MEAM 510 Lab 01

1. Choose an LED and a resistor from the ministore. What range of resistance will work well and why? (Hint: use the LED datasheets from DAIGIKEY.).Calculate the smallest resistor that you can safely use with this LED in series with 5V power. Submit a copy of the page from the datasheet you used highlighting the specification you used to determine this value along with the calculation you used to get it.

We choose 160-1133-ND 5mm yellow LED. The datasheet of this type of LED is shown as below. The specification is highlighted in the datasheet as well.

LITEON LITE-ON ELECTRONICS, INC.

Property of Lite-On Only

Absolute Maximum Ratings at TA=25℃

Parameter	Maximum Rating	Unit
Power Dissipation	60	mW
Peak Forward Current (1/10 Duty Cycle, 0.1ms Pulse Width)	80	mA
Continuous Forward Current	20	mA
Derating Linear From 50℃	0.25	mA/℃
Reverse Voltage	5	V
Operating Temperature Range	-55℃ to + 100℃	
Storage Temperature Range	-55℃ to + 100℃	
Lead Soldering Temperature [1.6mm(.063") From Body]	260℃ for 5 Seconds	

Part No. : LTL-4253 | Page : 2 of 4

BNS-OD-C131/A4

LITEON LITE-ON ELECTRONICS, INC.

Property of Lite-On Only

Electrical / Optical Characteristics at T_A=25℃

Parameter	Symbol	Min.	Typ.	Max.	Unit	Test Condition
Luminous Intensity	I_V	5.6	19		mcd	I_F = 10mA Note 1,4
Viewing Angle	$2\theta_{1/2}$		36		deg	Note 2 (Fig.6)
Peak Emission Wavelength	λ_P		585		nm	Measurement @Peak (Fig.1)
Dominant Wavelength	λ_d		588		nm	Note 3
Spectral Line Half-Width	$\Delta\lambda$		35		nm	
Forward Voltage	V_F		2.1	2.6	V	I_F = 20mA
Reverse Current	I_R			100	μA	V_R = 5V
Capacitance	C		15		pF	V_F = 0 , f = 1MHz

Note: 1. Luminous intensity is measured with a light sensor and filter combination that approximates the CIE (Commission International De L'Eclairage) eye-response curve.

2. $\theta_{1/2}$ is the off-axis angle at which the luminous intensity is half the axial luminous intensity.

3. The dominant wavelength, λ_d is derived from the CIE chromaticity diagram and represents the single wavelength which defines the color of the device.

4. The Iv guarantee should be added ± 15%.

LITEON LITE-ON ELECTRONICS, INC.

Property of Lite-On Only

Typical Electrical / Optical Characteristics Curves

(25℃ Ambient Temperature Unless Otherwise Noted)

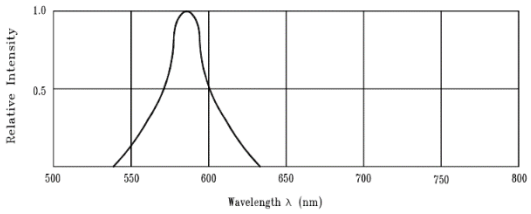


Fig.1 Relative Intensity vs. Wavelength

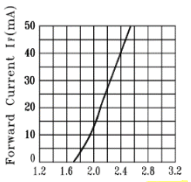


Fig.2 Forward Current vs. Forward Voltage

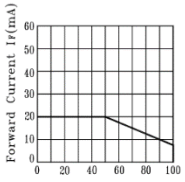


Fig.3 Forward Current Derating Curve

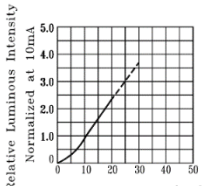


Fig.4 Relative Luminous Intensity vs. Forward Current

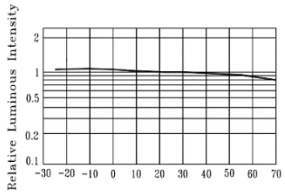


Fig.5 Luminous Intensity vs. Ambient Temperature

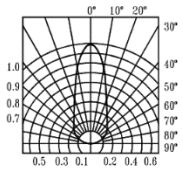


Fig.6 Spatial Distribution

Part No. : LTL-4253	Page : 4 of 4

BNS-OD-C131/A4

With the 5V power, the smallest resistor should be:

$$\mathrm{R_{min}} = \frac{(5-2.1)V}{20mA} = 145\Omega$$

The largest resistor should be:

$$\mathrm{R_{max}} = \frac{(5-1.7)V}{1mA} = 3300\Omega$$

In fact, the maximum resistor could be larger than 3300Ω, because when $\mathrm{V_F} = 1.7V$, the forward current is very close to 0, which means the maximum resistor is also close to infinity.

