Hacking the Copter

Dominic Spill & Michael Ossmann

Great Scott Gadgets
3D ROTATION

ITEM NO. CX-10A SIZE OF PRODUCT 4*4*2.2 CM
SIZE OF COLOR BOX 7.5*10.7*9.5
SIZE OF CARTON 64*24.5*84
N.W./G.W. 19/15KGS

FEATURES 4-AXIS RC QUADCOPTER WITH 6-AXIS GYRO ROLLING FUNCTION

2.4GHZ INFRARED TECHNOLOGY
Starts on 2402 MHz
1 Mbps GFSK
~500 ms packets every 6 ms
710f552f7d872649 e9 e290a704 7377ed96 d5b f fcc 2 801 5 30d 9 cacc 9649 50

---------- ----------- ----------- ----------- ----------- ----------- -----------
sync word  ph  CID  VID  rol  pit  thr  yaw  CRC  trailer

8 bits alternating

known whitening:

................ ............... ............... ee1  c76  ...  0b7  ...... ....

In [41]: a = BitStream('0xee1')
In [42]: b = BitStream('0xd5b')
In [43]: c = BitStream('0x6e1')
In [44]: d = BitStream('0xf9d')
In [45]: ab = a ^ b
In [46]: ab.reverse()
In [47]: ab.int
Out[47]: 1500
In [48]: ac = a ^ c
In [49]: ac.reverse()
In [50]: ac.int
Out[50]: 1
In [51]: ad = a ^ d
In [52]: ad.reverse()
In [53]: ad.int
Out[53]: 1000
replay
demo
Welcome, Guest

Username: [input] Password: [input] Remember me [ ] Login

Forgot your password? Forgot your username? Create an account

Forum Development Protocol Development JD 395 cx-10

**TOPIC: JD 395 cx-10**

**JD 395 cx-10** 17 Jul 2014 14:39

**kamueone**

The new green Board on the Cheerson cx-10 uses the same protocol as the JD 395.

Is there a way to use that protocol with a devo 7e?

Thank you for your help,

**JD 395 cx-10** 17 Jul 2014 15:59

**SeByDocKy**

kamueone wrote:

The new green Board on the Cheerson cx-10 uses the same protocol as the JD 395.

Is there a way to use that protocol with a devo 7e?

Thank you for your help,
Sturdy palm tree
No description or website provided.

- **Dominicgs / sturdy-palm-tree**
  - 7 commits
  - 3 branches
  - 0 releases
  - 1 contributor

Branch: **master**

- **SturdyPalmTree**: Fix arbitrary channel/modem settings for sniffing
  - Latest commit: cf10f59 4 days ago
  - 4 days ago
- **.gitignore**: Initial commit
  - A month ago
- **LICENSE**: Initial commit
  - A month ago
pitch = unwhitened[143:127:-1].read('uint:16')
throttle = unwhitened[159:143:-1].read('uint:16')
yaw = unwhitened[171:159:-1].read('uint:12')
flip = unwhitened[175:171:-1].read('uint:4')
special = unwhitened[191:175:-1].read('uint:4')
crc = packet[192:208].read('uint:16')

if crc == cx10a_crc(packet):
    print "%02x %08x %08x %5d %5d %5d %5d %5d %01x %04x %04x" \
    % (mode, cid, vid, roll, pitch, throttle, yaw, flip, special, crc)

def find_cx10a_packet(symbols):
    # search for whitened sync word
    if symbols.find('0xf7d872649', 0, symbols.len - 216):
        decode_cx10a(symbols[symbols.pos:symbols.pos + 216])
        return True, symbols[symbols.pos + 216:]
    else:
        return False, symbols[symbols.len - 216:]

def ubertooth_rx():
    dev = Radio(Radio, UBERTooth)
syncword = 0xf7d8726
dev.configure_radio(frequency=2402, freq_deviation=340, syncword=syncword)
symbol_stream = bitstring.ConstBitStream()
for metadata, pkt in dev.rx_pkts():
    print metadata
    print pkt.bin
    symbol_stream += pkt
    pkt_found, symbol_stream = find_cx10a_packet(symbol_stream)

if __name__ == '__main__':
    ubertooth_rx()
ubertooth snif/fing demo
Your-channel repeated frequency hopping sequence derived from the controller's unique identifier (CID). The controller transmits each packet on a new frequency in the hopping sequence, 5.25 ms after the start of the previous packet on the previous frequency.

2. Packet Payload Format

Packets in both binding phase and flying phase have the same format. The packet payload has a fixed length of 21 bytes divided into 9 fields. Each field is transmitted LSB first.

