DIGITAL CURRENT SENSOR APPLICATION NOTE

APPLICATION EXAMPLE ON ARDUINO

Winson reserves the right to make changes to improve reliability or manufacturability.

Digital Current Sensor Application Note

1. Communication Interface Format

Interface	UART TTL
Rate	9600 bps
Format	Parity bit: None , Data bit: 8 , Stop bit: 1

Interface	I2C
Rate	Standard Mode (100KHz)

2. Operating Mode

(2.1) Continuous mode: Transmit current data continuously. Reset need to pull low the RST pin to GND.

(2.2) AT Command mode: Measure current data and reset according to the command (software reset).

Setting Command	Command	Example	Return Parameter
Reset Current	AT+RST\r\n	"AT+RST\r\n"	"OK\r\n"(1)
0: DC 1: AC	AT+CURR\r\n	"AT+CURR,0\r\n" "AT+CURR,1\r\n"	"OK\r\n"(1)
Measure Current	AT+MEAS\r\n	"AT+MEAS\r\n"	<Current>
Measure Temperature	AT+TEMP\r\n	"AT+TEMP\r\n"	<Temperature>
Measure 120 Points of Current Data	AT+BUF\r\n	"AT+BUF\r\n"	<I1,I2,I3,…\r\n>
Change Baud Rate	AT+BR\r\n	"AT+BR,1\r\n"(2)	"OK\r\n"(1)

1. Command is error : return "Err\r\n" 。
2. Baud Rate (1: 9600, 2: 19200, 3: 38400, 4: 57600, 5: 125000 (bit/s))

(2.3) Modbus-RTU: Use device address for control and respond to commands.

Item	Address	Byte	R/W	Description
Reset	0x0000	2	Write	Write 0x0100 to Reset
Measuring Flag Data Valid Flag	0x0001	2	Write/ Read	**Write:** 0x0002: Measuring flag set **Read:** 0x0000: Measuring flag reset, data flag is invalid 0x0001: Measuring flag reset, data flag is valid 0x0002: Measuring flag set, data flag is invalid 0x0003: Measuring flag set, data flag is valid
Current	0x0002	4	Read	32-bit signed integers (Int32), Unit:0.001A Current= Int32/1000 (A)
Temperature	0x0004	4	Read	32-bit signed integers (Int32), Unit:0.1°C Temperature= Int32/10 (°C)
Slave Address	0x0010	2	Write	Default address: 1, Write address1~247
Baud Rate	0x0011	2	Write	Default: 1 (Baud Rate = 9600 bit/s) 0x0001: Baud Rate = 9600 bit/s 0x0002: Baud Rate = 19200 bit/s 0x0003: Baud Rate = 38400 bit/s 0x0004: Baud Rate = 57600 bit/s 0x0005: Baud Rate = 125000 bit/s
DC/AC	0x0020	2	Write	0: DC 1: AC

(2.4) I2C: Use device address for control and respond to commands.

Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Function	Range
00h	0	0	0	0	0	0	0	Auto	Control	0-1
01h	0	Address							Slave Address	-
02h	0	0	0	0	0	0	0	Reset	Current Reset	0-1
03h	0	0	0	0	0	0	Status	Valid	Status	0-3
04h~07h	Temperature								Temperature	-
08h~0Bh	+/-Current								DC Current	-
0Ch~0Fh	~Current								AC Current	-

Winson reserves the right to make changes to improve reliability or manufacturability.

3. Measurement Method (Continuous Mode)

(3.1) **AC measurement:** After power-on, the sensor will automatically reset the current value when no current passes through the sensor and the current value can be also manually reset to zero. Data update rate is 5 Data /sec.

(3.2) **DC measurement:** The residual magnetism of the sensor could affect the measurement accuracy. **When first use or switching the measurement direction, it is recommended to provide the test current first, and then reset the sensor when zero current pass.** Data update rate is 5 Data /sec.

(3.3) When measuring DC current, the sensor will generate an amount of remanence. If this remanence cause reading error, please re-reset it.

(3.4) The proper use of reset function will make the measurement more accurate.

(3.5) Current Data Output (Continuous):
Simultaneously measuring the AC and DC current signals, If the measured data is +DC “1.23”A and AC “1.23”A, then the output data is ‘**+**’, ‘1’, ‘**.**’, ‘2’, ‘3’, ‘0’, ‘**,**’, ‘**~**’, ‘1’, ‘**.**’, ‘2’, ‘3’, ‘0’, ‘\r’, ‘\n’, total of 15 bytes.

(3.6) Current Data Output (AT Command Version):

- If the measured data is AC “1.23”A, then the output data is ‘~’, ‘1’, ‘.’, ‘2’, ‘3’, ‘0’, ‘\r’, ‘\n’, total of 8 bytes. If the measured data is “10.45” A, then the output data is ‘~’, ‘1’, ‘0’, ‘.’, ‘4, ‘5, ‘\r’, ‘\n’, total of 8 bytes.
- If the measured data is +DC “1.23”A, then the output data is ‘+’, ‘1’, ‘.’, ‘2’, ‘3’, ‘0’, ‘\r’, ‘\n’, total of 8 bytes. If the measured data is -DC “1.23”A, then the output data is ‘-’, ‘1’, ‘.’, ‘2’, ‘3’, ‘0’, ‘\r’, ‘\n’, total of 8 bytes.

(3.7) Temperature Data Output (AT Command Version):

If the measured data is 25.5˚C, then the output data is ‘2’, ‘5’, ‘.’, ‘5’, ‘\r’, ‘\n’, total of 6 bytes. If the measured data is 5.0˚C, then the output data is ‘5’, ‘.’, ‘0’, ‘\r’, ‘\n’, total of 5 bytes. If the measured data is -10.0˚C, then the output data is ‘-’, ‘1’, ‘0’, ‘.’, ‘0’, ‘\r’, ‘\n’, total of 7 bytes.

(3.8) Measure 120 Points of Current Data Output (AT Command Version):

The output data is "+1.234, +1.233, +10.23, +10.24, -1.234, -1.233…..\r\n", total of 120 bytes.

Winson reserves the right to make changes to improve reliability or manufacturability.

4. Measuring Method (Modbus-RTU)

(4.1) Read Holding Registers (Function code:03H)

※This function cannot be used in broadcast mode (0x00).

(4.1.1) Measuring Flag and Data Valid Flag

Master request: 01 03 00 01 00 01 D5 CA

Slave Address	Function Code	Start Address	No. of Registers	Check Code (CRC)
01H	03H	00H , 01H	00H , 01H	D5H, CAH

Slave response: 01 03 02 00 03 F8 45

Slave Address	Function Code	Byte Count	Data (2 Bytes)	Check Code (CRC)
01H	03H	02H	**00H , 03H**	F8H, 45H

Result: **(01)** sensor number 1,

(00 00): Measuring flag reset, data flag is invalid

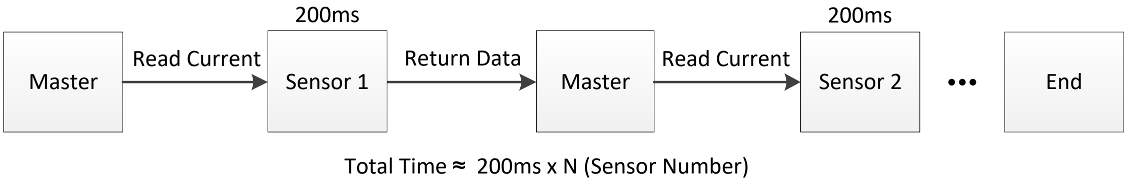
(00 01): Measuring flag reset, data flag is valid

(00 02): Measuring flag set, data flag is invalid

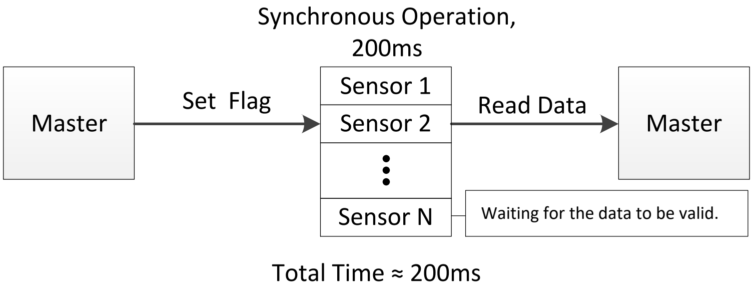
(00 03): Measuring flag set, data flag is valid

1. Measure Method

General measurement methods:



Use measurement flags:



Winson reserves the right to make changes to improve reliability or manufacturability.

(4.1.2) Current

Master request: 01 03 00 02 00 02 65 CB

Slave Address	Function Code	Start Address	No. of Registers	Check Code (CRC)
01H	03H	00H , 02H	00H , 02H	65H, CBH

Slave response: 01 03 04 00 00 04 D2 78 AE

Slave Address	Function Code	Byte Count	Data	Check Code (CRC)
01H	03H	04H	**00H , 00H , 04H , D2H**	78H, AEH

Result: **(01)** sensor number 1, **(00 00 04 D2)** current=1234/1000 = 1.234A

(4.1.3) Temperature

Master request: 01 03 00 04 00 02 85 CA

Slave Address	Function Code	Start Address	No. of Registers	Check Code (CRC)
01H	03H	00H , 04H	00H , 02H	85H, CAH

Slave response: 01 03 04 00 00 01 2C FA 7E

Slave Address	Function Code	Byte Count	Data	Check Code (CRC)
01H	03H	04H	**00H , 00H , 01H , 2CH**	FAH, 7EH

Result: **(01)** sensor number 1, **(00 00 01 2C)** temperature=300/10 = 30.0°C

(4.2) Write Holding Registers (Function code:06H)

※Broadcast mode (0x00) will not respond any value or error code.

(4.2.1) Reset

Master request: 01 06 00 00 01 00 88 5A

Slave response: 01 06 00 00 01 00 88 5A

Slave Address	Function Code	Start Address	Data	Check Code (CRC)
01H	06H	00H , 00H	**01H , 00H**	88H, 5AH

Result: **(01)** sensor number 1, **(01 00)** write 256 to **reset**

(4.2.2) Measuring Flag and Data Valid Flag

Master request: 01 06 00 01 00 02 59 CB

Slave response: 01 06 00 01 00 02 59 CB

Slave Address	Function Code	Start Address	Data	Check Code (CRC)
01H	06H	00H , 01H	**00H , 02H**	59H, CBH

Result: **(01)** sensor number 1, **(00 02)** write 2 to set measuring flag

(4.2.3) Write Address

Master request: 01 06 00 10 00 01 49 CF

Slave response: 01 06 00 10 00 01 49 CF

Slave Address	Function Code	Start Address	Data	Check Code (CRC)
01H	06H	00H , 10H	**00H, 01H**	49H, CFH

Result: **(01)** sensor number 1, default address 1, **(00 01)** write **address** 1

(4.2.4) Change Baud Rate

Master request: 01 06 00 11 00 01 18 0F

Slave response: 01 06 00 11 00 01 18 0F

Slave Address	Function Code	Start Address	Data	Check Code (CRC)
01H	06H	00H , 11H	**00H, 01H**	18H, 0FH

Result: **(01)** sensor number 1, default 1, **(00 01)** change baud rate to 9600bit/s

(00 01): 9600, (00 02): 19200, (00 03): 38400, (00 04): 57600, (00 05): 125000 (bit/s)

(4.2.5) Set Measurement Method (AC / DC)

Master request: 01 06 00 20 00 01 49 C0

Slave response: 01 06 00 20 00 01 49 C0

Slave Address	Function Code	Start Address	Data	Check Code (CRC)
01H	06H	00H , 20H	**00H, 01H**	49H, C0H

Result: **(01)** sensor number 1, set measurement method to AC **(00 01)** /DC **(00 00)**.

Winson reserves the right to make changes to improve reliability or manufacturability.

(4.3) Exception Code

(4.3.1) Function Code Exception

Master request: 01 01 00 00 00 00 3C 0A

Slave Address	Function Code	Start Address	No. of Registers	Check Code (CRC)
01H	01H	00H , 00H	00H , 00H	3CH, 0AH

Slave response: 01 81 01 81 90

Slave Address	Function Code	Exception Code	Check Code (CRC)
01H	**81H**	**01H**	81H, 90H

Result: **(01)** sensor number 1, **(81)**=0X80(exception) + 0X01(function code), **(01)** Exception Code

(4.3.2) Address Exception

Master request: 01 03 FF FF 00 04 44 2D

Slave response: **01 83 02** C0 F1

Result: **(01)** sensor number 1, **(83)**=0X80(exception) + 0X03(function code), **(02)**Exception Code

(4.3.3) Data Exception

Master request: 01 03 00 00 FF FF 44 7A

Slave response: **01 83 03** 01 31

Result: **(01)** sensor number 1, **(83)**=0X80(exception) + 0X03(function code), **(03)**Exception Code

(4.3.4) Slave Device Busy

Master request: 01 03 00 01 00 01 D5 CA

Slave response: **01 83 06** C1 32

Result: **(01)** sensor number 1, **(83)**=0X80(exception) + 0X03(function code), **(06)**Exception Code

Restore Slave Address to Factory State (0x01)

(1) Broadcast **(0x00)**: Set Slave Address to 0x01

Master request: **00** 06 00 10 00 01 48 1E

Slave response: write only, not respond

Winson reserves the right to make changes to improve reliability or manufacturability.

