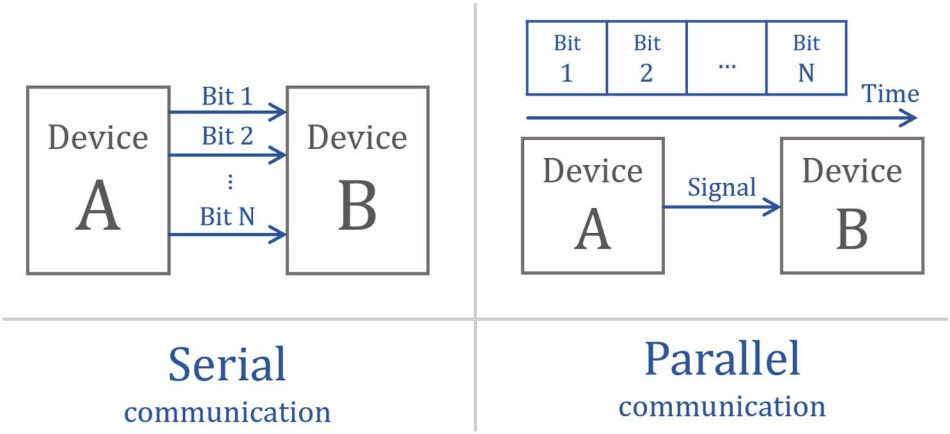
實驗操作已於學期初 Demo

REPORT

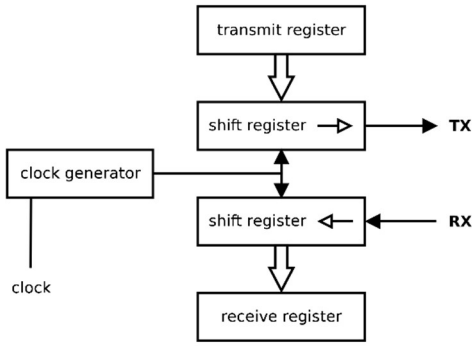
Serial communication

Serial communication is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus. This is in contrast to parallel communication, where several bits are sent as a whole, on a link with several parallel channels. The advantages of serial communication is that it use less wire than parallel communication, so the complexity of the communication circuit is lower than the parallel one. But serial communication usually takes more time than parallel communication. Hence, serial communication is more suitable for small data.

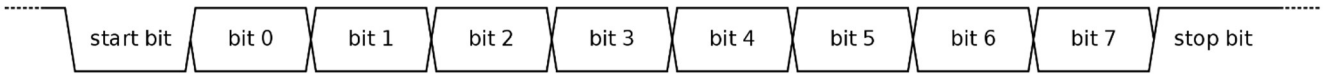


Experiment 1: UART protocol

UART protocol



A **Universal Asynchronous Receiver/Transmitter** (UART) is a hardware device for asynchronous serial communication. It's data format and transmission speeds are configurable. It was one of the earliest computer communication devices. Also, it's simple and easy to understand.



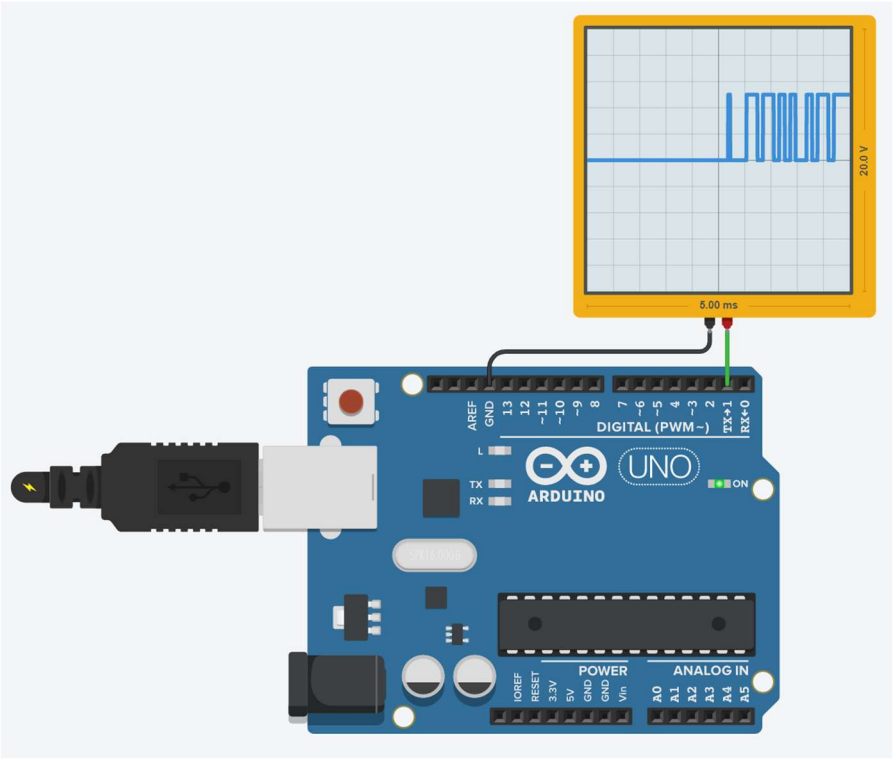
Serial.begin(speed [, config])

speed : Setting Baud, it's a integer parameter that decide the transmitting and receiving speed.
config : Determining the data format, the default setting is SERIAL_8N1.

Serial.print(str)

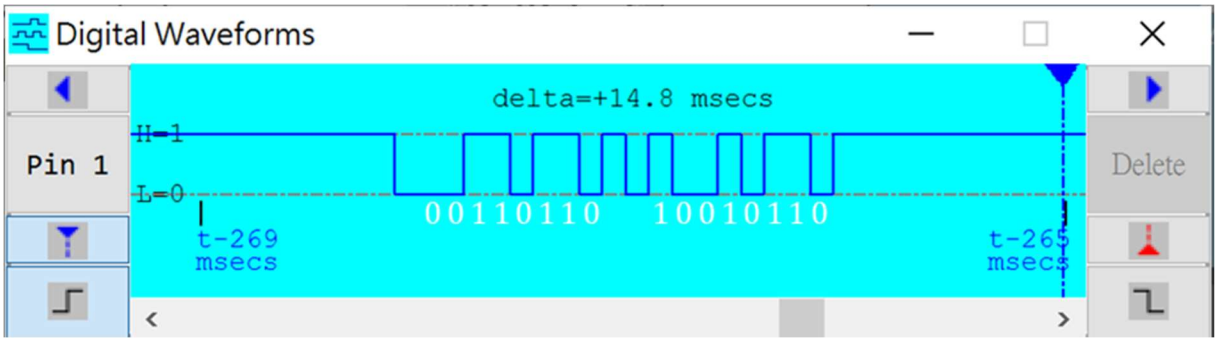
str : A string, which would be printed on serial monitor.

Circuit Analysis



We use the oscilloscope to observe the transient signal sent by Pin 1, which is the **transport pin** of serial communication of Arduino.

