Embedded Processors UESTC
Homework Assignment 2

Author: Changgang Zheng UoG ID: 2289258Z UESTC ID: 2016200302027

Q1. **[15 points]** Assume you have a 12-bit signed ADC. Let Vin be the analog voltage in volts and N be the digital ADC output. The input range of -5<=Vin<=+5V. The ADC digital output range is -2048<=N<=+2047. First, write a linear equation that relates Vin as a function of N. Next, rewrite the equation in fixed-point math assuming Vin is represented as a decimal fixed-point number with delta = 0.001V.

$$N_{total} = 2^M = 2^{12} = 4096$$

$$E_{FSR} = V_{RefHi} - V_{RefLow} = +5V - (-5V) = 10V$$

$$Q = \frac{E_{FSR}}{N_{total}} = \frac{10}{4096}$$

$$V_{in} = (2048 + N)Q + V_{RefLow} = (2048 + N)\frac{10}{4096} - 5 = \frac{5}{2048}N \;\; V$$

$$Fixed\ point\ number = I \cdot \Delta$$

$$\Delta = 0.001\text{v}$$

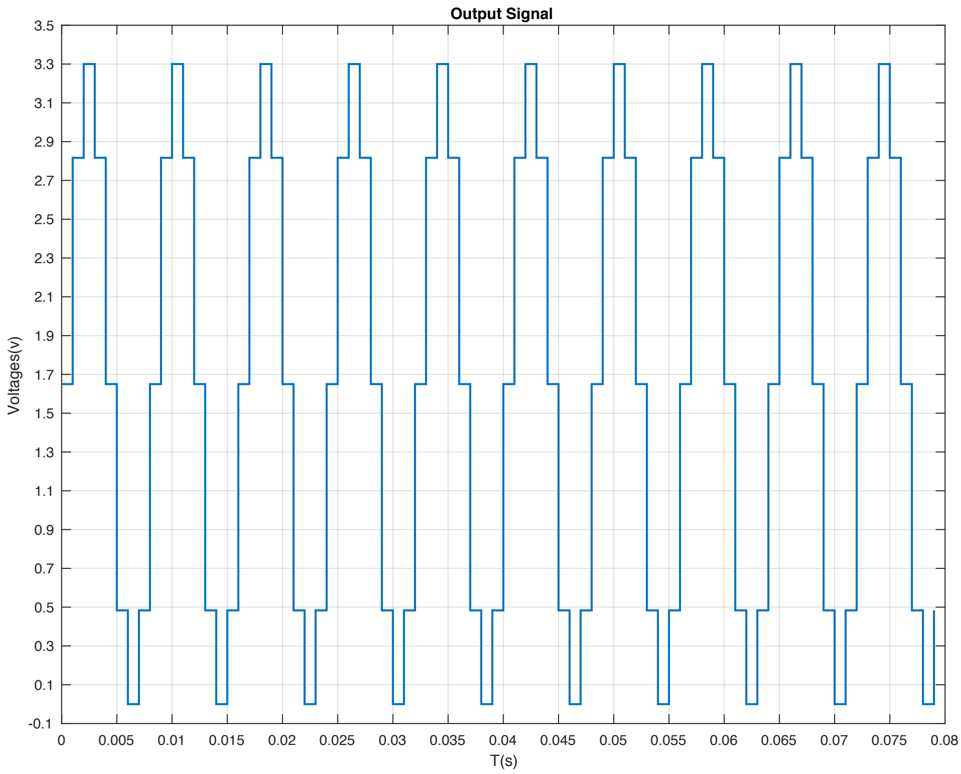
$$V_{in} = \frac{5}{2048}N = \left[\frac{5}{2048}N \cdot 1000\right] \cdot \Delta = \left[\frac{625}{256}N\right] \cdot \Delta = [2.44 \times N] \cdot \Delta \;\; V$$

('[]' means to apply 'Round', and 'V' means unit is volt)

Q2. **[15 points]** The following program produces a sine waveform through the DAC of a mbed development board. Sketch the output waveform.

```
#include "mbed.h"
AnalogOut Aout(p18);
float i;
int main()
{
    while(1)
    {
        for(i=0; i<2; i=i+0.25)
        {
            Aout = 0.5 + 0.5*sin(i*3.14159);
            wait(0.001);
         }
    }
}
```