5. Measuring Method (I2C)

(5.1) Register Configuration

- Register Initial Values:

Register	Reset(Power on)	Register	Reset(Power on)
00h	0000 0001	08h	0000 0000
01h	0101 0011	09h	0000 0000
02h	0000 0000	0Ah	0000 0000
03h	0000 0000	0Bh	0000 0000
04h	0000 0000	0Ch	0000 0000
05h	0000 0000	0Dh	0000 0000
06h	0000 0000	0Eh	0000 0000
07h	0000 0000	0Fh	0000 0000

- Control Register(00h)

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	Auto Mode
R/W	-	-	-	-	-	-	-	R/W
POR	-	-	-	-	-	-	-	1

Bit 7~1 Unimplemented, read as "0"

Bit 0 Auto Mode: Automatically refresh the data of measuring temperature and current

0: Manual, set in the status register (refer to the **Status Register(03h)**)

1: Automatic

- Slave Address Register (01h)

Bit	7	6	5	4	3	2	1	0
Name	-	Slave Address						
R/W	-	R/W						
POR	-	1	0	1	0	0	1	1

Bit 7 Unimplemented, read as "0"

Bit 6~0 Slave Address: Initial slave address is 0x53

- Current Reset Register (02h)

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	Reset
R/W	-	-	-	-	-	-	-	R/W
POR	-	-	-	-	-	-	-	0

Bit 7~1 Unimplemented, read as "0"

Bit 0 Reset: Current reset flag

0: Reset

1: Set (zeroing)

This bit will be automatically cleared after zeroing.

- Status Register (03h)

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	Status	Valid
R/W	-	-	-	-	-	-	R/W	R
POR	-	-	-	-	-	-	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1 Status: Measuring status flag / function

0: Disable

1: Enable, start measuring current and temperature

Bit 0 Valid: Data valid flag

0: The measurement has not been completed and the value is invalid.

1: The measurement has been completed and the value is valid.

1. **Manually set the status bit to start measuring current, wait for the valid bit to be set to 1 before reading the measured value.**
2. **After reading the current or temperature, the significant bit is cleared.**

- Measuring Data Registers

Calculation: Each set of data consists of 4 bytes, arranged from high to low bytes into a set of 32-bit signed integers, and converted to actual values using the following formula.

(1)Temperature Registers (04h~07h)

Register (04h)

Bit	31	30	29	28	27	26	25	24
Name	D31	D30	D29	D28	D27	D26	D25	D24

Register (05h)

Bit	23	22	21	20	19	18	17	16
Name	D23	D22	D21	D20	D19	D18	D17	D16

Register (06h)

Bit	15	14	13	12	11	10	9	8
Name	D15	D14	D13	D12	D11	D10	D9	D8

Register (07h)

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0

Temperature = D[31:0] / 10 (°C)

(2)DC Current Registers (08h~0Bh)

Register (08h)

Bit	31	30	29	28	27	26	25	24
Name	D31	D30	D29	D28	D27	D26	D25	D24

Register (09h)

Bit	23	22	21	20	19	18	17	16
Name	D23	D22	D21	D20	D19	D18	D17	D16

Register (0Ah)

Bit	15	14	13	12	11	10	9	8
Name	D15	D14	D13	D12	D11	D10	D9	D8

Register (0Bh)

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0

Current = D[31:0] / 1000 (A)

(3)AC Current Registers (0Ch~0Fh)

Register (0Ch)

Bit	31	30	29	28	27	26	25	24
Name	D31	D30	D29	D28	D27	D26	D25	D24

Register (0Dh)

Bit	23	22	21	20	19	18	17	16
Name	D23	D22	D21	D20	D19	D18	D17	D16

Register (0Eh)

Bit	15	14	13	12	11	10	9	8
Name	D15	D14	D13	D12	D11	D10	D9	D8

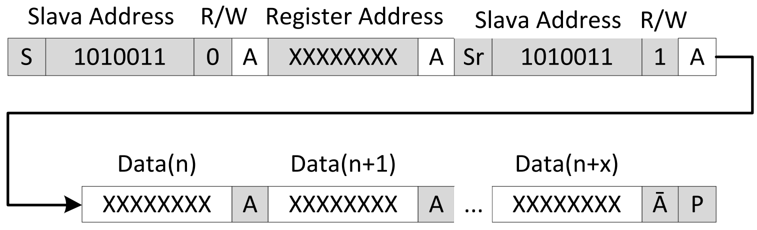
Register (0Fh)

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0

Current = D[31:0] / 1000 (A)

(5.2) Send "Read" Command

※This function cannot be used in broadcast mode (0x00).



	Slava Address	R/W		Register Address			Slava Address	R/W	
S	1010011	0	A	XXXXXXXX	A	Sr	1010011	1	A

Data(n)		Data(n+1)			Data(n+x)		
XXXXXXXX	A	XXXXXXXX	A	...	XXXXXXXX	Ā	P

S -Start

A -Acknowledge(Ack)

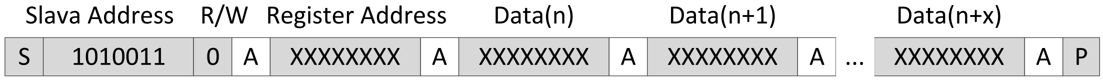
Ā -Not Acknowledge(Nack)

P -Stop

R/W 1:Read/0:Write

▨ Master to Slave □ Slave to Master

(5.3) Send "Write" Command



	Slava Address	R/W		Register Address		Data(n)		Data(n+1)			Data(n+x)		
S	1010011	0	A	XXXXXXXX	A	XXXXXXXX	A	XXXXXXXX	A	...	XXXXXXXX	A	P

S -Start

A -Acknowledge(Ack)

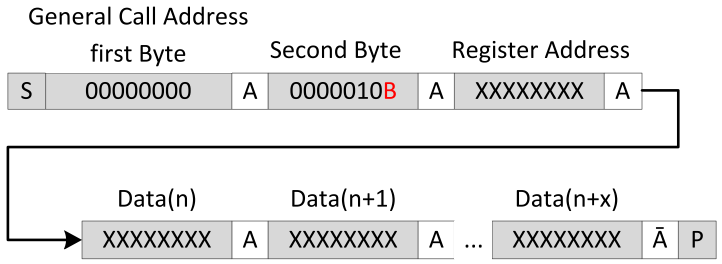
Ā -Not Acknowledge(Nack)

P -Stop

R/W 1:Read/0:Write

▨ Master to Slave □ Slave to Master

(5.3) Broadcast Mode(0x00)



S -Start

A -Acknowledge(Ack)

Ā -Not Acknowledge(Nack)

P -Stop

R/W 1:Read/0:Write

First Byte : General Call Address

Second Byte(04h) : System does not reset when writing data

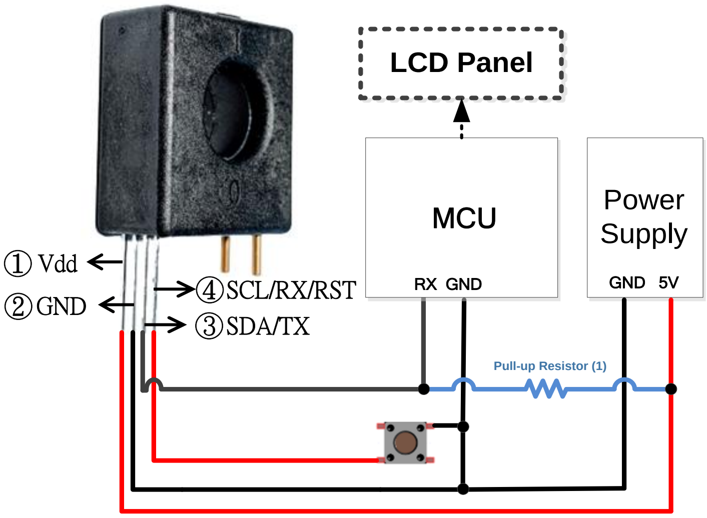
▨ Master to Slave □ Slave to Master

- The lowest bit B of the second byte is 0:
  Data can be written to the slave device through the second byte (04h). The hardware will not be reset during the process and the data sent by the master can be received.

  Data can be written to the slave device through the second byte (06h). The hardware will be reset during the process and the data sent by the master can be received. (not use)
- The lowest bit B of the second byte is 1: It is hardware broadcast (not used)

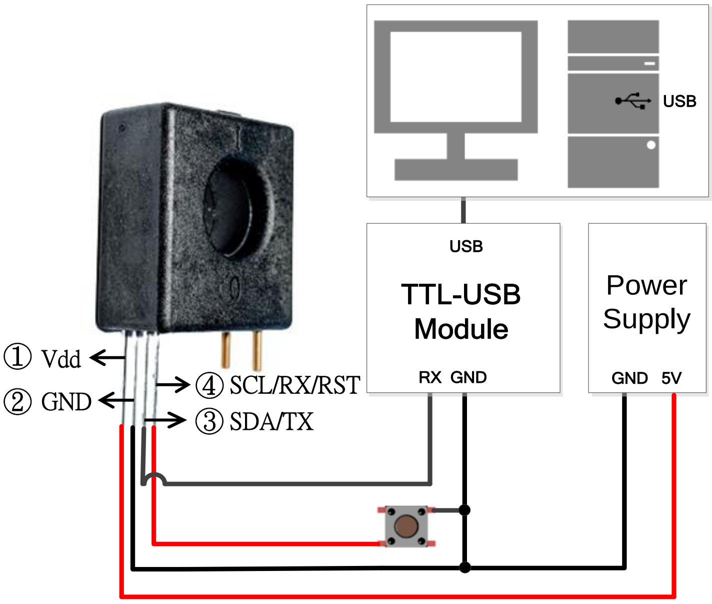
6. Application Diagram (Continuous Mode)

- **MCU Connection Diagram**



(1) DWCS **TX** pin is **open drain**, and pull-up resistor must be used. If the MCU **RX** pin has been internally pulled up, the resistor can be removed.

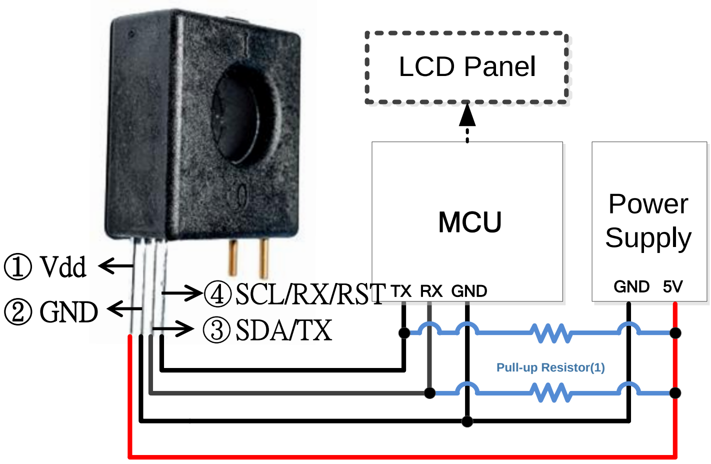
- **TTL to USB Connection Diagram**



Winson reserves the right to make changes to improve reliability or manufacturability.

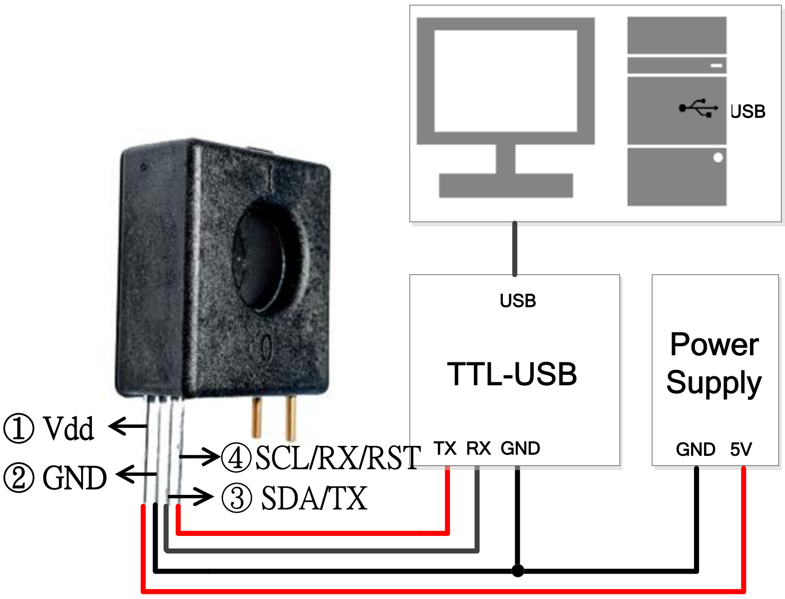
7. Application Diagram (AT Command & Modbus-RTU)

● MCU Connection Diagram



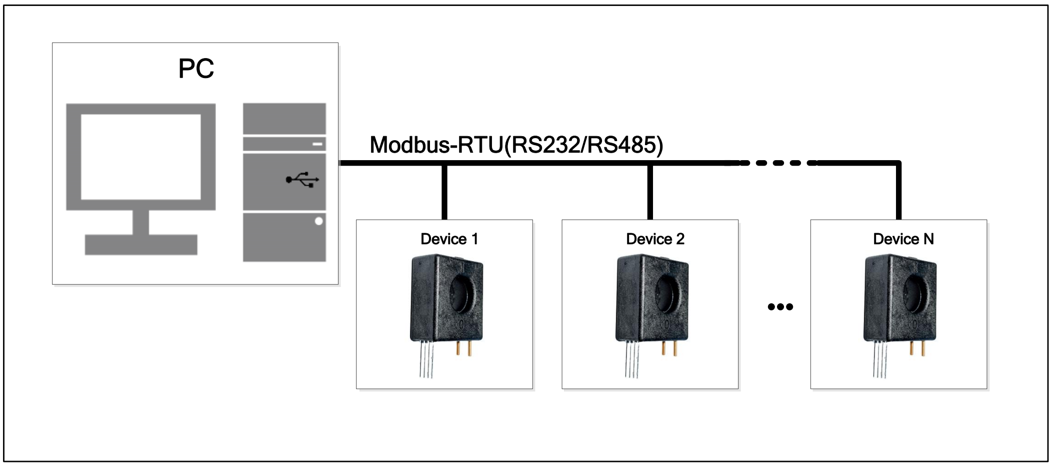
(1) DWCS **TX/RX** pin is **open drain**, and pull-up resistor must be used. If the MCU **TX/RX** pin has been internally pulled up, the resistor can be removed.

