PS2 handle working Principle

1-2 PS2 module and receiver

1. Introduction of PS2 controller and MCU communication:

The PS2 consists of two parts: the handle and the receiver. The handle is mainly responsible for sending key information. When the power is turned on and the handle switch is turned on, the handle and the receiver are automatically paired.

When the pairing is not successful, the green light of the receiver flashes, and the light on the handle also flashes.

After the pairing is successful, the green light on the receiver is light on. the light on the handle also light on. You can press the “MODE” button to select the handle transmission mode.

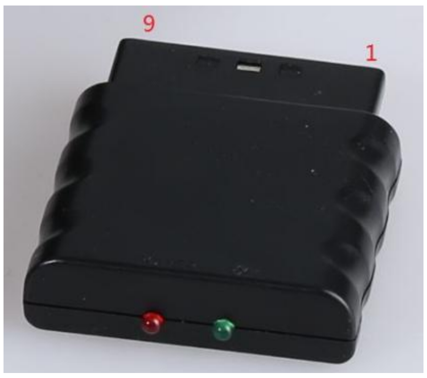
Red light mode: analog output value of the joystick;

Green light mode: The remote lever corresponds to the above four buttons, and only four extreme directions correspond.

The receiver is connected to the host (single chip microcomputer) to realize communication between the host and the handle.

1	2	3	4	5	6	7	8	9
DI/DAT	DO/CMD	NC	GND	VDD	CS/SEL	CLK	NC	ACK

Table 1: receiver pin output



3-1 receiver

DI/DAT: Signal flow direction, from the handle to the host, this signal is an 8-bit serial data that is transmitted synchronously in the falling edge of clock. The reading of the signal is done during the course of the clock from high to low.
DO/CMD: Signal flow direction, from the host to the handle. This signal is opposite to DI. The signal is an 8-bit serial data that is synchronously transmitted on the falling edge of the clock.
NC: empty port.
GND: power ground.
VDD: receiver working power supply, power supply range 3~5V;
CS/SEL: Used to provide the handle trigger signal. During communication, at a low level;
CLK: clock signal, sent by the host to keep data synchronized;
NC: empty port;
ACK: The response signal from the handle to the host.

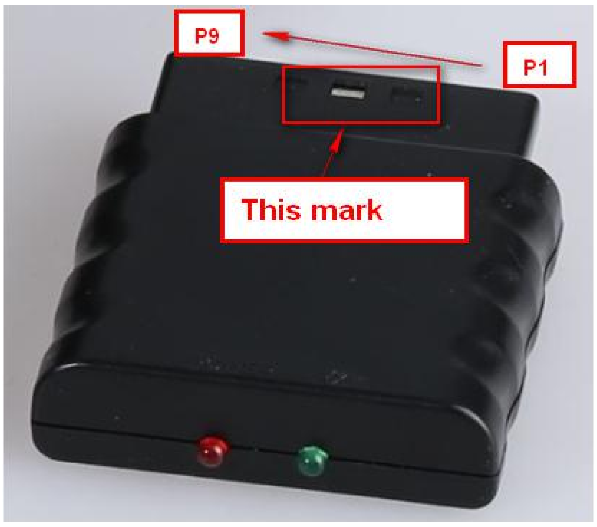
order	MDO	MDI	Bit0、Bit1、Bit2、Bit3、Bit4、Bit5、Bit6、Bit7、
0	0X01	idle	
1	0x42	ID	
2	idle	0x5A	
3	idle	data	SELECT、L3、R3、START、UP、RIGHT、DOWN、LEFT
4	idle	data	L2、R2、L1、R1、△、○、╳、□
5	idle	data	PSS_RX（0x00=left、0xFF=right）
6	idle	data	PSS_RY（0x00=up、0xFF=down）
7	idle	data	PSS_LX（0x00=left、0xFF=right）
8	idle	data	PSS_LY（0x00=up、0xFF=down）

