Francisco Ferrer

Département Technique

HELB Ilya Prigogine

# Projet électronique

AT-BLASTER v1.0

Gilles Vico
2EA HEFF

Gilles VICO

Année académique 2020-201

# Table des matières

# 1. Introduction

M'intéressant au monde du loisir et cherchant de nouvelles façons de m'amuser avec mes amis tout en respectant la distanciation sociale, j'ai décidé de concevoir mon propre système de Laser Game.

Les systèmes existants sur le marché sont généralement chers et ne répondent pas à mes attentes, ce projet me permet donc d'obtenir quelque chose de plus modulable que je pourrai adapter en fonction de mes besoins.

Je pourrais même en faire un jeu de tir sur cibles sans avoir à modifier le hardware.

Chaque joueur possède un pistolet. Les pistolets communiquent leur « tir » via de la lumière infrarouge : des LED au bout du pistolet, et un capteur positionné sur le torse du joueur.

Je me suis fixé l'objectif de pouvoir faire un suivi des scores, de façon à pouvoir déterminer un joueur gagnant à la fin de la partie. Pour ce faire, je me suis servi de modules radio, qui permettront aux pistolets de communiquer entre eux de façon non-directionnelle.

L'objectif de départ était d'avoir un simple système de pistolets, avec une LED servant à indiquer le gagnant en fin de partie.

Des fonctions bonus ont également été envisagées en fonction du temps restant disponible :

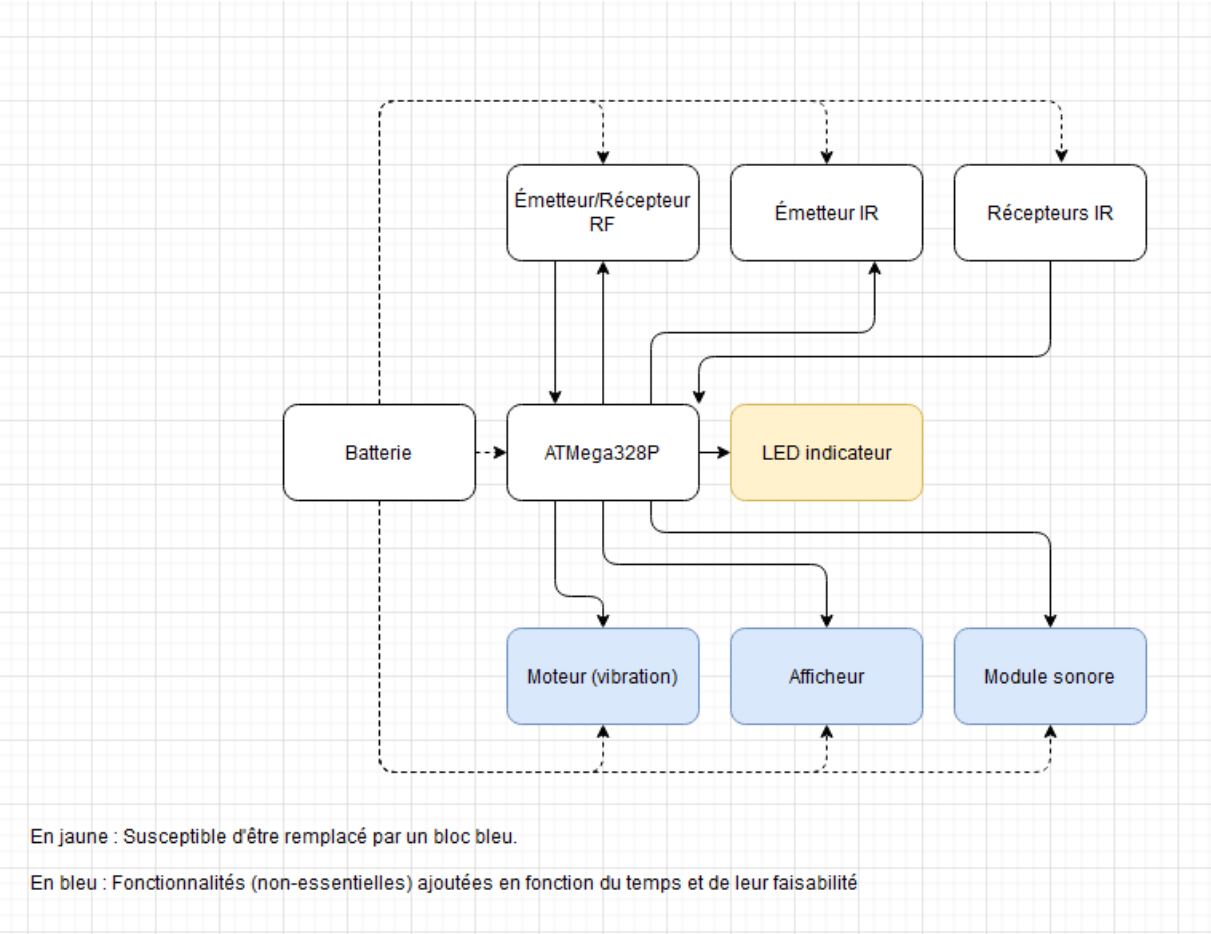
- Un écran LCD pour afficher les scores en temps réel
- Un buzzer pour émettre un signal sonore quand le pistoler tire
- Un moteur permettant de faire vibrer le pistolet quand il se fait toucher

# 2. Spécifications techniques

- Microcontrôleur : **ATmega328P U-TH**
- Transmission infrarouge : **LEDs infrarouge**
- Réception infrarouge : **Module VS1838B**
- Transmission et réception radio : **Module HC-12**
- Affichage : **LCD 16x2 caractères**
- Alimentation : **Pile 9V interchangeable**

- Dimensions du PCB : **50*176mm**
- Boîtier imprimé en 3D
- Prix des composants d'un pistolet (PCB non-inclus) : **43,51 €**

# 3. Schéma bloc




Le LCD a finalement remplacé la LED indicateur.