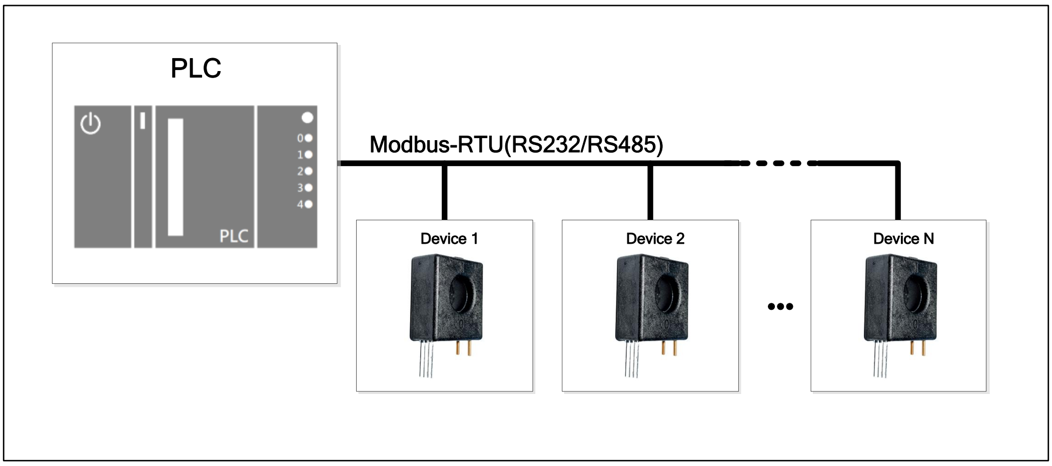
● TTL to USB Connection Diagram

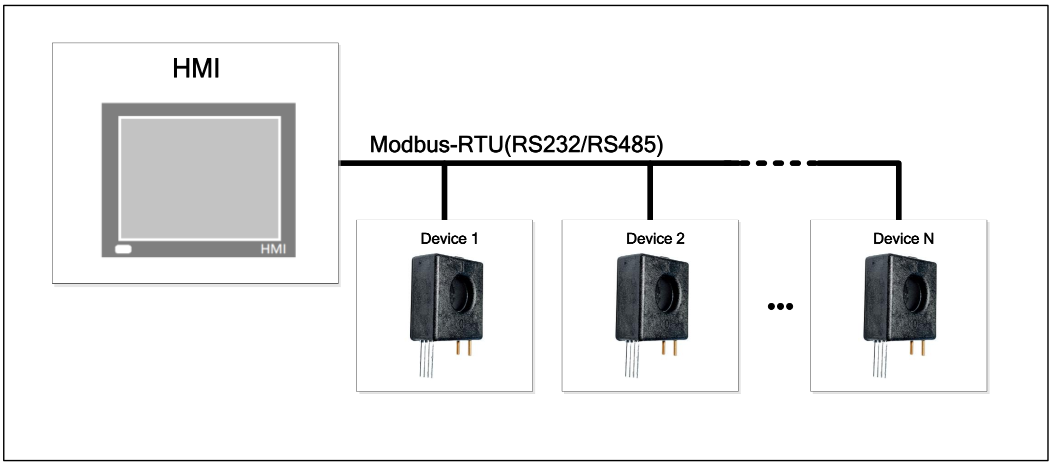


Winson reserves the right to make changes to improve reliability or manufacturability.

- **Modbus-RTU Architecture Diagram:**

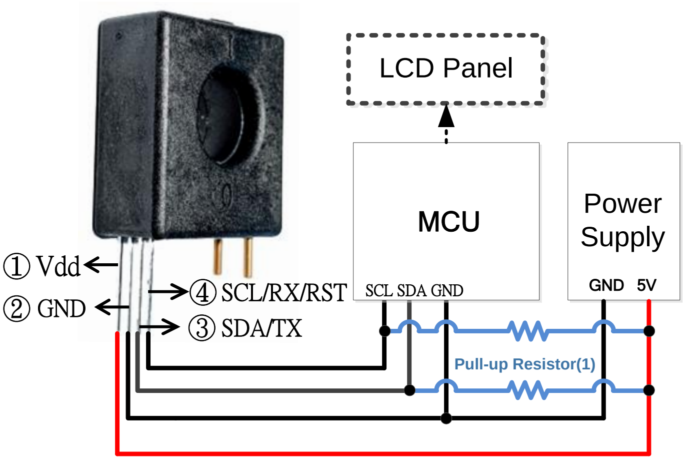






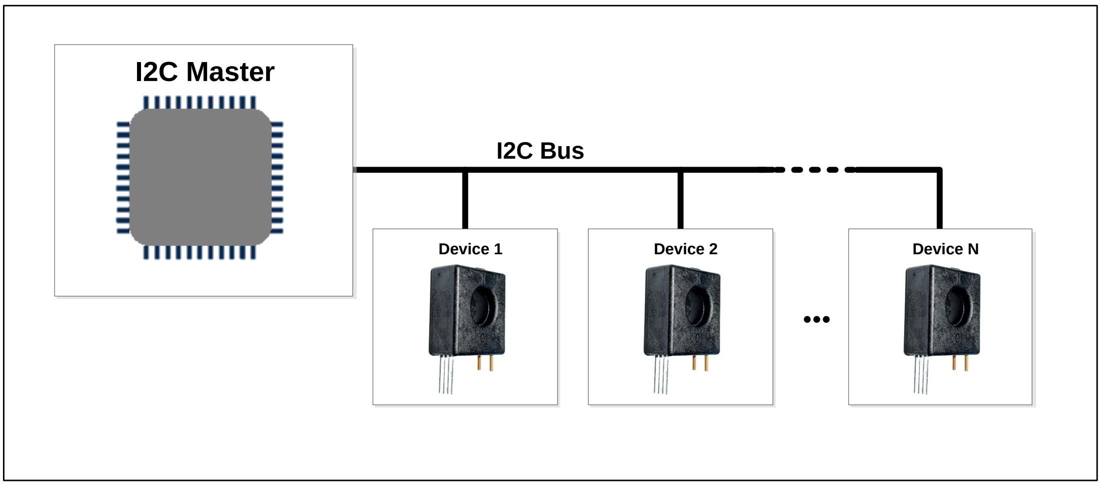
8. Application Diagram (I2C)

- **I2C Connection Diagram**



(1) DWCS **SDA/SCL** pin is **open drain**, and pull-up resistor must be used. If the MCU **SDA/SCL** pin has been internally pulled up, the resistor can be removed.

- **I2C Architecture Diagram:**

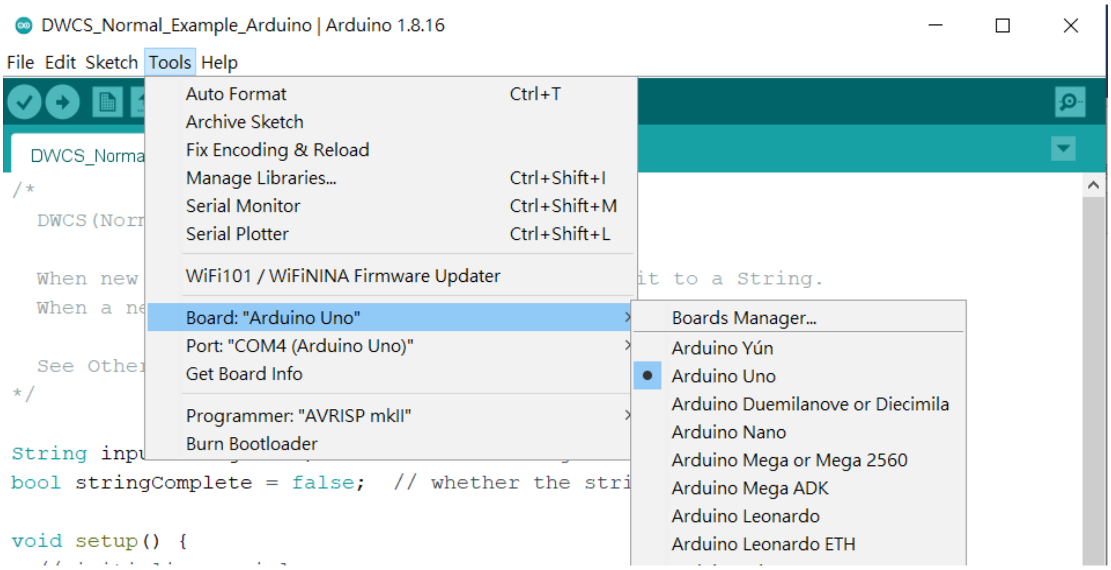


Winson reserves the right to make changes to improve reliability or manufacturability.

Application Example on Arduino

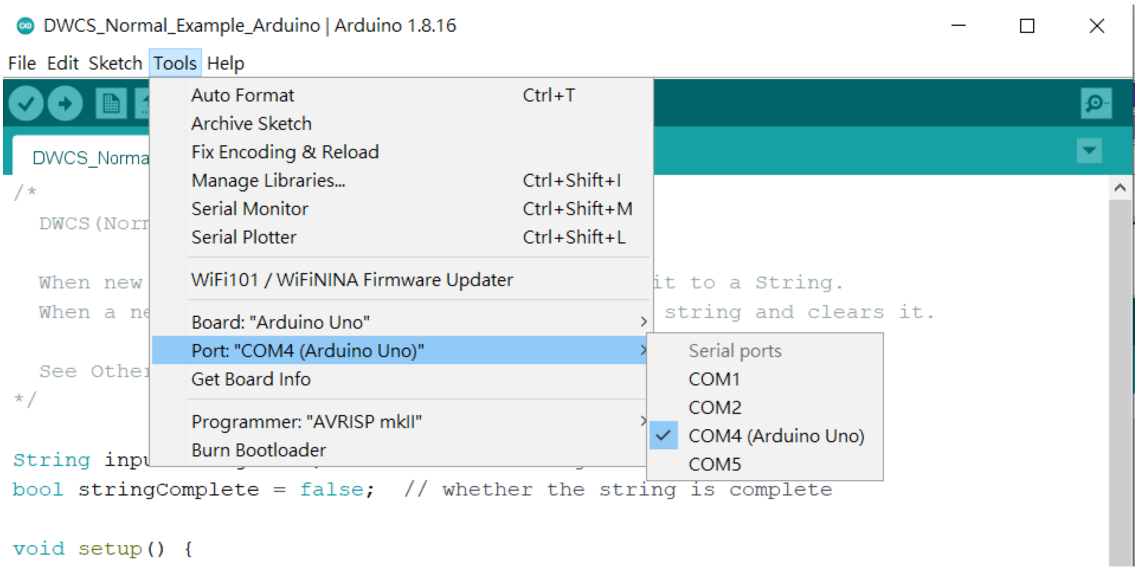
9. Instructions for Arduino

(1). Check the type of board is correct.


DWCS_Normal_Example_Arduino | Arduino 1.8.16
File Edit Sketch Tools Help
DWCS_Norma
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port: "COM4 (Arduino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino Yún
● Arduino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino Leonardo ETH

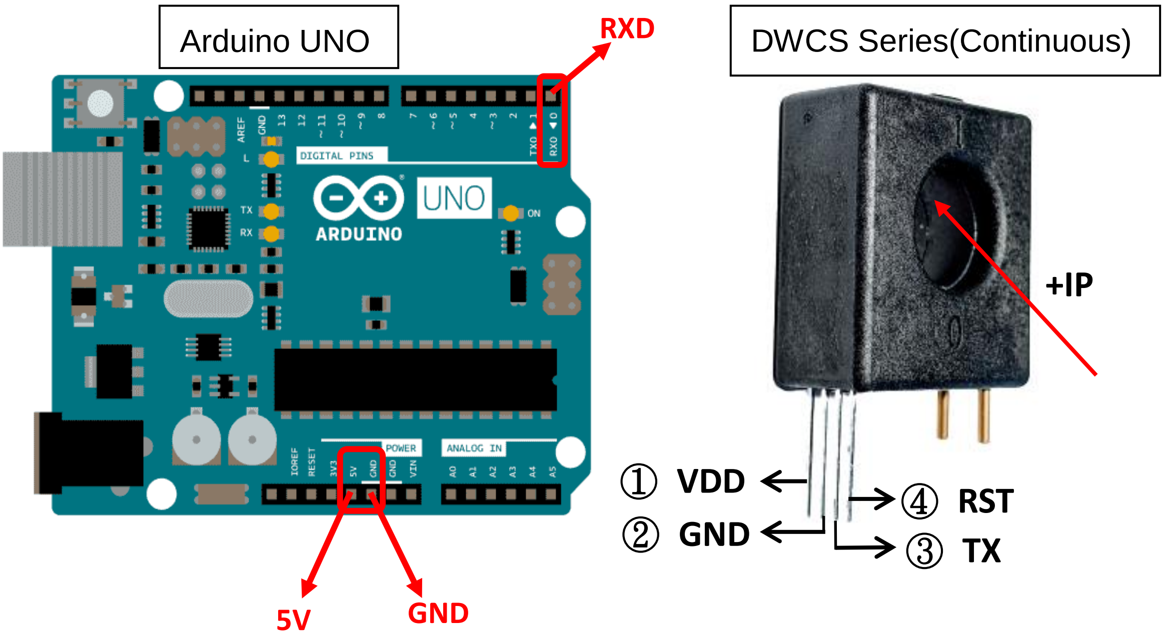
```
/*
  DWCS(Norr

  When new                                                it to a String.
  When a ne

  See Other
*/

String inpu
bool stringComplete = false;  // whether the stri

void setup() {
```

(2). Check the port of Arduino is connected and selected correctly.

