. SERVO AND IO EXPANDER

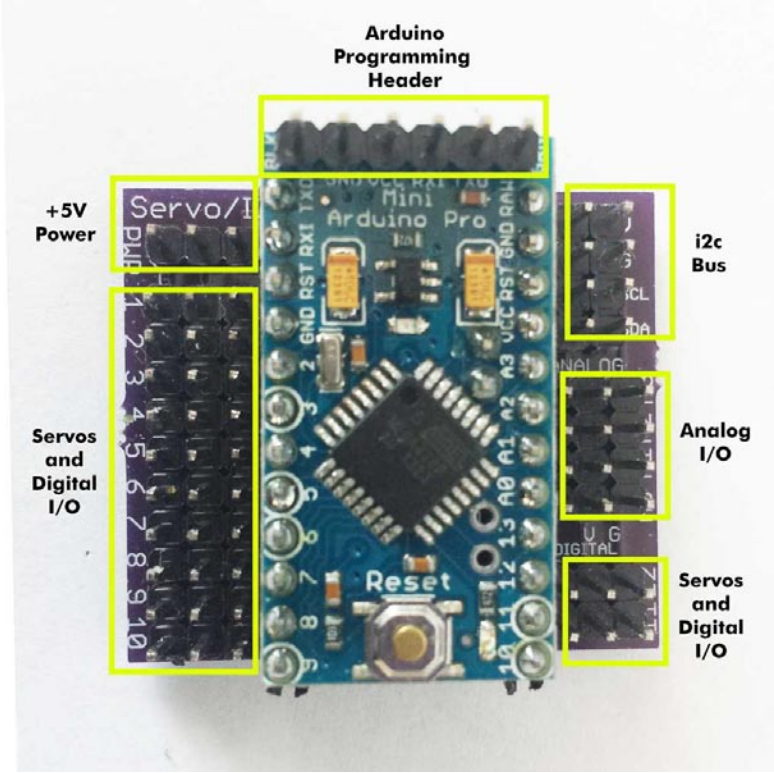
6.1. OVERVIEW

The Servo I/O Expander is basically an Arduino Pro Mini (ATMEGA 328) module with the addition of header pins to easily connect and power servos and other IO devices. It also breaks out the i2c bus to communicate with the main Stealth Controller or other i2c devices.

Connections:

- 12 Servos/Digital IO pins 1-12 (maps to Arduino DIO 2-13)
- 4 Analog IO pins (maps to Arduino A0-A3)
- i2c header/bus (2 connections, each with V,G, SCL and SDA)
- PWR
- Arduino Programming Header (using FTDI Serial cable/adaptor)

6.2. PIN OUTS



6.2.1. PWR

Input power for the Servos and also to the Arduino Pro Mini's RAW connection, which in turn connects to the Arduino's onboard 5V DC Converter. The Arduino can also be powered by the i2c V connection.

Pin	Description
S	Not connected
V	+5V (usually red wire)
G	Ground (usually black or brown wire)

6.2.2. SERVOS - S1-12 (OR DIGITAL I/O)

Pin	Description
S	Digitial or Servo PPM Signal (usually white or yellow wire)
V	+5V (usually red wire)
G	Ground (usually black or brown wire)

These pins map to the Arduino's Digital Pins D2 thru D13. Note that D13 on the Arduino Pro Mini has an LED and resistor attached and is generally used as a status pin/indicator.

6.2.3. ANALOG - A0 - A3

Pin	Description
S	Analog I/O
V	+5V (usually red wire)
G	Ground (usually black or brown wire)

These pins map to the Arduino's Analog Pins A0 thru A3.

6.2.4. I2C BUS

Pin	Description
V	+5V / VCC
G	Ground
SCL	i2c Clock Signal
SDA	I2c Data Signal

Used to connect to the main Stealth Controller Receiver or other Servo Expander Modules.

The Arduino can also be powered by the V in connection and is connect to VCC of the Arduino Pro Mini Board and should not exceed +5V.