Simulation



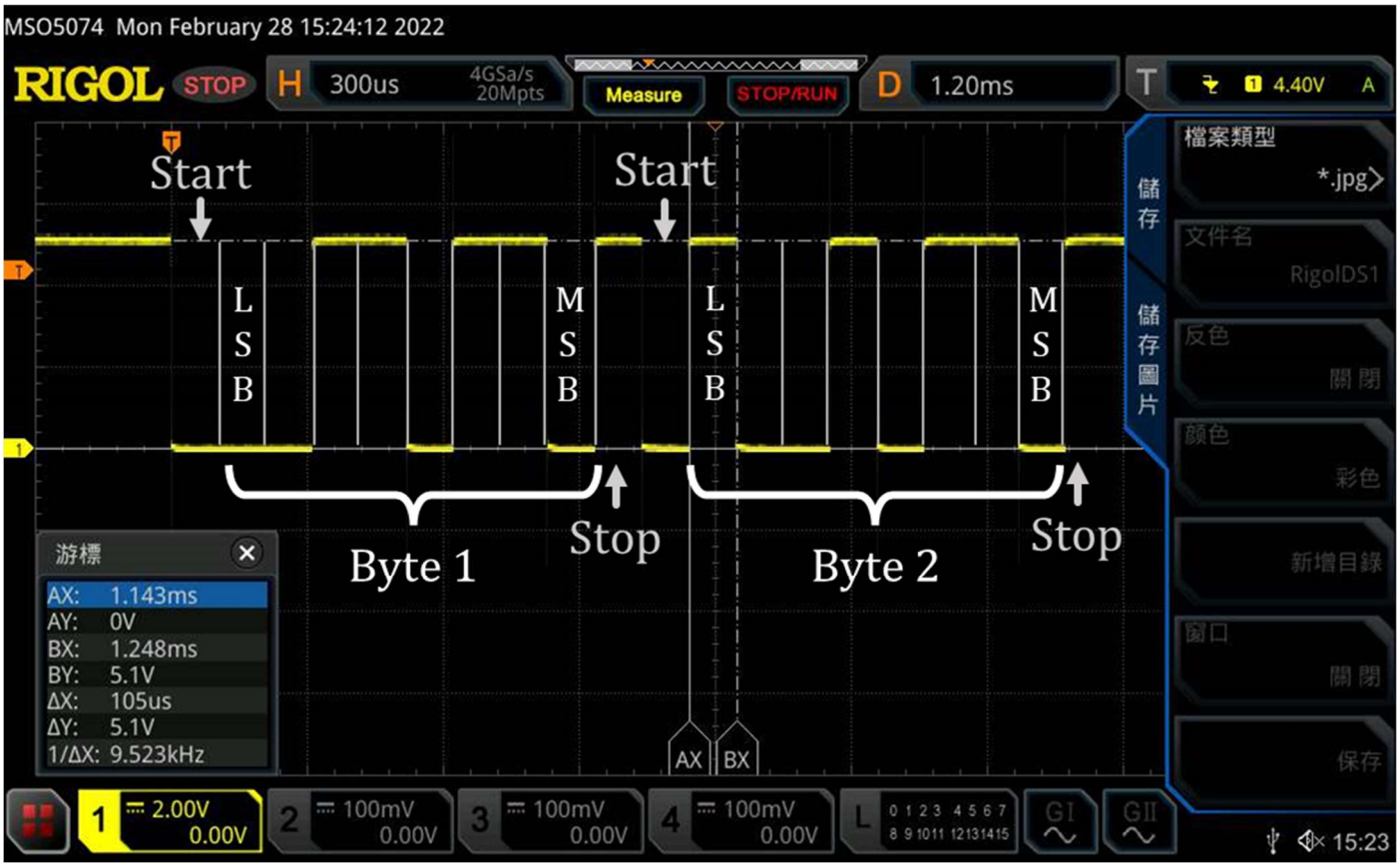
The corresponding alphabets of the binary ASCII code shown above is “li”. Since bit 7(MSB) is the rightmost bit and bit 0 is the leftmost bit, we need to read the binary numbers **from right to left** in each frame.

Sketch

```
void setup()
{Serial.begin(9600);}
void loop()
{
  Serial.print("li");
  delay(500);
}
```

Data

1. UART frame waveform



2. Frame content (Fill the blank with 0 or 1)

	START	Bit0 (LSB)	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7 (MSB)	STOP
1st frame	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**
2nd frame	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**

3. The interval of a bit is **108u** (second) which means the Baud rate is equal to **9524** (bps).

Baud rate is equals to the frequency of one bit.

Problem 1. The observed waveform is not complete.

Solution 1.

The output signal of UART protocol would keep high while there's no data transforming. In order to observe the start bit, it's better to change the trigger mode to **falling edge trigger** mode and adjust the trigger voltage and time shift properly.

Problem 2. How to measure the time interval of 1 bit.

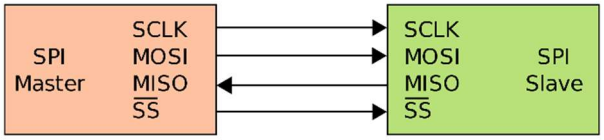
Solution 2.

Compare the observed waveform and the waveform of corresponding binary ASCII code to determine the position of every bit, then use the cursor to measure. Also, we could estimate it from the Baud rate.

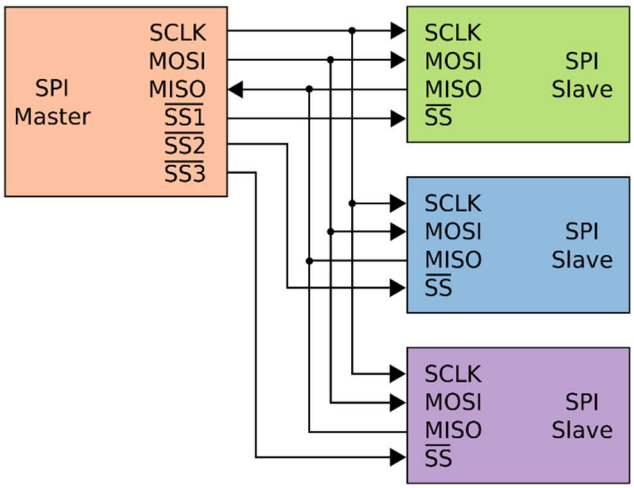
Experiment 2: SPI protocol

SPI protocol

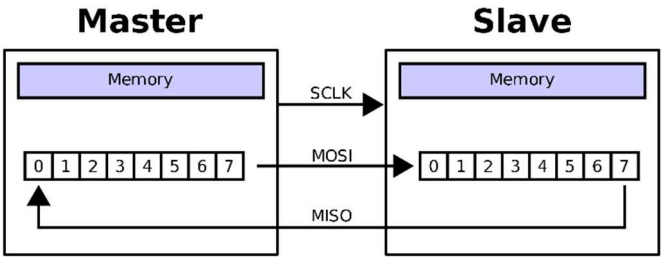
The **Serial Peripheral Interface (SPI)** is a synchronous serial communication. In comparison with UART, it can have multiple slave devices. It was developed by Motorola in the mid-1980s and it has become a de facto standard, which means it's commonly used.