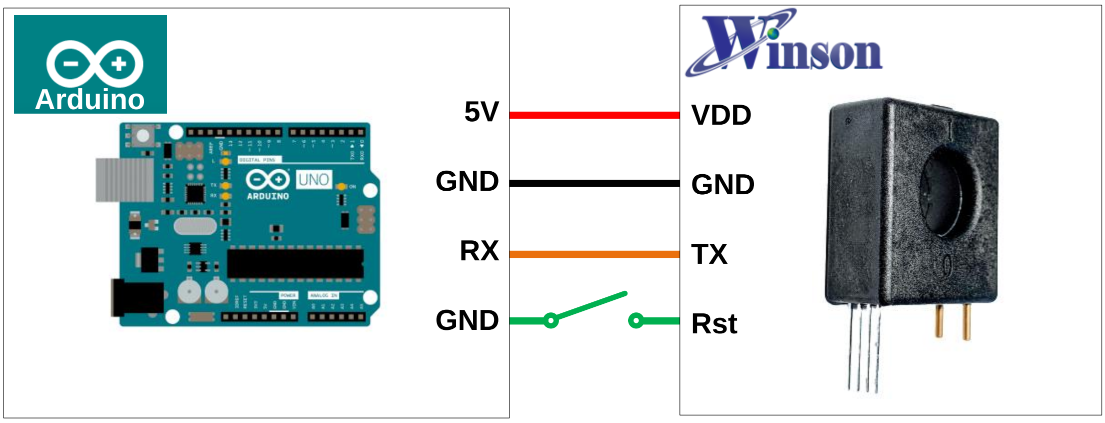

DWCS_Normal_Example_Arduino | Arduino 1.8.16
File Edit Sketch Tools Help
DWCS_Norma
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port: "COM4 (Arduino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
COM2
✓ COM4 (Arduino Uno)
COM5

```
/*
  DWCS(Norr

  When new                                                it to a String.
  When a ne                                                string and clears it.

  See Other
*/

String inpu
bool stringComplete = false;  // whether the string is complete

void setup() {
```

10. Continuous Mode

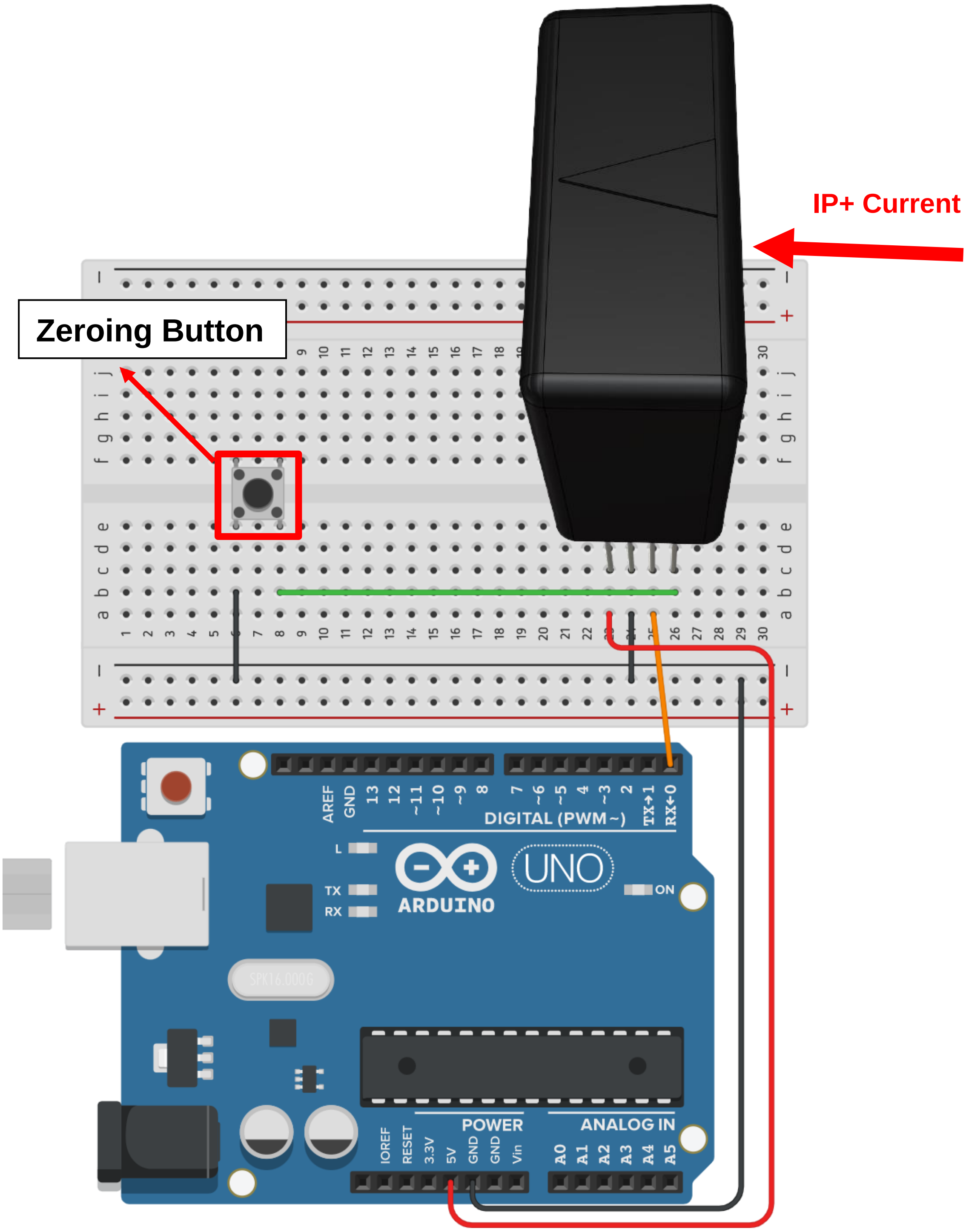


- **Schematic Diagram**



Winson reserves the right to make changes to improve reliability or manufacturability.

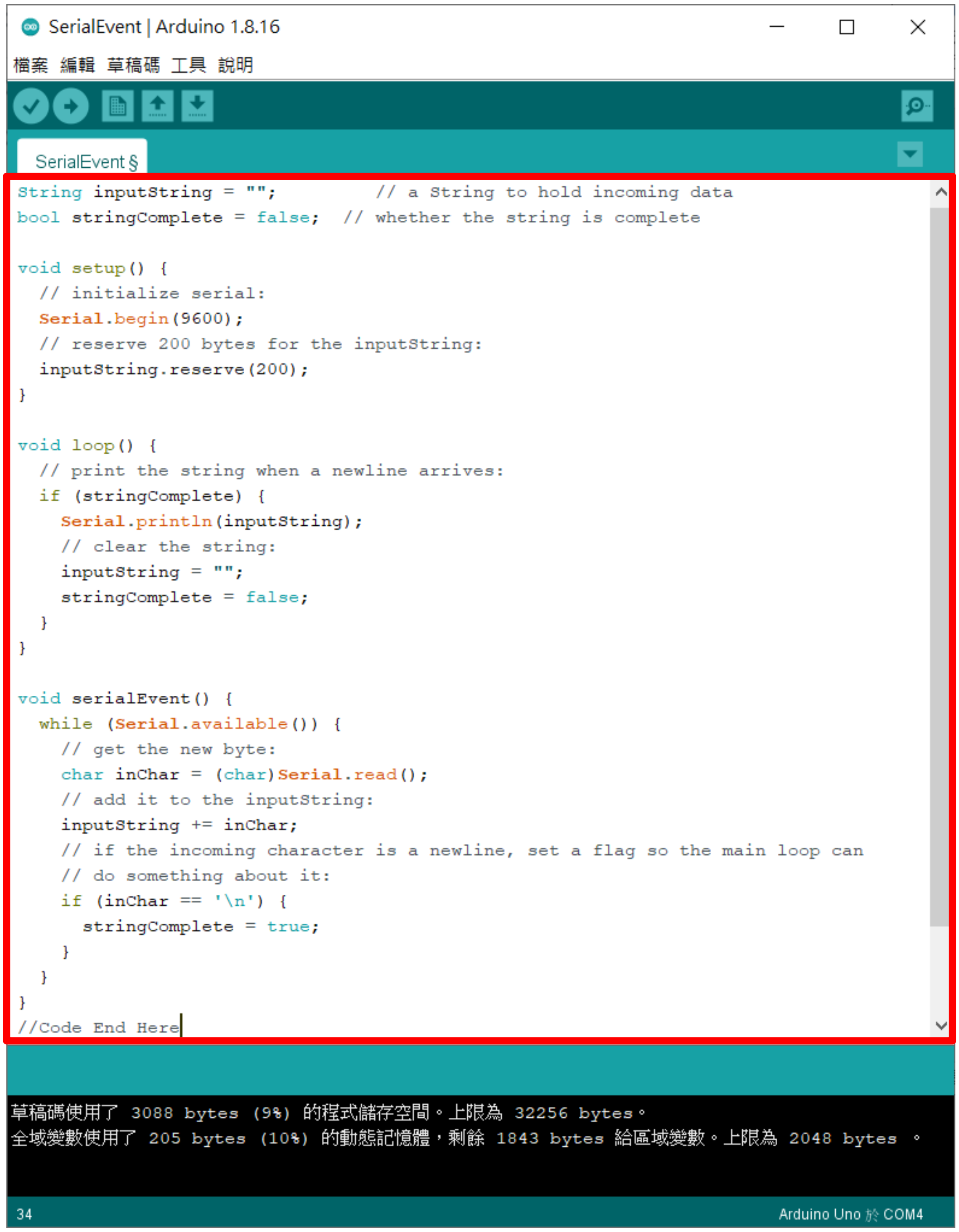
- **Wiring Diagram**



Winson reserves the right to make changes to improve reliability or manufacturability.

● Software & Program

(1). Example code can be download at:http:/www.winson.com.tw/Product/156



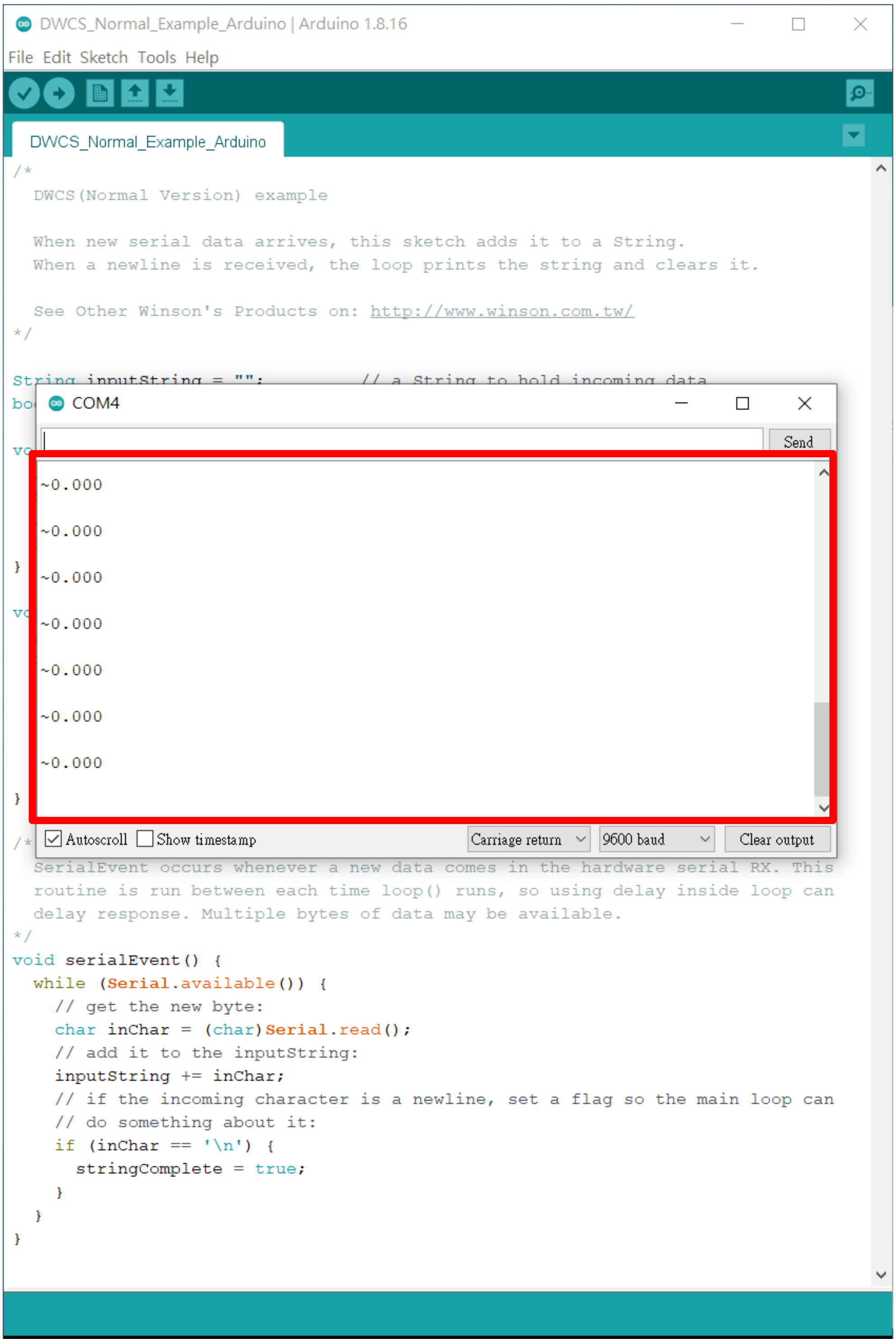
```
String inputString = "";         // a String to hold incoming data
bool stringComplete = false;  // whether the string is complete

void setup() {
  // initialize serial:
  Serial.begin(9600);
  // reserve 200 bytes for the inputString:
  inputString.reserve(200);
}

void loop() {
  // print the string when a newline arrives:
  if (stringComplete) {
    Serial.println(inputString);
    // clear the string:
    inputString = "";
    stringComplete = false;
  }
}

void serialEvent() {
  while (Serial.available()) {
    // get the new byte:
    char inChar = (char)Serial.read();
    // add it to the inputString:
    inputString += inChar;
    // if the incoming character is a newline, set a flag so the main loop can
    // do something about it:
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
//Code End Here
```

```
草稿碼使用了 3088 bytes (9%) 的程式儲存空間。上限為 32256 bytes。
全域變數使用了 205 bytes (10%) 的動態記憶體，剩餘 1843 bytes 給區域變數。上限為 2048 bytes 。
```

※CAUTION!! To prevent upload failure of Arduino, please insert DWCS after upload process.

Winson reserves the right to make changes to improve reliability or manufacturability.