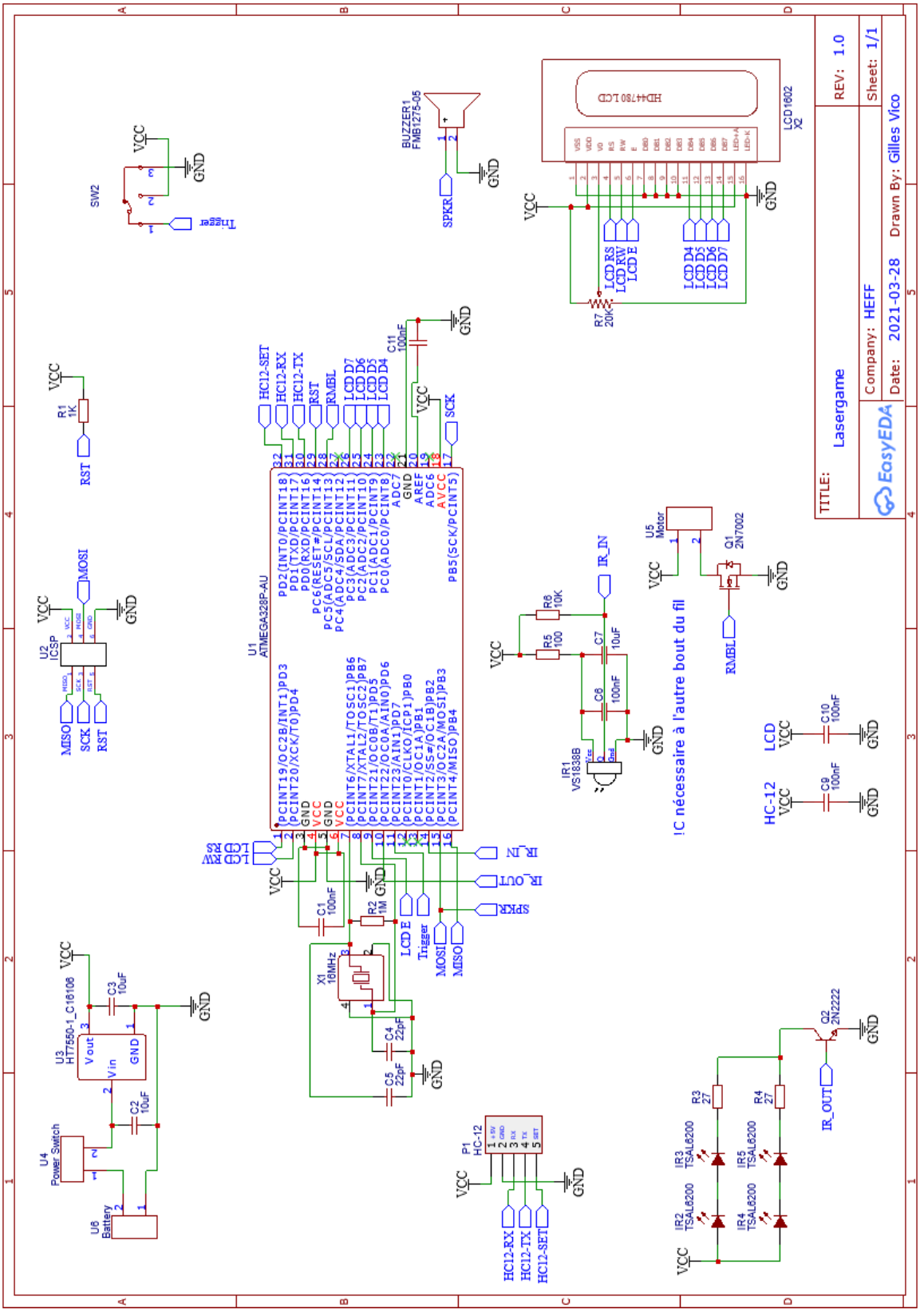
Le buzzer et le moteur ont aussi été prévus sur le PCB.

Un régulateur s'assure que la tension Vcc est de 5V.

La transmission IR assure la détection des tirs.

La transmission RF assure la communication entre les pistolets.

# 4. Schéma électronique



# 5. Choix de composants

La plupart des composants (dont le microcontrôleur et son cristal) ont été choisis dans leur version CMS, afin de gagner de la place.

Pour l'émission infrarouge, le choix n'est pas bien compliqué et les LEDs infrarouges s'imposent. Un système de laser aurait été techniquement possible, mais en plus de coûter beaucoup plus cher il est dangereux : la lumière du laser occasionne des dégâts aux yeux, sans même qu'on ne puisse s'en rendre compte, celle-ci étant invisible.

Plusieurs LEDs ont été utilisées ici afin de renforcer la puissance de l'émission.

Pour la transmission et réception radio, plusieurs modules ont été considérés :

- Tout d'abord des **nrf24l01**, mais ceux-ci utilisaient un protocole limitant les possibilités (notamment en ne permettant la communication qu'entre un nombre limité d'appareils), et s'avèrent compliqués à l'utilisation.

- Des modules génériques ont aussi été testés, leur prix les rendait intéressants mais leurs performances étaient particulièrement médiocres, la communication s'interrompant après moins d'un mètre.

- Finalement, les modules retenus sont les **HC-12**, qui sont performants, ont un prix raisonnable et sont faciles à utiliser. Etant de la même famille que les HC-05 vus en cours de systèmes à microcontrôleurs, il était d'autant plus simple de travailler avec ceux-ci.

Pour l'afficheur, un LCD 16x2 caractères identique à celui utilisé en cours de systèmes à microcontrôleurs a été utilisé, car celui-ci est connu et donc plus simple à utiliser.

La programmation de l'ATmega328P se fait directement via ICSP. Un branchement en USB n'est pas nécessaire, et le bus UART est de toute façon déjà utilisé par le module HC-12.

Un régulateur est présent pour assurer une tension de 5V, d'autres tensions (comme 3.3V) ne sont pas requises.

# 6. Mise en œuvre de la CEM

Voir les images du PCB pour illustration.

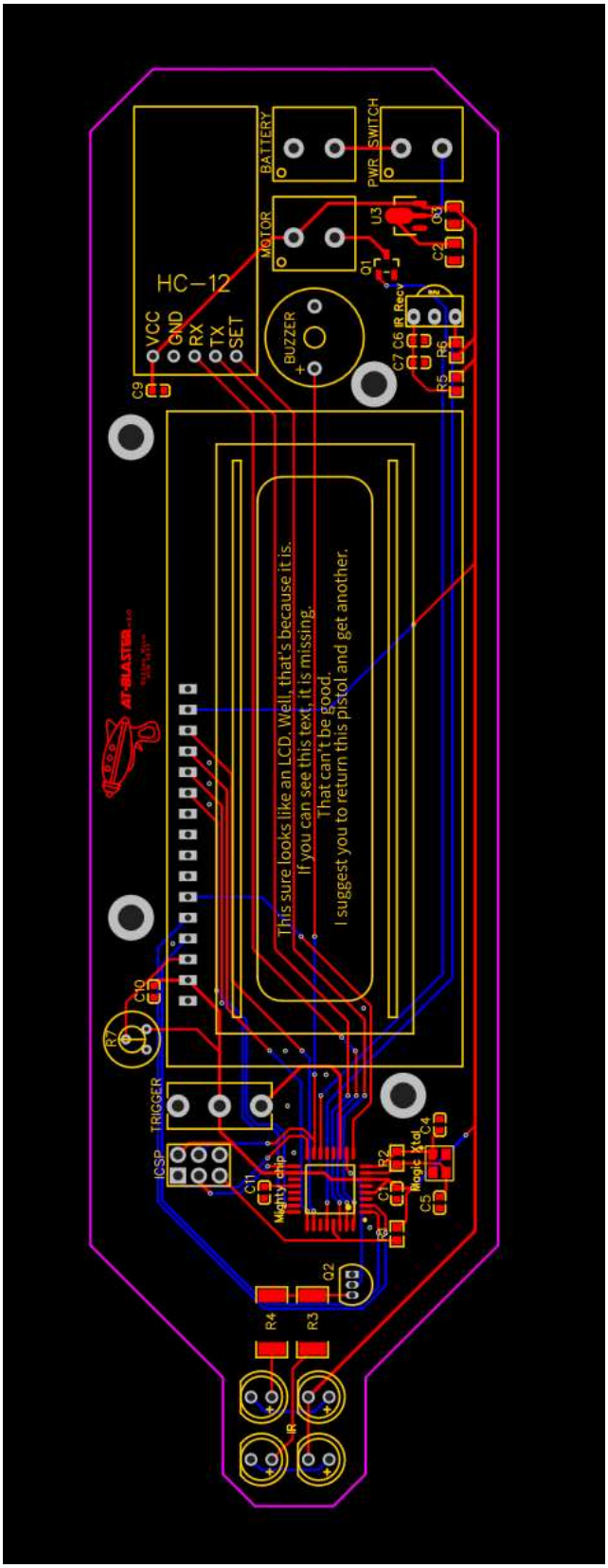
Afin de réduire au maximum le bruit électrique dans le circuit, plusieurs précautions ont été prises :

- Un plan de masse est présent sur les deux faces du PCB.
- Des condensateurs de découplage de 100nF sont placés entre Vcc et la masse, au plus proche de chaque composant actif.
- Les pistes ne font jamais d'angle autre que 45°.
- La tension Vcc est distribuée le long d'une piste principale.
- Les fils utilisés pour transporter la tension et le signal du module VS1838B, éloigné du PCB pour être sur le torse du joueur, sont blindés.