2. **Use a breadboard and hook up the LED you chose in series with a resistor that has a**

value close to but higher than the value you calculated above. Connect it to 5V and ground and observe the brightness and verify that the LED does not get too hot, smoke or explode. Increase the resistance (using more resistors in series or resistors with larger values) and find what value of resistance will give a very dim but still visible light. Submit the value of resistance you observed and calculate the current that gives this dim light.

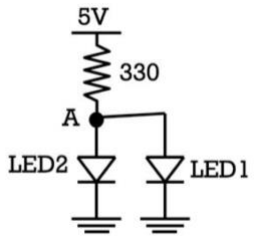
We first connect a 150Ω resistor in series with the LED, the LED does not get too hot, smoke or explode. It was lightened stably.

As we add the resistors in series to 8.950kΩ, the LED give a very dim but still visible light. In this case, the forward voltage of LED is very close to 1.7V. Therefore we have,

$$\mathrm{I_F} = \frac{(5 - 1.7)V}{8.95k\Omega} = 369\mu\mathrm{A}$$

So the current that gives this dim light should be around at 369μA.

3. **Take two LED's with different colors, Put them in parallel with a 330 ohm resistor as in the figure below. Measure the voltage at point A. Submit what you observe: include which LED's you chose; which LED would you expect to be brighter based on the specifications; which LED is brighter; what values on the two different datasheets for each LED explains the voltage at point A?**



We take these two type of LED:

a) 160-1133-ND 5mm yellow LED

b) 160-1782-ND 3mm white LED

The voltage at A point is 1.950V.

From the datasheet, we obtain that the forward voltage of a) is 2.1V, the forward voltage of b) is 3.3V. As $\mathrm{V_{Fa}} < V_{Fb}$, we expect that the yellow one would be brighter. In fact, only the yellow LED is lightened and the white LED is not lightened in such circuit.

Since $\mathrm{V_{Fa}} < V_{Fb}$, when the yellow LED is lightened, the forward voltage of the white LED is lower than $\mathrm{V_{Fb}}$. In this case, as it is shown in the image below, the forward current in the white LED is very close to 0, so the white LED would not be lightened.

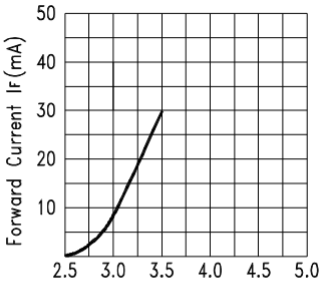
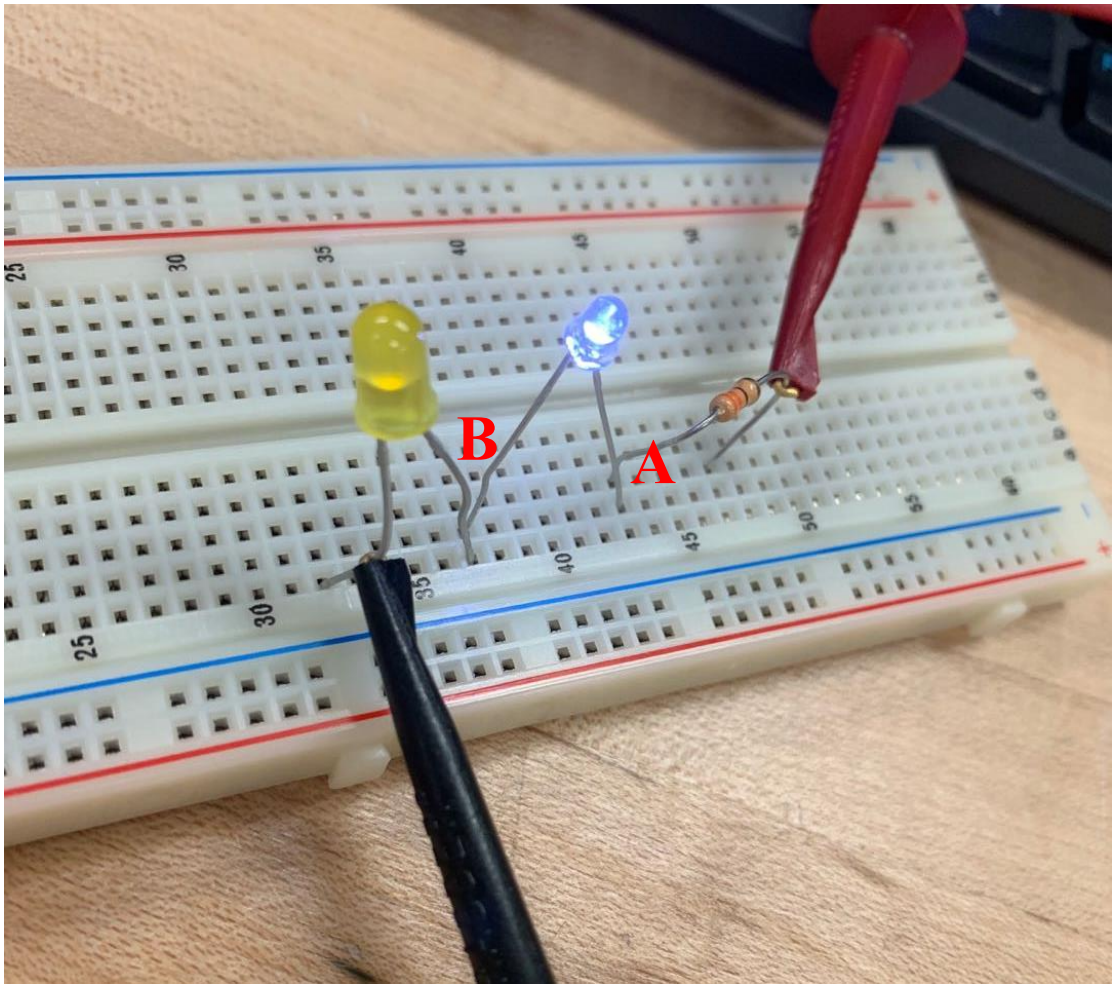