(2). Upload the example code and open Serial Monitor to display the measured current.


DWCS_Normal_Example_Arduino | Arduino 1.8.16
File Edit Sketch Tools Help
DWCS_Normal_Example_Arduino
COM4
Send

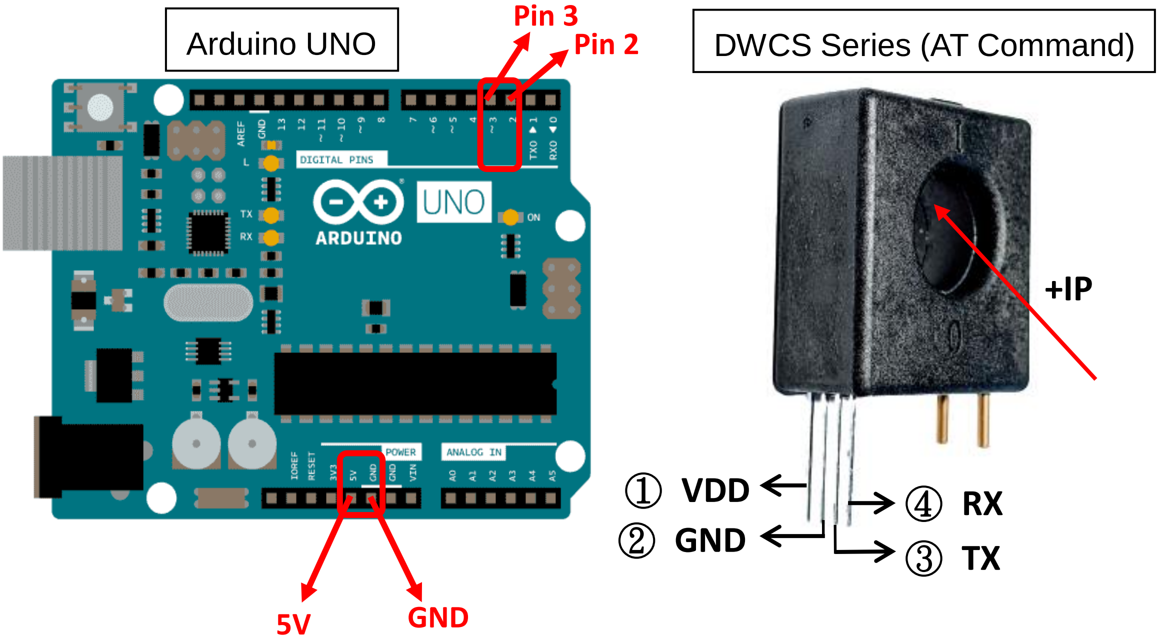
```
/*
  DWCS(Normal Version) example

  When new serial data arrives, this sketch adds it to a String.
  When a newline is received, the loop prints the string and clears it.

  See Other Winson's Products on: http://www.winson.com.tw/
*/

String inputString = "";          // a String to hold incoming data
bo

vo



}

vo





}

/*
  SerialEvent occurs whenever a new data comes in the hardware serial RX. This
  routine is run between each time loop() runs, so using delay inside loop can
  delay response. Multiple bytes of data may be available.
*/
void serialEvent() {
  while (Serial.available()) {
    // get the new byte:
    char inChar = (char)Serial.read();
    // add it to the inputString:
    inputString += inChar;
    // if the incoming character is a newline, set a flag so the main loop can
    // do something about it:
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
```

```
~0.000

~0.000

~0.000

~0.000

~0.000

~0.000

~0.000
```

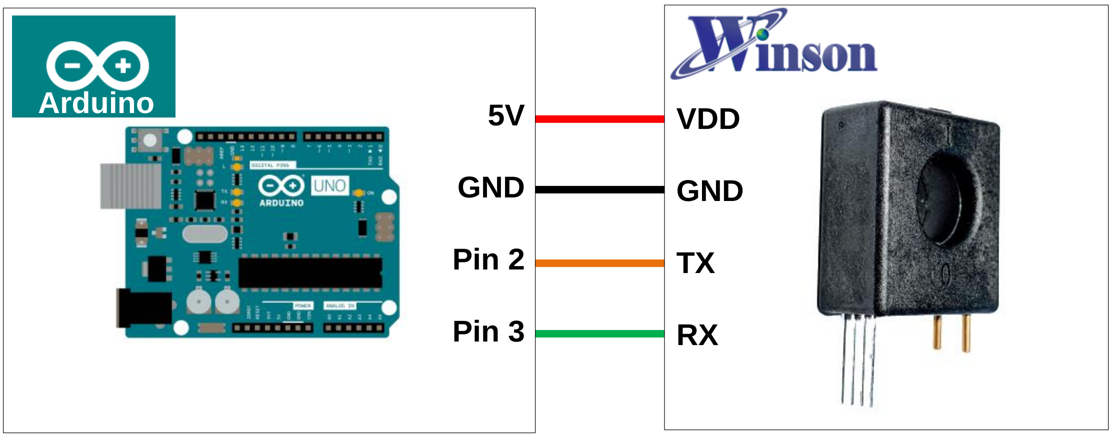
☑ Autoscroll □ Show timestamp Carriage return 9600 baud Clear output

Winson reserves the right to make changes to improve reliability or manufacturability.

11. AT Command Mode

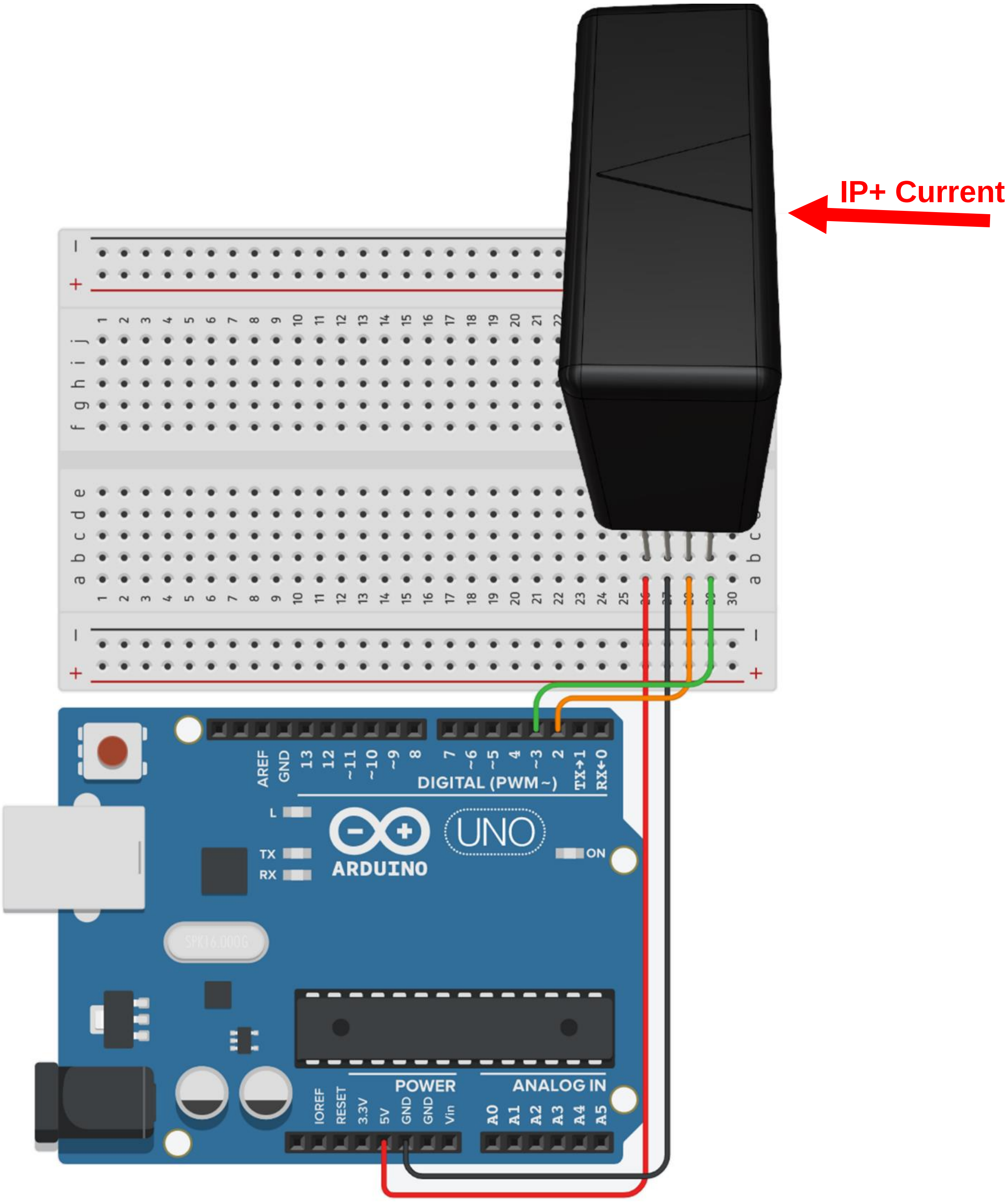


- **Schematic Diagram**



Winson reserves the right to make changes to improve reliability or manufacturability.

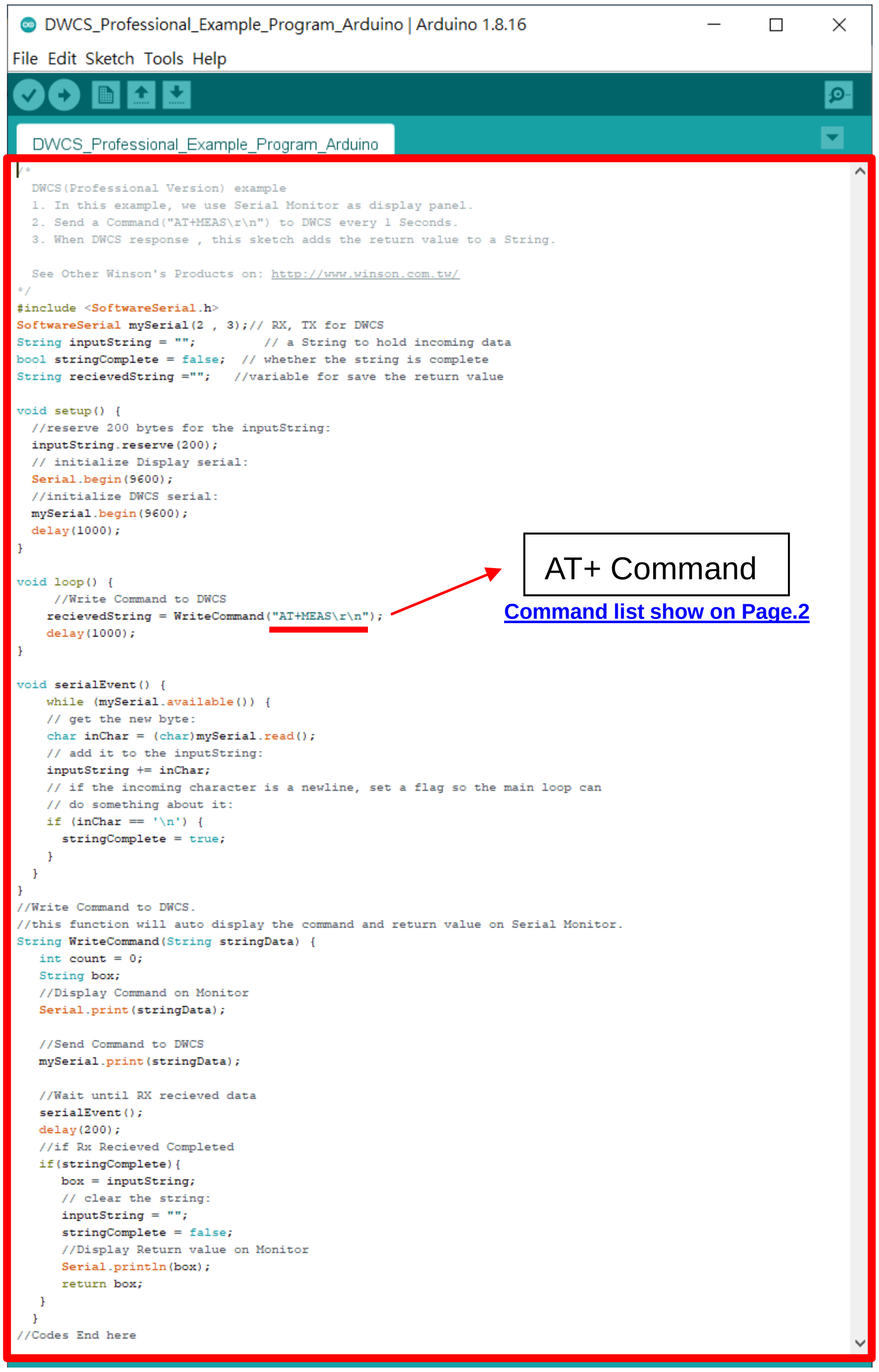
- **Wiring Diagram**



Winson reserves the right to make changes to improve reliability or manufacturability.