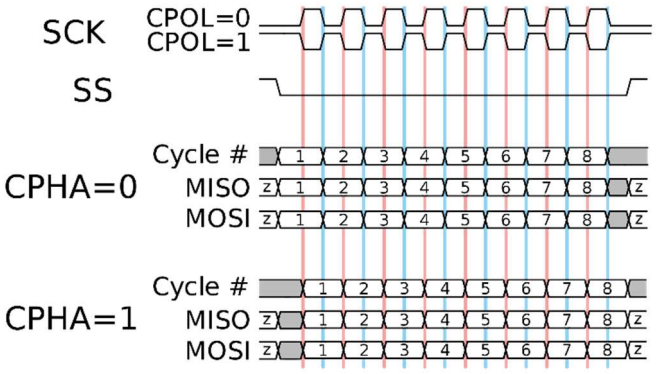


One slave device



Multiple slave devices





Serial Clock (SCLK) :pin13

Clock signal sent by Master for synchronizing.

Master Output, Slave Input (MOSI) :pin11

The data would be sent from the Master to **all** Slaves.

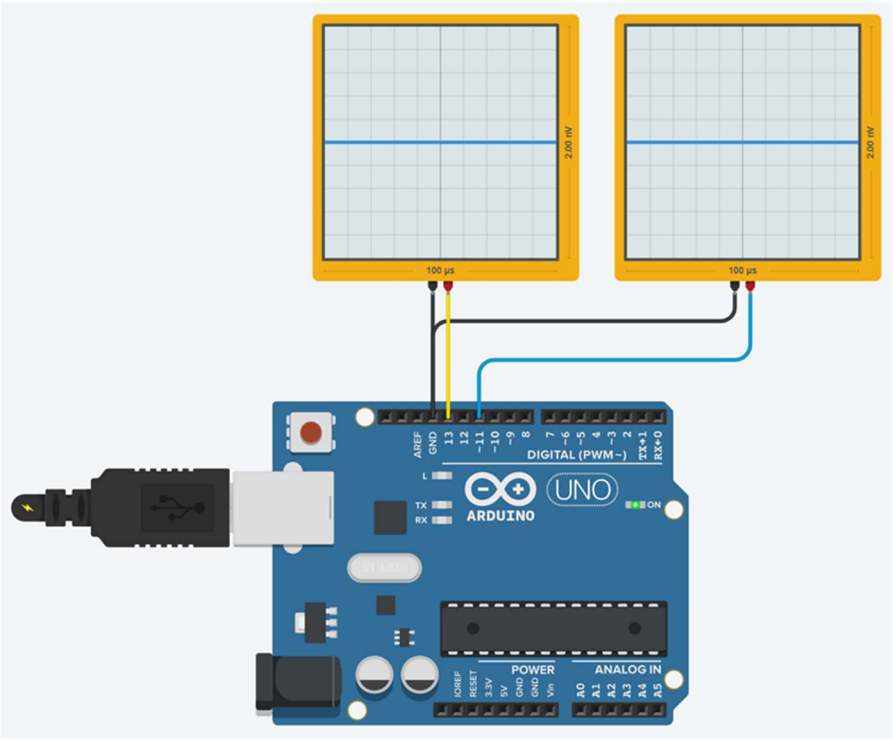
Master Input, Slave Output (MISO) :pin12

The data would be sent from Slaves to the Master.

Slave Select (SS) :pin10

This port would determine if the corresponding Slave would **read** the data sent by Master.
(It's active low, usually.)

Circuit Analysis



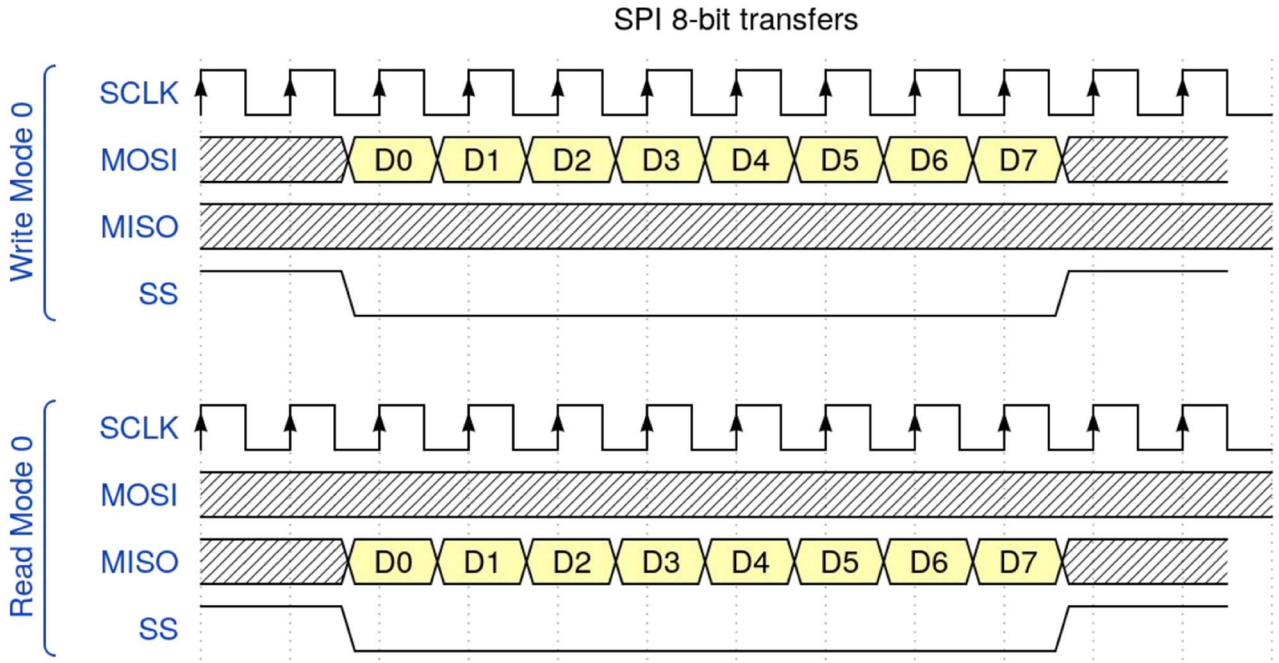
CH1: SCLK
Connect to pin 13.

CH2: MOSI
Connect to pin 11.

Problem 3. How to read the output signal of the master of SPI.

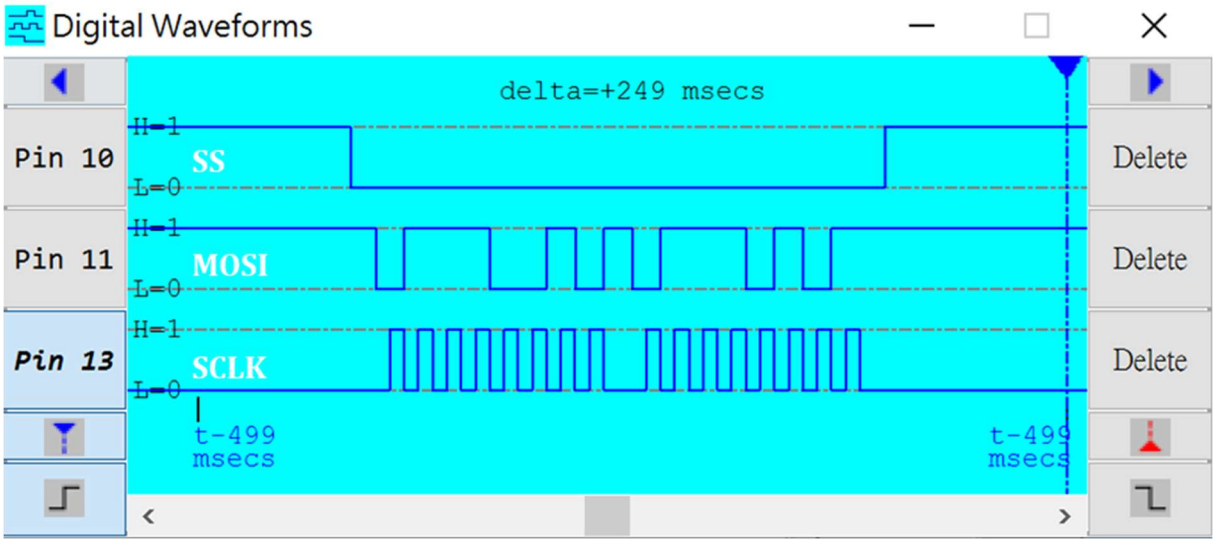
Solution 3.