Fig.2 Forward Current vs. Forward Voltage

4. **Take the same circuit, move LED1 to be in series with LED2. Measure the voltage at point A and the point between the two LED's. Submit what you observe: include the values you read and the values you expect based on the datasheet.**



The voltage between of A and B is 2.617V.

The forward current vs. forward voltage graph of the yellow LED is shown below,

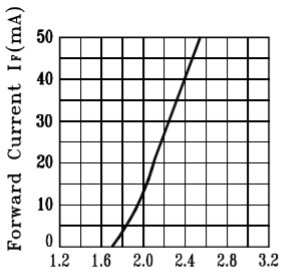


Fig.2 Forward Current vs. Forward Voltage

The forward current vs. forward voltage graph of the white LED is shown in Q3.

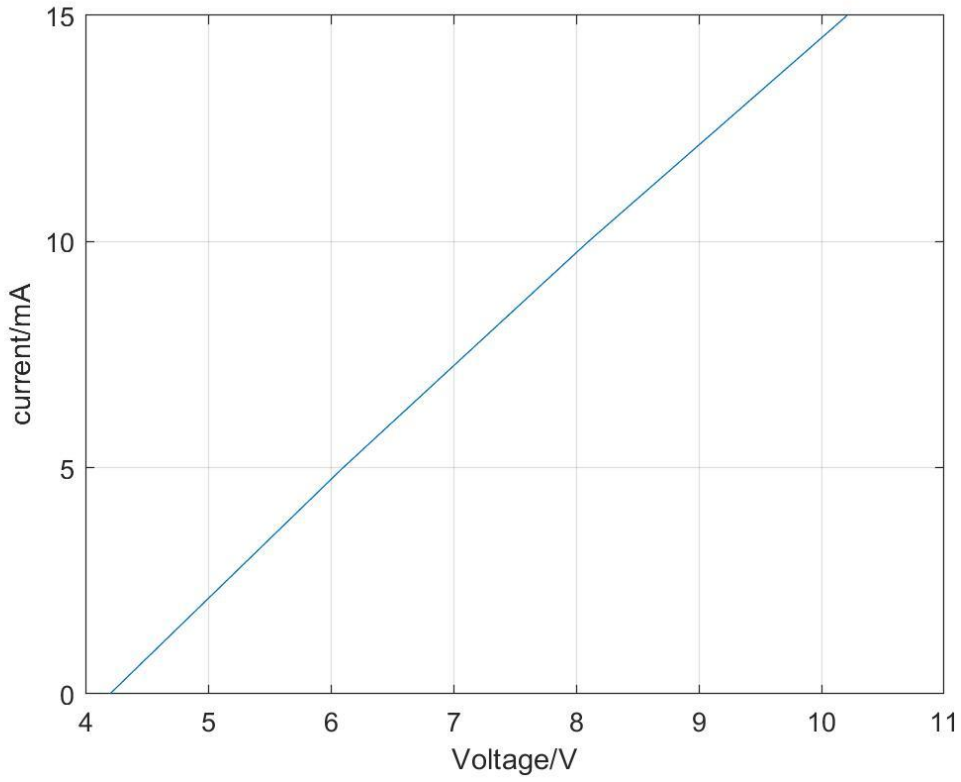
From Kirchhoff law, we know that in series circuit the current equals everywhere. So for the entire circuit, we have the relationship,