# 7. Liste de matériel (BOM)

| **BOM : AT-Blaster v1.0** | | | |
|---|---|---|---|
| 1 | ATmega328P U-TH | 3,25 € | 3,25 € |
| 1 | HC-12 | 9,90 € | 9,90 € |
| 1 | LCD 16x2 | 11,03 € | 11,03 € |
| 1 | Micro-switch | 2,70 € | 2,70 € |
| 1 | Switch | 1,32 € | 1,32 € |
| 2 | LED IR | 1,32 € | 2,64 € |
| 1 | VS1838B | 0,44 € | 0,44 € |
| 1 | Header 1x30 | 1,17 € | 1,17 € |
| 1 | Batterie 9V | 7,47 € | 7,47 € |
| 1 | HCT7750 | 0,17 € | 0,17 € |
| 1 | 2N2222 | 0,48 € | 0,48 € |
| 1 | 2N7002 | 0,09 € | 0,09 € |
| 1 | 75H 20K | 1,01 € | 1,01 € |
| 1 | 16MHz Xtal | 1,01 € | 1,01 € |
| 3 | 10µF céramique | 0,10 € | 0,30 € |
| 2 | 22pF céramique | 0,04 € | 0,08 € |
| 5 | 100nF céramique | 0,02 € | 0,10 € |
| 1 | 1K 0805 | 0,02 € | 0,02 € |
| 1 | 1M 0805 | 0,25 € | 0,25 € |
| 2 | 27E 2512 | 0,02 € | 0,04 € |
| 1 | 100E 0805 | 0,02 € | 0,02 € |
| 1 | 10K 0805 | 0,02 € | 0,02 € |
| Total : | | | 43,51 € |

# 8. Conception du PCB



Dans cette vue, le plan de masse n'est pas activé afin que tout soit visible.

Le PCB a été réalisé avec EasyEDA.

Ce afin d'expérimenter d'autres logiciels de CAO, et de pouvoir profiter du service d'assemblage de JLCPCB, société par laquelle sont commandés les PCB.

Les composants ont été placés en gardant à l'esprit la forme du pistolet : le PCB devait donc être allongé, avec le LCD au centre, la connectique externe majoritairement à l'arrière (de façon à pouvoir faire passer des fils dans la poignée du pistolet), et les LEDs infrarouges à l'avant.

Des trous ont également été prévus afin de pouvoir fixer le PCB au boitier avec des vis.

# 9. Code C

main.c

```
#include "Drivers/Headers/IO_Config.h"
#include "Drivers/Headers/UART_Config.h"

#include "Application/Headers/Globals.h"
#include "Application/Headers/Defines.h"
#include "Application/Headers/LCD.h"
#include "Application/Headers/HC-12.h"
#include "Application/Headers/Lasergame.h"

#include <avr/interrupt.h>
#include <avr/io.h>
#include <util/delay.h>
#include <string.h>
#include <stdlib.h>



int main(void)
{
	io_config();
	lcd_init();
	uart_config();

	sei();

	lcd_Write_String("#  AT-BLASTER  #","# HW1.0  SW1.0 #");
	_delay_ms(2000);
	if (TRIGGER_PULLED)
		HC12_init();  // Only initializes the HC-12 when the trigger is pulled on
boot
	lcd_clear();

	lcd_Write_Line("# Score :      #",2);
	while (game_on)
	{
		if (rx_string[0] == 'S')
			pistol_hit();
		else if (rx_string[0] == 'H' && rx_string[3] == UNIT_ID)
		{
			points_i += 50;
			lcd_Set_Cursor(2, 10);
			lcd_Write(itoa(points_i,points_a,10));
		}
		else if (rx_string[0] == 'E')     // Endgame if someone reaches 1000
points (20 successful shots)
			game_on = 0;
		rx_string[0] = 0;

		if (TRIGGER_PULLED)
		{
			pistol_shoot();
		}
```

```
		if (points_i == 1000)
			pistol_endgame();
	}

	while (1)
	{
		if (points_i == 1000)
			lcd_Write_Line("#  Victoire !  #", 1);
		else
			lcd_Write_Line("#  Game Over   #", 1);

		_delay_ms(1000);
		lcd_Write_Line("#              #", 1);
		_delay_ms(500);
	}
}
```

## IO_Config.c

```
#include "../Headers/IO_Config.h"
#include <avr/io.h>

void io_config(void)
{
	/* IO direction definition - 1 for output*/
	DDRB = 0b00001000;
	DDRC = 0b00101111;
	DDRD = 0b01111110;

	/* Input pull-up control - 0 to disable*/
	/* Output value control - pull-up always disabled in this case*/
	PORTB = 0b00110011;
	PORTC = 0b00010000;
	PORTD = 0b00000100;

	/* Input value */
	/* Writing a 1 toggle PORTx value (pull-up ctrl if input or output value)*/
	PINB = 0x00;
	PINC = 0x00;
	PIND = 0x00;

	/* Global pull-up control on inputs - 1 to disable*/
	MCUCR = 0x00;
}
```

## UART_Config.c

```
#include <avr/io.h>

void uart_config(void)
{
	UCSR0A = 0x00;			//single speed - multiprocessor off
	UCSR0B = 0b10011000;		//RX int ON - TX int OFF - RX enable - TX enable
	UCSR0C = 0b00000110;		//Asynchronous - No Parity - 8 Data bits - 1 Stop
bit

	UBRR0 = 103;			//9600bps
}

void uart_send(char data)
{
	while (!(UCSR0A & (1<<UDRE0)));

	UDR0 = data;
}

void uart_send_string(char *data)
{
	while (*data)
		uart_send(*data++);
}
```

## Globals.c

```
#include "../Headers/Globals.h"

volatile char rx_string[8];
volatile char rx_i = 0;

char game_on = 1;
int points_i = 0;
char points_a[5];
```

Defines.c

```
#ifndef DEFINES_H_
#define DEFINES_H_

#ifndef F_CPU
#define F_CPU 16000000UL  // 16 MHz
#endif

#ifndef byte
#define byte  unsigned char
#endif

#ifndef uint
#define uint  unsigned int
#endif

#ifndef ulong
#define ulong unsigned long
#endif

#ifndef UNIT_ID
#define UNIT_ID 0x02// Unique ID to identify each pistol
#endif

#ifndef IR_ON
#define IR_ON PORTD |= 0b01000000
#endif

#ifndef IR_OFF
#define IR_OFF PORTD &= 0b10111111
#endif

#ifndef INCOMING_IR
#define INCOMING_IR (PINB & (1 << PINB2)) != (1 << PINB2)    // PINB2 low when IR
input
#endif

#ifndef TRIGGER_PULLED
#define TRIGGER_PULLED (PIND & (1 << PIND7)) == (1 << PIND7) // PIND7 high when
trigger pulled
#endif

#endif /* DEFINES_H_ */
```