The output bits are aligned with the **rising edge**s of SCLK. While implementing this experiment, use the cursor or adjusting the trigger level position would help a lot. Since the bit 7 is the left most bit and bit 0 is the rightmost bit, we need to read the binary numbers **from left to right**.



#REF: https://ithelp.ithome.com.tw/articles/10245910

Simulation



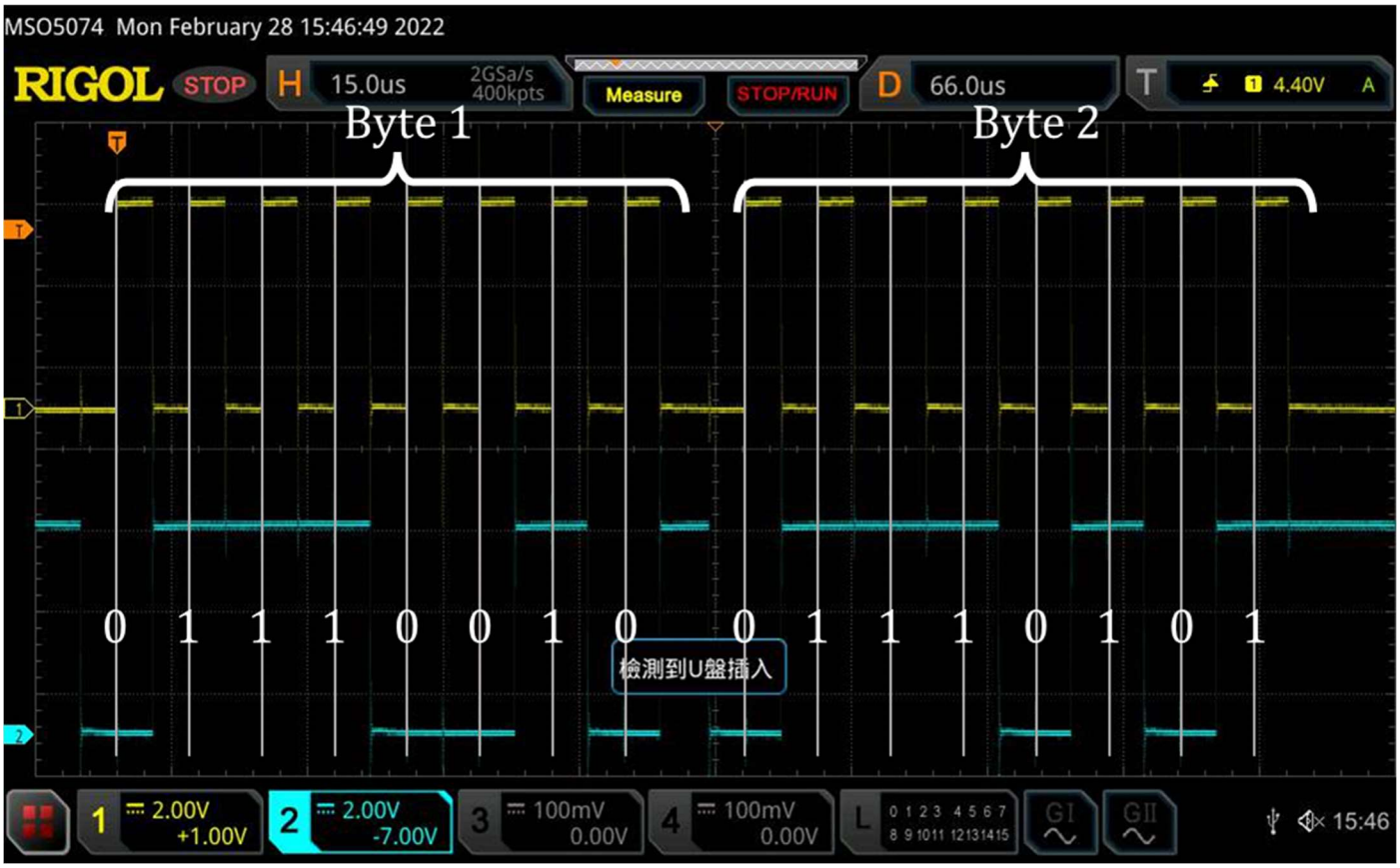
Result: 01110010 01110101 => “ru”

Sketch

```
#include<SPI.h>

void setup()
{
  Serial.begin(115200);
  digitalWrite(SS,HIGH);
  SPI.begin();
  SPI.setClockDivider(SPI_CLOCK_DIV128);
}

void loop()
{
  char c;
  digitalWrite(SS,LOW);
  for(const char *p="ru"; c = *p;  p++)
  {
    SPI.transfer(c);
    Serial.print(c); //check if it works from serial monitor
  }
  digitalWrite(SS,HIGH);
  delay(2000);
}
```

Data

1. SPI frame waveform



2. Frame content (Fill the blank with 0 or 1)

	Bit7 (MSB)	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 (LSB)
1st frame	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**0**
2nd frame	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**1**

3. The frequency of the SCK is equal to **116.9k** Hz

Problem 4. Is it possible to trigger properly without the SCLK signal.

Solution 4.

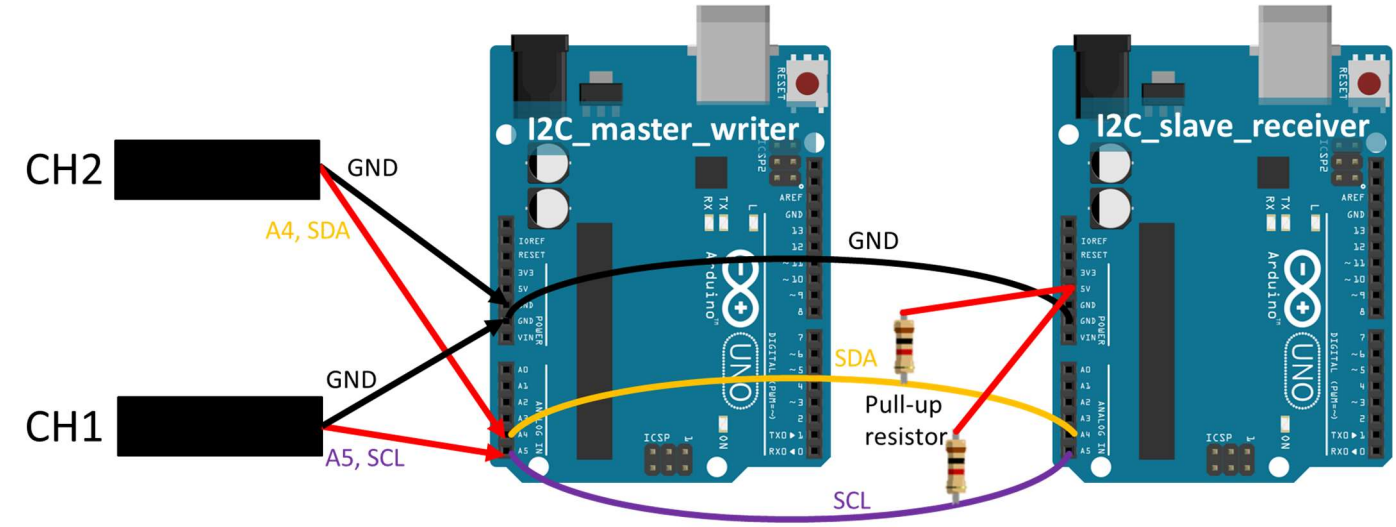
The **SS** also told us the starting position of the signal. Hence, there are at least 2 ways to set the trigger conditions which are easy to operate.