Table 3-1 Data meaning comparison table

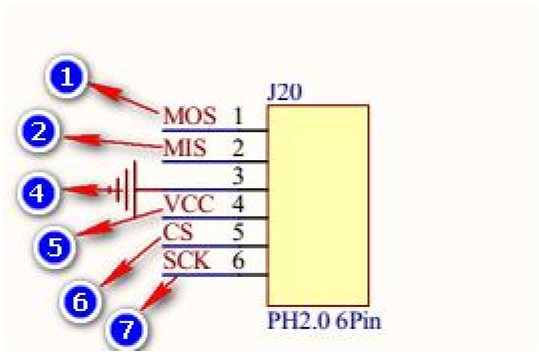
When a button is pressed, the corresponding bit is "0" and the other bits are "1", for example, when the key "SELECT" is pressed, Data[3]=11111110B.

Red light mode: the left and right joysticks send analog values, between 0x00 and 0xFF, and the key values of the joystick are pressed L3, R3 are effect;
Green light mode: the analog values of the left and right joysticks are invalid.

When pushed to the limit, the corresponding UP、RIGHT、DOWN、LEFT、△、○、╳、□ buttons L3, R3 are invalid.

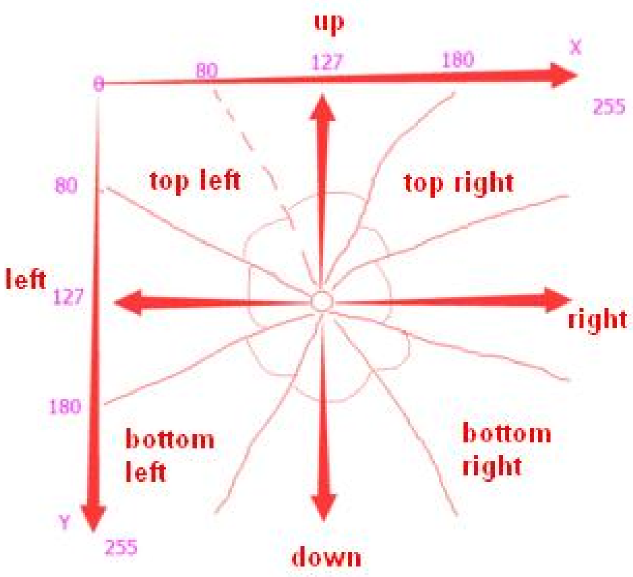


3-2-1 Receiver pins



3-2-2 Receiver pins

The joystick analog value corresponds to the XY coordinate map:



2. Principle of SPI communication

Let's briefly explain the SPI timing:

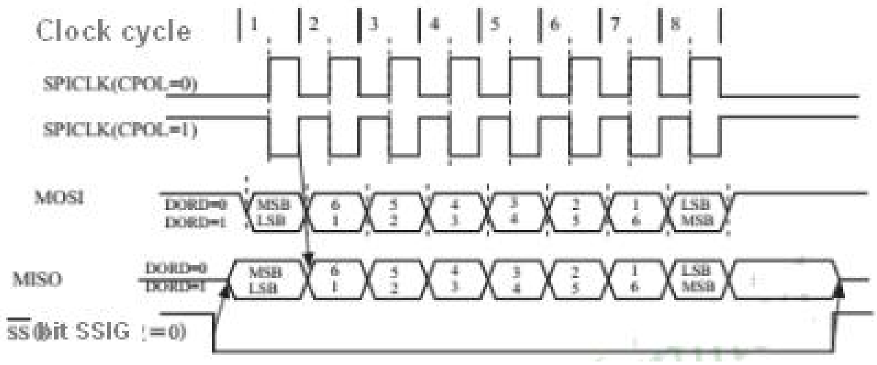
The SPI bus is a three-wire synchronous interface introduced by Motorola. It communicates synchronously in 3-wire mode: one clock line SCK, one data input line MOSI, and one data output line MISO; it is used for full duplex of CPU and various peripheral devices, synchronous serial communication.