$$\mathrm{V} = \mathrm{V_{Fa}} + V_{Fb} + iR$$

Thus, we have:

$$\mathrm{i} = \frac{\mathrm{V} - \mathrm{V_{Fa}} - V_{Fb}}{R}$$

The composition graph of the circuit could be drawn as below.



Therefore, from the composition graph we expect that the current to be around 1.9mA.

When the forward current in the circuit is 1.9mA, the voltage difference between point A and point B is 2.6V, which is very close to the value we measure.

5. **Pretend you are designing an LED powered light source. This could be a flashlight (tight beam) or a desk lamp (wide area). Choose a type of light you want for your lamp – Color brightness, (wide or tight beam), cost, size. Go to www.digikey.com. Find the brightest or cheapest or smallest that suits your needs. Submit a write up that describes the reason you chose it, and how it is optimal (either bright cheap or small).**

   I would like to design an LED desk lamp. Therefore, it is better to choose a white- warm color LED , with wide viewing angle and lowest price.

   First, I type LED in the search bar. Then click 'LED lighting - white' and choose the filter 'white-warm' and 'bulk'/'tube' to search for the LEDs. Sort the list in descending order of viewing angle. Finally, pick the cheapest one in the stock.

   What I pick is XPGDWT-01-0000-00HF6-ND, with a viewing angle of 125° and unit price USD of $0.56560.

   The most suitable package for LED desk lamp is bulk or tube. The LED bulk/tube with lowest price and largest viewing angle is XPGDWT-01-0000-00HF6-ND.

1.2 Digital Output

1.2.1 What register are you changing and why?

I choose PC6 as my output. And therefore, I have to change the register 'DDRC' and 'PORTC'.

First of all there are 25 GPIO pins on the Teensy board, including ports B, C, D and F. All pins of ports B and D are available, but only C and F ports can be used. Therefore, I choose PC6 as my output.

To use PC6 as output, I have to modify the following registers:

DDRC – DDRC means Data Direction Register of pin C. By setting C6 to 1 and the others to 0, it will configure the pin C6 for output and enable driver of teensy. To set C6 as output, we have:

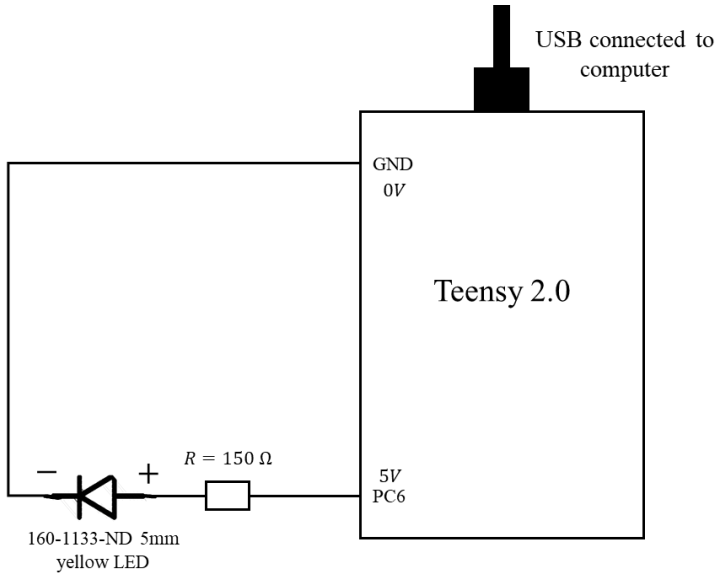
0	1	0	0	0	0	0	0
C7	C6	C5	C4	C3	C2	C1	C0

So, in the code we write: DDRC = 0x40; or set(DDRC, 6);

PORTC – As DDRC is set to 0x40, then clearing the relative PORTC bit will take the output low, while setting the PORTC bit will output high. To change the low/high output of PC6, in the code we could write: toggle(PORTC, 6) or PORTC ^= 0x40, etc.