LCD.c

```
#include "../Headers/LCD.h"
#include "../Headers/Globals.h"
#include "../Headers/Defines.h"

#include <util/delay.h>

/*****************************************************
Function:           lcd_init
Receive :           nothing
Send:               nothing
Description:  LCD initialization
!! SHALL BE CALLED ONCE BEFORE USING LCD TO CONFIGURE
IT PROPERLY !!
*****************************************************/

void lcd_init(void)
{
	RS_CLR;
	Data_PORT_CLR;

	_delay_ms(5);
	Data_PORT |= 0x20 >> 4;
	EN_SET;
	_delay_ms(5);
	EN_CLR;

	lcd_cmd(0x28);		// 4-bit mode - 2 line - 5x7 font.
	lcd_cmd(0x01);		// Clear display
	lcd_cmd(0x0c);		// Display no cursor - no blink.
	lcd_cmd(0x06);		// Automatic Increment - No Display shift.
	lcd_cmd(0x80);		// Address DDRAM with 0 offset 80h.
}

/*****************************************************
Function:           lcd_cmd
Receive :           command to send to LCD
Send:               nothing
Description:  used to send commands to LCD.

SHOULD ONLY BE USED IN LCD.C
*****************************************************/

void lcd_cmd(char cmd)
{
	RS_CLR;
	Data_PORT_CLR;
	//Data_PORT |= (cmd & 0xf0);
	Data_PORT |= (cmd & 0xf0)>>4;
	EN_SET;
	_delay_ms(5);
	EN_CLR;
	Data_PORT_CLR;
	//Data_PORT |= ((cmd<<4) & 0xf0);
	Data_PORT |= ((cmd<<4) & 0xf0)>>4;
	EN_SET;
	_delay_ms(5);
	EN_CLR;
}
```

```
/*************************************************
Function:            lcd_data
Receive :            data to send to LCD
Send:                nothing
Description:  used to send data to LCD in 4bits mode

CAN BE USED OUTSIDE LCD.C TO SEND 8bits DATA TO LCD
*************************************************/

void lcd_data(char data)
{
       RS_SET;
       Data_PORT_CLR;
       Data_PORT |= (data & 0xf0)>>4;
       EN_SET;
       _delay_ms(5);
       EN_CLR;
       Data_PORT_CLR;
       Data_PORT |= ((data<<4) & 0xf0)>>4;
       EN_SET;
       _delay_ms(5);
       EN_CLR;
}

/*************************************************
Function:            lcd_clear
Receive :            nothing
Send:                nothing
Description:  used to clear LCD screen

CAN BE USED OUTSIDE LCD.C TO CLEAR LCD SCREEN
*************************************************/

void lcd_clear(void)
{
       lcd_cmd(0x01);       // Clear display
}

/*************************************************
Function:            lcd_Set_Cursor
Receive :            a, position on first LCD line
                            b, position on second LCD line
Send:                nothing
Description:  used position cursor on LCD screen

CAN BE USED OUTSIDE LCD.C TO POSITION CURSOR ON SCREEN
*************************************************/

void lcd_Set_Cursor(char a, char b)
{
       if(a == 1)lcd_cmd(0x80 + b);
       else if(a == 2)lcd_cmd(0xC0 + b);
}

/*************************************************
Function:            lcd_Write_String
Receive :            a, text on first LCD line
                            b, text on second LCD line
Send:                nothing
Description:  used to write text on LCD screen
```

```
USED OUTSIDE LCD.C TO POSITION CURSOR ON SCREEN
************************************************/

void lcd_Write_String(char *a,char *b)
{
	int i;

	lcd_clear();

	lcd_Set_Cursor(1,0);
	for(i=0;a[i]!='\0';i++)	lcd_data(a[i]);

	lcd_Set_Cursor(2,0);
	for(i=0;b[i]!='\0';i++) lcd_data(b[i]);
}

/*****************************************************
Function:		lcd_Write_Line
Receive :		s, text on LCD line
			b, LCD line select
Send:			nothing
Description:  used to write text on one line of the LCD screen without clearing it
*****************************************************/

void lcd_Write_Line(char *s,char l)
{
	int i;

	lcd_Set_Cursor(l,0);
	for(i=0;s[i]!='\0';i++)	lcd_data(s[i]);
}

/*****************************************************
Function:		lcd_Write
Receive :		s, text on LCD line
Send:			nothing
Description:  used to write text when cursor position is already defined
*****************************************************/

void lcd_Write(char *s)
{
	int i;
	for(i=0;s[i]!='\0';i++)	lcd_data(s[i]);
}
```

## HC-12.c

```c
#include "../Headers/Defines.h"
#include "../../Drivers/Headers/UART_Config.h"

#include <avr/io.h>
#include <util/delay.h>

/*
Initial configuration of the HC-12.
Does not have to be called at each boot: settings are stored in an EEPROM inside the
HC-12 unit.
*/
void	HC12_init(void)
{
	PORTD &= 0b11111011; // HC-12 SET to 0 (config mode)

	_delay_ms(50);
	uart_send_string("AT+DEFAULT\r\n");	// Settings to default
	_delay_ms(50);
	uart_send_string("AT+C012\r\n");  // Setting channel to 12
	_delay_ms(50);

	PORTD |= 0b00000100; // HC-12 SET to 1 (transmitting mode)
	_delay_ms(50);
}
```

## Lasergame.c

```c
#include "../Headers/Defines.h"
#include "../Headers/Globals.h"
#include "../Headers/LCD.h"
#include "../../Drivers/Headers/UART_Config.h"

#include <util/delay.h>

/*
	RF data memento:
	----------------
	x being the ID of the unit
	y being the ID of another unit

	- Sx : "I am shooting"
	- Hx : "I am hit"
	- By : "I was shot by y"
	- Ey  : Endgame, y won
	- #  : Terminating character
*/
```

```
void pistol_hit(void)
{
	if (INCOMING_IR)
	{
		_delay_ms(5);
		if (INCOMING_IR)	// IR signal must be maintained for 5ms to avoid
false detections due to ambient light
		{
			lcd_Write_Line("#   Ouille !   #",1);

			uart_send('H');
			uart_send(UNIT_ID);
			uart_send('B');
			uart_send(rx_string[1]);	// The ID of the other unit
			uart_send('#');

			_delay_ms(2000);	// 2 seconds of immunity when hit
			lcd_Write_Line("#              #",1);
		}
	}
}

void pistol_shoot(void)
{
	char i = 0;

	IR_ON;
	for (i=0;i<3;i++)	// Sends message 3 times to ensure proper transmission
(keep?)
	{
		uart_send('S');
		uart_send(UNIT_ID);
		uart_send('#');

		_delay_ms(5);
	}
	_delay_ms(20);
	IR_OFF;

	_delay_ms(1000);	// Automatic shot, max 1/second
}

void pistol_endgame(void)
{
	uart_send('E');
	uart_send(UNIT_ID);
	uart_send('#');
	game_on = 0;
}
```

Int_Vectors.c

```c
#include "../Headers/Globals.h"
#include "../Headers/Defines.h"
#include "../../Drivers/Headers/UART_Config.h"

#include <avr/interrupt.h>
#include <avr/io.h>



ISR(USART_RX_vect)
{
	rx_string[rx_i] = UDR0;

	if (rx_string[rx_i] == '#')// RF commands must end with '#'
		rx_i = 0;
	else
		rx_i++;
}
```

# 10. Explication du fonctionnement

Au démarrage, si nécessaire, le module HC-12 est configuré. Une configuration systématique n'est pas nécessaire, le module disposant d'une EEPROM pour sauvegarder les paramètres.

Les autres configurations sont ensuite faites.

Pendant le jeu, le programme vérifie en continu au cas où un message serait reçu par radio, ou si la gâchette est tirée.

Si la gâchette est tirée :

- Les LEDs infrarouge s'allument.
- Le programme envoie un message radio indiquant que le pistolet est en train de tirer.