The output signal is as follow:



Q3. [40 points] Describe the main differences between UART and SPI. Sketch the output waveform of pin 9 of the UART port when both switches are pressed, based on the following program.

```
/* Sending its own switch positions, and displaying those of the
other. */
#include “mbed.h"
Serial async port(p9, p10);      //set up TX and RX
DigitalOut red led(p25);
DigitalOut green led(p26);
DigitalOut strobe(p7);     //a strobe to trigger the scope
DigitalIn switch ip1(p5);
DigitalIn switch ip2(p6);
char switch word ;          //the word we will send

char recd_val;                 //the received value
int main() {
  async port.baud(9600);           //set baud rate to 9600
  while (1){
       switch word=0xa0;  //set up a recognizable output pattern
      if (switch ip1==1)
       switch word=switch word|0x01;   //OR in lsb
    if (switch ip2==1)
      switch word=switch word|0x02;   //OR in next lsb
      strobe =1;          //short strobe pulse
      wait_us(10);
      strobe=0;
      async_port.putc(switch_word); //transmit switch_word
     if (async_port.readable()==1)
         //is there a character to be read?
        recd_val=async_port.getc(); //if yes, then read it
        //set LEDs according to incoming word
        red_led = 0; //preset both to 0
        green_led = 0;
        recd_val=recd_val&0x03; //AND out unwanted bits
       if (recd_val==1)
         red_led=1;
      if (recd_val==2)
         green_led=1;
      if (recd_val==3){
          red_led=1;
         green_led=1;}
    }
}
```

Main difference:

The main difference is that for UART, no clock information is conveyed through the serial line. Before the transmission starts, the transmitter and receiver must agree on a set of parameters in advance: The baud-rate; number of bits per second; number of data bits and stop bits; use of parity bit or not. SPI based on the synchronous communication. SPI has Two types of devices, masters and slaves. Clock signal start when the transform begins, which means it should be master or the selected slave and there is no start bit and stop bit when data is transmitted.

UART receiver has a clock running at a multiple of the baud rate. UART has one connection for transmitted data, usually called TX, and another for received data, called RX. UART takes a parallel data stream and funnels it down to a serial data stream at the transmitter end and then returns the data stream to a parallel signal at the receiver end.

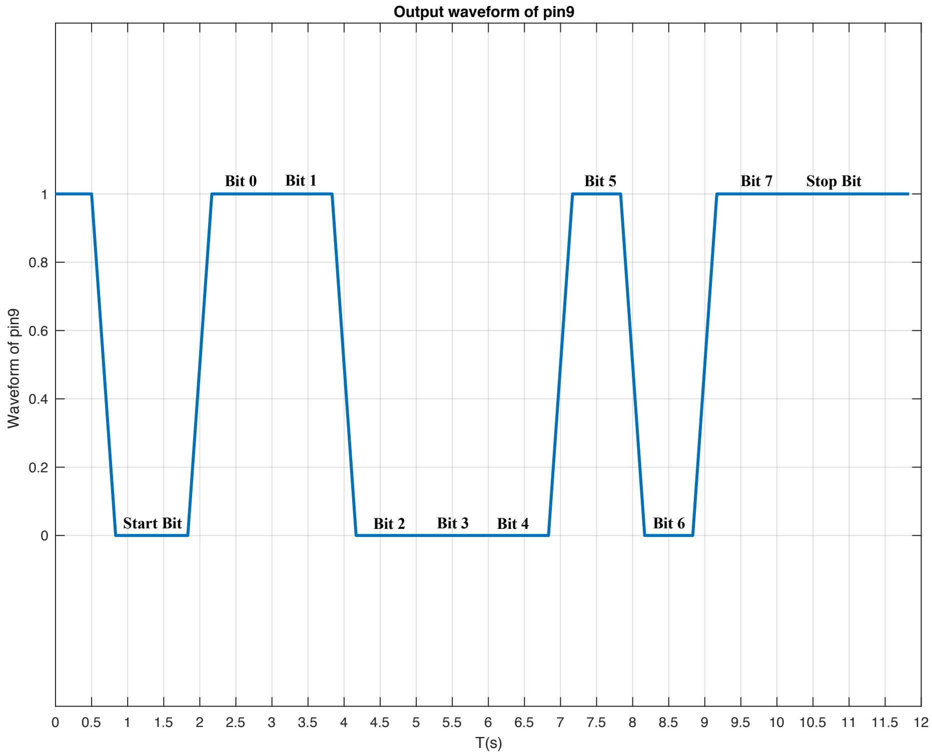
SPI is a based on the synchronous communication: Data is synchronized to the clock. When there is no data being sent, there is no signal on the clock line; Every time the clock pulses, one bit is output by the transmitter and should be read by the receiver; The receiver synchronizes its reading of the data with an edge of the clock. It has Two types of devices, masters and slaves. When the master initiates communication and selects a slave device, data can be transferred in both directions simultaneously. The SPI protocol uses four signals: SCLK, Serial Clock, set by Master; MOSI: master output, slave input; MISO: master input, slave output; SS (or CS): slave select or clock select. SPI is simple, convenient and low-cost, but not appropriate for complex or high reliable systems.