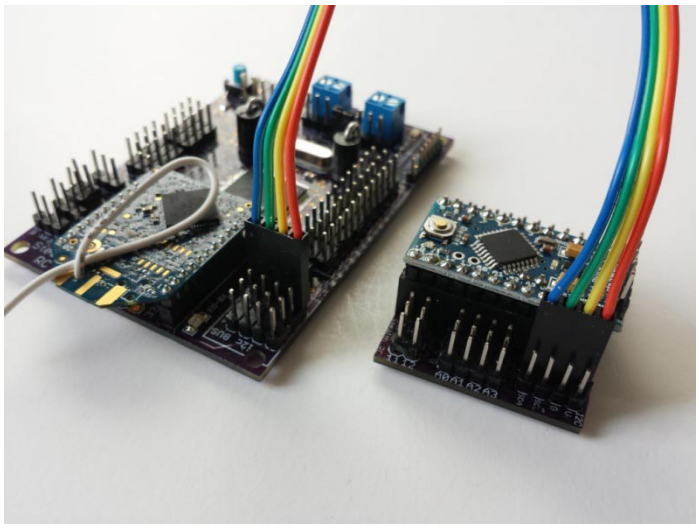
6.2.5. JOINV

Pin	Description
VCC	Internal +5V of the Arduino and i2c bus.
VIN	External +V power to the servos via PWR Header, which also connects to the Arduino's DC Converter.

In most cases this connection is not used, but in rare cases you may want to power the Servos from the i2c +5V supply. This jumper connects the internal Arduino +5V Vcc to the VIN/Servo Power line.

Use with caution.

6.3. I2C CONNECTION

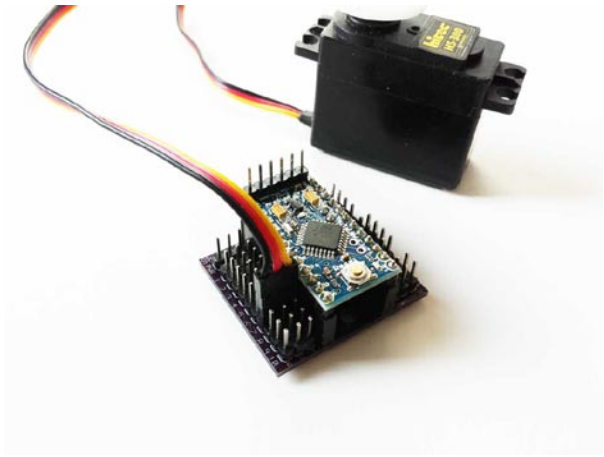


Connecting the Expander to the main Stealth Controller's i2c bus

When implementing a more complex setup using the Servo Expanders (or other Arduino/i2c devices) it's recommend you initially monitor the Stealth Controller Receivers serial output to make sure it can see the i2c devices.

In rare instances the Stealth Controller can hang if you missing configure the i2c bus. Try disconnecting all i2c devices if on bootup all the status LED stay on indefinitely.

6.4. SERVO CONNECTIONS



Inner most pin is the servo signal

6.5. PROGRAMMING AND COMMUNICATION

In the Stealth RC environment communication to the Servo Module is via i2c. Each device on the i2c bus is given a unique address. By default the Stealth Controller Receiver is '0' but can be changed using the myi2c parameter in the configuration file. Each Servo IO Module is also given a unique address.

The Servo Module can be programmed through the normal Arduino Integrated development environment (IDE.)

It's beyond the scope of this guide to fully explain how to program Arduino's and how i2c works. Please see sample code, or go to the main Arduino website for tutorials.

http://arduino.cc/en/Guide/ArduinoProMini

There's also a good book by O'Reilly, Getting Started with Arduino by Massimo Banzi.

6.6. EXAMPLE CODE

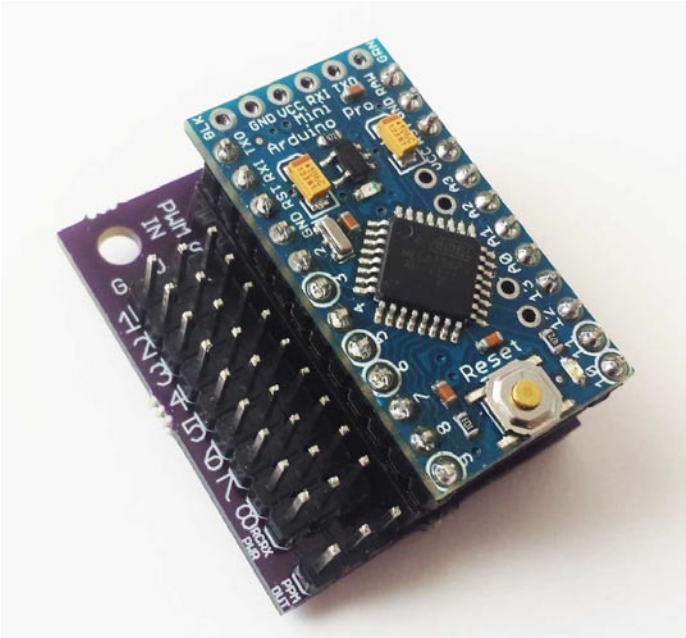
More example code will be posted to the support page, but here's a minimal outline to get you started.