Si un message est reçu, plusieurs choses peuvent arriver en fonction du message :

- Si quelqu'un tire, le programme vérifie si un signal infrarouge est reçu en même temps. Si c'est le cas et qu'il est maintenu pendant 5ms (pour éviter les faux positifs), c'est que le joueur se fait tirer dessus. Le programme l'affiche donc, et envoie un message indiquant qu'il s'est fait tirer dessus et relayant son numéro d'identification ainsi que celui du tireur.
- Si quelqu'un s'est fait toucher par le pistolet, le joueur gagne 50 points. Si le joueur totalise 1000 points, il gagne. Le programme communique l'information aux autres unités : le jeu s'arrête.
- Si un autre joueur a gagné la partie, le jeu s'arrête.

A la fin de la partie, l'écran du pistolet gagnant indique « Victoire ! », alors que celui des autres indique « Game Over ».

# 11. Dépannage hardware

Les PCB fabriqués ont deux principaux défauts :

- Il manque une résistance de 470 Ohms entre Vcc et l'alimentation du rétro-éclairage du LCD. Sans cela, il fonctionne quand-même mais consomme beaucoup d'énergie et voit sa durée de vie réduite. La piste d'alimentation a donc été coupée sur chaque pistolet et des résistances SMD de 470 Ohms ont été rajoutées.

- Les LEDs infrarouge consomment trop de courant. Comme il s'agit de deux jeux de deux LEDs en série, le problème peut être résolu en en retirant un de ces deux jeux, et en ajustant la valeur de la résistance limitant le courant qui traverse les LEDs.

Par souci de temps et de ressources, le buzzer et le moteur n'ont pas pu être implémentés.

Par conséquent il est difficile de savoir si leur implémentation hardware possède des défauts. Néanmoins, aucune anomalie n'a été relevée au cours des inspections du PCB.

# 12. Dépannage software

Les avaries du software sont trop nombreuses pour être citées, et même retenues, mais plusieurs aspects du programme ont dû être revus au cours du développement.

Il était notamment prévu au départ que le tireur communique son numéro d'identification via infrarouge, mais il s'est finalement avéré plus simple de communiquer uniquement par radio et d'utiliser l'infrarouge comme un simple indicateur.

La gestion de l'interrupt sur RX a dû être revue plusieurs fois aussi, car elle provoquait dans certains cas un ralentissement significatif du fonctionnement du programme, rendant le pistolet inutilisable. La forme utilisée dans le programme actuel est finalement la plus optimisée parmi celles qui ont été tentées, l'expérimentation a donc été plutôt bénéfique.

# 13. Dessin du boîtier

Le boîtier dispose de trous pour visser le PCB, et la poignée en dispose pour placer l'interrupteur servant à l'alimentation et le micro-switch servant de gâchette.

Le bas de la poignée est amovible afin de pouvoir insérer et changer la batterie.

# 14. Conclusion

Je suis plutôt content de ce que j'ai pu produire, je m'attendais à avoir plus de problèmes que je n'en ai eu réellement et le résultat est assez bon.

Je prévois d'améliorer encore à l'avenir mes pistolets, en diversifiant les modes de jeu ou en centralisant les résultats à l'aide par exemple d'un Raspberry Pi. Je pourrais également réaliser les fonctions bonus que je n'avais pas pu faire cette fois-ci, à savoir le buzzer et le moteur.

Une révision du hardware ne serait même pas forcément nécessaire dans l'immédiat, les problèmes de cette première version étant assez faciles à corriger. Les PCBs que je n'ai pas monté pourront donc me servir.

# 15. Datasheets

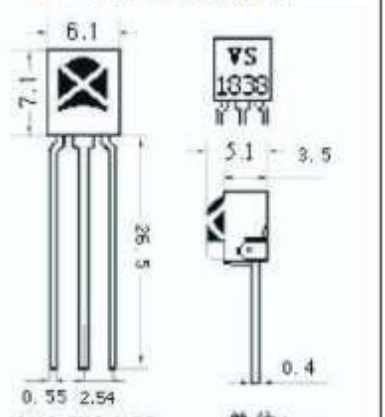


## Infrared Receiver Module 红外线接收器

型号：VS1838B

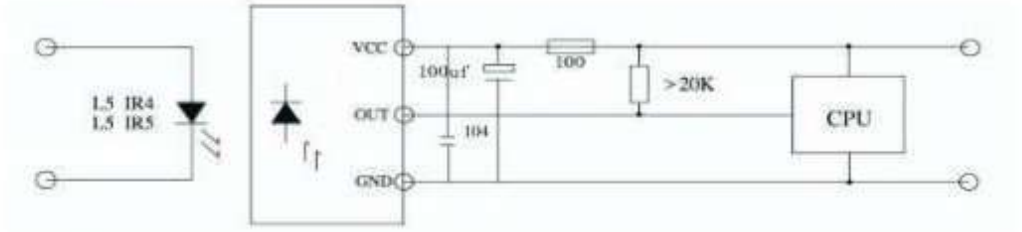
1. 特性
   - ●小型设计；
   - ●内置专用 IC；
   - ●宽角度及长距离接收；
   - ●抗干扰能力强；
   - ●能抵御环境光线干扰；
   - ●低电压工作；

外型尺寸及引脚排列图

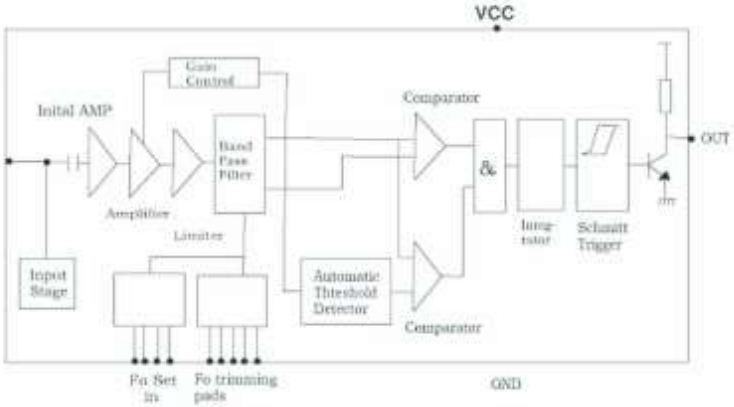



2. 应用：
   - ■视听器材(音响，电视，录影机，碟机)
   - ■家用电器 (冷器机，电风扇，电灯)
   - ■其它无线电器遥控产品；

3. 应用电路图：




4. 原理图：



## Infrared Receiver Module 红外线接收器

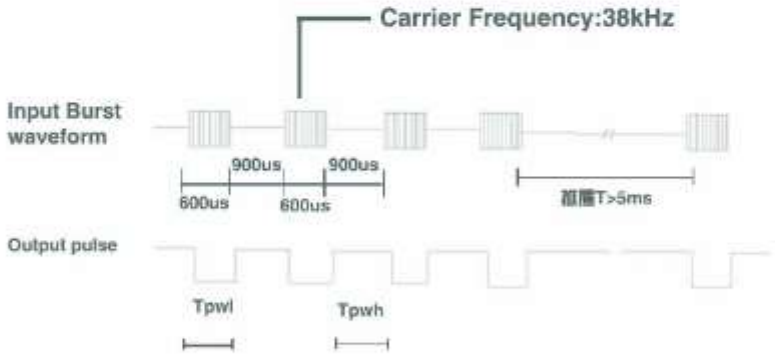
型号：VS1838B

光电参数(T=25℃ Vcc=5v $f_0$=38KHZ)