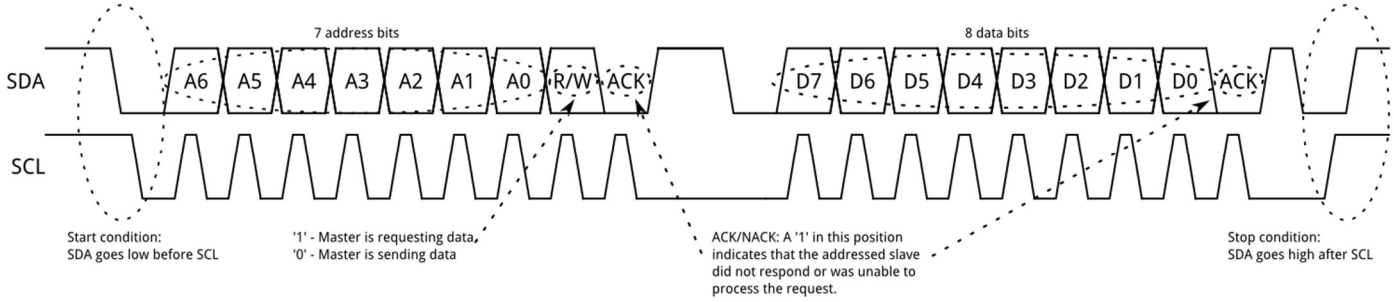
Method 1. SCLK as the trigger signal.
Using the **rising edge trigger mode**.

Method 2. SS as the trigger signal.
Using the **falling edge trigger mode**. (If SS is active low)

Experiment 3: I2C protocol

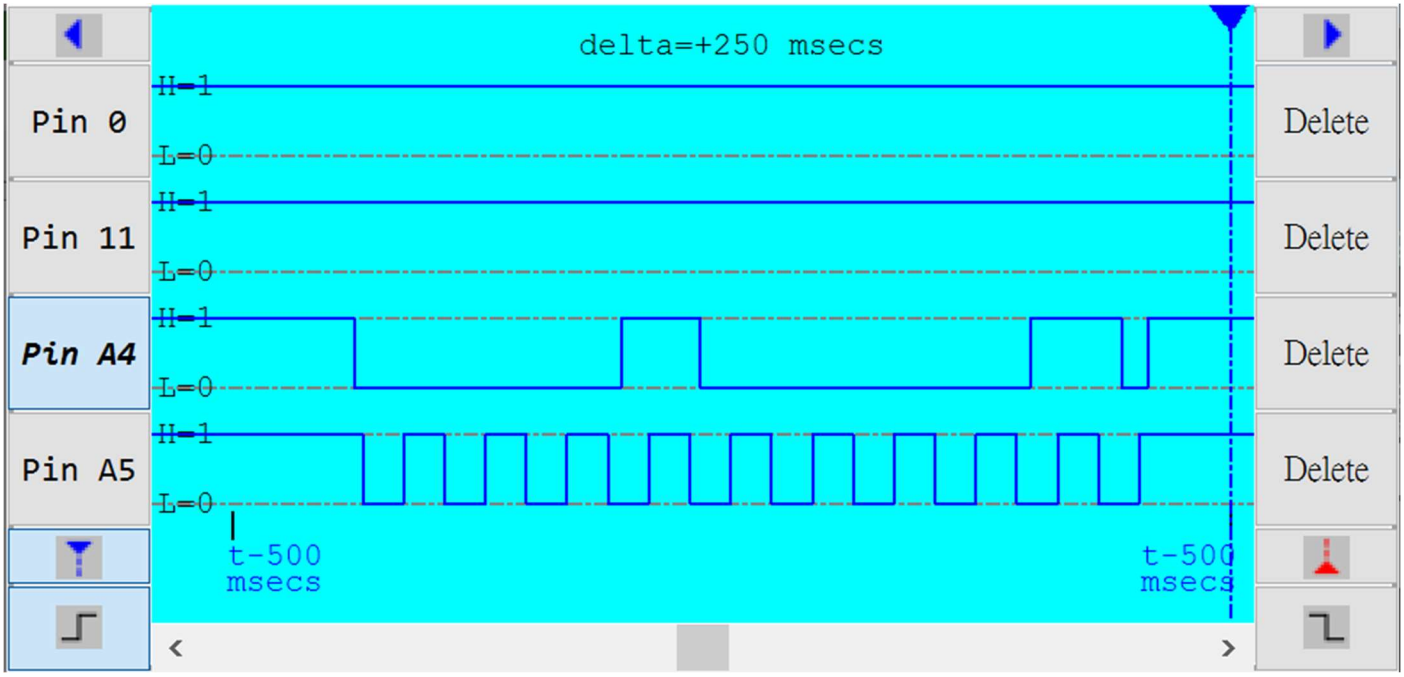
Circuit





The first byte of I²C Bus is consist of 7 address bits and a Read(1)/Write(0) bit, the following bytes are the data.