6.6.1. MINIMAL EXAMPLE

```
//---------------------------------------------------------------
// Generic Servo Expander

#include <Servo.h>
#include <Wire.h>

// My i2c Address
#define MYADDRESS 3
int i2cCommand = 0;

// Status
#define STATUS_LED 13 // Arduino built in LED

// Servo Bits
#define NBR_SERVOS 2
#define FIRST_SERVO_PIN 2
Servo Servos[NBR_SERVOS];
#define NEUTRAL 90

//---------------------------------------------------------------
void setup() {
  Serial.begin(57600); // DEBUG
  Serial.println("CJ Stealth RC Servo Expander 1.1 - 1.31.14");

  pinMode(STATUS_LED, OUTPUT); // Enable Status LED Pin
  digitalWrite(STATUS_LED, LOW); // Turn it off

  Serial.print("Activating Servos");
  // Attach and center Servos
  for(int i =0; i <  NBR_SERVOS; i++) {
    Serial.print(".");
    Servos[i].attach(FIRST_SERVO_PIN +i);
    Serial.print(".");
    Servos[i].write(NEUTRAL);
  }
  delay(500); // Wait a bit to make sure they're all centered
```

```
  // Detach from servo to safe power and stop jitter
  for(int i =0; i <  NBR_SERVOS; i++) {
    Serial.print(".");
    Servos[i].detach();
  }
  Serial.println("");

  Serial.print("My i2c address:");
  Serial.println(MYADDRESS);

  Wire.begin(MYADDRESS); // Start I2C Bus as a Slave
  Wire.onReceive(receiveEvent); // routine we call when we get a commandregister event

  i2cCommand = -1; // Make sure i2cCommand isn't set
  Serial.println("Listening for i2c Command");
}

//-------------------------------------------------------------
// receive Event
// this is called every time we detect an i2c command for us
//-------------------------------------------------------------
void receiveEvent(int howMany) {
  i2cCommand = Wire.read();    // receive byte as an integer
  Serial.print("i2c Command = ");
  Serial.println(i2cCommand);
}

//-------------------------------------------------------------
// Perform this loop indefinitely
//-------------------------------------------------------------
void loop() {

  if (i2cCommand == 0) {
    digitalWrite(STATUS_LED, LOW);
    i2cCommand=-1;
  }

  //-------------------------------------------------------------
  // Status / Reset from Main Controller
  if (i2cCommand == 1) {
  // "1" is a special startup Stealth RC command.
  // It tells us the main Controller Receiver reset.
  // Typically we've just rebooted/reset too.
  // Status LED is set on to show we got the reset message

    Serial.println("Got reset message");

    // Turn on Status LED so we can visually see that we got a reset command
    digitalWrite(STATUS_LED, HIGH);

    // Always reset i2cCommand to -1 or command will loop/repeat forever
    i2cCommand=-1;
  }

  //-------------------------------------------------------------
  // Command 1
  if (i2cCommand==2) {
    Serial.println("COMMAND 2!");

    // Turn on Status LED to show we're in the command
    digitalWrite(STATUS_LED, HIGH);

    // Always reset i2cCommand to -1 or command will loop/repeat forever
    i2cCommand=-1;

    // Turn off Status LED
    digitalWrite(STATUS_LED, LOW);

  }
}
```

7. PPM/PWM CONVERTER



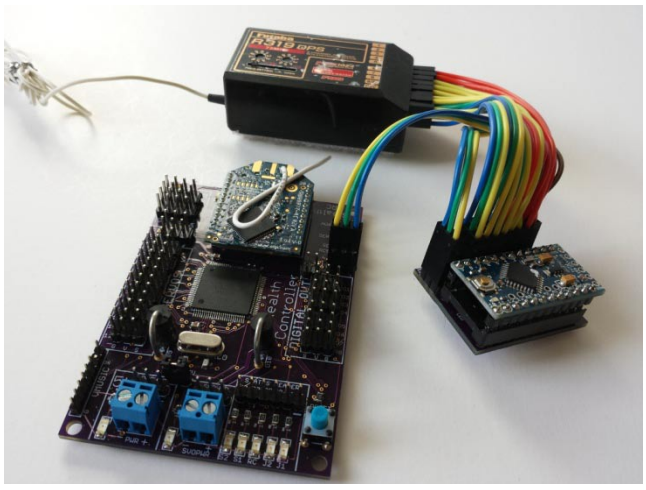
The **PPM/PWM Converter** allows you to use a standard RC Receiver (RX) as a backup control device. A minimum of 4 channels are required for normal operation, but 6 or 8 are typical. 8 being the maximum.

