Milight new protocol is used in the new Milight lightbulbs and controllers. The protocol is different from both the RGBW and CW WW the so called CCT bulbs. Thanks to Chris Mullins the Milight new protocol has been decoded including the new encryption method. I have used this to be able to program the nRF24L01 with the correct syncword etc and to decode the protocol.

Bits on Air	010101010101010101010101010111100010100011010001101001001010111000000000110101001111010101110110111000001000100000000101000101000100101000111111
Value hex	5 5 5 5 5 5 5 E 2 8 D 1 A 4 A E 0 0 D 4 F 5 7 6 E 0 8 8 0 5 1 4 4 A 3 F
Value (little endian)	101010101010101010101010101001110100000110111000010100100101011100000000101100101111101011100110011100000001000100001010100000100010010111001111
Value hex	A A A A A A A 7 4 1 B 8 5 2 5 7 0 0 B 2 F A E 6 7 0 1 1 0 A 8 2 2 5 C F

AS Pl1167	3 Bytes	4 Bytes	4 bits	1 Byte	7 Bytes	2 Bytes
Field	Preamble	Sync	Trailer	Length	Payload	CRC
Value	0xAAAAAA	0x258B147A	0x5	0x07	0xB0 0xF2 0xEA 0x76 0x10 0x01 0x8A	0xC522

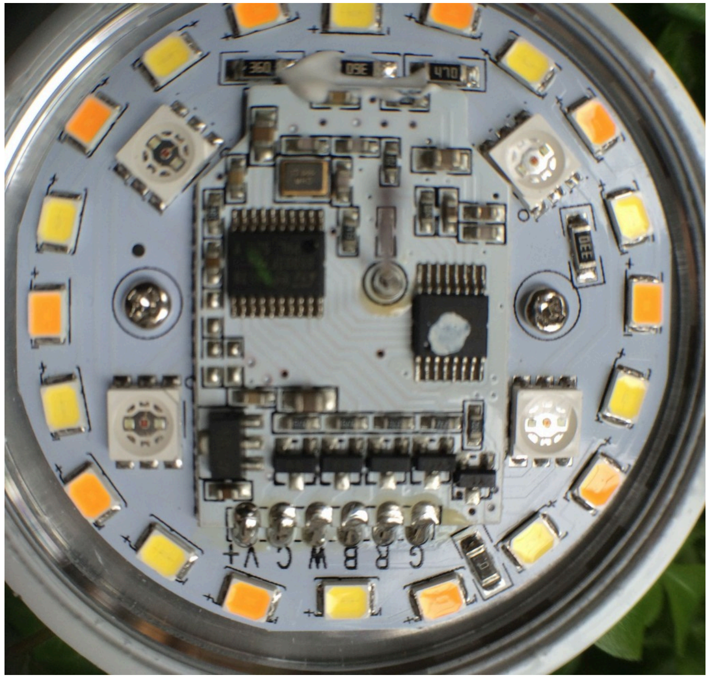
AS nRF24L01	3 Bytes	5 Bytes	12 Bytes
Field	Preamble	Adress	Payload
Value	0x55	0xD1 0x28 0x5E 0x55 0x55	0xA4 0xAE 0x00 0xD4 0xF5 0x76 0xE0 0x88 0x05 0x14 0x4A 0x3F