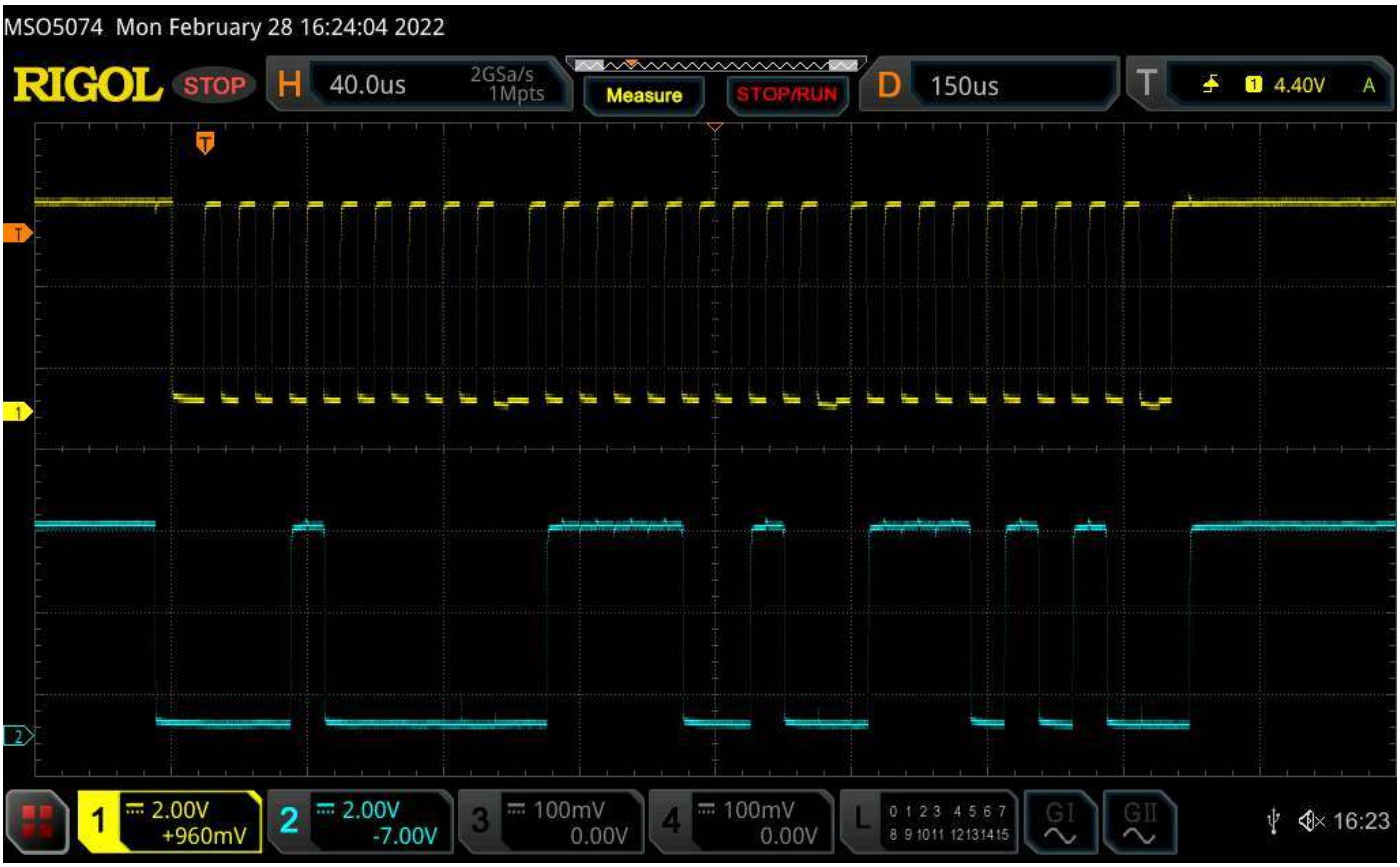
Simulation



This is the first byte, but the simulation software seems to be unable to simulate I²C write mode, so the data bytes didn't appear.

Data

1. I2C frame waveform



3 Bytes, read from left to right, SCL falling edge trigger.

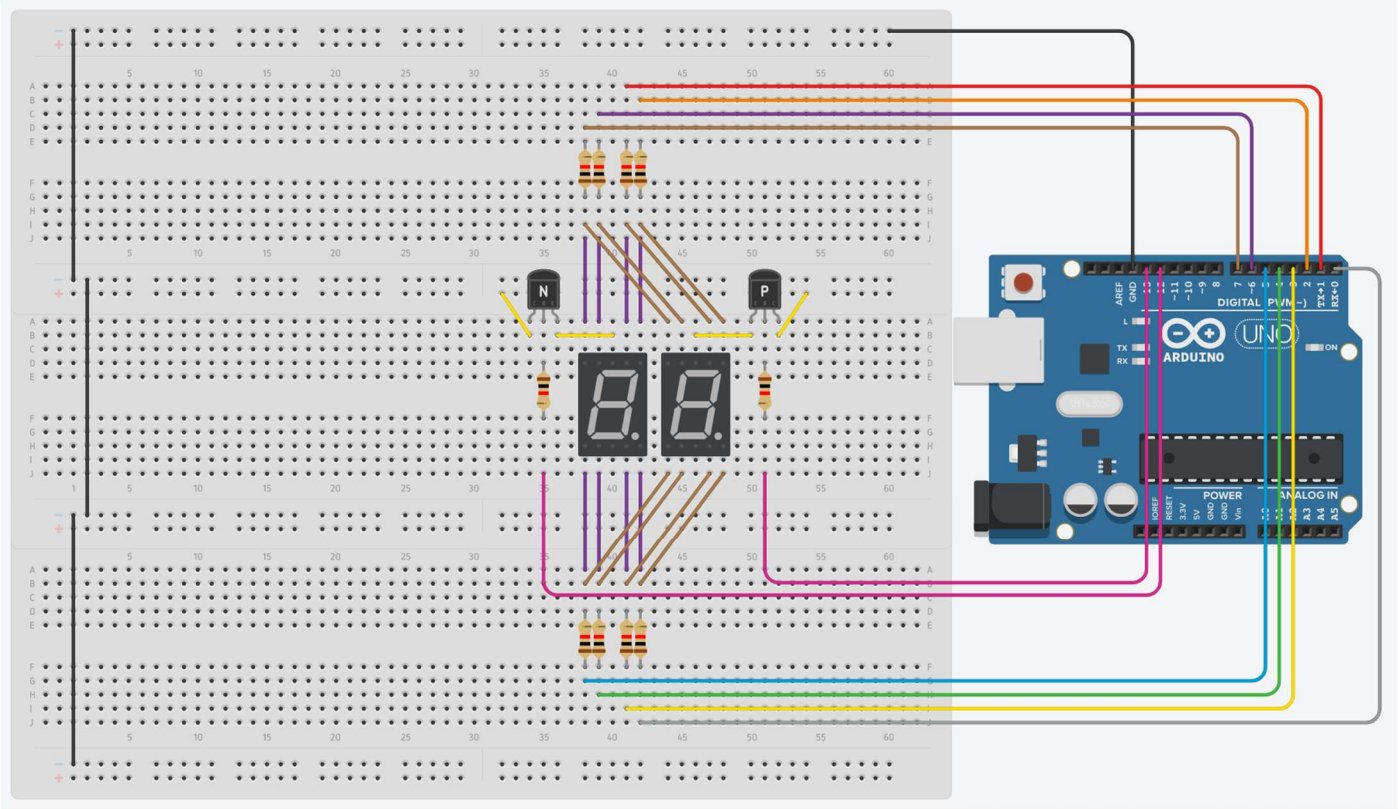
2. Frame content (Fill the blank with 0 or 1)

	Address6	Address5	Address4	Address3	Address2	Address1	Address0	Read /Write	ACK
1st frame	0	0	0	1	0	0	0	0	0
	Bit7 (MSB)	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 (LSB)	ACK
2nd frame	0	1	1	1	1	0	0	1	0
3rd frame	0	1	1	1	0	1	0	1	0

3. The frequency of the SCL is equal to **100k** Hz

Experiment 4: 2-digit 7-segment display

Circuit Analysis



Here I use Pin 0 to control Pin DP of the 7-segment display, and Pin 1~8 is for Pin A~F of that. Pin 12 and Pin 13 with the transistors and resistors plays the role of switch, they control the VCC of two 7-seg display. This circuit is kind of an application of SPI protocol, the 7-seg display could be seen as the slave devices, and the Arduino is the master devices. Pin 1~8 are the MOSI pins, and Pin 12 and Pin 13 are SS pins.