As a bonus you can perform gestures and trigger sounds via your RC Transmitter.

The number radio channels should be configured in CONFIG.TXT using the ***rcchn*** parameter (*rcchn=6, or rcchn=8*). See 5.7.1. General Parameters.

(It may look like it's an Arduino Pro Mini, but it's running custom firmware to interpret the PWM signals and map them to a single PPM output stream.)

7.1. CONNECTING TO A RC RECEIVER



In summary, you connect the RC Receivers 6 or 8 channels using the short 3-pin jumper cables.

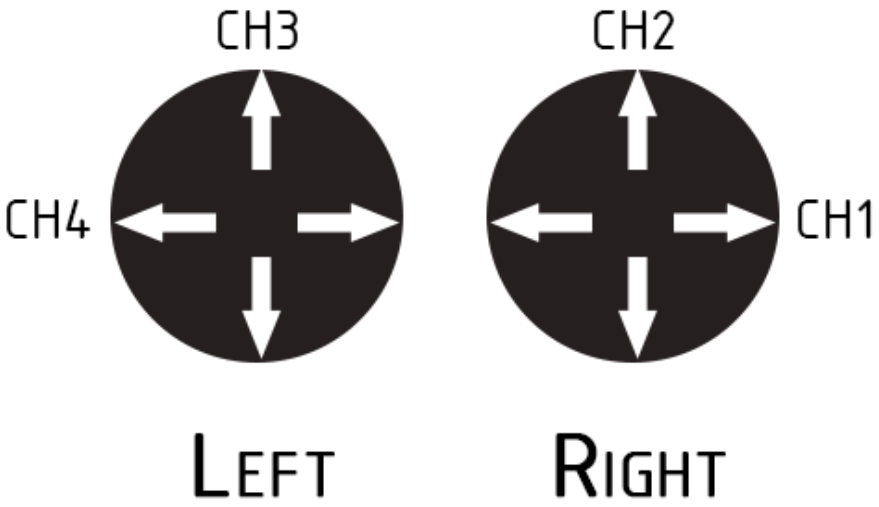
Then use a longer 3-pin jumper cable to connect the **PPMOUT** to **PPMIN** on the Stealth Controller.

Depending on your receiver, you may also need to connect **RCRXPWR** from the PPM Convert to your RC receivers Battery Power connection.

You can also reduce the cables needed by just connecting the data/signal pins instead of repeating the ground/voltage lines for each channel. But you will still need at least one ground/voltage connection.

CAUCTION: Not all RC Receivers outputs are labelled the same nor will they necessarily be in the same order on the RC Receiver.

Before we connect anything we need to first understand how the **Stealth Remotes** are mapped to the **Stealth Controller Receivers** servo channel outputs. They follow the standard RC convention, which may not seem logical, but that's the standard



.

Futaba typically labels their RC Receiver channels numerically, and JR/Spektrum uses the terms Aileron, Elevator, Rudder, Throttle, Gear and Auxiliary. To confuse matters, RC transmitters can come in "Mode 1" and "Mode 2". Most RC airplane remotes in the USA are Mode 2, and that's what this guide follows.

It's very important that all joysticks/channels (both conventional RC and Stealth RC) logically map the same way, otherwise your robot may respond differently depending on which remote you use. This includes any reverse setting you may have configure in the RC transmitter. This need to be duplicated in the servo parameters in CONFIG.TXT.

Stealth RC PPM #	Futaba Mode 2		JK/Spektrum Mode 2	
	Label	Function	Label/Function	Position
1	1	Aileron	Aileron	2
2	2	Elevator	Elevator	3
3	3	Throttle	Throttle	1
4	4	Rudder	Rudder	4
5	5	Gear	Gear *	5
6	6	-	Aux 1	6
7	7	-	Aux 2 *	7
8	8	-	Aux 3*	8

* label may vary on different receiver model

Because we're intercepting the RC signals we can remap them to perform Stealth RC operations.

Depending on the number of channels you have on your RC radio , you can assign channels 5-8 to either switches, knobs or dials - which can in turn map to volume, sound delay and to trigger sounds or start gestures.