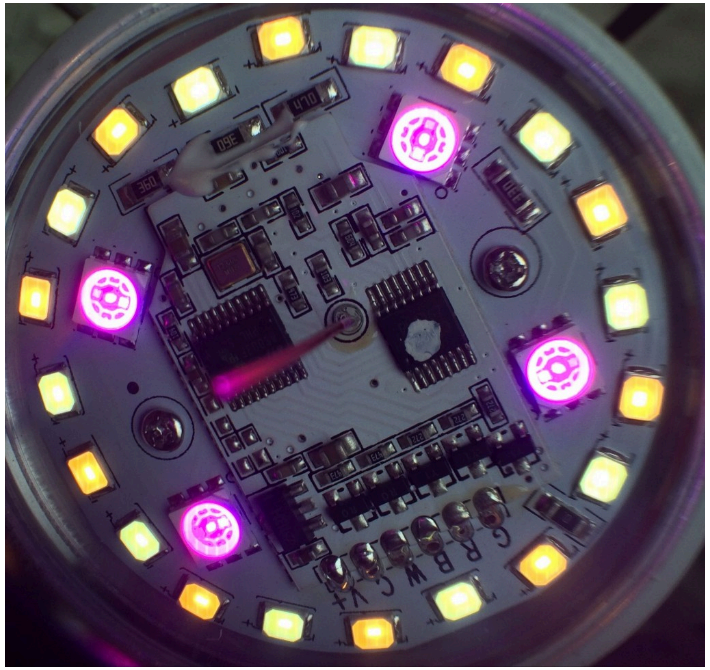
Bits on Air	0101010101010101010101010110110001001110100100000001100010101001000001111000001000010110110000110111011110001000000001010001010001001010001101000100101000111111
Value hex	5 5 5 5 5 5 6 C 4 E 9 0 1 8 A 9 0 7 8 2 1 6 C 3 7 7 8 8 0 5 1 4 4 A 3 4 4 A 3 F
Value (little endian)	1010101010101010101010100110001100100111100100001000000101011001000011100001010010000110001111001110111001100101010010111000110100101111110100111000000110101111
Value hex	A A A A A A 6 3 2 7 9 0 8 1 5 9 0 E 1 4 8 6 3 C E E 6 5 4 B 8 D 2 F D 3 8 1 A F

AS Pl1167	3 Bytes	4 Bytes	4 bits	1 Byte	9 Bytes	2 Bytes
Field	Preamble	Sync	Trailer	Length	Payload	CRC
Value	0xAAAAAA	0x18097236	0x5	0x09	0xB0 0xF2 0xEA 0x76 0x10 0x01 0x8A 0x22 0xC5	0xC522

AS nRF24L01	3 Bytes	5 Bytes	14 Bytes
Field	Preamble	Adress	Payload
Value	0x55	0x90 0x4E 0x6C 0x55 0x55	0x18 0xA9 0x00 0xD4 0xF5 0x76 0xE0 0x88 0x05 0x14 0x4A 0x34 0x4A 0x3F

The new lightbulb is 8W and contains RGB and both warm white and cold white leds compared to only one type of white in the old bulbs. It also reacts to more commands from the new remote than the old ones.





The most righthand picture shows that it is possible to switch on all the LEDs at the same time which is not possible with the 6W RGBW lightbulb.

To control this milight bulb you need all 5 channels to control the RGB but also the CW and WW signals. It was this that i had in mind when designing the controller PCB for the 6W 4 channel milight bulb. This PCB is published on the RF page but this time the few additional components on the backside of the print are needed to control all 5 channels. Optionally these (exept one 10k resistor) can be left out if you want to use the controller PCB for the 6W RGBW milight bulb. You can see that there are 6 connections for the milight bulb print above. This conform the requirement of the RGB CW and WW milight bulb printed circuit board.
Below you can find the new code for descrambling the new protocol using arduino code. Now that this is possible i want to get cracking with making my own milight RGB CCT bulb but based on the smaller 6W variety.

[spoiler effect="blind" show="RGB CCT Sniffer" hide="hide me"]

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h> //http://tmrh20.github.io/RF24/
#include "PL1167_nRF24.h"
#include "MiLightRadio.h"

// define connection pins for nRF24L01 shield on www.arduino-projects4u.com
#define CE_PIN 9   //ESP8266 2
#define CSN_PIN 10 //ESP8266 15

#define V2_OFFSET(byte, key, jumpStart) (
  V2_OFFSETS[byte-1][key%4]
    +
  ((jumpStart > 0 && key >= jumpStart && key <= jumpStart+0x80) ? 0x80 : 0)
)

#define V2_OFFSET_JUMP_START 0x54

uint8_t const V2_OFFSETS[][4] = {
  { 0x45, 0x1F, 0x14, 0x5C },
  { 0x2B, 0xC9, 0xE3, 0x11 },
  { 0xEE, 0xDE, 0x0B, 0xAA },
```