1.2.2 Attached an LED to this pin with appropriate additional hardware. Submit a drawing of your circuit and justify your additional hardware choice.

The circuit diagram is shown as below



The LED I choose is 160-1133-ND 5mm yellow LED. From 1.1, we know that when the LED is connected to 5V and ground, to ensure the LED work normally, the smallest resistor connected to LED in series is 145Ω, so that the LED does not get too hot, smoke or smoke or explode.

So I choose a resistor of 150Ω. The forward current vs. forward voltage graph of this kind of LED is shown below:

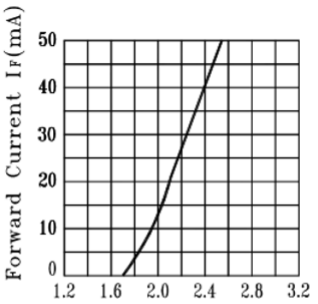


Fig.2 Forward Current vs. Forward Voltage

When the 150Ω resistor is in series with the LED, using the oscilloscope I measured that the forward voltage of LED is 2.16V. From the datasheet we know that relative forward current is 22mA, and therefore the LED could work stably in such case.

1.2.3 Create a variable in your code that you can use to change the frequency the LED blinks. At what frequency does the LED start to appear to be continually on even though it is blinking very quickly?

The LED start to be continually on even though it is blinking very quickly when the teensy delays 5ms, i.e. at a frequency of $\frac{1}{5\text{ms}} \div 2 = 100\text{Hz}$.

```
/* Name: main.c
 * Author: Yuchen Sun
 * Copyright: <insert your copyright message here>
 * License: <insert your license reference here>
 */

#include "teensy_general.h"  // includes the resources included in the teensy_general.h file

int main(void)
{
    set(DDRC, 6); // set PC6 as output
    teensy_clockdivide(0); // set the clock speed to 16MHz
    int t; // create a variable that change the time teensy wait
    t = 5; // set the cycle time to 5ms
    /* insert the teensy initialization here*/

    for(;;)
    {
        /* insert your main loop here*/
        toggle(PORTC,6); // change the state of teensy
        teensy_wait(t); // The teensy delay t ms
    }
    return 0; /* never reached*/
```