● Software & Program

(1). Example code can be download at:http:/www.winson.com.tw/Product/156


DWCS_Professional_Example_Program_Arduino | Arduino 1.8.16
File Edit Sketch Tools Help
DWCS_Professional_Example_Program_Arduino

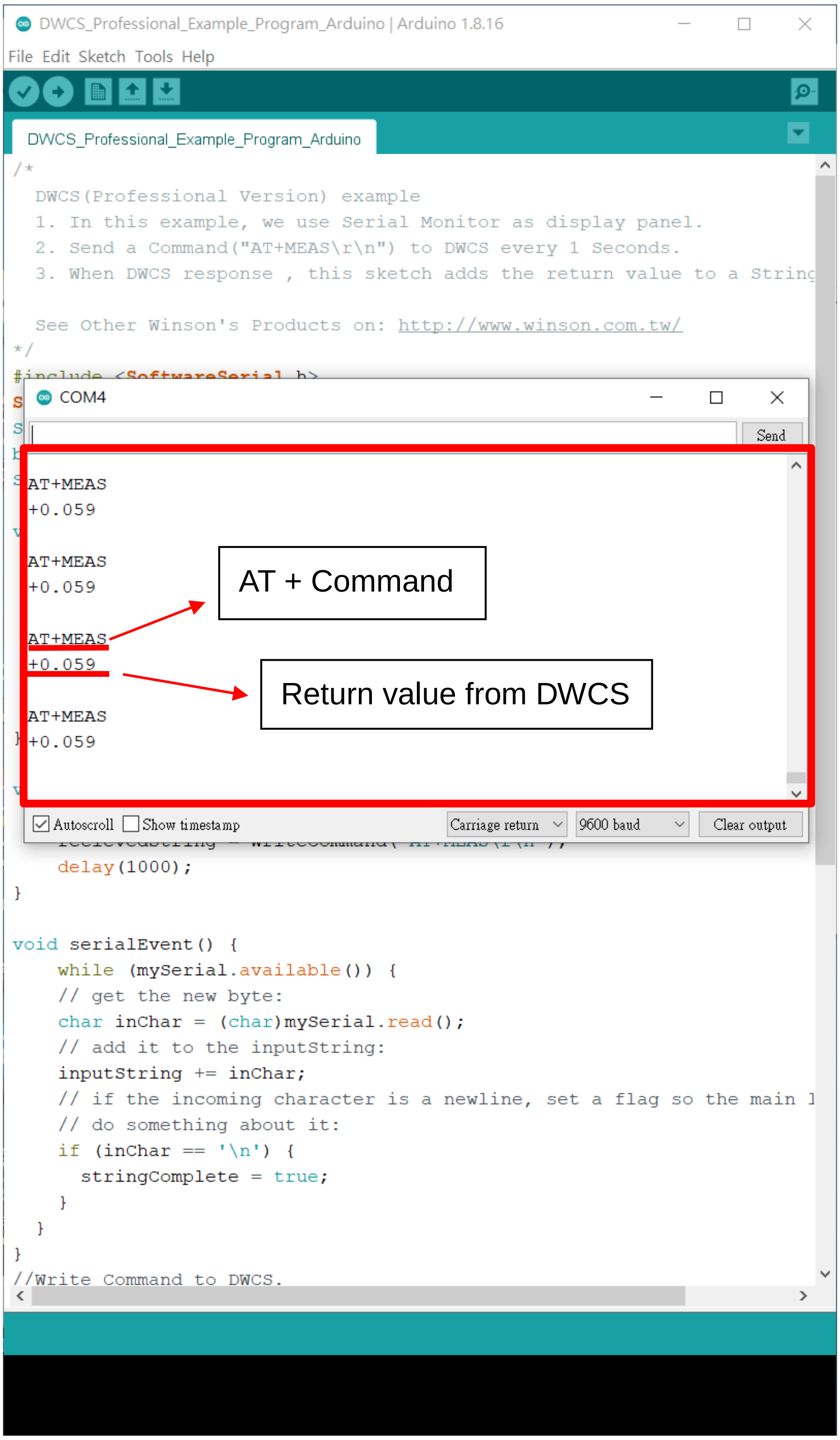
```
/*
  DWCS(Professional Version) example
  1. In this example, we use Serial Monitor as display panel.
  2. Send a Command("AT+MEAS\r\n") to DWCS every 1 Seconds.
  3. When DWCS response , this sketch adds the return value to a String.

  See Other Winson's Products on: http://www.winson.com.tw/
*/
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2 , 3);// RX, TX for DWCS
String inputString = "";         // a String to hold incoming data
bool stringComplete = false;  // whether the string is complete
String recievedString ="";   //variable for save the return value

void setup() {
  //reserve 200 bytes for the inputString:
  inputString.reserve(200);
  // initialize Display serial:
  Serial.begin(9600);
  //initialize DWCS serial:
  mySerial.begin(9600);
  delay(1000);
}

void loop() {
     //Write Command to DWCS
    recievedString = WriteCommand("AT+MEAS\r\n");
    delay(1000);
}

void serialEvent() {
    while (mySerial.available()) {
    // get the new byte:
    char inChar = (char)mySerial.read();
    // add it to the inputString:
    inputString += inChar;
    // if the incoming character is a newline, set a flag so the main loop can
    // do something about it:
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
//Write Command to DWCS.
//this function will auto display the command and return value on Serial Monitor.
String WriteCommand(String stringData) {
   int count = 0;
   String box;
   //Display Command on Monitor
   Serial.print(stringData);

   //Send Command to DWCS
   mySerial.print(stringData);

   //Wait until RX recieved data
   serialEvent();
   delay(200);
   //if Rx Recieved Completed
   if(stringComplete){
      box = inputString;
      // clear the string:
      inputString = "";
      stringComplete = false;
      //Display Return value on Monitor
      Serial.println(box);
      return box;
   }
  }
//Codes End here
```

AT+ Command

[Command list show on Page.2]()

※CAUTION!! To prevent upload failure of Arduino, please insert DWCS after upload process.

(2). Upload the example code and open Serial Monitor to display the return value. ◦


DWCS_Professional_Example_Program_Arduino | Arduino 1.8.16
File Edit Sketch Tools Help
DWCS_Professional_Example_Program_Arduino
COM4
Send
Clear output
AT + Command
Return value from DWCS

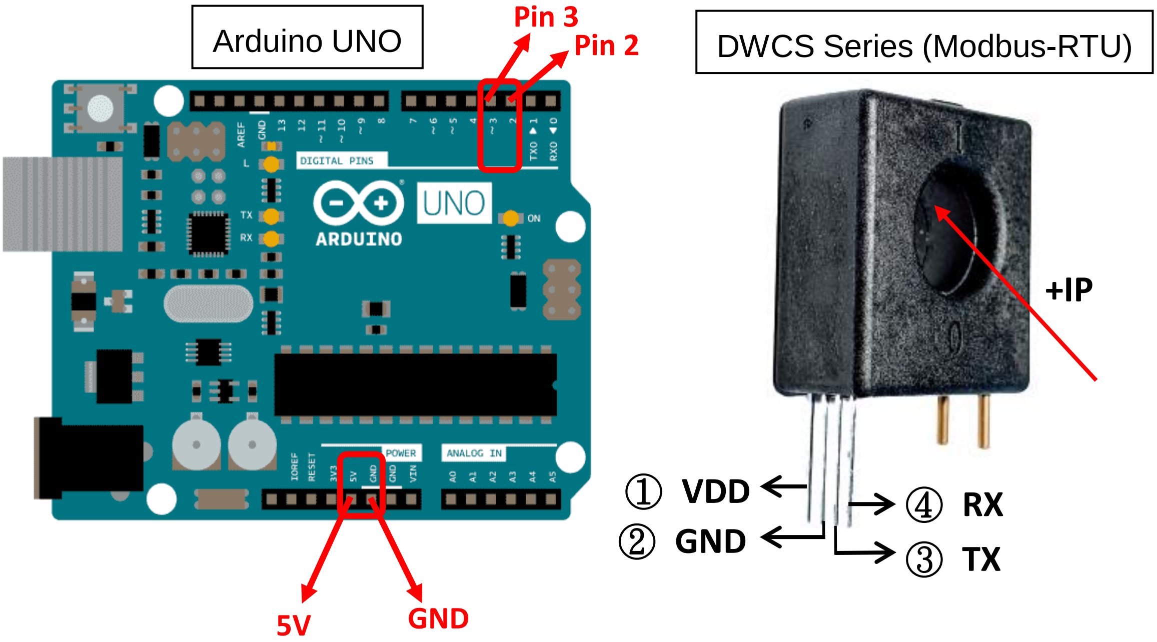
```
/*
  DWCS(Professional Version) example
  1. In this example, we use Serial Monitor as display panel.
  2. Send a Command("AT+MEAS\r\n") to DWCS every 1 Seconds.
  3. When DWCS response , this sketch adds the return value to a String

  See Other Winson's Products on: http://www.winson.com.tw/
*/
#include <SoftwareSerial.h>
```

```
AT+MEAS
+0.059

AT+MEAS
+0.059

AT+MEAS
+0.059

AT+MEAS
+0.059
```

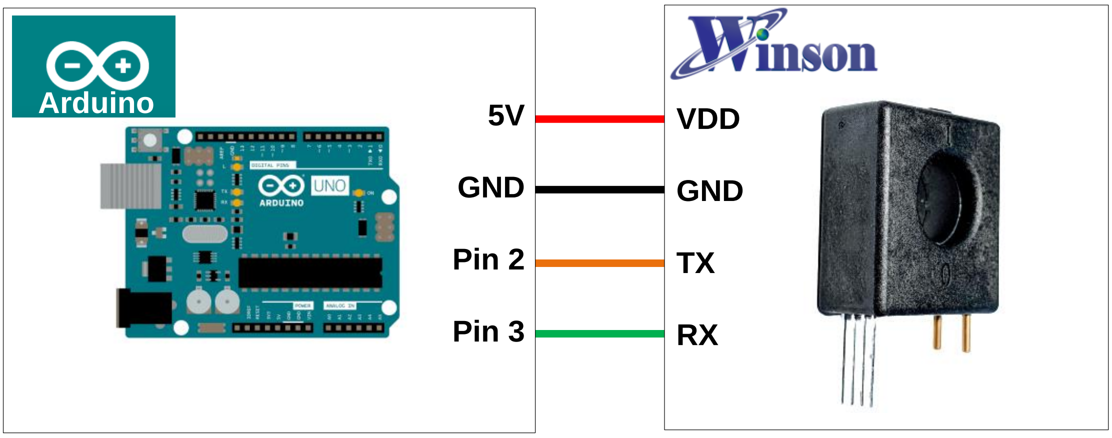
☑ Autoscroll □ Show timestamp Carriage return 9600 baud

```
    delay(1000);
}

void serialEvent() {
    while (mySerial.available()) {
    // get the new byte:
    char inChar = (char)mySerial.read();
    // add it to the inputString:
    inputString += inChar;
    // if the incoming character is a newline, set a flag so the main l
    // do something about it:
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
//Write Command to DWCS.
```

12. Modbus-RTU (Single Device Communication)

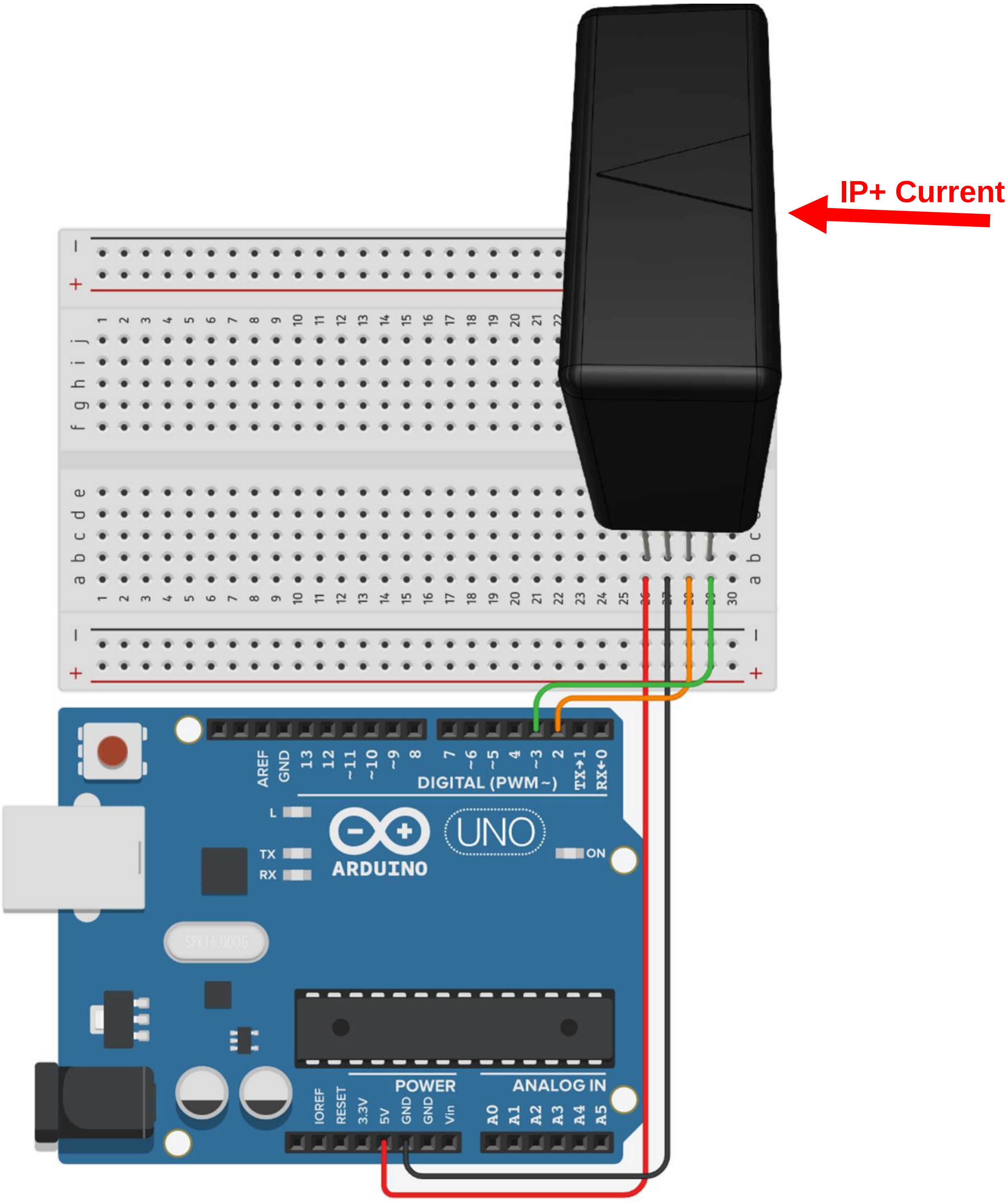


- **Schematic Diagram**



Winson reserves the right to make changes to improve reliability or manufacturability.