Considering the persistence of vision, we could switch the 7-seg display quickly so that it seems to be able to display 2 numbers simultaneously.

Data

The sketch of your design: (copy from the Arduino IDE window and paste here)

```
int i,m,n;
int d1=12;
int d2=13;
float x;
int ID = 8;
int cycle = 100;
byte pinA=1, pinB=2, pinC=3, pinD=4, pinE=5, pinF=6, pinG=7, pinP=0;
```

```
void setup()
{
  pinMode(pinA, OUTPUT); pinMode(pinB, OUTPUT); pinMode(pinC, OUTPUT);
  pinMode(pinD, OUTPUT); pinMode(pinE, OUTPUT); pinMode(pinF, OUTPUT);
  pinMode(pinG, OUTPUT); pinMode(pinP, OUTPUT);
  pinMode(d1, OUTPUT);   pinMode(d2, OUTPUT);
}

void showSevenSeg(byte A, byte B, byte C, byte D, byte E, byte F, byte G, byte P) {
  digitalWrite(pinA, A); digitalWrite(pinB, B); digitalWrite(pinC, C);
  digitalWrite(pinD, D); digitalWrite(pinE, E); digitalWrite(pinF, F);
  digitalWrite(pinG, G); digitalWrite(pinP, P);
}

void num(int n)
{
  switch(n)
  {
    case 0:
      showSevenSeg(0,0,0,0,0,0,1,1);
      break;
    case 1:
      showSevenSeg(1,0,0,1,1,1,1,1);
      break;
    case 2:
      showSevenSeg(0,0,1,0,0,1,0,1);
      break;
    case 3:
      showSevenSeg(0,0,0,0,1,1,0,1);
      break;
    case 4:
      showSevenSeg(1,0,0,1,1,0,0,1);
      break;
    case 5:
      showSevenSeg(0,1,0,0,1,0,0,1);
      break;
    case 6:
      showSevenSeg(0,1,0,0,0,0,0,1);
      break;
    case 7:
      showSevenSeg(0,0,0,1,1,1,1,1);
```

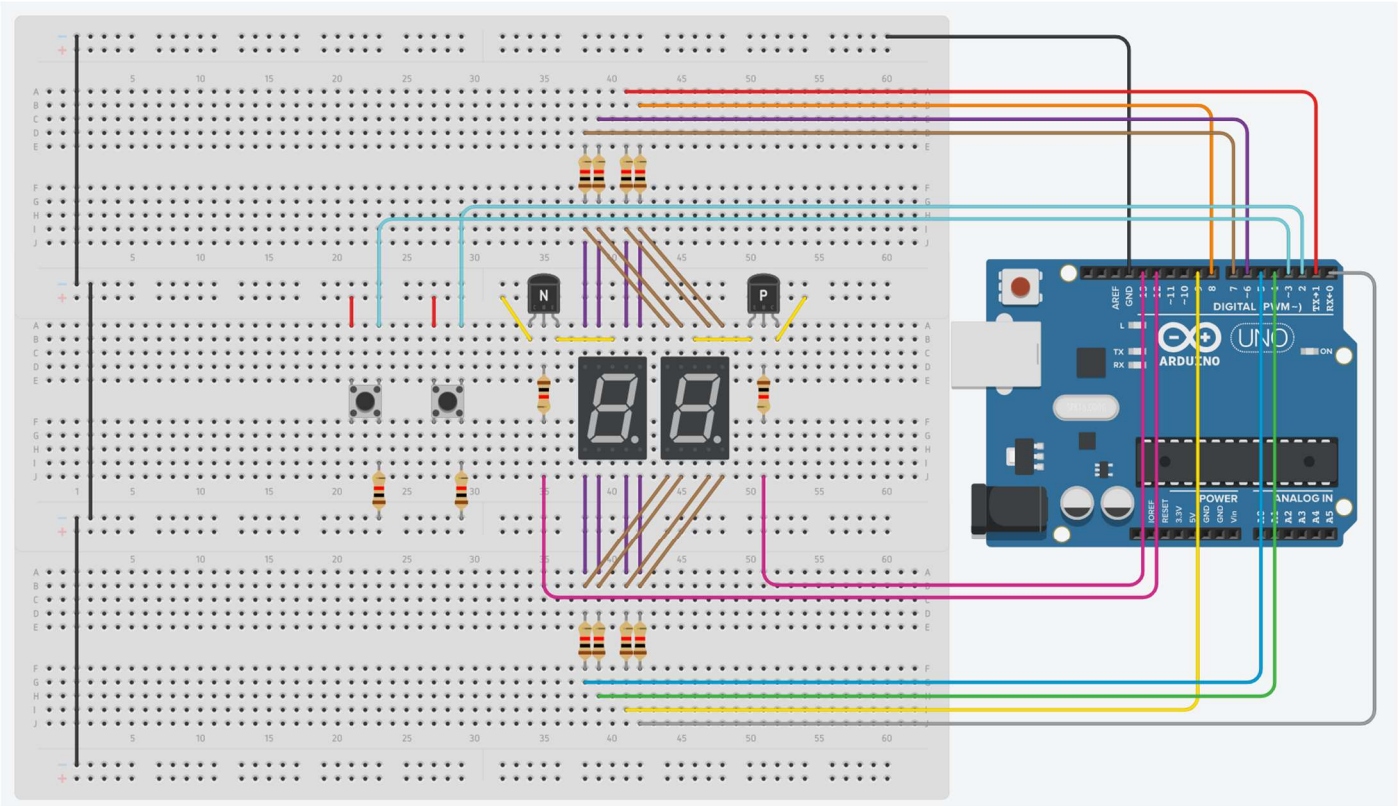
```
      break;
    case 8:
      showSevenSeg(0,0,0,0,0,0,0,1);
      break;
    case 9:
      showSevenSeg(0,0,0,0,1,0,0,1);
      break;
    case 10:
      showSevenSeg(1,1,1,1,1,1,1,1);
      break;
  }
}

void Display(int tmp)
{
  i=cycle;
  m=tmp/10;
  n=tmp%10;
  while(i--)
  {
    digitalWrite(d1,HIGH);
    digitalWrite(d2,LOW);
    num(m);
    delay(5);
    digitalWrite(d1,LOW);
    digitalWrite(d2,HIGH);
    num(n);
    delay(5);
  }
}
void loop()
{
  Display(ID);
}
```