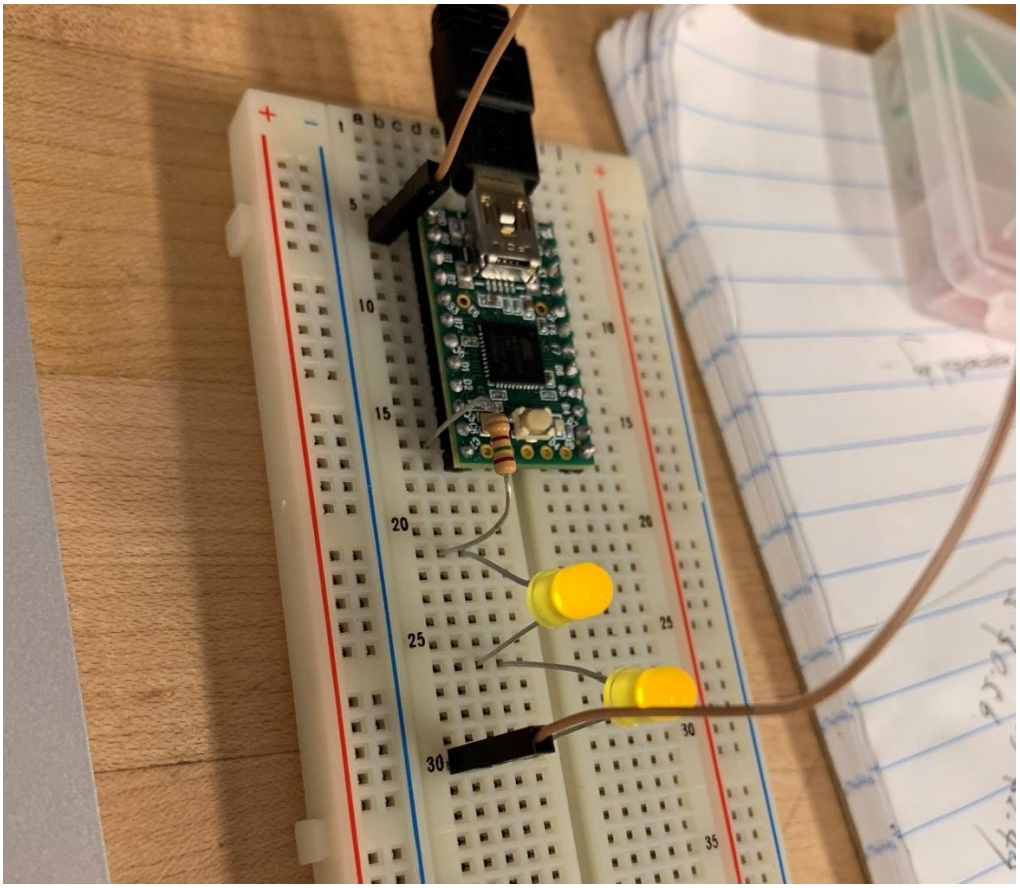
```
}
```

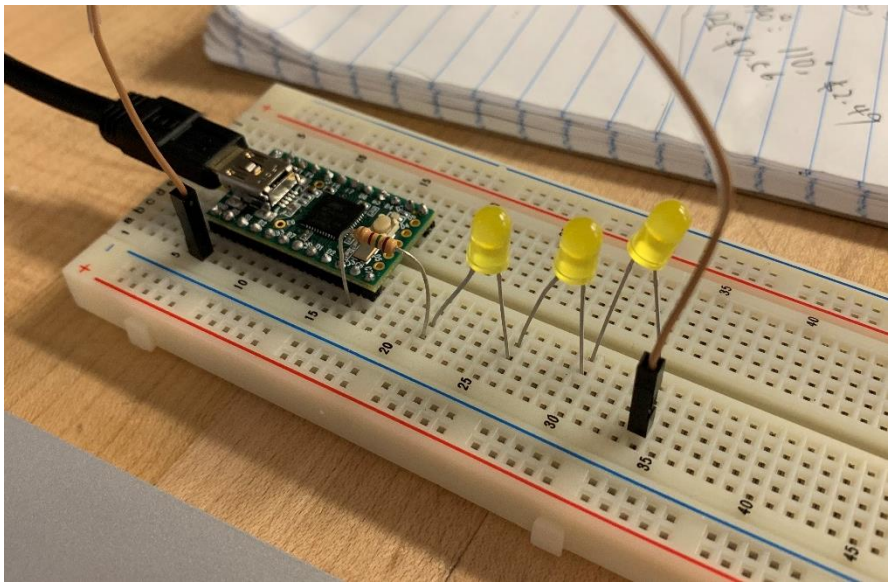
1.2.4 Create a variable in your code so you can easily change the duty cycle (100% duty cycle is always on, 0% duty cycle is always off, 50% is half on half off).

```
/* Name: main.c
 * Author: Yuchen Sun
 * Copyright: <insert your copyright message here>
 * License: <insert your license reference here>
 */

#include "teensy_general.h"  // includes the resources included in the teensy_general.h file

int main(void)
{
    set(DDRC, 6); // set PC6 as output
    teensy_clockdivide(0); // set the clock speed
    int t; // create a variable that change the time teensy wait
    t = 3000; // set the cycle time
    float duty_circle; // create a variable that change the duty circle
    duty_circle = 0.2; // set duty circle
    /* insert the teensy initialization here*/

    for(;;)
    {
        /* insert your main loop here*/
        toggle(PORTC,6); // change the state of teensy, Turn ON the LED
        teensy_wait((duty_circle*t); // The teensy ON  for (duty_circle)*t ms
        toggle(PORTC,6); // change the state of teensy, Yurn OFF the LED
        teensy_wait((1-duty_circle)*t); // The teensy OFF for (1-duty_circle)*t ms
    }
    return 0; /* never reached*/
}
```

