USB HOST SHIELD EB0009

The USB equipment is consisted of two equipments: HOST & SLAVE. The data transmission won't succeed until a set of HOST connect with a SLAVE. In a nutshell, if a digital equipment is USB HOST supported, it can obtain data from another USB equipment. The serial communication of UNO and NANO is supported by those smart phones with Android 4.0 system or above. Therefore, through OTG connecting wire and applying the software, you can read and send data directly. The baud rate of Arduino serial port must be 9600!

OTG wire enable the Android smart phone to power up Arduino, so the external power source is no longer needed.

The Android smart phone needs to download the Arduino Uno Communicate in the Google e-market. (Download things from the Google play store is assuring without accessing to Trojan horse virus. However, downloading from other market is hard to judge.)

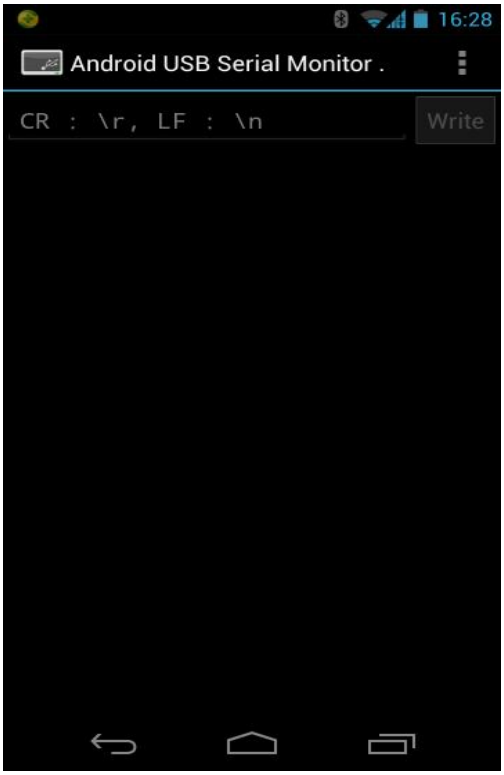
When insert the OTG wire into the Arduino, it will display:

The computational result after inserting into the Arduino:

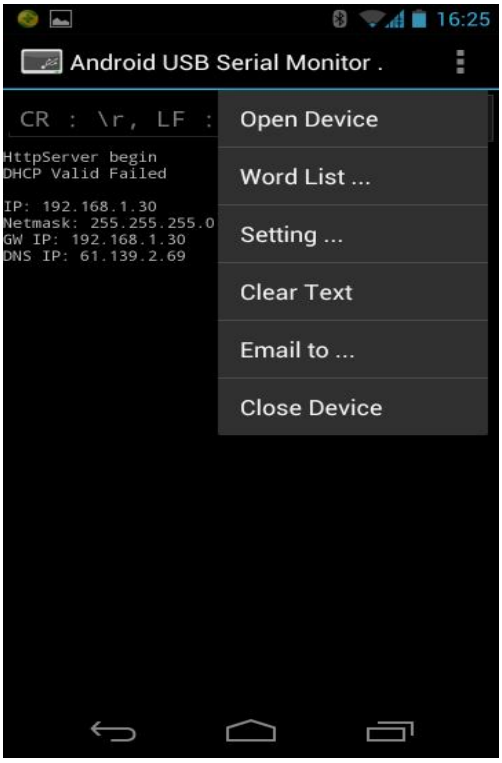
Now, we can check the Arduino serial output, but we can't data-in. Besides, the "Enter" has turned into a "dot". However, checking the procedure's work log is enough. So, is there any better software? Yes, there is.

USB Serial Monitor Lite from the Google E-Market:

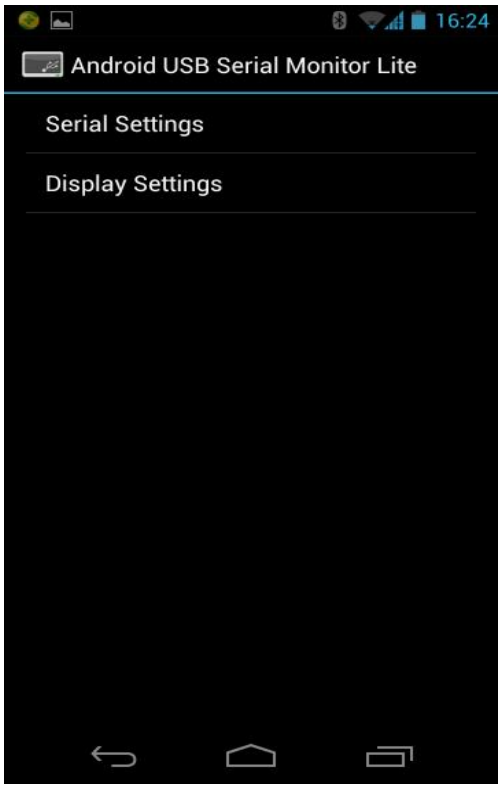
Main interface of the procedure:



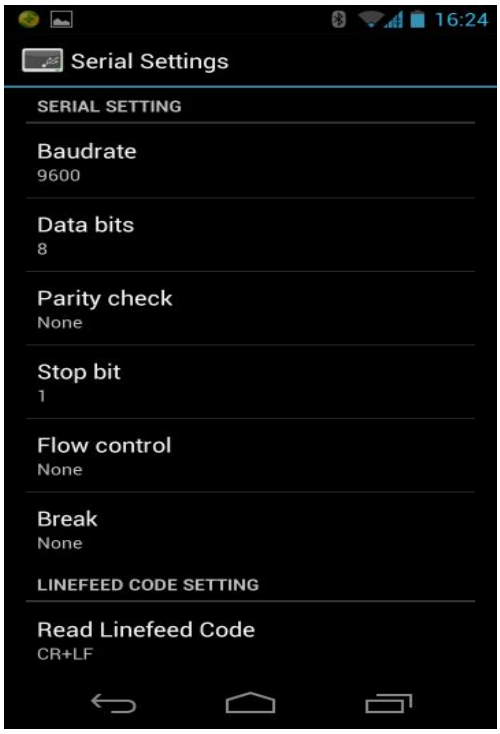
Setting: Setup menu. Choose “setting”



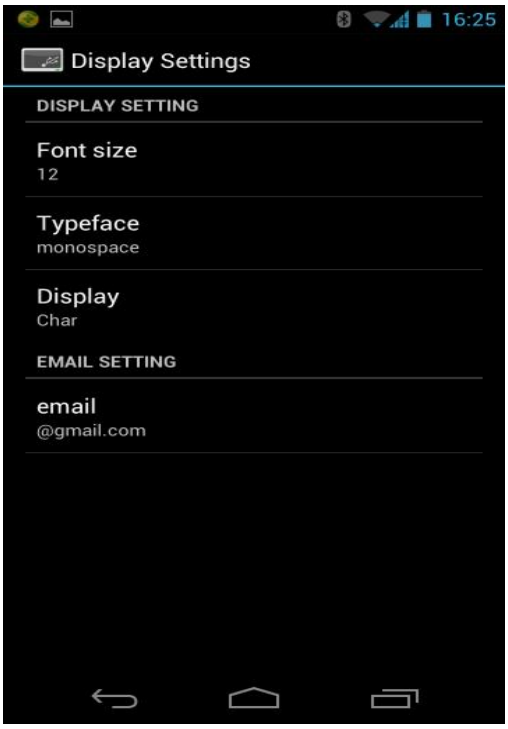
Setup menu, there are serial settings and display settings.



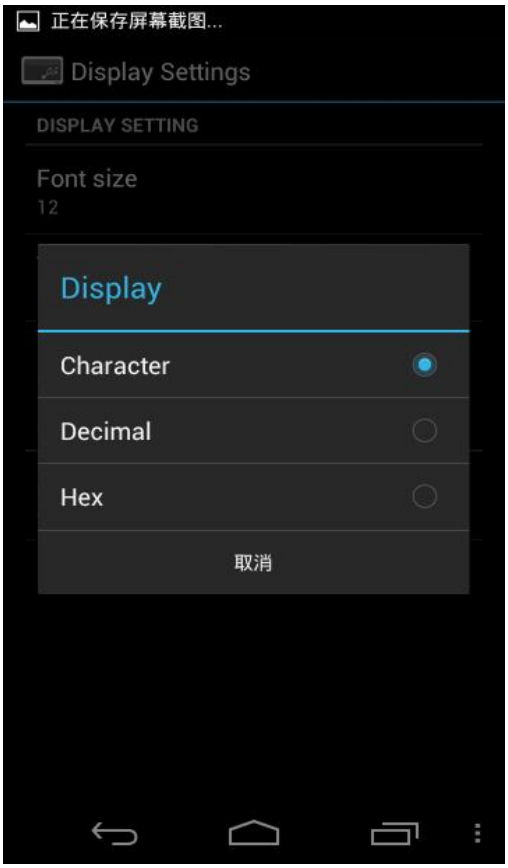
Set the serial baud rate to 9600:



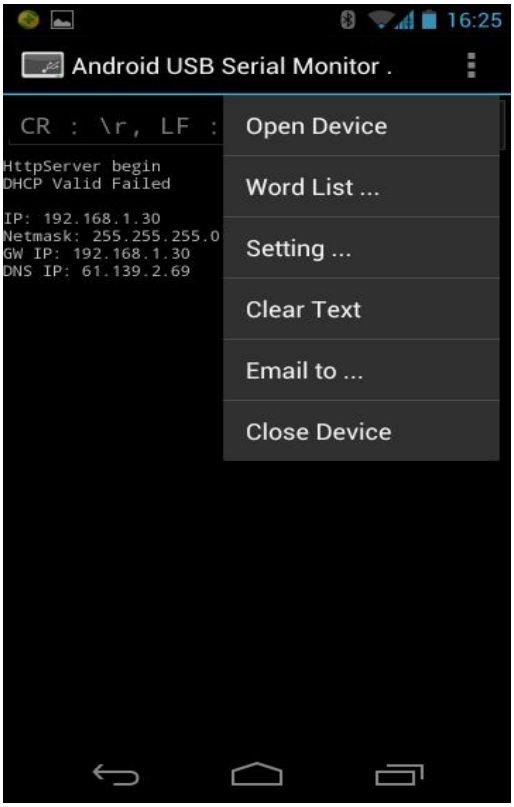
Display can be default or set by yourself:



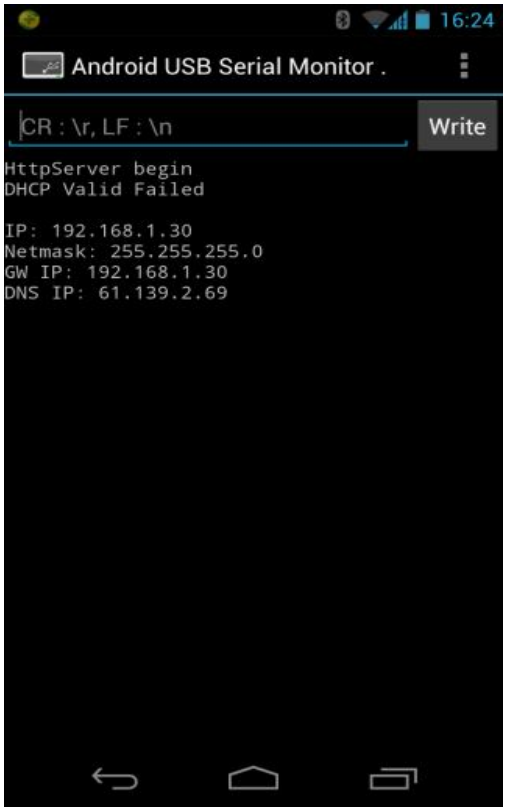
Do you need display “characters”, “decimal” or “hex”?



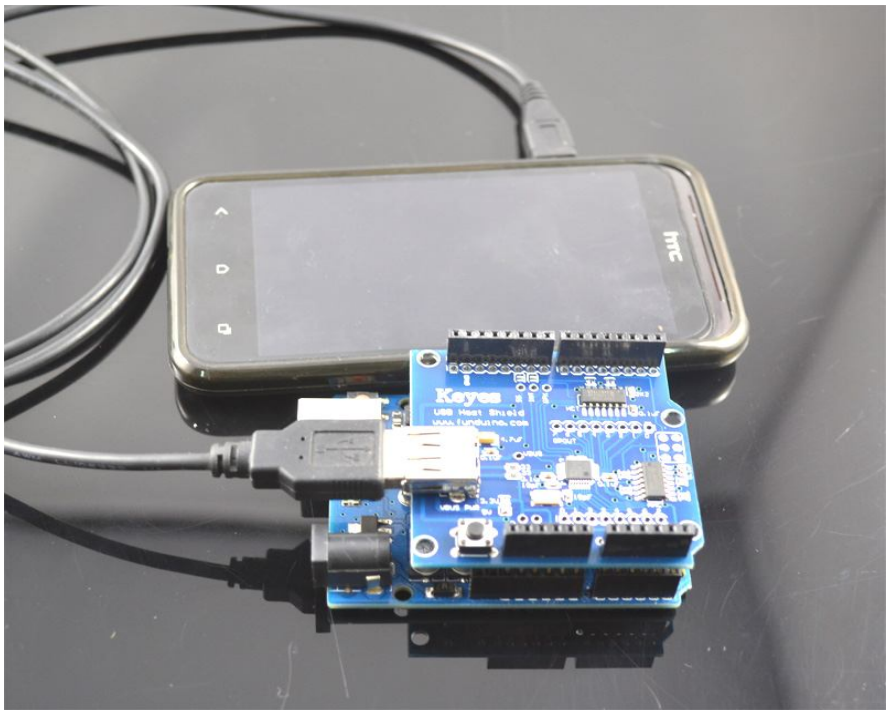
Open Device. After setting and connecting the Arduino, select “Open Device”.