6 Channel Radios:

- Assign Channels 5 and 6 to ***on-off*** toggle switches. On the Stealth side we map these to ***Remote 1 - Buttons 5 and Remote 2 Button 10***.This allows us to trigger random sounds and enter gestures.
- Unfortunately, we don't have enough channels for volume.

This configuration has been tested with a Futaba 6EX.

8 Channel Radios:

Most 8 channel radios have a wide range of dials and switches, so we have some flexibility.

- Assign Channels 6 and 7 to a knob or dial. These will map to volume and random sound delay(***Remote 2 - Thumb Wheels 3 and 4***)
- Assign Channel 8 to an ***on-off-on*** toggle switch. ***Up*** maps to Gesture/Select (***Remote 2 - Buttons 10***), ***Down*** maps to ***Remote 1 - Button 5***.
- Channel 5 is a pass thru channel, useful for controlling a direct attached servo.

This configuration has been tested on a Futaba 10G.

If you're 8 Channel RC radio does not have dials or an on-off-on toggle switch you may need to stick with just using 6 channels.

Most droids are driven in "tank mode" where channels 1 and 2 are mixed. This should be done in the Speed Controller and not in the RC radio.

Failsafes in RC Mode are handled in your RC Receiver.

7.2. PIN OUTS

Pins	Description
PWM 1-8	Connect to the RC Receiver outputs (G, V, Signal)
RCRXPWR	Connects to the RC Receiver Battery connection. **Do not accidentally power the RC Receiver from two different power sources at the same time.**
PPMOUT	Connects to the Stealth Controller Receiver PPMIN. Passes PPM signal to Controller and also powers the PPM Converter from the Controller.

See section **Error! Reference source not found.**. **Error! Reference source not found.** on how to recommendations how to configuration the radio.

7.3. STATUS LED

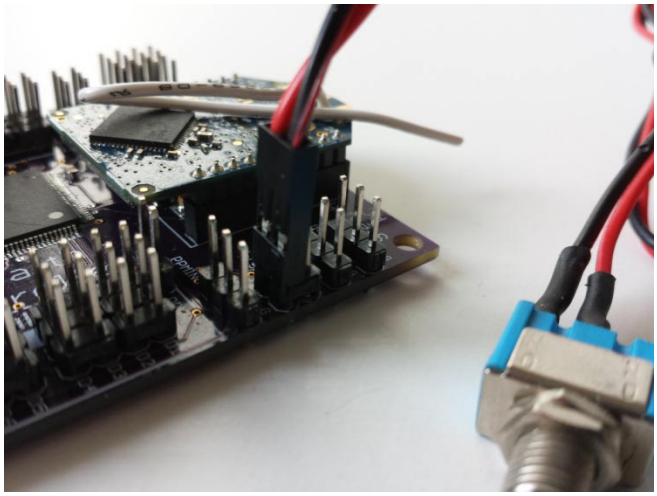
The PPM Converter has two LEDs, one is power and the second (green) is RC signal status.

Solid green means there is no valid RC signal detected (checked via channel 1). When no signal is detected the PPM Converter outputs neutral/centered values for all 8 channels.

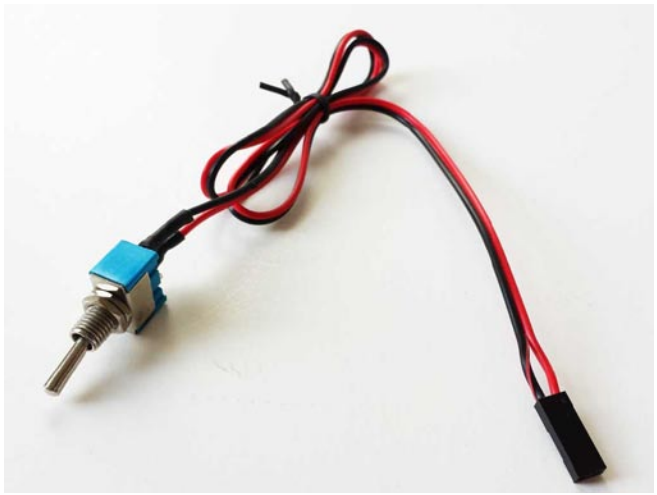
Blink means we have a good signal (checked via channel 1) and all connected channels are being output via the PPM connection.

Remember failesafes are managed in the RC Transmitter/Receiver.