If both of the switches are pressed, the following are the output from pin9 (UART):

$$\text{switch\_word} = (0\text{xa}0|0\text{x}01)|0\text{x}02 = \mathbf{1010\ 0011}$$

Before the transformation, the waveform is relative high and the transformation begin with a 1.5 length relative low start bit. The end of the transformation is after 1.5 length relative high stop bit and the waveform rice back to relative high.

The output waveform of pin9 is as follow:



Output waveform of pin9
Waveform of pin9
T(s)
Start Bit
Bit 0
Bit 1
Bit 2
Bit 3
Bit 4
Bit 5
Bit 6
Bit 7
Stop Bit

Sketch the output of pin 11 and pin 13 of the SPI port when both switches are pressed, based on the following program. Show their relationship.

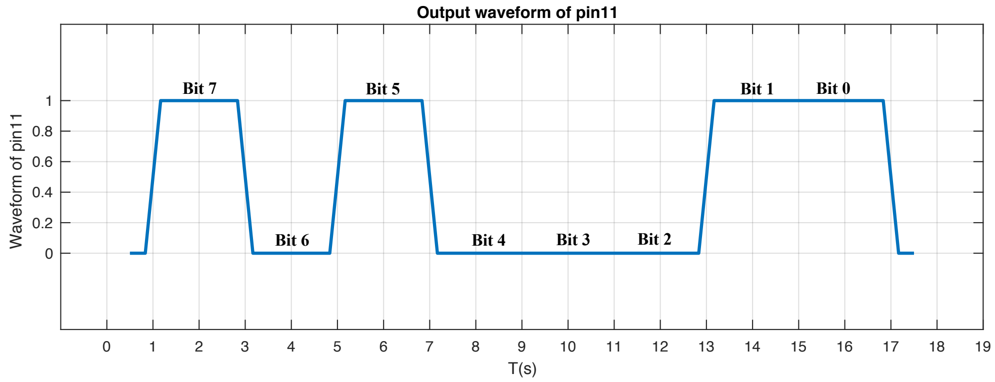
```
#include "mbed.h"
SPI  ser port(p11,p12,p13);   //mosi, miso, sclk
DigitalOut     red led(p25);       //red led
DigitalOut     green led(P26);          //green led
DigitalOut cs(p14);           //this acts as "slave select"
DigitalIn switch ip1(p5);
DigitalIn  switch ip2(p6);
char switch word;             //word we will send

char recd_val;                //value return from slave
int main(){
     while(1){
          //default settings from SPI Master chosen, no need
     for further //configuration
          //Set up the word to be sent, by testing switch inputs
          switch_word = 0xa0;
          //set up a recognizable output pattern
          if(switch_ip1==1)
               switch_word = switch_word|0X01; //OR in lsb
          if(switch_ip2==1)
               switch_word = switch_word/0x02;//OR in next
          1sb cs = 0;
          recd_val = ser_port.write(switch_word);
               //send switch_word and receive
               data
          cs = 1;
          wait(0.01);

          //set leds according to incoming word from slave
          red_led =0; //preset both to 0
          green_led = 0;
          recd_val = recd_val & 0x03; //And out unwanted bits
          if(recd_val ==1)
                red led = 1;
           if(recd val == 2)
                green led= 1;
          if(recd_Val ==3)
          {
               red_led = 1;
               green_led = 1;
          }
     }
}
```