```
  { 0xAF, 0x03, 0x1D, 0xF3 },
  { 0x1A, 0xE2, 0xF0, 0xD1 },
  { 0x04, 0xD8, 0x71, 0x42 },
  { 0xAF, 0x04, 0xDD, 0x07 },
  { 0xE1, 0x93, 0xB8, 0xE4 }
};
uint8_t xorKey(uint8_t key) {
  // Generate most significant nibble
  const uint8_t shift = (key & 0x0F) < 0x04 ? 0 : 1;
  const uint8_t x = (((key & 0xF0) >> 4) + shift + 6) % 8;
  const uint8_t msn = (((4 + x) ^ 1) & 0x0F) << 4;
  // Generate least significant nibble
  const uint8_t lsn = ((((key & 0x0F) + 4)^2) & 0x0F);
  return ( msn | lsn );
}
uint8_t decodeByte(uint8_t byte, uint8_t s1, uint8_t xorKey, uint8_t s2) {
  uint8_t value = byte - s2;
  value = value ^ xorKey;
  value = value - s1;
  return value;
}
uint8_t encodeByte(uint8_t byte, uint8_t s1, uint8_t xorKey, uint8_t s2) {
  uint8_t value = byte + s1;
  value = value ^ xorKey;
  value = value + s2;
  return value;
}
void decodeV2Packet(uint8_t *packet) {
  uint8_t key = xorKey(packet[0]);
  for (size_t i = 1; i <= 8; i++) {
    packet[i] = decodeByte(packet[i], 0, key, V2_OFFSET(i, packet[0], V2_OFFSET_JUMP_START));
  }
}
RF24 radio(CE_PIN, CSN_PIN);
```

```
PL1167_nRF24 prf(radio);
MiLightRadio mlr(prf);

void setup()
{
  Serial.begin(115200);
  Serial.println();
  delay(1000);
  Serial.println("# OpenMiLight Receiver/Transmitter starting");
  mlr.begin();
}


static int dupesPrinted = 0;
//static bool receiving = true; //set receiving true or false
//static bool escaped = false;
//static uint8_t outgoingPacket[7];
//static uint8_t outgoingPacketPos = 0;
//static uint8_t nibble;
//uint8_t crc;
static enum {
  IDLE,
  HAVE_NIBBLE,
  COMPLETE,
} state;

void loop()
{
    if (mlr.available()) {
      uint8_t packet[9];
      size_t packet_length = sizeof(packet);
      Serial.println();
      Serial.print("<-- ");
      if (packet_length<0x10) Serial.print("0");
```

```
      Serial.print(packet_length,HEX);
      Serial.print(" ");
      mlr.read(packet, packet_length);
      for (int i = 0; i < packet_length; i++) {
        if (packet[i]<0x10) Serial.print("0");
        Serial.print(packet[i],HEX);
        Serial.print(" ");
      }
      Serial.print("Decoded package = ");
      uint8_t key = xorKey(packet[0]);
      uint8_t sum = key;
      Serial.print(key,HEX);Serial.print(" ");
      for (size_t i = 1; i <= 7; i++) {
        packet[i] = decodeByte(packet[i], 0, key, V2_OFFSET(i, packet[0], V2_OFFSET_JUMP_START));
        sum += packet[i];
        if (packet[i]<0x10) {Serial.print("0");}
        Serial.print(packet[i],HEX);Serial.print(" ");
      }
    }
    int dupesReceived = mlr.dupesReceived();
    for (; dupesPrinted < dupesReceived; dupesPrinted++) {
    Serial.print(".");
    }
}
```

[/spoiler]

The buttons on the remote have the following codes:

Button	Command	Arguement
ALL ON	01	00

ALL OFF	01	05
ZONE 1 ON	01	01
ZONE 2 ON	01	02
ZONE 3 ON	01	03
ZONE 4 ON	01	04
ZONE 1 OFF	01	06
ZONE 2 OFF	01	07
ZONE 3 OFF	01	08
ZONE 4 OFF	01	09
S+	01	0A
S-	01	0B
White	03	B0-FF – 00-AF
Colourwheel	02	00-FF
Brightness	04	80-FF
Saturation	04	00-7F
Mode	05	0,1,2,3,4,5,6,7,8

Finally the output from the program showing a RGB CCT remote talking to the arduino. First it shows the scrambled code then the decoded package.
You can see the key lead byte 0x20 followed by the ID1 and ID2 bytes. Then the command argument and sequence bytes and finally the group byte. I have not shown the checksum byte.