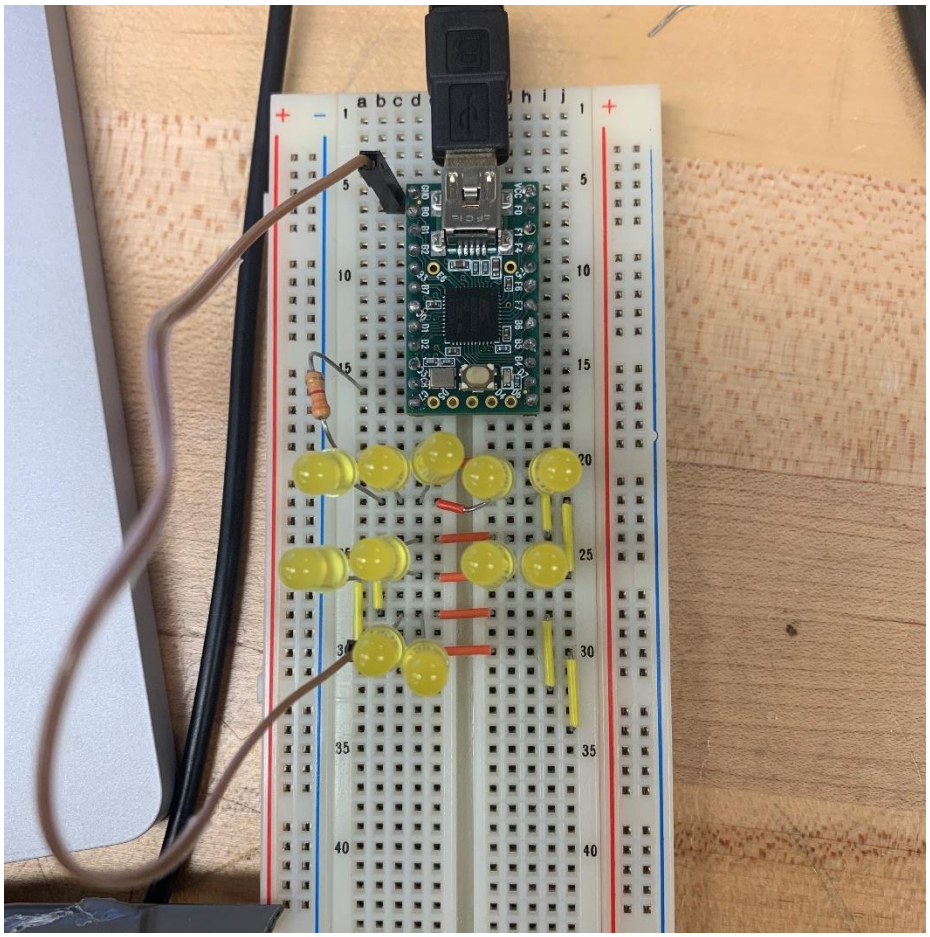
1.2.5 How many LEDs can you turn on from one pin of the Teensy? Submit a picture of your set up and a description of the issues limiting the number of LEDs.

If the LEDs are connected in series with a 150Ω resistor, Teensy can at most turn on 2 LEDs at most, the set up of the circuit is shown as below





If the LEDs are connected in parallel with a 3.3kΩ resistor in series, Teensy could at most turn on 11 LED from one pin, the set up of the circuit is shown below.



The teensy output voltage measured by oscilloscope is 5.36V. When LEDs are connected in series, the total LED forward voltage drop measured by the oscilloscope is shown in the table below,

Number of LED	Total LED Forward Voltage/V	Forward Current/mA
1	2.16	21.33
2	4.08	8.53
3	5.28	0.53

When there are 3 LED connected in series in the circuit, the forward current is 0.53mA, which is quite small to light up the LED. Therefore, when LED are connected in series, a pin could at most light up 2 LED.

When LEDs are connected in parallel with a 3.3kΩ resistor(the exact value is 3.239kΩ), the total LED forward voltage drop measured by the oscilloscope is shown in the table below.

Number of LED	Total LED Forward Voltage/V	Forward Current for each LED/mA
1	1.92	1.03
2	1.88	0.52
3	1.84	0.35
4	1.84	0.26
5	1.78	0.21
6	1.78	0.18

7	1.76	0.15
8	1.76	0.14
9	1.76	0.12
10	1.76	0.10
11	1.74	0.09

When there are 11 LED connected in parallel, the forward current for each LED is only 0.10mA, which is so small that the LED only gives a very dim light Therefore, when LED are connected in parallel, a pin could at most light up 11 LED.

Above all, the issue limiting the number of LED is the Forward Current vs Forward Voltage curve of the LED. For example, the cut-off voltage of 160-1133-ND 5mm yellow LED 1.7V, which means that the LED won't work when the voltage added to it is below 1.7V. However, the cut-off voltage of 160-1782-ND 3mm white LED is 2.5 V. As the yellow LED could work under the voltage of 2.1V, the white LED won't work. Therefore, theoretically, if it fixes the output voltage, more yellow LED could be connected in series.