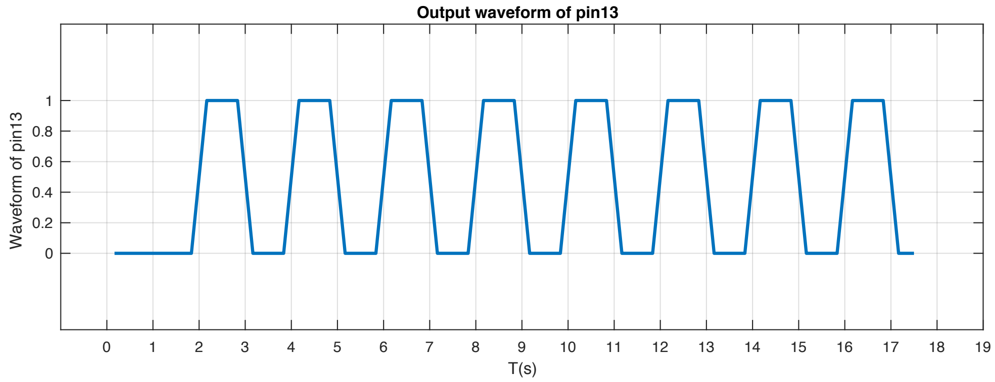
If both of the switches are pressed, the following are the output from pin11 (SPI):

$$\text{switch\_word} = (0xa0|0x01)|0x02 = \mathbf{1010\ 0011}$$

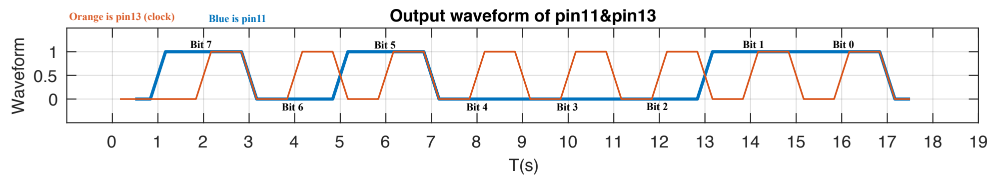
pin13 will be the clock select.

The output waveform of pin11 and pin13 (clock) are as follow:





The result can also be plotted together:



Q4. [15 points] Answer the following two questions related to timers and interrupts.

1) Briefly describe the definitions of hardware timers and interrupts, and the relationship between a timer and an interrupt.

2) The input of a 16-bit digital counter is connected to a clock signal. An interrupt occurs at the moment of overflow. If the frequency of the clock signal is 1MHz, and the interrupt frequency is 200Hz, to what value should the digital counter be initialized?

Question1:

Timer: A digital circuit that allows us to measure time, and hence make things happen when a certain time has elapsed.

Interrupts: A mechanism whereby a running program can be interrupted, with the CPU then being required to jump to some other activity.

I remember that professor mentioned an example in the lecture. Timer is something like you wake up every 2 minutes during the night to check if there are bad guys break into your house. Interrupts is a kind of alarm that if the bad buys come to your house, it would wake you up. The above are the definition and relationship of the interrupt and timer.