<table>
<thead>
<tr>
<th>field name</th>
<th>byte offset</th>
<th>length in bytes</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>phase</td>
<td>0</td>
<td>1</td>
<td>operating phase</td>
</tr>
<tr>
<td>CID</td>
<td>1</td>
<td>4</td>
<td>controller identifier</td>
</tr>
<tr>
<td>VID</td>
<td>5</td>
<td>4</td>
<td>vehicle identifier</td>
</tr>
<tr>
<td>aileron</td>
<td>9</td>
<td>2</td>
<td>aileron (roll) control</td>
</tr>
<tr>
<td>elevator</td>
<td>11</td>
<td>2</td>
<td>elevator (pitch) control</td>
</tr>
<tr>
<td>throttle</td>
<td>13</td>
<td>2</td>
<td>throttle control</td>
</tr>
<tr>
<td>rudder</td>
<td>15</td>
<td>1,5</td>
<td>rudder (yaw) control</td>
</tr>
<tr>
<td>flip</td>
<td>16, 16.5</td>
<td>0.5</td>
<td>flip control</td>
</tr>
<tr>
<td>mode</td>
<td>17</td>
<td>2</td>
<td>flight control mode</td>
</tr>
<tr>
<td>CRC</td>
<td>19</td>
<td>2</td>
<td>cyclic redundancy check</td>
</tr>
</tbody>
</table>

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|
| **i** | **pha** | **l** | **e** | **C** | **I** | **D** | **V** | **I** | **D** | **V** | **I** | **D** | **V** | **I** | **D** | **V** | **I** | **D** | **V** | **I** | **D** |

2.1 Phase Field
binding request

reply

confirmation

ack

2402 MHz
Frequency hopping

CID: \( \text{\texttt{0x555556FED0}} \)

\[
\begin{align*}
2403 + \text{\texttt{0x0}} &= 2403 \text{ MHz} \\
2422 + \text{\texttt{0xE}} &= 2436 \text{ MHz} \\
2445 + \text{\texttt{0xF}} &= 2460 \text{ MHz} \\
2464 + \text{\texttt{0x6}} &= 2470 \text{ MHz}
\end{align*}
\]
STM32F031x4 STM32F031x6

ARM®-based 32-bit MCU with up to 32 Kbyte Flash, 9 timers, ADC and communication interfaces, 2.0 - 3.6 V

Features

- Core: ARM® 32-bit Cortex®-M0 CPU, frequency up to 48 MHz
- Memories
  - 16 to 32 Kbytes of Flash memory
  - 4 Kbytes of SRAM with HW parity
- CRC calculation unit
- Reset and power management
  - Digital and I/Os supply: 2.0 to 3.6 V
  - Analog supply: $V_{DDA} =$ from $V_{DD}$ to 3.6 V
  - Power-on/Power-down reset (POR/PDR)
  - Programmable voltage detector (PVD)
  - Low power modes: Sleep, Stop and Standby
  - $V_{BAT}$ supply for RTC and backup registers
- Clock management
  - 1 to 32 MHz crystal oscillator

- 1 x 16-bit timer, with IC/OC and OCN, deadtime generation, emergency stop and modulator gate for IR control
- 1 x 16-bit timer with 1 IC/OC
- Independent and system watch(og) timers
- SysTick timer: 24-bit downcounter

- Calendar RTC with alarm and periodic wakeup from Stop/Standby
- Communication interfaces
  - 1 x I²C interface, supporting Fast Mode Plus (1 Mbit/s) with 20 mA current sink, SMBus/PMBus, and wakeup from Stop mode
图1 XN297系统结构方框图
Black Magic Probe

$70.00

Black Magic Probe by Black Sphere Technologies is a JTAG and SWD Adapter used for programming and debugging ARM Cortex MCUs. Its the best friend of any ARM microcontroller developer.

Features:

- GDB server port without the need of special PC side software.
- TTL level serial interface
- SWD and JTAG support
- Semihosting support
- Works on Linux, Mac and Windows
- Works with Eclipse and other Integrated Development Environments
- Supports STM32, LPC11, LM3S - full support list
- DroneCode compatible

Dimensions:

- 33mm (1.3 in) x 15mm (0.6 in)
- 2.4g (0.85oz)
project ideas
dump/reverse firmware
fix full throttle bug
game controller/HID
autopilot (remote or local)
follow the leader
add crypto to protocol
flying ibeacon
flying bluetooth sniffer
POV
LED/RF/audio positioning
jam/hijack
sturdy palm tree
frequency hopping
binding
firmware functions
name
slide advance demo