Computational result:



After connecting USB HOST and cell phone:



Procedure code:

```
#include <Wire.h>
#include <Servo.h>
#include <Max3421e.h>
#include <Usb.h>
#include <AndroidAccessory.h>
#define   LED1_RED          3
#define   LED1_GREEN        5
#define   LED1_BLUE         6
#define   SERVO1            2
#define   RELAY1            4
#define   LIGHT_SENSOR      A2
#define   TEMP_SENSOR       A3
#define   BUTTON1           A4
#define   BUTTON2           A5
#define   JOYSTICK_X        A0
#define   JOYSTICK_Y        A2

AndroidAccessory acc("Google, Inc.",
                     "DemoKit",
                     "DemoKit Arduino Board",
                     "1.0",
                     "http://www.android.com",
                     "0000000012345678");
Servo servos[1];

void setup();
void loop();

void init_buttons()
```

```
{
	pinMode(BUTTON1, INPUT);
	pinMode(BUTTON2, INPUT);
	// enable the internal pullups
	digitalWrite(BUTTON1, HIGH);
	digitalWrite(BUTTON2, HIGH);
}

void init_relays()
{
	pinMode(RELAY1, OUTPUT);
}

void init_leds()
{
	pinMode(LED1_RED, OUTPUT);
	pinMode(LED1_GREEN, OUTPUT);
	pinMode(LED1_BLUE, OUTPUT);
	analogWrite(LED1_RED, 0);
	analogWrite(LED1_GREEN, 0);
	analogWrite(LED1_BLUE, 0);
}
byte b1, b2, c;
void setup()
{
	Serial.begin(115200);
	Serial.print("\r\nStart");
	init_leds();
	init_relays();
	init_buttons();
	servos[0].attach(SERVO1);
	servos[0].write(90);
	b1 = digitalRead(BUTTON1);
	b2 = digitalRead(BUTTON2);
	c = 0;
	acc.powerOn();
}

void loop()
{
	byte err;
	byte idle;
	static byte count = 0;
	byte msg[3];
	long touchcount;
	if (acc.isConnected()) {
		int len = acc.read(msg, sizeof(msg), 1);
		int i;
		byte b;
		uint16_t val;
		int x, y;
		char c0;
```

```
if (len > 0) {
	// assumes only one command per packet
	if (msg[0] == 0x2) {
		if (msg[1] == 0x0)
			analogWrite(LED1_RED, msg[2]);
		else if (msg[1] == 0x1)
			analogWrite(LED1_GREEN, msg[2]);
		else if (msg[1] == 0x2)
			analogWrite(LED1_BLUE, msg[2]);
		else if (msg[1] == 0x10)
			servos[0].write(map(msg[2], 0, 255, 0, 180));
	} else if (msg[0] == 0x3) {
		if (msg[1] == 0x0)
			digitalWrite(RELAY1, msg[2] ? HIGH : LOW);
	}
}
msg[0] = 0x1;
b = digitalRead(BUTTON1);
if (b != b1) {
	msg[1] = 0;
	msg[2] = b ? 0 : 1;
	acc.write(msg, 3);
	b1 = b;
}
b = digitalRead(BUTTON2);
if (b != b2) {
	msg[1] = 1;
	msg[2] = b ? 0 : 1;
	acc.write(msg, 3);
	b2 = b;
}
switch (count++ % 0x10) {
case 0:
	val = analogRead(TEMP_SENSOR);
	msg[0] = 0x4;
	msg[1] = val >> 8;
	msg[2] = val & 0xff;
	acc.write(msg, 3);
	break;
case 0x4:
	val = analogRead(LIGHT_SENSOR);
	msg[0] = 0x5;
	msg[1] = val >> 8;
	msg[2] = val & 0xff;
	acc.write(msg, 3);
	break;
case 0x8:
	read_joystick(&x, &y);
	msg[0] = 0x6;
	msg[1] = constrain(x, -128, 127);
	msg[2] = constrain(y, -128, 127);
	acc.write(msg, 3);
```

```
                    break;
                }
        } else {
                // reset outputs to default values on disconnect
                analogWrite(LED1_RED, 255);
                analogWrite(LED1_GREEN, 255);
                analogWrite(LED1_BLUE, 255);
                servos[0].write(90);
                digitalWrite(RELAY1, LOW);
        }
        delay(10);
}

void read_joystick(int *x, int *y)
{
        *x = analogRead(JOYSTICK_X);
        *y = analogRead(JOYSTICK_Y);
}       // End
```