The main features of SPI:

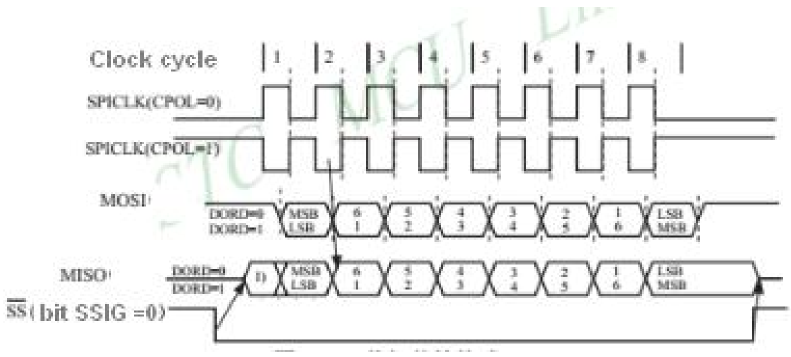
1) it can send and receive serial data at the same time;
2) it can be used as master or slave;
3) provide frequency programmable clock;
4) send end interrupt flag;
5) write conflict protection;
6) bus competition protection.

The SPI bus has four modes of operation (SP0, SP1, SP2, SP3), the most widely used of which are SPI0 and SPI3. In order to exchange data with peripherals, the SPI module can configure the output serial synchronous clock polarity and phase according to the peripheral operation requirements. The clock polarity (CPOL) has no significant impact on the transmission protocol.

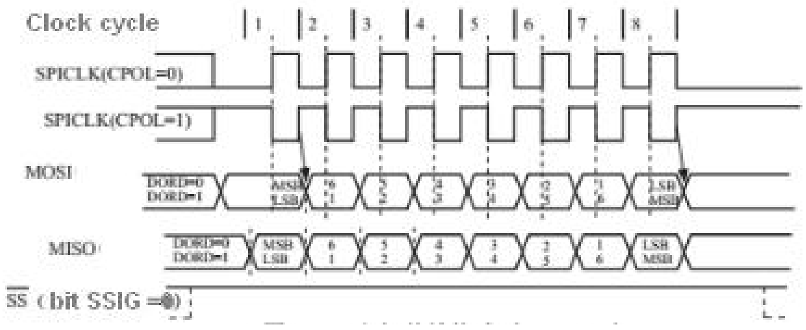
When CPOL=0, the idle state of the serial synchronous clock is low;
When CPOL = 1, the idle state of the serial synchronous clock is high.
The Clock Phase (CPHA) can be configured to select one of two different transmission protocols for data transmission:
When CPHA=0, the data is sampled on the first edge of the serial synchronous clock (rising or falling);
When CPHA=1, the data on the second edge of the serial synchronous clock (rising or falling) is sampling.



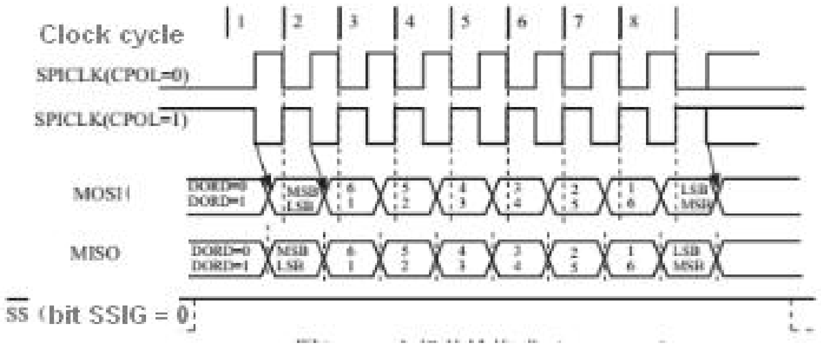
3-4 SPI slave transfer format（CPHA=0）



3-5 SPI slave transfer format（CPHA=1)