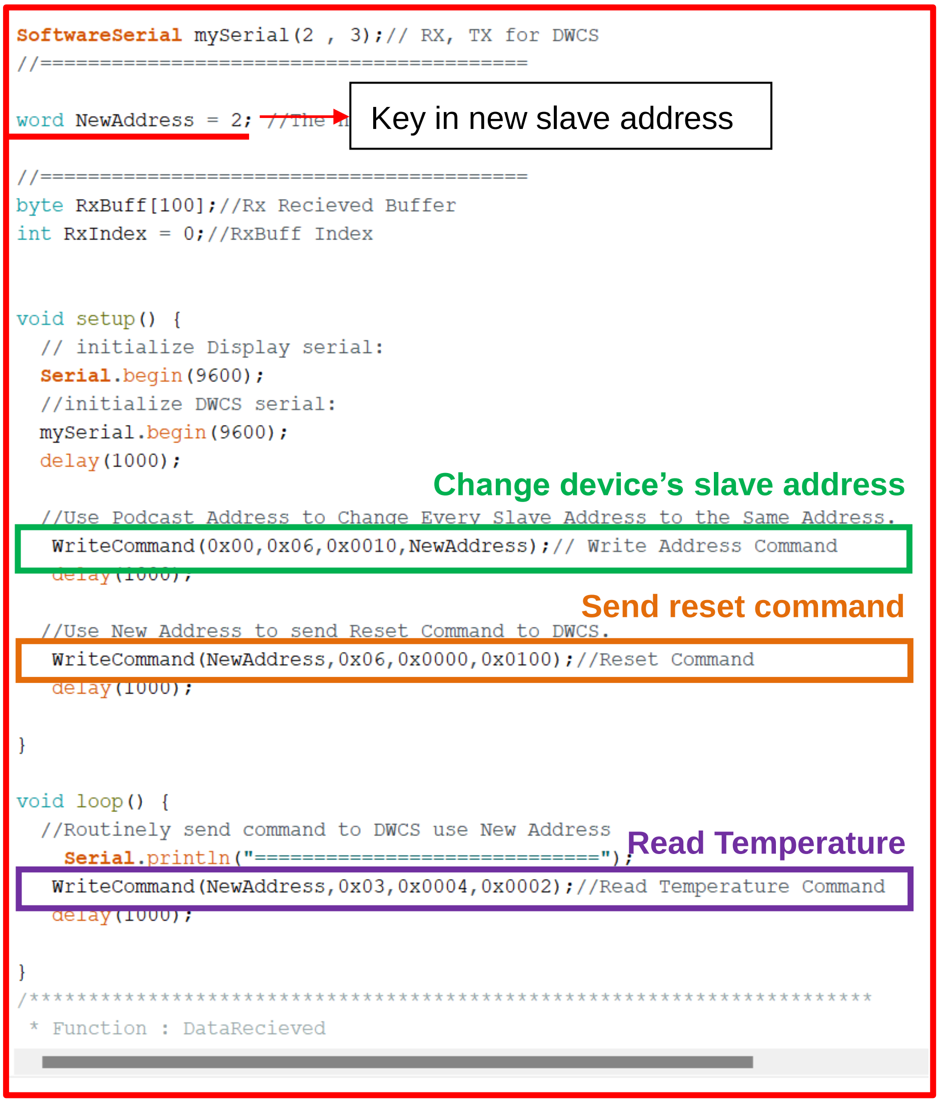
- **Wiring Diagram**



Winson reserves the right to make changes to improve reliability or manufacturability.

● Software & Program

(3). Example code can be download at:http:/www.winson.com.tw/Product/156



```
SoftwareSerial mySerial(2 , 3);// RX, TX for DWCS
//=========================================

word NewAddress = 2; //The n

//=========================================
byte RxBuff[100];//Rx Recieved Buffer
int RxIndex = 0;//RxBuff Index


void setup() {
  // initialize Display serial:
  Serial.begin(9600);
  //initialize DWCS serial:
  mySerial.begin(9600);
  delay(1000);

  //Use Podcast Address to Change Every Slave Address to the Same Address.
   WriteCommand(0x00,0x06,0x0010,NewAddress);// Write Address Command
   delay(1000);

  //Use New Address to send Reset Command to DWCS.
   WriteCommand(NewAddress,0x06,0x0000,0x0100);//Reset Command
   delay(1000);

}

void loop() {
  //Routinely send command to DWCS use New Address
    Serial.println("=============================");
   WriteCommand(NewAddress,0x03,0x0004,0x0002);//Read Temperature Command
   delay(1000);

}
/***********************************************************************
 * Function : DataRecieved
```

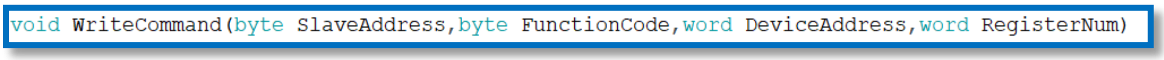
Key in new slave address

Change device's slave address

Send reset command

Read Temperature

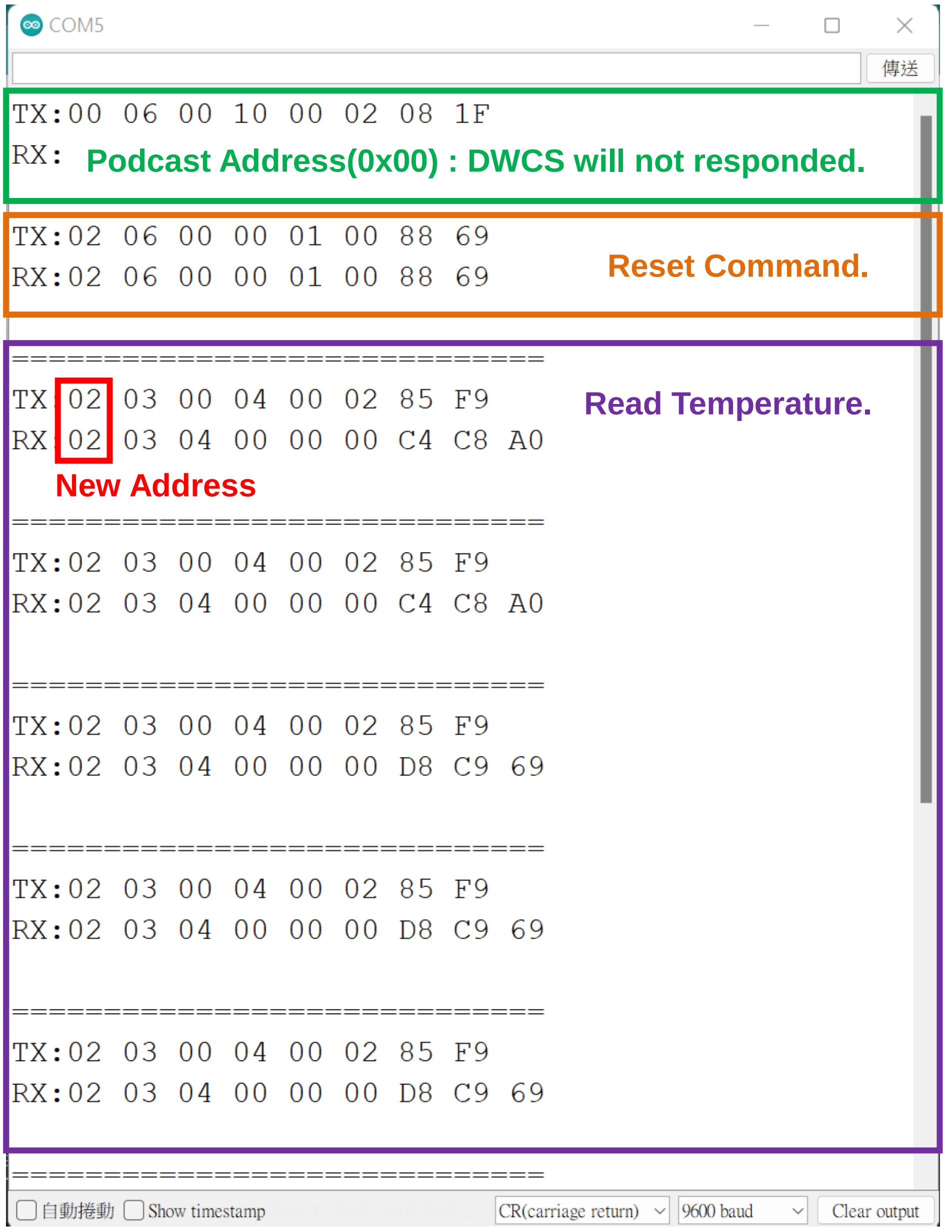
Write Command [Read(03H) / Write(06H)]:



```
void WriteCommand(byte SlaveAddress,byte FunctionCode,word DeviceAddress,word RegisterNum)
```

※CAUTION!! To prevent upload failure of Arduino, please insert DWCS after upload process.

(4). Upload the example code and open Serial Monitor to display the return value.



```
TX:00 06 00 10 00 02 08 1F
RX:
```

Podcast Address(0x00) : DWCS will not responded.

```
TX:02 06 00 00 01 00 88 69
RX:02 06 00 00 01 00 88 69
```

Reset Command.

```
==============================
TX:02 03 00 04 00 02 85 F9
RX:02 03 04 00 00 00 C4 C8 A0
```

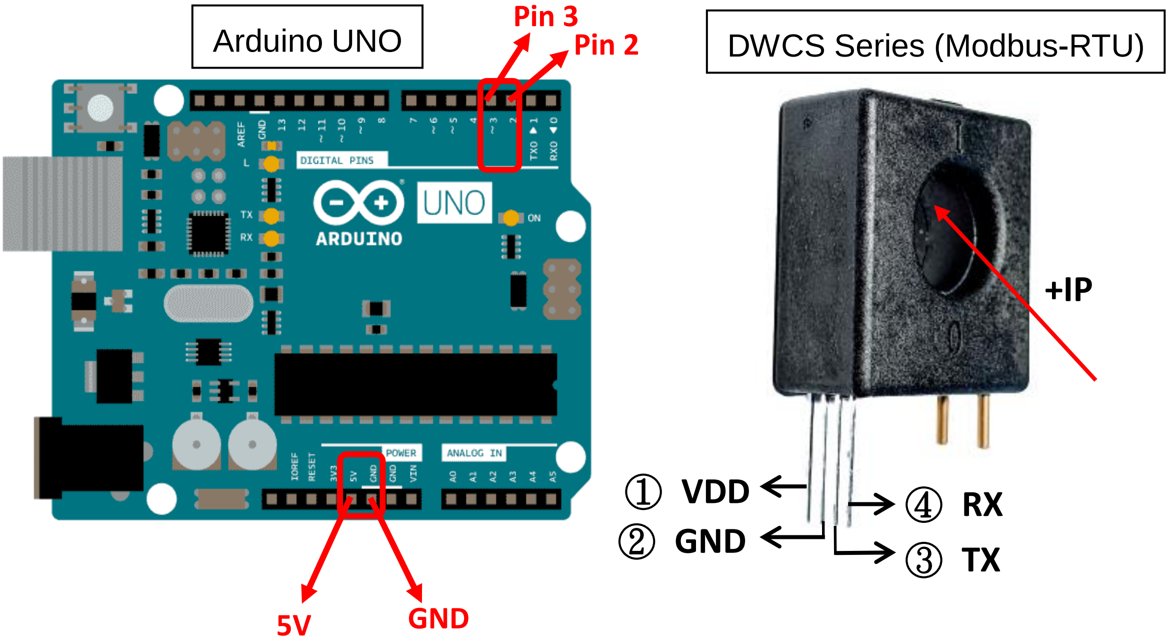
New Address

Read Temperature.

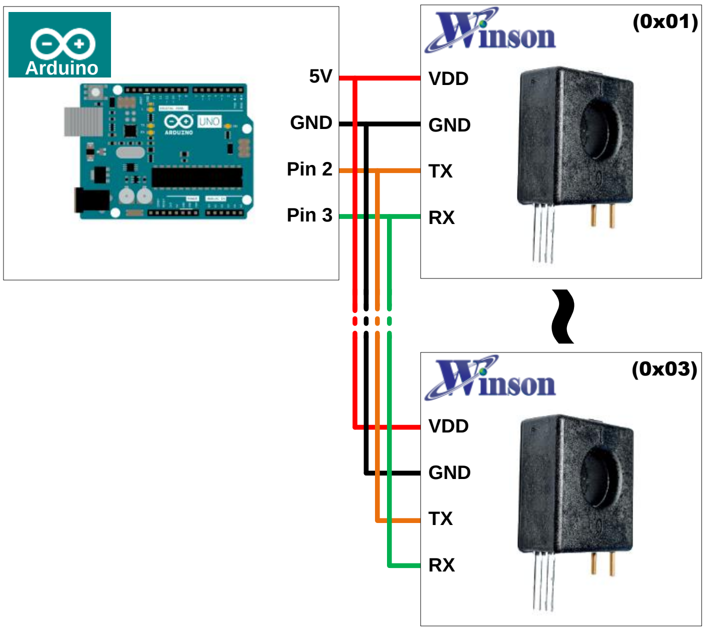
```
==============================
TX:02 03 00 04 00 02 85 F9
RX:02 03 04 00 00 00 C4 C8 A0

==============================
TX:02 03 00 04 00 02 85 F9
RX:02 03 04 00 00 00 D8 C9 69

==============================
TX:02 03 00 04 00 02 85 F9
RX:02 03 04 00 00 00 D8 C9 69

==============================
TX:02 03 00 04 00 02 85 F9
RX:02 03 04 00 00 00 D8 C9 69

==============================
```

Winson reserves the right to make changes to improve reliability or manufacturability.