7.4. RC SELECT SWITCH



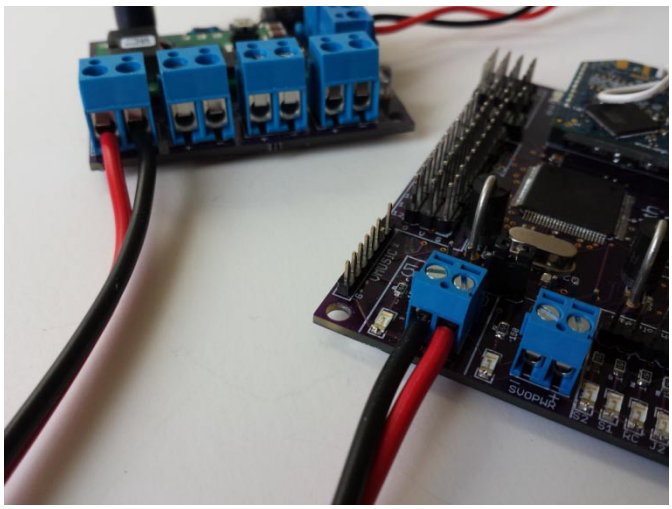
RC Mode is selected by enabling via the ***RCSEL*** header/jumper to use your conventional RC setup.



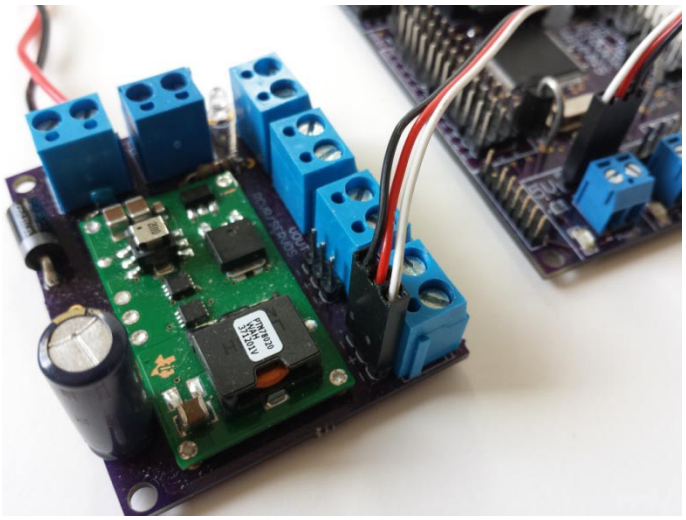
A simple two wire cable with a header plug on one end and a on/off toggle on the other is easily made up and mounted in a convenient location.

8. DC/DC CONVERTER

The Stealth RC setup requires a good +5V supply to operate. The CJ DC Converter is a highly efficient voltage regulator. And designed to power all the devices in your Stealth RC setup, including the main Stealth Controller Receiver, Servo Expander Modules and a large number of servos.



You can either connect your +5V supply using regular wire to the ***PWR*** terminal block (up to 16 AWG)



or using a 2 or 3 pin jumper wire to the ***PWRHDR*** header behind the ***PWR*** terminal block

9. ADDITIONAL COMPONENTS

9.1. SPEED CONTROLLERS

Stealth RC has been tested with the following Speed Controllers:

- Dimension Engineering Syren 10 (Dome)
- Dimension Engineering Sabertooth 2x25 (Drive, mixed/tank mode)
- RoboteQ AX2500 (Drive, mixed/tank mode)
- RoboteQ AX2250 (Drive, mixed/tank mode)

Dimension Engineering speed controllers need to be on a firmware release that supports DEScribe. This will allow you to fine tune power curves and set mixed/tank mode operation.

http://www.dimensionengineering.com/software/DEScribe/html/

Other speed controllers, for example from Vantec should work as they offer tank driving and adjustable power curves. But we've not tested them as of yet.

9.2. AMPLIFIERS

You will need an amplifier and speaker setup to use the MP3 Module. It's beyond the scope of this document to make recommendations beyond the caveat that some amplifiers may need a ground loop isolator to minimize interference and possible damage to the VMUSIC2 MP3 Module.

10. TROUBLESHOOTING/FAQ

10.1. STEALTH POCKET REMOTES

10.1.1. CAN I HOT-SWAP BETWEEN POCKET REMOTES AND RC MODE?

No, you have to power down. Flip your RC Select switch and then power back on.

10.1.2. I LOST THE SCREW / BUTTONS AFTER I OPENED UP THE CASE - WHERE CAN I GET MORE?

Please be careful not to lose the small buttons or screws when you open up the case. The remotes were not really designed to be opened in the field and the small parts are easily lost.

The small screw that holds the case together is a #0 18-8 1/4" Flat Head Sheet Metal Screw . McMaster-Carr part number 90065A020.