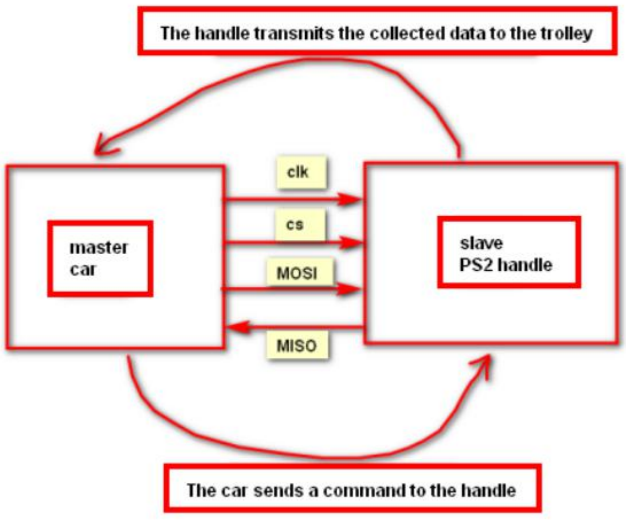


3-6 SPI master transfer format（CPHA=0）

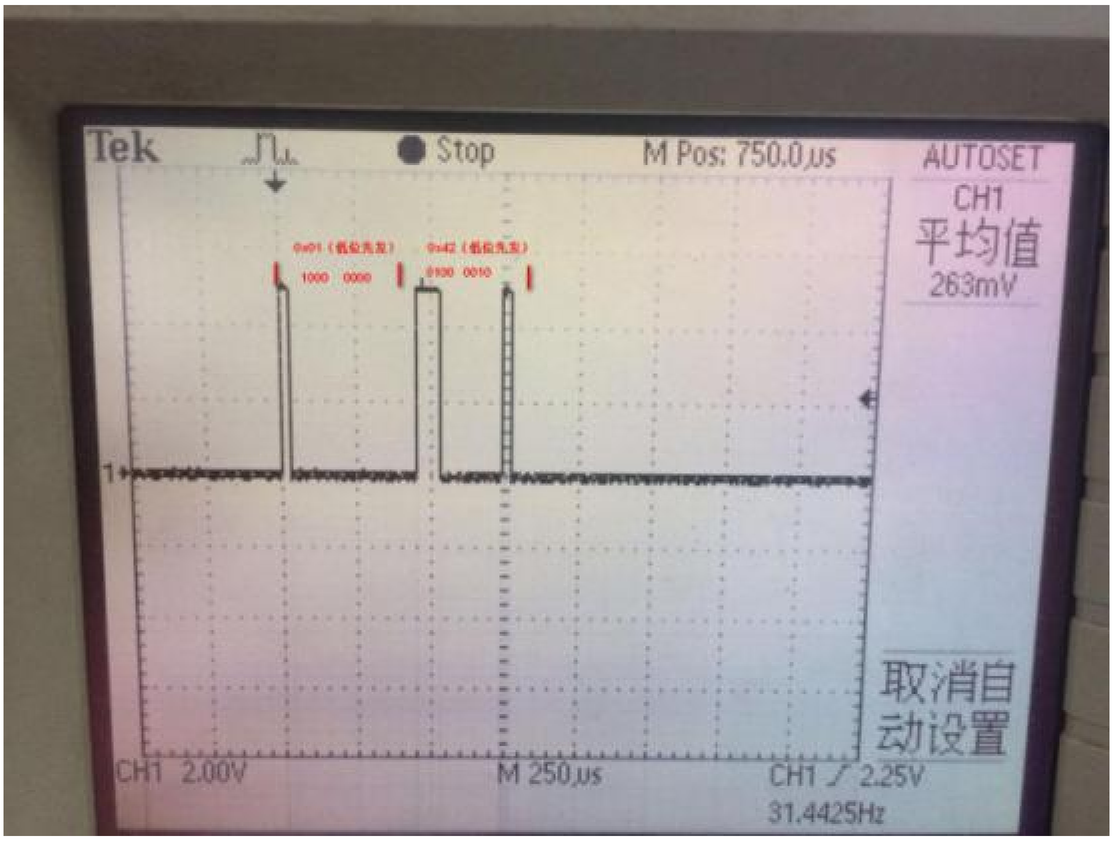


3-7 SPI master transfer format（CPHA=1）

Some the knowledge of the PS2 handle control car:


The handle transmits the collected data to the trolley
clk
cs
MOSI
MISO
master car
slave PS2 handle
The car sends a command to the handle

Next, we briefly analyze the communication process of spi:
The first is that we need to pull the chip select line low,it is effective for the communication between the master and slave devices. Next, we input the 8-bit level signal of 0x01 (first transfer low bit) to the MOSI command line. Then send an 8-bit level signal of 0X42 (first transfer low bit).
The waveform is as follows:
1000 0000 0100 0010

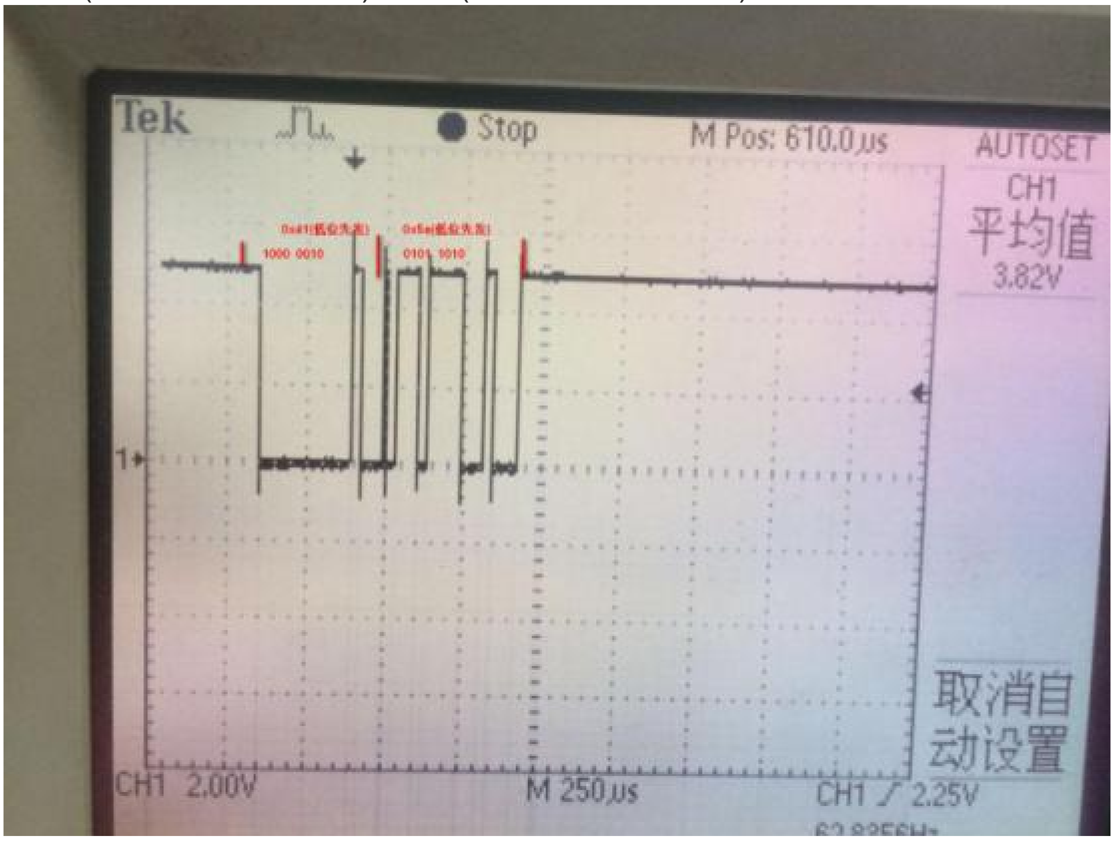


At the same time, the MISO(PS2 handle) will return the data information of the data button after receiving the command:
The first thing that will be sent to the car is ID "0x41=analog green light, 0x73=analog red light", then send 0x5a, indicating that the data is coming, please car start receiving the button information returned by the PS2 handle:

Green light mode:

1000 0010 0101 1010

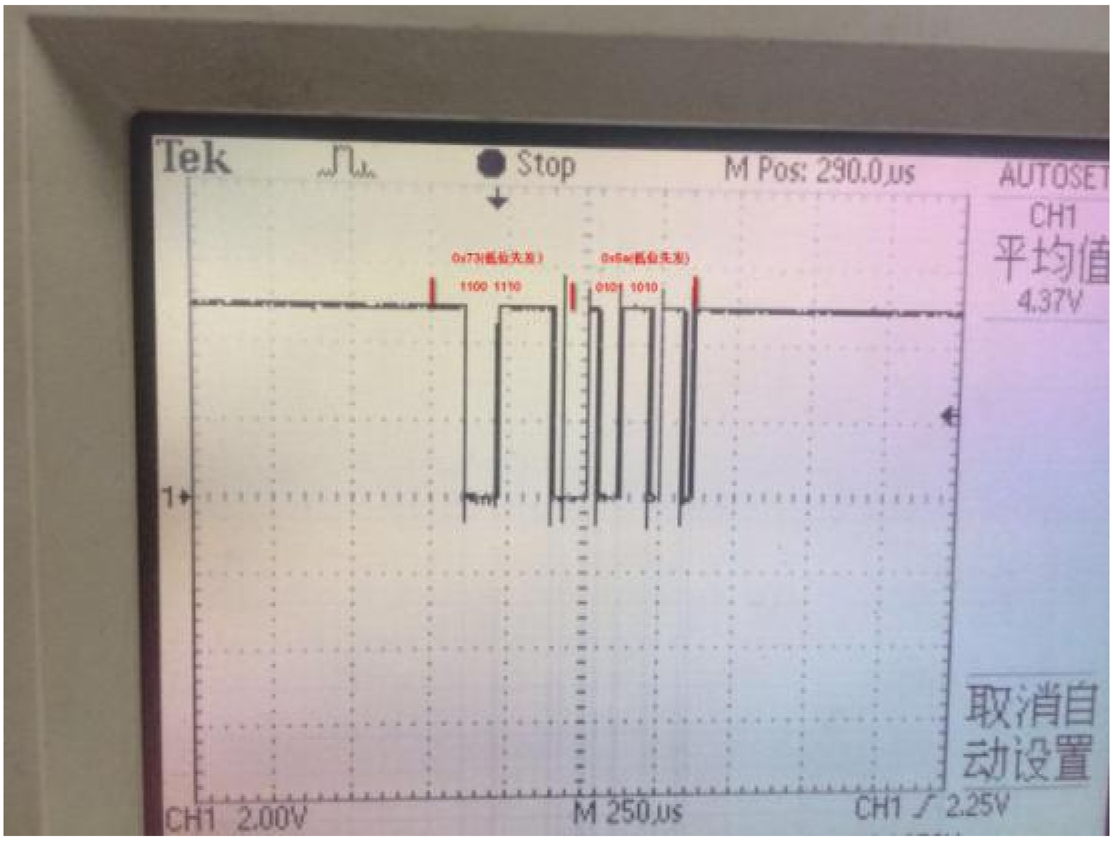
0x41 (first transfer low bit) 0x5a (first transfer low bit)