| 参数 | 符号 | 测试条件 | Min | Typ | Mnx | 单位 |
|---|---|---|---|---|---|---|
| 工作电压 | Vcc | | 2.7 | | 5.5 | V |
| 接收距离 | L | L51R=300MA (测试信号) | 18 | 20 | | M |
| 载波频率 | $f_0$ | | 38K | | | HZ |
| 接收角度 | 01/2 | 距离衰减 1/2 | | +/-45 | | Deg |
| BMP 宽度 | $F_{BW}$ | -3Db andwidth | 2 | 3.3 | 5 | kHz |
| 静态电流 | Icc | 无信号输入时 | —— | 0.4 | 1.5 | mA |
| 低电平输出 | $V_{OL}$ | Vin=0V Vcc=5V | | 0.2 | 0.4 | V |
| 高电平输出 | $V_{OH}$ | Vcc=5V | 4.5 | | | V |
| 输出脉冲 宽 度 | $T_{PWL}$ | Vin=500μVp-p※ | 500 | 600 | 700 | μs |
| | $T_{PWH}$ | Vin=50mVp-p※ | 500 | 600 | 700 | μs |

※光轴上测试，以宽度为 600/900μs 为发射脉冲，在 5CM 之接收范围内，取 50 次接收脉冲之平均值

5. 测试波型：



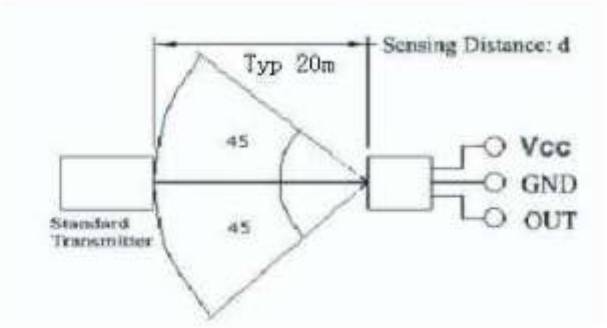

6. 极限参数：

| 项目 | 符号 | 规格 | 单位 |
|---|---|---|---|
| 供应电压 | Vcc | 6.0 | v |
| 工作温度 | Topr | -20-85 | ℃ |
| 储存温度 | Tstg | -40-125 | ℃ |
| 焊接温度 | Tsol | 240 | ℃ |

## Infrared Receiver Module 红外线接收器

**型号：** VS1838B

7. 接收角度：




8. 推荐使用条件：

| 项目 | 符号 | Min | Typ | Mnx | 单位 |
|---|---|---|---|---|---|
| 工作电压 | Vcc | 2.7 | —— | 5.5 | V |
| 输入频率 | FM | | 38 | | kHz |
| 工作温度 | Topr | -20 | 25 | 80 | ℃ |

9. 使用注意

1) 在无任何外加压力及影响品质的环境下储存及使用；

2) 在无污染性气体或海风（含盐份）的环境下储存及使用；

3) 在低湿度环境下储存及使用；

4) 在规定的条件下焊接引线管脚，焊接后，请勿施加外力；

5) 请勿清洗本产品，使用前，请先用静电带将作业员及电烙铁连接落地线；

# HC-12 Wireless Serial Port Communication Module

User Manual version 2.3B
(updated from v1.1 English and v2.3 Chinese)

## Product Applications

- Wireless sensor
- Community building security
- Robot wireless control
- Industrial remote control and telemetry
- Automatic data acquisition
- Container information management
- POS system
- Wireless acquisition of gas meter data
- Vehicle keyless entry system
- PC wireless networking

Document History:

| | | |
|---|---|---|
| 2012/10 | version 1.1 | English, www.seeedstudio.com |
| 2014/09 | version 2.3 | Chinese, www.wavesen.com |
| 2016/01 | version 2.3B | Translated v2.3 online and merged with v1.1 by RR |

## Product Features

- Long-distance wireless transmission (FU3: 1000m in open space, baud rate 5000bps in the air. FU4: 1800m in open space, baud rate 500bps in the air)
- Working frequency range (433.4-473.0MHz, with 100 communication channels)
- Maximum 100mW (20dBm) transmitting power (8 levels of power can be set)
- Four working modes, adapted to different application situations
- Built-in MCU performs communication with external device through serial port, no programming or configuration required for basic use
- The number of bytes transmitted continuously is unlimited (FU1 and FU3 modes only)
- Update software version through the serial port

## Product Introduction

The HC-12 wireless serial port communication module is a new generation of multi-channel embedded wireless data transmission module. Its wireless working frequency band is 433.4-473.0MHz. Multiple channels can be set, with a channel stepping of 400kHz and a total of 100 channels. The maximum transmitting power of the module is 100mW (20dBm), the receiving sensitivity is -117dBm at a baud rate of 5000bps in the air. Communication distance is 1000m (FU3 mode at 4800bps serial speed) in open space, 1800m in FU4 mode at reduced baud rate and volume of data.

The module uses stamp hole packaging to allow for patch soldering, with dimensions of 27.8mm ×14.4mm ×4mm (including antenna cap, excluding spring antenna), making it is very convenient for incorporate into user specific applications. There is a PCB antenna socket ANT1 on the module, so an external 433MHz frequency band antenna can be attached via a coaxial cable; there is also an antenna solder eyelet ANT2 on the module, convenient to solder a spring antenna to. Select one of these antenna options according to usage requirements.

The module has an onboard MCU, eliminating the need for user to program the radio section separately, with transparent half-duplex serial transmission provided for receiving and sending serial port data. This making the HC-12 very easy to interface with. The module adopts multiple serial port transparent transmission modes that are user selected by AT commands according to usage requirements. The average working current of the four modes FU1, FU2, FU3, and FU4 in idle state are: 3.6mA, 80uA, 16mA, and 16mA respectively, while the maximum working current in any mode is 100mA (in the transmitting state).

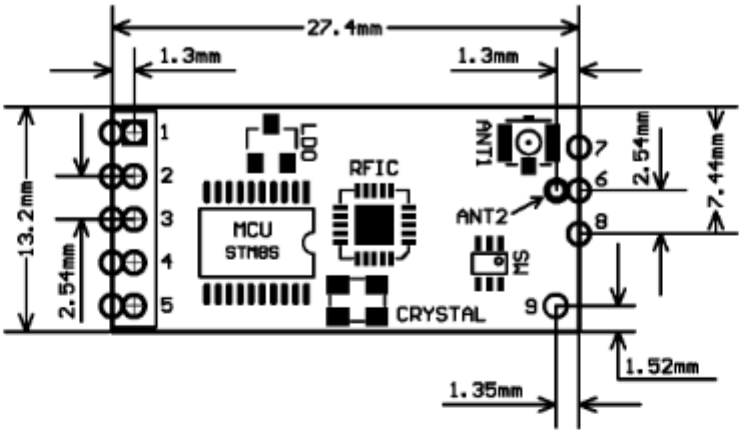
## Product Configuration

Standard configuration of the HC-12 module only contains one 433MHz-frequency-band wireless communication module with IPEX20279-001E-03 standard RF socket. Optional accessories are 433MHz frequency band spring antenna, or IPEX-to-BNC coaxial cable and matching 433MHz frequency band omni-directional rubber antenna with BNC connector base. The user can purchase these according to their application requirements.