Take a photo of your display content.
https://youtu.be/vghNlIEzciM

BONUS 1: 2-bit Counter Timer

Demo: https://youtu.be/4Xa7UOvNkbQ

Circuit Analysis



Sketch

```
int i,m,n;

//ctrl
int d1=12;
int d2=13;

//button (INTR)
int b1=2;
int b2=3;

float x;
bool sw1,sw2,en;
int dt = 1;
int Time = 0;
int cycle = 100; //T = 10ms => 100 cycles = 1sec;
byte pinA=1, pinB=8, pinC=9, pinD=4, pinE=5, pinF=6, pinG=7, pinP=0;
//B,C move to pin 8,9 cuz 2,3 is for INTR
```

```
void rev(){if(en){dt=-dt;en=!en;}}

void setup()
{
  pinMode(pinA, OUTPUT); pinMode(pinB, OUTPUT); pinMode(pinC, OUTPUT);
  pinMode(pinD, OUTPUT); pinMode(pinE, OUTPUT); pinMode(pinF, OUTPUT);
  pinMode(pinG, OUTPUT); pinMode(pinP, OUTPUT);
  pinMode(d1, OUTPUT);   pinMode(d2, OUTPUT);   pinMode(b1, INPUT);   pinMode(b2, INPUT);
  attachInterrupt(digitalPinToInterrupt(b1), rev, RISING);
  en = 1;
}

void showSevenSeg(byte A, byte B, byte C, byte D, byte E, byte F, byte G, byte P)
{
  digitalWrite(pinA, A); digitalWrite(pinB, B); digitalWrite(pinC, C);
  digitalWrite(pinD, D); digitalWrite(pinE, E); digitalWrite(pinF, F);
  digitalWrite(pinG, G); digitalWrite(pinP, P);
}

void num(int n)
{
  switch(n)
  {
    case 0:
      showSevenSeg(0,0,0,0,0,0,1,1);
      break;
    case 1:
      showSevenSeg(1,0,0,1,1,1,1,1);
      break;
    case 2:
      showSevenSeg(0,0,1,0,0,1,0,1);
      break;
    case 3:
      showSevenSeg(0,0,0,0,1,1,0,1);
      break;
    case 4:
      showSevenSeg(1,0,0,1,1,0,0,1);
      break;
    case 5:
      showSevenSeg(0,1,0,0,1,0,0,1);
      break;
```

```
    case 6:
      showSevenSeg(0,1,0,0,0,0,0,1);
      break;
    case 7:
      showSevenSeg(0,0,0,1,1,1,1,1);
      break;
    case 8:
      showSevenSeg(0,0,0,0,0,0,0,1);
      break;
    case 9:
      showSevenSeg(0,0,0,0,1,0,0,1);
      break;
    case 10:
      showSevenSeg(1,1,1,1,1,1,1,1);
      break;
  }
}

void Display(int tmp,int x)
{
  m=tmp/10;
  n=tmp%10;
  while(x--)
  {
    digitalWrite(d1,HIGH);
    digitalWrite(d2,LOW);
    num(m);
    delay(5);
    digitalWrite(d1,LOW);
    digitalWrite(d2,HIGH);
    num(n);
    delay(5);
  }
}

void count()
{
  Time+=dt;
  en = 1;
  if(Time>=60)Time-=60;
  else if(Time<0)Time+=60;
```

```
}

void loop()
{
  Display(Time,100);
  count();
}
```

BONUS 2: 2-bit Stopwatch

Demo: https://youtu.be/6oMMVZipc9w

Circuit Analysis

Same as BONUS1.

Sketch

```
int i,m,n;

//ctrl
int d1=12;
int d2=13;

//button (INTR)
int b1=2;
int b2=3;

float x;
bool sw1,sw2,en;
bool dt = 1;
int Time = 0;
int cycle = 100; //T = 10ms => 100 cycles = 1sec;
int rec;

byte pinA=1, pinB=8, pinC=9, pinD=4, pinE=5, pinF=6, pinG=7, pinP=0;
//B,C move to pin 8,9 cuz 2,3 is for INTR

void runstop(){if(en){dt=!dt;en=!en;}}

void setup()
{
  pinMode(pinA, OUTPUT); pinMode(pinB, OUTPUT); pinMode(pinC, OUTPUT);
```

```
  pinMode(pinD, OUTPUT); pinMode(pinE, OUTPUT); pinMode(pinF, OUTPUT);
  pinMode(pinG, OUTPUT); pinMode(pinP, OUTPUT);
  pinMode(d1, OUTPUT);   pinMode(d2, OUTPUT);   pinMode(b1, INPUT);   pinMode(b2, INPUT);
  attachInterrupt(digitalPinToInterrupt(b1), runstop, RISING);
  en = 1;
}
void showSevenSeg(byte A, byte B, byte C, byte D, byte E, byte F, byte G, byte P)
{
  digitalWrite(pinA, A); digitalWrite(pinB, B); digitalWrite(pinC, C);
  digitalWrite(pinD, D); digitalWrite(pinE, E); digitalWrite(pinF, F);
  digitalWrite(pinG, G); digitalWrite(pinP, P);
}
void num(int n)
{
  switch(n)
  {
    case 0:
      showSevenSeg(0,0,0,0,0,0,1,1);
      break;
    case 1:
      showSevenSeg(1,0,0,1,1,1,1,1);
      break;
    case 2:
      showSevenSeg(0,0,1,0,0,1,0,1);
      break;
    case 3:
      showSevenSeg(0,0,0,0,1,1,0,1);
      break;
    case 4:
      showSevenSeg(1,0,0,1,1,0,0,1);
      break;
    case 5:
      showSevenSeg(0,1,0,0,1,0,0,1);
      break;
    case 6:
      showSevenSeg(0,1,0,0,0,0,0,1);
      break;
    case 7:
      showSevenSeg(0,0,0,1,1,1,1,1);
      break;
    case 8:
```

```
      showSevenSeg(0,0,0,0,0,0,0,1);
      break;
    case 9:
      showSevenSeg(0,0,0,0,1,0,0,1);
      break;
    case 10:
      showSevenSeg(1,1,1,1,1,1,1,1);
      break;
  }
}
void Display(int tmp,int x)
{
  m=tmp/10;
  n=tmp%10;
  while(x--)
  {
    digitalWrite(d1,HIGH);
    digitalWrite(d2,LOW);
    num(m);
    delay(5);
    digitalWrite(d1,LOW);
    digitalWrite(d2,HIGH);
    num(n);
    delay(5);
  }
}
void count()
{
  Time+=dt;
  en = 1;
  if(Time>=60)Time-=60;
  else if(Time<0)Time+=60;
  sw1=digitalRead(b1);
  if(sw1) rec++;
  if(rec>=5) {rec = 0;Time=0;}
}
void loop()
{
  Display(Time,100);
  count();
}
```

Reference

[1] Universal asynchronous receiver-transmitter, Wikipedia.
https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter

[2] Arduino 串列埠測試(UART), Web blog.
https://info.tcivs.tc.edu.tw/TeaPage_Open_Doc.aspx?UID=tungchii&GUID=b0cedc28-75ad-4971-bde2-14193aa49c76

[3] Serial Peripheral Interface, Wikipedia.
https://en.wikipedia.org/wiki/Serial_Peripheral_Interface

[4] De facto standard, Wikipedia.
https://en.wikipedia.org/wiki/De_facto_standard

[5] Arduino - SPI 通訊, Web blog.
http://www.tastones.com/zh-tw/tutorial/arduino/arduino_serial_peripheral_interface/

[6] I2C, Wikipedia.
https://zh.wikipedia.org/zh-tw/I%C2%B2C

心得

這次實驗主要著重在通訊的部分，很多都是以前沒碰過的東西，包含模擬軟體等等。實驗過程中有遇到一些問題，幸好大部分都有順利解決。在查詢資料時，花了很多時間去理解各種訊協定，雖然不能說瞭解得很透徹，但也有學到不少東西。尤其是後面用少少的線就可以控制很多 LED 的部分真的蠻有趣的，雖然程式碼要寫得好維護的話要寫一大堆 QQ。阿因為前面找資料花太多時間了，所以後面的內容就有點雜亂 Orz