```
# OpenMiLight Receiver/Transmitter starting

<-- 09 64 6F 43 B6 BA 24 0C B9 EF Decoded package = 8A 20 12 C2 01 00 02 00 .
<-- 09 A8 33 07 FA 7E E8 D1 7D E8 Decoded package = 4E 20 12 C2 01 00 03 00 .
<-- 09 BF 2D F4 DD E3 C5 37 F8 E7 Decoded package = 71 20 12 C2 01 05 04 00 .
<-- 09 BC 17 0B 1E A2 11 F8 A1 E8 Decoded package = 72 20 12 C2 01 05 06 00
<-- 09 E4 EF C3 36 3A A4 91 39 E4 Decoded package = 8A 20 12 C2 01 00 07 00
<-- 09 57 95 1C 85 0B ED 52 20 EA Decoded package = 99 20 12 C2 01 05 09 00
<-- 09 B2 F8 B9 11 E2 BE 3A A1 CF Decoded package = 44 20 12 C2 01 0A 0D 00
<-- 09 FE A4 85 7D CE AA 2F 8D 28 Decoded package = B0 20 12 C2 01 0A 0E 00
<-- 09 55 5A D2 B7 1D F3 E3 1F C9 Decoded package = 9B 20 12 C2 01 0A 10 00 .
<-- 09 01 B6 6E 53 B9 9F 7E BB 10 Decoded package = B7 20 12 C2 01 0A 11 00
<-- 09 22 88 29 A1 72 4F B8 31 F6 Decoded package = 54 20 12 C2 01 0B 13 00
<-- 09 7F 6D 34 9D 23 0B 67 38 5D Decoded package = B1 20 12 C2 01 0B 14 00
<-- 09 36 6C 4D C5 96 6C DF 59 AE Decoded package = 78 20 12 C2 01 04 16 04
<-- 09 60 7B 2F C2 C6 2C 12 C1 19 Decoded package = 96 20 12 C2 01 04 18 04 ..
```

It is also possible to decode several differenmt protocols at the same time.
This requires a slightly modified version of the Henryk Plötz version. It allows the automatic switching between keywords, channels and length of the messages so that you can read any milight remote and get the codes. This version includes the new RGB + CCT remotes alswell as the older CCT, RGBW and RGB remotes.

Below you can see the ouput of the program showing several different remotes with different protocols being received on the fly at the same time without changing any code.

```
# OpenMiLight Receiver/Transmitter starting

<-- 09 47 A4 85 63 5B 39 82 6F B3 Decoded package = 69 21 1D D0 01 01 29 01 ....
<-- 09 D0 0B 1F 12 96 01 FD 96 FA Decoded package = 66 20 12 C2 01 01 1F 01
<-- 09 13 E0 C9 1F 97 75 D1 AB 2A Decoded package = A5 21 1D D0 01 01 2A 01 .
<-- 07 B0 73 13 00 01 03 04
<-- 07 5A 8D 11 01 08 0E 16
<-- 07 B0 73 13 00 01 03 05 .
<-- 07 B0 73 13 00 01 04 06
<-- 09 DA D0 71 69 BA 88 2D 75 D8 Decoded package = 9C 20 12 C2 01 04 20 04
<-- 09 D4 FF B3 46 4A AD BC 4D 3C Decoded package = 9A A0 92 42 81 89 A2 84
```

Recent Comments

- hothyundai on Contact
- Vuk Petrovic-Mijic on Contact
- hothyundai on Contact

Tag cloud

DCF77 & MSF60 Shields Uncategorized Wii

Meta

- Log in
- Entries feed
- Comments feed
- WordPress.org