Question2: I assume that we do not reserve One bit in the API object as a sign bit:

$$x = initial\ value$$

$$y = input\ signal\ frequency = 1MHz$$

$$There\ is\ an\ 'n'\ bit\ digital\ counter.$$

$$n = 16$$

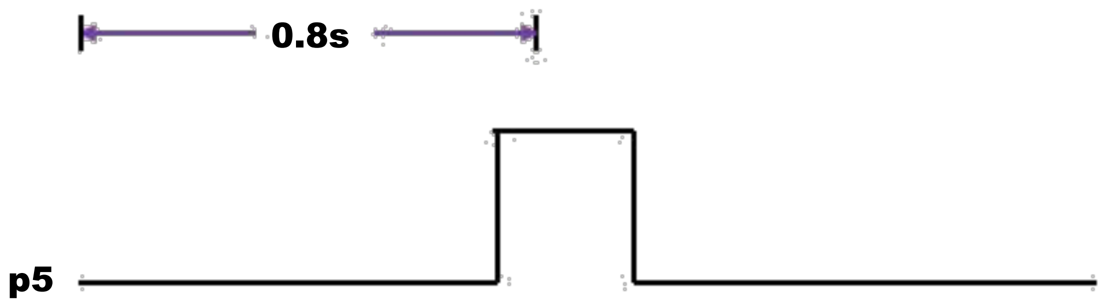
$$Interrupt\ frequency = \frac{y}{(2^n - x)} = 200Hz$$

$$200 = \frac{1\times10^6}{(2^{16} - x)}$$

$$x = \frac{65536\times200 - 1\times10^6}{200} = 60536 = binary\ number: 1110110001111000$$

Q5. [15 points] Sketch the output waveforms of LED1, LED2 and LED3, based on the following program. The input of pin5 is illustrated as follows.

```
#include "mbed.h"
void blink_end(void);
void blink(void);
void ISR1(void);
DigitalOut                    led1(LED1);
DigitalOut                    led2(LED2);
DigitalOut                    led3(LED3);
Timeout Response;
Timeout Response_duration;
InterruptIn                   button(p5);
void blink(){
     led2 = 1;
     Response_duration.attach(&blink_end, 0.5);
}
void blink_end(){
     led2 = 0;
}
void ISR1(){
     led3 = 1;
     Response.attach(&blink, 0.3);
}
int main(){
     button.rise(&ISR1);
     while(1){
          led3 = 0;
          led1 = !led1;
          wait(0.2);
     }
}
```



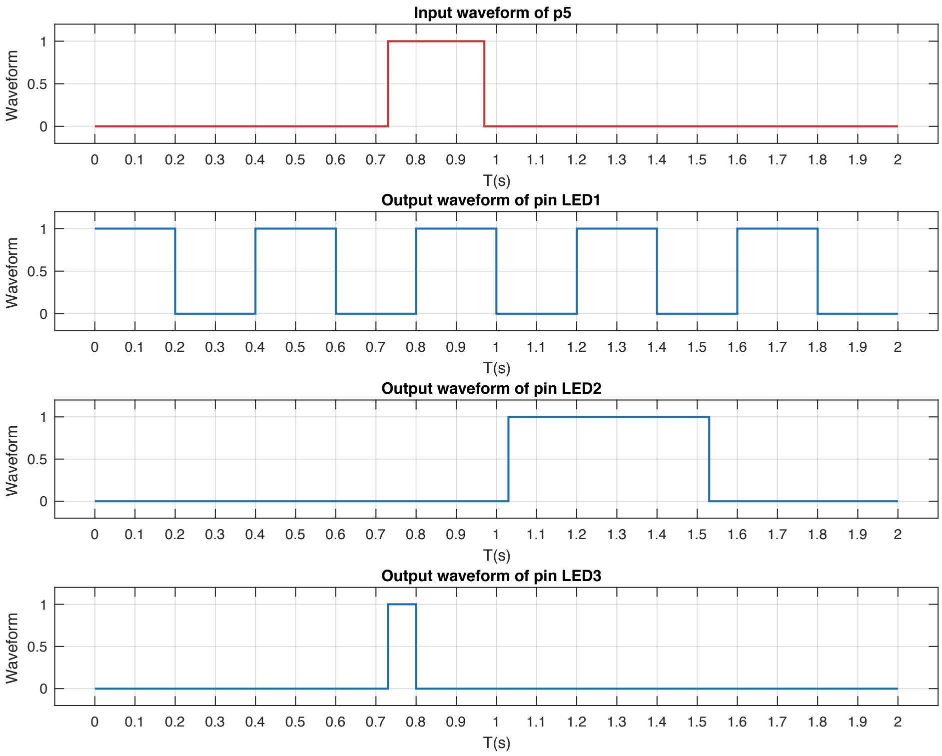
Q5. timers and interrupts.