Spare buttons will be available to purchase soon.

10.1.3. WHAT BATTERY IS USED IN THE REMOTE?

The remotes use a small 3.7V 400mAh Li-Po batteries. See section 4.4.4 Replacing internal Batteries.

10.1.4. HOW LONG WILL BE REMOTES RUN FOR ON A FULL CHARGE?

Depending on the XBEE radios installed and condition/age of the battery you should get around 4 hours of use on 900MHz radios and 6 hours on 2.4GHz radios.

10.1.5. WHAT'S THE LIFE EXPECTANCY OF A BATTERY?

It really depends on how well you look after it and how often you use/charge it. Over time the charge the battery can hold will diminish and you'll get shorter and shorter run times between needing to recharge.

10.1.6. HOW LONG DOES IT TAKE TO CHARGE THE BATTERY?

From "empty", allow 2-3 hours to charge the battery fully - but time will vary depending on usage, age of battery and the amperage available from the USB power source.

10.1.7. THE BATTERY IS DEAD AND I CAN'T TELL IF THE REMOTE IS TURNED OFF OR ON

Looking at the back of the remote, the remote is turned on with the switch all the way to the left or off when it's to the right.

Or look for the little notch underneath that signifies the on position

10.1.8. THE YELLOW CHARGE LED IS BLINKING WHAT DOES IT MEAN?

When charging, the yellow status LED should be on constantly, and will turn off when done.

If it's blinking, it probably means the battery isn't hold a charge and the built in charger is cycling between detecting a full charge and needing to charge state.

You could try charging the battery on an external Li-Po charge but it's probably time to replace your battery.

10.1.9. CAN I USE THE REMOTES WHILE CHARGING?

You can use the remotes while charging, and it's handy to have an external booster battery to run your remotes when they're running low.

But there's a catch. First the remotes have to be charge to a certain level before they will turn on - this should only take a few minutes.

Second, once the battery is full, the built in charger turns its self off. It will not recheck the status of the battery until you unplug and reconnect the USB cable. So if you continue to use the remotes tethered to an external battery don't assume it will continue to charge the battery.

10.1.10. THE BATTERY IS SWOLLEN OR LEAKING - WHAT SHOULD I DO?

You should not attempt to charge the battery. Disconnect it immediately and dispose of correctly.

10.1.11. CAN I REPLACE THE XBEE RADIOS?

Yes, but you will need to reconfigure the radios and set the unique XBEE addresses in the CONFIG.TXT file. Documentation on how to do this will come later.

10.1.12. CAN I SWITCH MY XBEE RADIOS FROM 2.4GHZ TO 900MHZ OR VICE-VERSA?

Yes, but you will need to replace the radios. They're fixed at either 900MHz or 2.4GHz. As above you will need to reconfigure the replacement radios. Plus you may need to adjust some solder bridges in the remotes. Documentation on how to do this will come later.

10.1.13. THE J1 AND J2 LEDS KEEP ON FLASHING ON AND OFF

You may want to check your remote batteries. If power is too low to maintain a connection they will fail to sync with the receiver and eventually give up.

10.2. STEALTH CONTROLLER RECEIVER

10.2.1. STARTUP HANGS WHEN I HAVE AN SERVO EXPANDER ATTACHED

Make sure you don't have conflicting i2c addresses set. Best to assign the Servo Expander(s) and address above 5 to be on the safe side.

10.2.2. WHEN WILL THE J.E.D.I. CONTROLLER BE SUPPORTED?

The J.E.D.I. controller is scheduled to get a software upgrade to version 4.X and the target is to try and make the Stealth Controller compatible with that release, but there's no guarantee.

10.2.3. THE THUMBWHEELS ARE STIFF/HARD TO TURN

Over time they will loosen up and you can accelerate this by simply playing with them. It may be easier to do this with the top of the case removed.

If you have big thumbs, you can also try filing down the opening in the case - but the reason they're recessed so much is to avoid accidentally turning them.

10.3. DRIVE SYSTEM

10.3.1. MY DROID IS VERY ERRATIC WHEN I TRY AND I CAN'T CONTROL HIM

The joysticks are very sensitive and they'll take getting used to, and you will need to fine tune the "ramp time" or speed/acceleration power curves configured on your speed controller.

Alternatively, the Stealth Controller offers rudimentary servo speed/acceleration capability that can be changed in the configuration file (see section 5.7.6 Servos and Channels.)

However, it's highly recommended that you try to first tune your speed controller as they have a wider range of options.