13. Modbus-RTU (one-to-many communication)



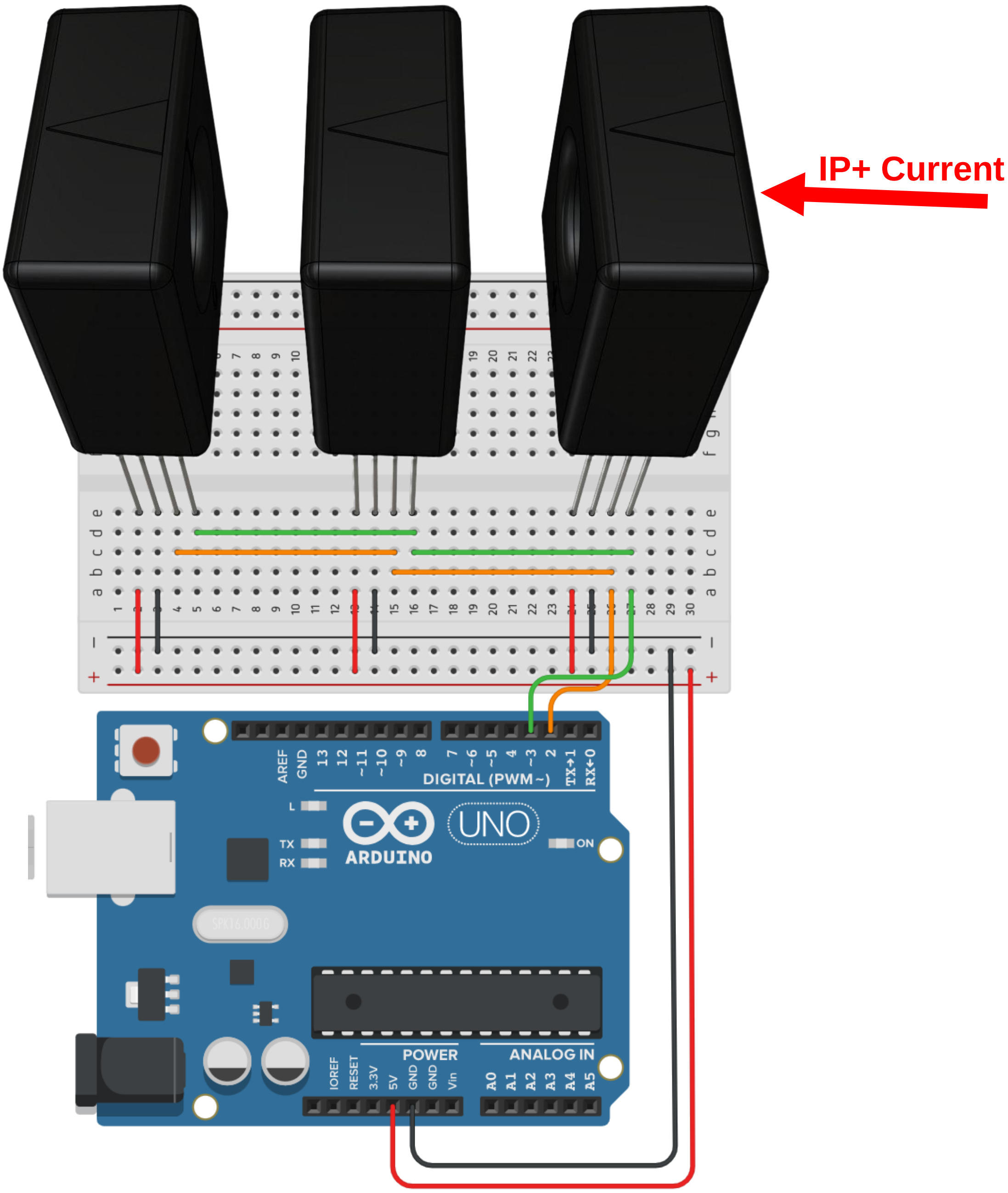
- **Schematic Diagram**



※Each DWCS should have its own unique slave address.
(Change DWCS slave address see previous example.)

Winson reserves the right to make changes to improve reliability or manufacturability.

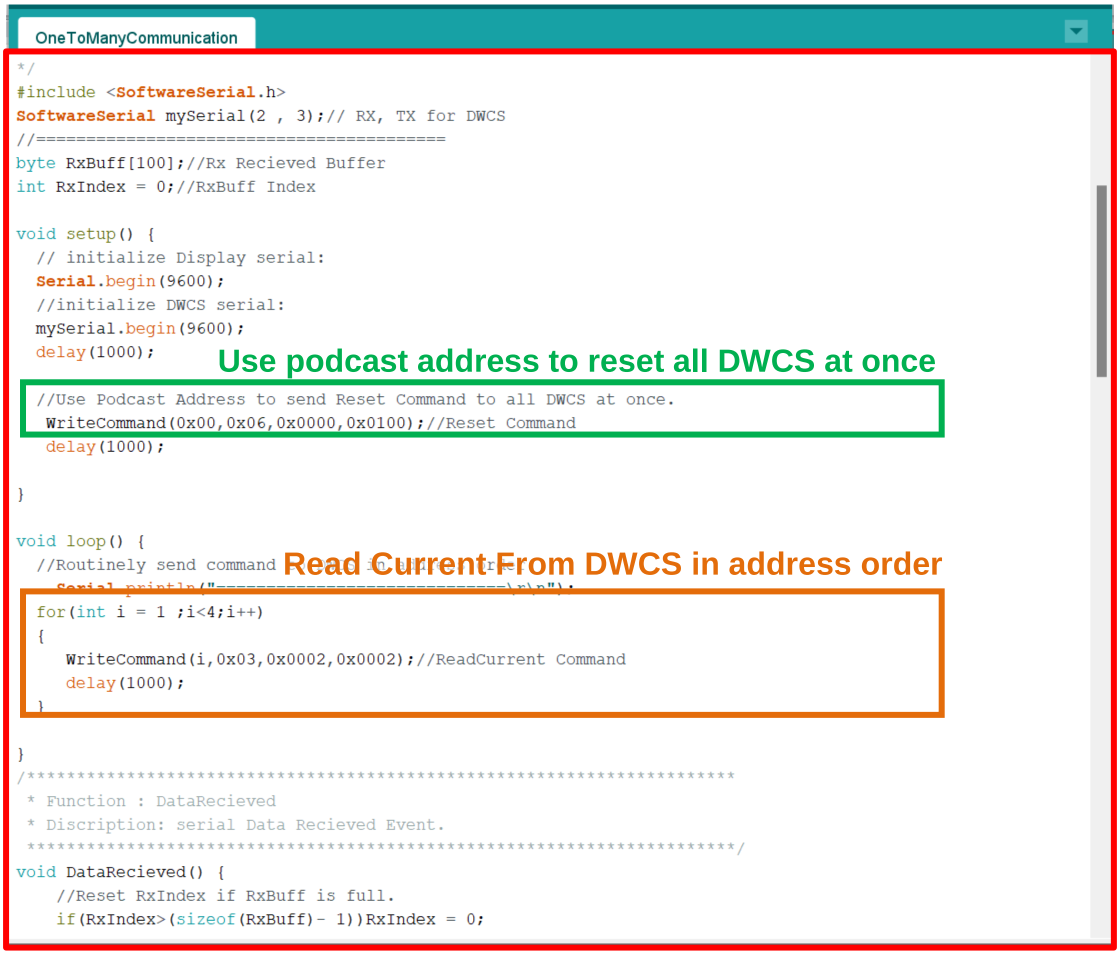
- **Wiring Diagram**



Winson reserves the right to make changes to improve reliability or manufacturability.

● Software & Program

(5). Example code can be download at:http:/www.winson.com.tw/Product/156



```
*/
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2 , 3);// RX, TX for DWCS
//==========================================
byte RxBuff[100];//Rx Recieved Buffer
int RxIndex = 0;//RxBuff Index

void setup() {
  // initialize Display serial:
  Serial.begin(9600);
  //initialize DWCS serial:
  mySerial.begin(9600);
  delay(1000);
```

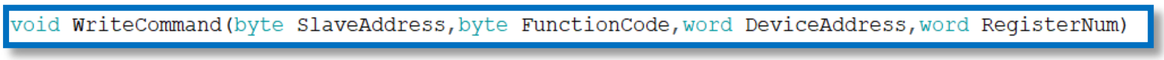
Use podcast address to reset all DWCS at once

```
  //Use Podcast Address to send Reset Command to all DWCS at once.
   WriteCommand(0x00,0x06,0x0000,0x0100);//Reset Command
   delay(1000);

}

void loop() {
  //Routinely send command to DWCS in address order
    Serial.println("=============================\r\n");
```

Read Current From DWCS in address order

```
  for(int i = 1 ;i<4;i++)
  {
    WriteCommand(i,0x03,0x0002,0x0002);//ReadCurrent Command
    delay(1000);
  }

}
/*************************************************************************
 * Function : DataRecieved
 * Discription: serial Data Recieved Event.
 *************************************************************************/
void DataRecieved() {
    //Reset RxIndex if RxBuff is full.
    if(RxIndex>(sizeof(RxBuff)- 1))RxIndex = 0;
```

Write Command [Read(03H) / Write(06H)]:

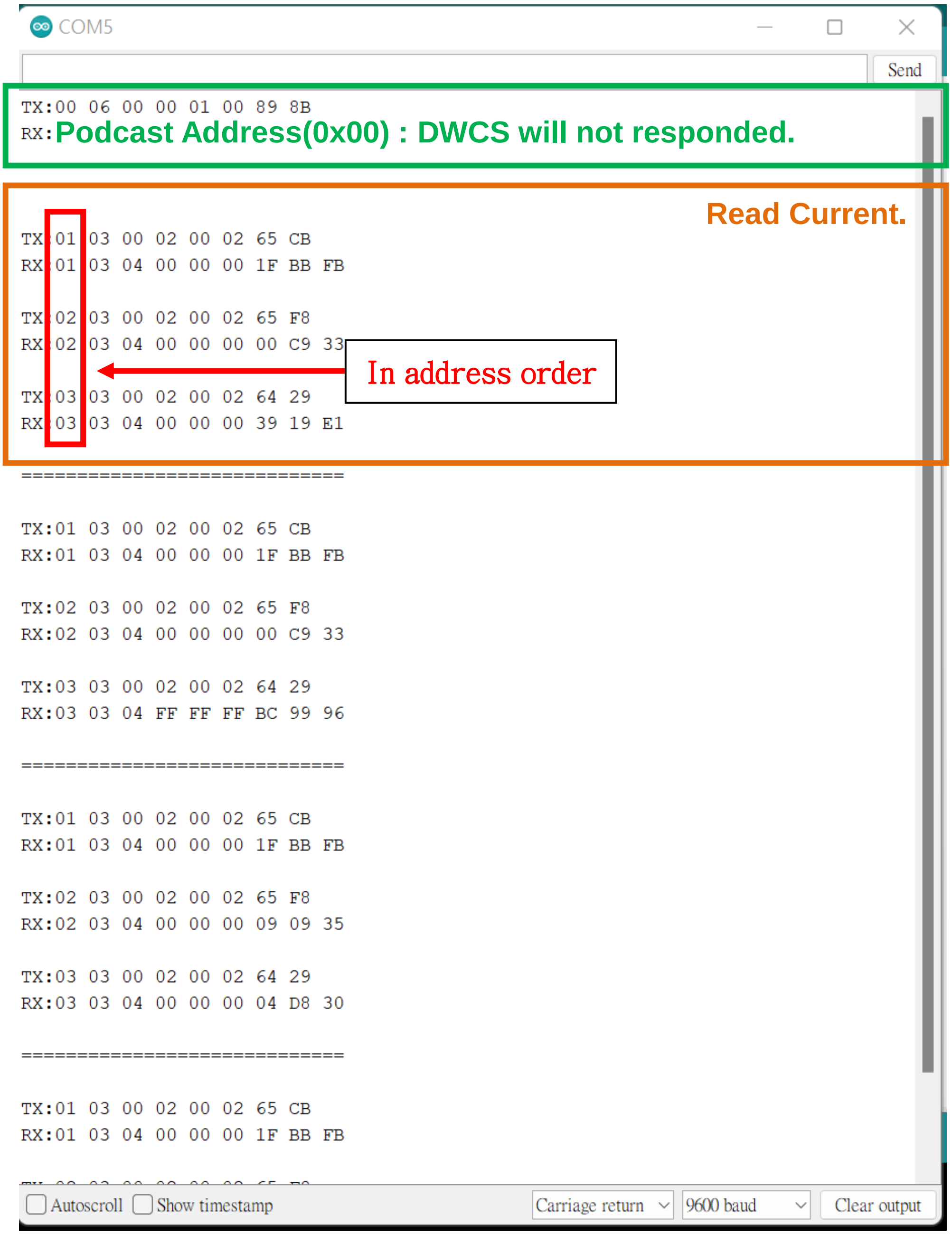


```
void WriteCommand(byte SlaveAddress,byte FunctionCode,word DeviceAddress,word RegisterNum)
```

※CAUTION!! To prevent upload failure of Arduino, please insert DWCS after upload process.

Winson reserves the right to make changes to improve reliability or manufacturability.

(6). Upload the example code and open Serial Monitor to display the return value. 。



```
TX:00 06 00 00 01 00 89 8B
RX:

TX:01 03 00 02 00 02 65 CB
RX:01 03 04 00 00 00 1F BB FB

TX:02 03 00 02 00 02 65 F8
RX:02 03 04 00 00 00 00 C9 33

TX:03 03 00 02 00 02 64 29
RX:03 03 04 00 00 00 39 19 E1

=============================

TX:01 03 00 02 00 02 65 CB
RX:01 03 04 00 00 00 1F BB FB

TX:02 03 00 02 00 02 65 F8
RX:02 03 04 00 00 00 00 C9 33

TX:03 03 00 02 00 02 64 29
RX:03 03 04 FF FF FF BC 99 96

=============================

TX:01 03 00 02 00 02 65 CB
RX:01 03 04 00 00 00 1F BB FB

TX:02 03 00 02 00 02 65 F8
RX:02 03 04 00 00 00 09 09 35

TX:03 03 00 02 00 02 64 29
RX:03 03 04 00 00 00 04 D8 30

=============================

TX:01 03 00 02 00 02 65 CB
RX:01 03 04 00 00 00 1F BB FB
```

Podcast Address(0x00) : DWCS will not responded.

Read Current.

In address order