[Note] The relationships between these outputs need to be demonstrated your figure.

Relationship:

From the program, we can know that before it begins the 'while' loop in the main function, it sets an interrupt, when it goes through the 'while' loop, the output waveform of LED1 rotate 1(high), 0 (low), 1, 0 … every 0.2s. After the interrupt event happened (upper edge of the p5 at t=7.3), the progress jumps from the main function and processes the function 'ISR1' (only at that moment and the time duration for processing is extremely small). Function 'ISR1' light the LED3 and initialize a ticker to trigger function 'blink'. At this time, the waveform of LED3 is high and it keeps this for only 0.07s until the 'while' loop in the 'main' function spin to the 'led3=0' and the waveform of LED3 become low. As the 'while' loop in the 'main' function continuous to spin. The time will be up and the ticker call the function 'blink'. At this time, LED2 becomes high for 0.5s ('while' loop in the 'main' function continuous to spin) and this function also initialize a ticker to trigger function 'blink_end'. After this, 'while' loop in the 'main' function continuous to spin and the function 'blink_end' will be triggered and the wave form of LED2 becomes low and the progress back to the 'main' function, and LED1 will continuous to rotate ('while' loop in the 'main' function continuous to spin).

The output waveforms are as follow:



Appendix

Program (used to generate all the wave form) based on the MATLAB are shown as follow:

```
clear;
clc;

%% problem 2
n=10;
Aout=zeros(1,8);
for k=1:n
    j=1;
    for i=0:0.25:1.75
        Aout(j)=0.5 + 0.5*sin(i*pi);
        j=j+1;
    end
    if k==1
        out=Aout;
    else
        out=[out Aout];
    end
end
t=[0:0.001:0.008*n-0.001];

%figure;
%scatter(t,out,'.')



figure;
stairs(t,(0.5+0.5*sin(t*pi*250))*3.3,'LineWidth',1)
grid on;
set(gca,'FontSize',7);
ylim([-0.1 3.5]);
xlim([0 0.08]);
xlabel('T(s)');
ylabel('Voltages(v)');
title('Output Signal');
set(gca,'XTick',0:0.005:0.08);
set(gca,'YTick',-0.1:0.2:3.5);
saveas(gca,'2.pdf');
%% problem 3Ôºç1
clear;

% t=[1:11];
% u=[ 1 1 0 1 0 0 0 1 1 0 1 ];

%{
figure;
stairs(t,u,'b','LineWidth',2);
ylim([-1 2])
```

```
hold on;
stairs(t(1:2),u(1:2),'r','LineWidth',2);
hold on;
stairs(t(10:11),u(10:11),'r','LineWidth',2);
%}

clear;

t=[1:12];
u=[ 1 1 1 0 0 0 1 0 1 1 1 1];

%{
figure;
stairs(t,u,'b','LineWidth',2);
ylim([-1 2])
hold on;
stairs(t(1:2),u(1:2),'r','LineWidth',2);
hold on;
stairs(t(10:11),u(10:11),'r','LineWidth',2);
%}

[u,t]=prosess(u,t);
t=[t(2)-1.6666 t(2)-1.3333 t(2)-0.3333 t(2:length(t)-1)];
t=t+0.5;
t=[0 t];
u=[1 1 0 0 u(2:length(u)-1)];

figure;
plot(t,u,'LineWidth',1.5);
ylim([-0.5 1.5]);
xlim([-0 12])
set(gca,'XTick',0:0.5:12);
set(gca,'YTick',0:0.2:1);
set(gca,'FontSize',7);