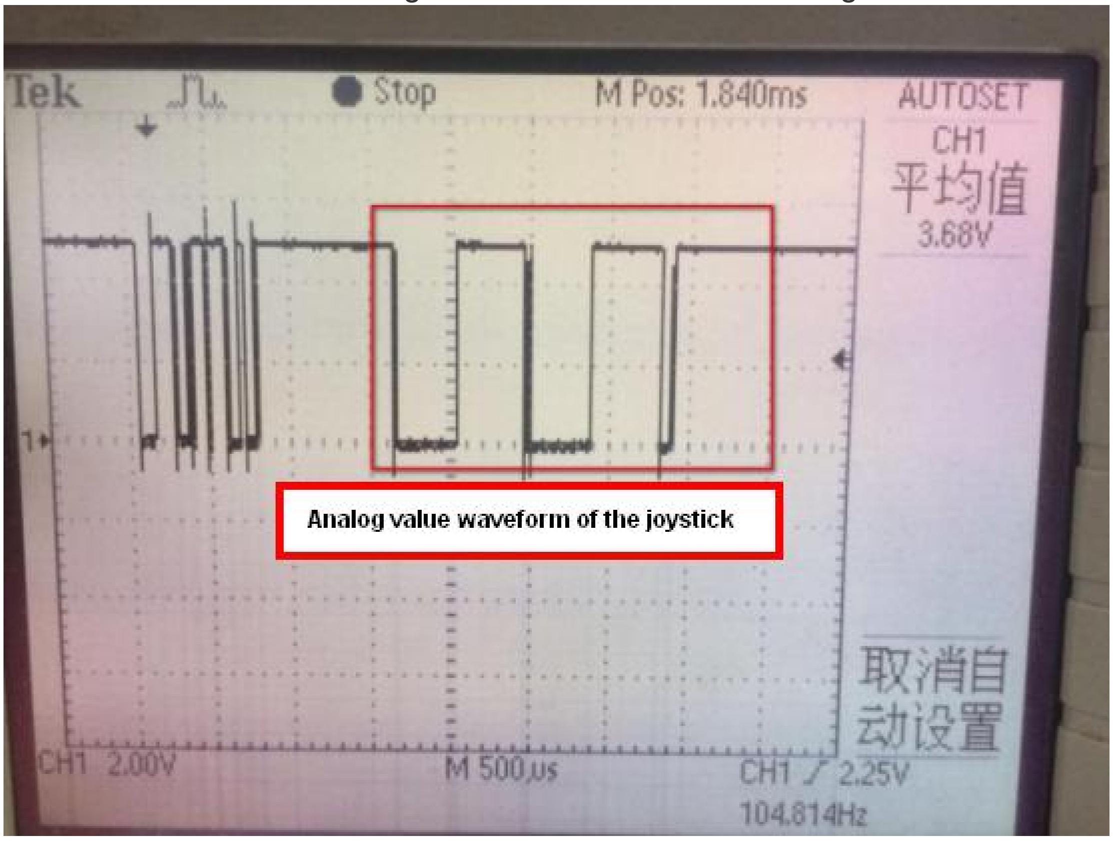
Red light mode:

1100 1110 0101 1010

0x73 (first transfer low bitl) 0x5a (first transfer low bit)

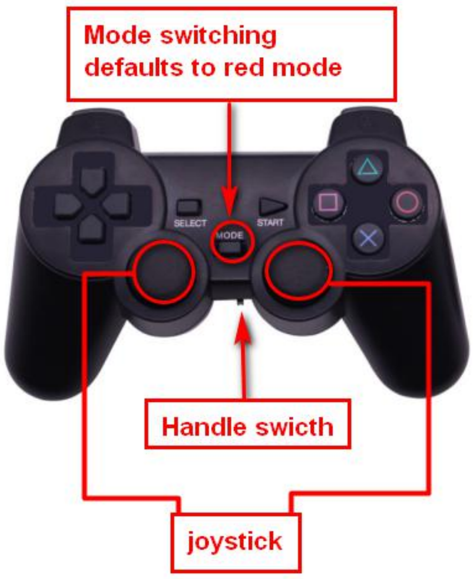


If we don't press the button, the analog value of the joystick can be read after the waveform in the above figure. This is because the red light mode is on.



Every time we change the level of the MOSI line, we need CLK to adjust, only when the rising edge comes, the level of MOSI will be read.

Handle key definition:



The function of each control button can be defined in the program by yourself. We will provide the Raspberry Pi code and Arduino code with the PS2 handle module.