This isn't too hard to do, but beyond the scope of this document to outline the steps in detail.

The Stealth RC system has been tested on RoboteQ and Dimension Engineering (DE) speed controllers, both of which have a serial/PC interface to allow adjustment of these parameters.

Only DE controllers with DEScribe enabled firmware support parameter tuning. If you bought your DE controller prior to 2013 there's a good chance you will need to upgrade your firmware. DE will do this for free if you send it to them after asking for an RMA via their website (http://www.dimensionengineering.com)

For details on DEScribe:

http://www.dimensionengineering.com/software/DEScribe/html

Vantec speed controllers also offer different speed/power curves, which can be set using DIP switches. At this time this has not speed fully tested.

10.3.2. MY DROID WILL NOT GO FORWARD AND TURNS IN STRANGE DIRECTIONS

Please check your wires to make sure that you've not connect channel 1 and 2 to the wrong speed controller channel. You can also reverse direction of the motor or motors by flipping the wires that connect to the motor from the speed controller. Be very carefully not to confuse this with the input power to your speed controller.

Direction can also be change by reversing the direction of one or both of servo channels 1 and 2. See section 5.7.6 Servos and Channels.

If you're also using your convention RC remote, you should match the Stealth RC forward and reverse options with your RC remote.

At this time the controller does not support channel mixing or tank mode. This has to be set in your speed controller.

10.3.3. MY DOME SPINS THE WRONG WAY

Either reverse the polarity of the wires connecting to the motor or configure the servo channel to be reversed.

10.4. SERVO I/O EXPANDER

10.4.1. I HAVE A SERVO EXPANDER LOADED WITH THE EXAMPLE CODE BUT IT'S NOT RESPONDING TO I2C COMMANDS

Using the "basic" example code, on power up the Servo Expander should turn on its green status LED when the Stealth Controller Receiver initializes.

Things to check:

- Do you have power going to the servo bus?
- Check it has a unique i2c address on your network.
- Try disconnect all other i2c devices and try again.
- Make sure you have at least one valid i2c gesture or button configuration command in CONFIG.TXT, and the i2c addresses matches the one assigned in your Servo Expander code.

10.4.2. CAN THE SERVO EXPANDER RANDOMLY MOVE A DOME HP?

Short answer is yes. Sample code will be release soon.

10.4.3. CAN I CONNECT MY JEDI HP LED BOARDS DIRECTLY TO THE STEALTH SYSTEM?

Yes in principle you could connect them to one or more of the Digital Out connections. It's best to drive them from the Stealth Servo I/O Expander rather than the Stealth Controller Receiver as you'd save on running additional connections up to the dome.

C**aution:** The center pin on JEDI HP LED board connect to GND and should be disconnected/cut/disabled. The center pin on all the Stealth I/Os outputs are +5V. Connecting these without disabling the center wire will most likely cause a short and damage components in the system.

10.4.4. WHAT ABOUT JEDI DISPLAY, CAN I TRIGGER THEM FROM THE STEALTH RC SETUP?

Again, yes in principle. You could connect the J.E.D.I. Display Controller to a serial out connection on the Servo I/O Expander Module, and send J.E.D.I. command which could be triggered by an i2c command via a gesture or button.

Example code and wiring diagram is in the works.

10.4.5. CAN I CONTROL A TEECES LOGIC DISPLAY FROM THE STEALTH RC SETUP?

The best way to do this is to load the J.E.D.I. compatibility Teeces firmware and use a serial connection and J.E.D.I. commands as above.

10.5. SOUND SYSTEM

10.5.1. THERE'S STATIC COMING FROM MY SPEAKERS

Do you have ground loop isolator between the amplifier's and VMUSIC2? If not, install one.

10.5.2. THERE'S A CLICKING COMING FROM SPEAKERS

If you turn of the radios and/or remote the XBEE radio does the noise still happen?

Some amplifiers are sensitive to the 2.4GHz radios. You will either need to move the XBEE radio as far away from the amplifier as possible or replace the amplifier.

If the noise is still there with the radios disconnected/off - then try a test with ear buds/headphones in place of the amplifier. Is the noise there too? If not you may have a ground loop problem.

11. APPENDIX

11.1. FIRMWARE

11.1.1. REVISIONS

0.15 (10-24-2013) - Limited BETA Release
1.0 (1-31-2014) - Expanded BETA Release

11.1.2. FIRMWARE UPGRADE PROCEDURE

To be added

12. REFERENCES

http://www.arduino.cc - good site to get a broader understand of the Arduino programming model used in Servo Expander Module.