xlabel('T(s)');
ylabel('Waveform of pin9');
title('Output waveform of pin9');
grid on;
saveas(gca,'3-1.pdf');


%% problem 3Ôºç2

c=[0.5:15.5];
C=[0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0];
[C,c]=prosess(C,c);

C=[0 0 C 0 0 1 1 0 0];
c=[c(1)-0.3333 c(1) c(2)-0.3333 c(2:length(c)-1) c(length(c))-0.3333
c(length(c)) c(length(c))+0.3333 c(length(c))+0.3333*2 c(length(c))+0.3333*4
c(length(c))+0.3333*5 c(length(c))+0.3333*6];


t=[1:18];
s=[0 1 1 0 0 1 1 0 0 0 0 0 0 1 1 1 1 0];
[s,t]=prosess(s,t);
```

```
s=[0 0 s(2:length(s)) 0];
t=[t(1) t(1)+0.3333 t(2:length(t)-1) t(length(t))-0.3333 t(length(t))];
t=t-(t(6)-c(7));


figure;
subplot(2,1,1);
plot(t,s,'LineWidth',1.5);
ylim([-0.5 1.5]);
xlim([-1 19]);
set(gca,'XTick',0:1:19);
set(gca,'YTick',0:0.2:1);
set(gca,'FontSize',7);
xlabel('T(s)');
ylabel('Waveform of pin11');
title('Output waveform of pin11');
grid on;


subplot(2,1,2);
plot(c,C,'LineWidth',1.5);
ylim([-0.5 1.5]);
xlim([-1 19]);
set(gca,'XTick',0:1:19);
set(gca,'YTick',0:0.2:1);
set(gca,'FontSize',7);
xlabel('T(s)');
ylabel('Waveform of pin13');
title('Output waveform of pin13');
grid on;

saveas(gca,'3-2.pdf');


figure;
plot(t,s,'LineWidth',1.5);
hold on;
plot(c,C,'LineWidth',0.8);
ylim([-0.5 1.5]);
xlim([-1 19]);
set(gca,'XTick',0:1:19);
set(gca,'YTick',0:0.5:1);
set(gca,'FontSize',7);
xlabel('T(s)');
ylabel('Waveform');
title('Output waveform of pin11&pin13');
grid on;


%% problem 4
p5=[0 1 0 0];
t5=[0 0.73 0.97 2];

l1=[1 0 1 0 1 0 1 0 1 0 0];
t1=[0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2];
```

```
l2=[0 1 0 0];
t2=[0 1.03 1.53 2];


l3=[0 1 0 0];
t3=[0 0.73 0.8 2];


figure;
subplot(4,1,1);
stairs(t5,p5,'Color',[0.8 0.2 0.2],'LineWidth',1);
ylim([-0.2 1.2]);
xlim([-0.1 2.1]);
set(gca,'FontSize',7);
set(gca,'XTick',0:0.1:2);
set(gca,'YTick',0:0.5:1);
title('Input waveform of p5');
xlabel('T(s)');
ylabel('Waveform');
grid on;


subplot(4,1,2);
stairs(t1,l1,'Color',[0.1 0.43 0.7],'LineWidth',1);
ylim([-0.2 1.2]);
xlim([-0.1 2.1]);
set(gca,'FontSize',7);
set(gca,'XTick',0:0.1:2);
set(gca,'YTick',0:0.5:1);
title('Output waveform of pin LED1');
xlabel('T(s)');
ylabel('Waveform');
grid on;

subplot(4,1,3);
stairs(t2,l2,'Color',[0.1 0.43 0.7],'LineWidth',1);
ylim([-0.2 1.2]);
xlim([-0.1 2.1]);
set(gca,'FontSize',7);
set(gca,'XTick',0:0.1:2);
set(gca,'YTick',0:0.5:1);
title('Output waveform of pin LED2');
xlabel('T(s)');
ylabel('Waveform');
grid on;

subplot(4,1,4);
stairs(t3,l3,'Color',[0.1 0.43 0.7],'LineWidth',1);
ylim([-0.2 1.2]);
xlim([-0.1 2.1]);
set(gca,'FontSize',7);
set(gca,'XTick',0:0.1:2);
set(gca,'YTick',0:0.5:1);
title('Output waveform of pin LED3');
xlabel('T(s)');
```

```
ylabel('Waveform');
grid on;
saveas(gca,'4.pdf');
```

After this, I find that maybe I should use python to plot as it is much easier than using the MATLAB.