## Technical Details

The HC-12 module uses a Silicon Labs Si4463 to provide the RF communications link. This is a high performance, low current, single-chip "EZRadioPRO" family transceiver with up to 20dBm (100mW) transmitting output power. The Si4463 communicates through an SPI bus with an STMicroelectronics STM8S003F3 8-bit MCU that runs the HC-12 firmware. The STM8S provides a transparent serial data interface for interfacing to the module, allowing two HC-12 modules to act like a wired TTL level serial cable without any attached hardware devices needing to be aware of the RF link. Serial port and transceiver configurations are held in onboard non-volatile flash memory.

## Product Dimensions




## Definition of Pins

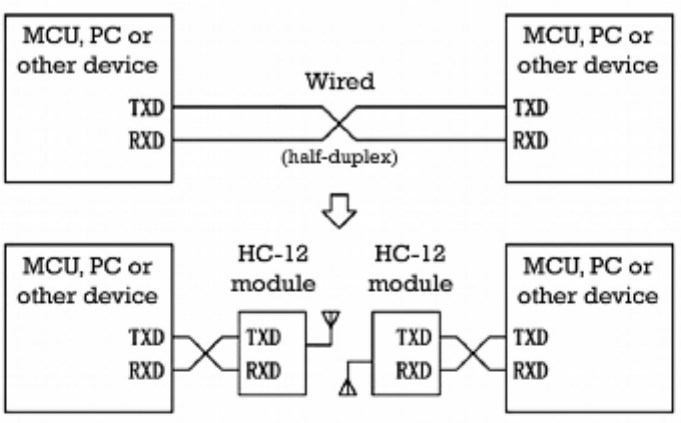
The HC-12 module can be patch soldered, or have a 2.54mm-spacing pin header attached and directly inserted onto the user's PCB. The module has nine pins in total, and one RF antenna socket (ANT1), with definitions as shown in the table below:

| Pin | Definition | I/O direction | Notes |
|---|---|---|---|
| 1 | Vcc | | Power supply input, DC3.2V-5.5V, with load capacity not less than 200mA. Note: if the module is working in the transmitting state for an extended time, it is suggested that a 1N4007 diode be connected in series if the supply voltage is greater than 4.5V, so as to avoid overheating the onboard LDO regulator |
| 2 | GND | | Common ground |
| 3 | RxD | Input (weak pullup) | UART data input, TTL level. 1k resistor connected in series inside the module |
| 4 | TxD | Output | UART data output, TTL level. 1k resistor connected in series inside the module |
| 5 | SET | Input (10k pullup) | Parameter setting control pin, active low level. 1k resistor connected in series inside the module |
| 6 | ANT | Input/Output | 433MHz antenna pin |
| 7 | GND | | Common ground |
| 8 | GND | | Common ground |
| 9 | NC | | No connection, used in mechanical fixing, compatible with HC-11 module pin position |
| ANT1 | ANT | Input/Output | IPEX20279-001E-03 antenna socket |
| ANT2 | ANT | Input/Output | 433MHz spring antenna solder eyelet |

Pins 1-6 each have two bonding pads, with the outer half-hole bonding pads intended for patch soldering. When the inner bonding pad ANT2 of Pin 6 is used for connection, the spring antenna can be soldered here by hand. The inner round-hole bonding pads of Pins 1-5 may then be used to solder a 2.54mm-spacing pin header that can be plugged into a PCB socket.

## Wireless Serial Port Transparent Transmission

### (1) Simple introduction of working principle




As shown in the above diagram, two HC-12 modules can be used in place of physical wiring to replace a wired half-duplex serial communications link carrying TTL level signals. The left device sends serial port data to the module, and after the RxD port of the left module receives the serial port data, it will automatically send the data over the air via radio wave. The right module receives the data, and restores the serial port data originally sent by the left device and sends it out TxD. It is the same from right to left. Only a half-duplex link is available between modules, as they can not receive and send data over the air at the same time.

### (2) Serial port transparent transmission

The HC-12 module has four serial port transparent transmission modes, expressed as FU1, FU2, FU3, and FU4. In operation, these modes hide all the details of wireless communications from attached devices. The factory default working mode of the system is FU3 full-speed mode, and in this mode the baud rate in this air is automatically adjusted according to baud rate that the serial port has been set to. The usable communication distance will be the farthest at the lowest baud rate. Different modes can not transmit data to each other, and the user should select the optimal mode according to practical circumstances.

The modules are usually operated in pairs, with data transmitted by means of a half-duplex link. For successful wireless transmission, the transparent transmission mode, serial port baud rate, and wireless communication channel of the two paired modules must be set the same. The factory default module setting are: FU3, 9,600bps (8N1: 8 data bits, no parity, 1 stop bit), CH001 (433.4MHz), 20dBm power (100mW).

The number of bytes that can be continuously sent to the serial port of the module is unlimited in modes FU1 and FU3. However, considering ambient interference and other factors, if thousands of data bytes are sent continuously, some number of bytes may be lost. Therefore, the attached devices at each end of the link should have some sort of response and resending mechanism to avoid information loss.

### (3) The four serial port transparent transmission modes

When the HC-12 module leaves the factory, its default serial port transparent transmission mode is FU3. In this mode the module remains in full-speed state, with an idle current of about 16mA. The module automatically adjusts the baud rate of wireless transmission in the air according to the serial port baud rate,

with the corresponding relationship as shown in the table below:

| Serial port baud rate | 1200 bps | 2400 bps | 4800 bps | 9600 bps | 19,200 bps | 38,400 bps | 57,600 bps | 115,200 bps |
|---|---|---|---|---|---|---|---|---|
| Baud rate in the air | 5000bps | | 15,000bps | | 58,000bps | | 236,000bps | |

To get the maximum communication distance, the serial port baud rate should be set to be low (1200bps or 2400bps). For short-time transmission of mass data, the serial port baud rate may be set high, but be aware that the communication distance will be reduced accordingly.

The receiving sensitivity of the module at different baud rates in the air is as shown in the table below:

| Baud rate in the air | 5000bps | 15,000bps | 58,000bps | 236,000bps |
|---|---|---|---|---|
| Wireless receiving sensitivity | -117dBm | -112dBm | -107dBm | -100dBm |

Generally, every time the receiving sensitivity is reduced by 6dB, the communication distance will be reduced by half.

When the "SET" pin of the module is pulled low, the serial port transparent transmission mode and other parameters can be set through AT commands (see the introduction in the following chapter for details).

FU1 mode is a moderate power saving mode, with an idle working current of about 3.6mA. In this mode, the module can also be set to any of the eight serial port baud rates shown in the above table, but the baud rate in the air is a uniform 250,000bps.

FU2 mode is an extreme power saving mode, with an idle working current of about 80uA. In this mode, the module only supports baud rates of 1200bps, 2400bps, and 4800bps, with the baud rate in the air uniform at 250,000bps. If the module is subsequently set to any other serial port baud rate, the module will not be able to conduct wireless communication normally.

When the module is set to FU2 mode, if the currently set baud rate exceeds 4800bps it will be automatically reduced to 4800bps. In FU2 mode, the sending time interval of data packets can not be too short, otherwise data will be lost. It is suggested that the sending time interval between data packets should be no less than 1 second.

FU4 mode is useful for maximum range, up to 1.8km. Only a single baud rate of 1200bps is supported, with the in the air baud rate reduced to 500bps for improved communication distance. This mode can only be used for small amounts of data (each packet should be 60 bytes or less), and the time interval between sending packets must not be too short (preferably no less than 2 seconds) in order to prevent loss of data.

The following table gives typical reference values for the various modes:

| Mode | FU1 | FU2 | FU3 | FU4 | Remarks |
|---|---|---|---|---|---|
| Idle current | 3.6mA | 80uA | 16mA | 16mA | Average value |
| Transmission time delay | 15-25mS | 500mS | 4-80mS | 1000mS | Sending one byte |
| Loopback test time delay 1 | 31mS | | | | Serial port baud rate 9600, sending one byte |

| | | | | | |
|---|---|---|---|---|---|
| Loopback test time delay 2 | 31mS | | | | Serial port baud rate 9600, sending 10 bytes |
| Operating range at full power (20dBm) | 100m | 100m | 600m at 9600bps<br>1000m at 2400bps | 1800m at 1200bps | Clear line of sight between modules under ideal conditions |

Note: Loopback test time delay means the round trip time taken for data that is sent to the input (RxD pin) of one module, to begin to emerge from the output (TxD pin) of the same module, where a second (remote) module has been configured with the TxD and RxD pins connected together.

**Module Parameter Setting AT Commands**

AT commands are used to set module parameters and switch between module functions when the module is in command mode. After being set, these changes will become valid only after exiting from command mode. Parameters are stored in onboard non-volatile flash memory, so will not be lost when power is removed.

**(1) Entering command mode**

There are two ways to enter command mode:

1. while energized, pull Pin 5 ("SET") low,
   wait 40ms for command mode to engage

2. disconnect the power supply,
   connect Pin 5 ("SET") to GND,
   re-energize the module

Note: pin 5 has a 10k pullup resistor connected internally, allowing the pin to be driven by an open-collector output from an attached device.

Either of the above two methods will place the module in command mode ready to accept AT commands; releasing pin 5 ("SET") in either case exits from command mode. If the module settings have changed after exiting from command mode, it will be switched to the new settings within 80ms.

When the second method (pin 5 "SET" tied to ground before power is applied), the module always enters command mode with the serial port configured for 9600bps, 8 data bits, no parity, 1 stop bit, irrespective of any previously configured settings.

**(2) Command instructions**

- AT

Test command. Send command "AT" to the module, and the module returns "OK".

- AT+Bxxxx

Change the serial port baud rate. The baud rate can be set to 1200bps, 2400bps, 4800bps, 9600bps, 19,200bps, 38,400bps, 57,600bps, or 115,200bps. The default value is 9600bps.

e.g: To set the serial port baud rate of the module to 19,200bps, send command "AT+B19200" to the module, and the module will return "OK+B19200". After exiting from command mode, the module will begin to communicate at 19,200bps.

- AT+Cxxx

Change wireless communication channel, selectable from 001 to 127 (for wireless channels exceeding 100, the communication distance cannot be guaranteed). The default value for the wireless channel is 001, with a working frequency of 433.4MHz. The channel stepping is 400KHz, and the working frequency of channel

100 is 473.0MHz.

e.g: To set the module to work on channel 21, send command "AT+C021" to the module, and the module will return "OK+C021". After exiting from command mode, the module will work on channel 21, with a working frequency of 441.4MHz.
Note: As the wireless receiving sensitivity of the HC-12 module is relatively high, when the serial port baud rate is greater than 9600bps five adjacent channels should be staggered for use. Even when the serial port baud rate is not greater than 9600bps, over short distances (less than 10m) also five adjacent channels should be staggered for use.

- AT+FUx

Change the serial port transparent transmission mode of the module. Four modes are available, namely FU1, FU2, FU3, and FU4. Only when the serial port speed, channel, and transparent transmission mode of two modules is set to be the same, can normal wireless communications occur. For more details, please see the above section "Wireless Serial Port Transparent Transmission".

e.g: Send command "AT+FU1" to the module, and the module returns "OK+FU1".

- AT+Px

Set the transmitting power of the module, with x selectable from 1 to 8. The corresponding transmitting power of the module is as shown below:

| x value | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Transmitting power of module | -1 dBm (0.8mW) | 2 dBm (1.6mW) | 5 dBm (3.2mW) | 8 dBm (6.3mW) | 11 dBm (12mW) | 14 dBm (25mW) | 17 dBm (50mW) | 20 dBm (100mW) |

The default value is 8, and the higher the transmitting power, the farther the possible wireless communication distance. When the transmitting power level is set to 1, the transmitting power is at the minimum. Generally speaking, every time the transmitting power is reduced by 6dB, the communication distance will be reduced by half.

e.g: Send command "AT+P5" to the module, and the module returns "OK+P5". After exiting from command mode, the transmitting power of the module will be set to 11dBm.

- AT+Ry

Obtain a single parameter from the module, where y is any letter among B, C, F, and P, respectively representing: baud rate, communication channel, serial port transparent transmission mode, and transmitting power.

Example 1:
Send command "AT+RB" to the module, and if the module returns "OK+B9600" it is confirmed that the serial port baud rate of the module is 9600bps.

Example 2:
Send command "AT+RC" to the module, and if the module returns "OK+RC001" it is confirmed that the communication channel of the module is 001.

Example 3:
Send command "AT+RF" to the module, and if the module returns "OK+FU3" it is confirmed that the module is working in serial port transparent transmission mode FU3.

Example 4:
Send command "AT+RP" to the module, and if the module returns "OK+RP:+20dBm" it is confirmed that the transmitting power of module is set to 20dBm (100mW).

- AT+RX

Obtain all parameters from the module. Returns serial port transparent

transmission mode, serial port baud rate, communication channel, and transmitting power in that order.

e.g: Send command "AT+RX" to the module, and the module returns "OK+FU3\r\n OK+B9600\r\n OK+C001\r\n OK+RP:+20dBm\r\n". ("\r\n" means return\newline)

- AT+Udps

Set data bits (d), parity (p), and stop bits (s) for serial port communication. For parity, N means none, O means odd check, and E means even check. For stop bits, 1 means one stop bit, 2 means two stop bits, and 3 means 1.5 stop bits.

e.g: To set the serial port format to eight data bits, odd parity, and one stop bit, send command "AT+U8O1" to the module. The module will return "OK+U8O1".

- AT+V

Request firmware version information from the module.

e.g: Send command "AT+V" to the module, and the module returns "HC-12_V2.3".

- AT+SLEEP

After receiving this command, the module will enter sleep mode upon exiting from command mode, with a working current of about 22uA. This mode doesn't allow serial port data transmission. Upon entering command mode again the module will exit from sleep mode automatically.

e.g: When wireless data transmission is not needed, to save power send command "AT+SLEEP" to the module, and the module will return "OK+SLEEP". Upon exit from command mode the working current will drop to about 22uA.

- AT+DEFAULT

Set serial port baud rate and configuration, communication channel, power, and serial port transparent transmission mode back to the factory default values.

e.g: Send command "AT+DEFAULT" to the module, and the module returns "OK+DEFAULT", with the factory default values restored. The factory default serial port baud rate is 9600bps, 8 data bits, no parity, 1 stop bit, communication channel is 001, transmitting power is 20dBm, and serial port transparent transmission mode is FU3.

- AT+UPDATE

Puts the module in the state of waiting for a software update. After receiving this command the module will not respond to any further AT commands until power has been cycled.

**Design Considerations**

- Do not connect a light-emitting diode and resistor directly to the module's TxD output as this may affect serial port communication.

- If using a PC or MCU to dynamically modify the module parameters, after pulling pin 5 ("SET") low wait at least 40ms before sending any AT commands to the module. After releasing pin 5 ("SET"), wait at least 80ms for the module to return to serial port pass-through mode.

- The HC-12 may require up to 100mA of current when transmitting. Ensure sufficient current is available – a USB bridge device may not be able to supply sufficient current. It is recommended that a reservoir capacitor be provided across the power supply of at least 22uF, preferably 1000uF.

edited by Robert Rozee, 15 January 2016