Second, the forward current of a certain LED is also an issue limiting the number of LED. For example, when LEDs are connected in parallel, the voltage difference changes quite slightly. However, as the number of LED increases, forward current for each LED get smaller and smaller until the LED become very dim.

Finally, when the LED are connected in parallel, the resistor connected in series could not be too small. As the total current equals to the sum of each LED's forward current, the total might current might be so large when the resistor is too small that the teensy might get burned.

1.3 Timer

1.3.1 Use the timer of the ATMEGA32u4 to get an LED to blink at 20Hz. Verify the driving frequency on a scope. Submit a photo of the oscilloscope screen showing the 20Hz signal and the code that generate this.

The pin for output is PC6, and from MEAM.Design we find that the timer 3 channel is multiplexed with PC6. The counter TCNT3 can count to 16bit, i.e. timer 3 can count to 65,635 times.

Timer 3

Configuration Details

OC3A	C6	output compare, timer 3, channel A
IPC3	C7	input capture, timer 3

The LED blinks at 20Hz, which means that the LED is 1/40 sec on and 1/40 sec off. Therefore, the timer should count 1/40 sec and then toggle pin. Since timer3 can only count up to 65,635 times, we should select an appropriate prescaler and an output compare value.

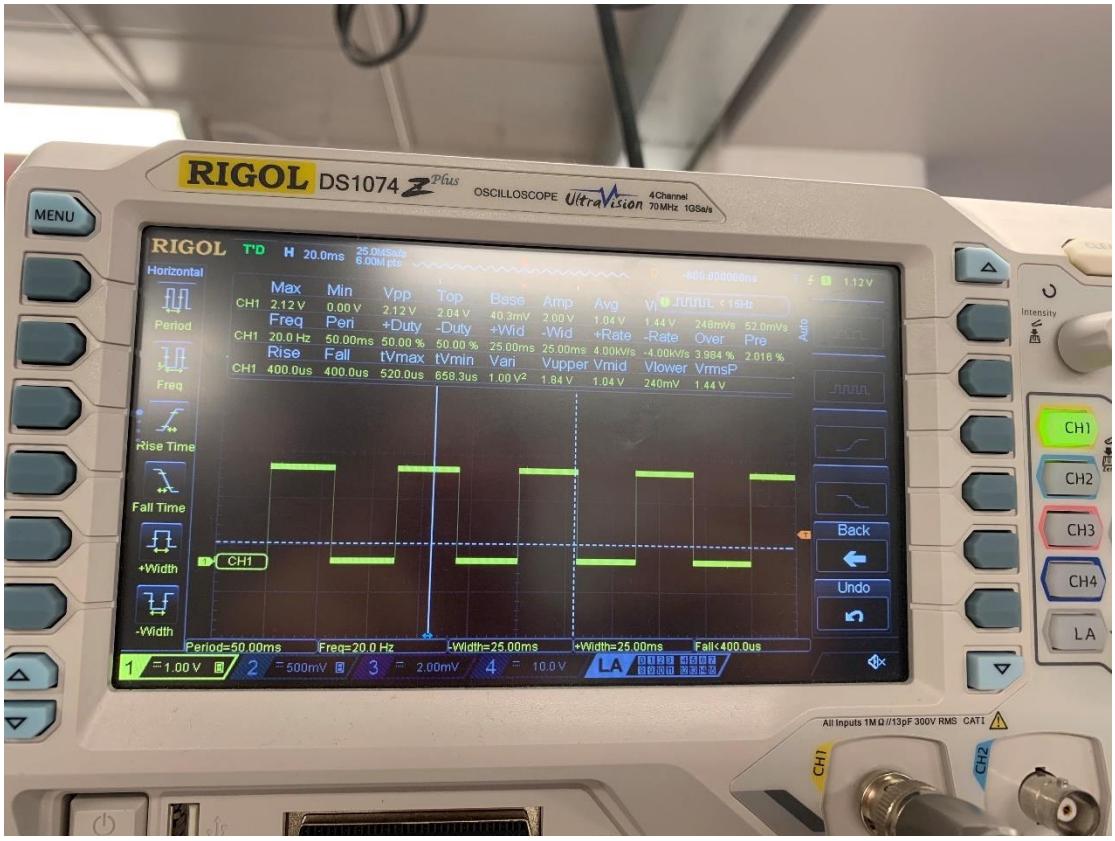
When the prescaler is set to /8, the clock speed of teensy is:

$$16\text{MHz} \div 8 = 2\text{MHz}$$

And timer3 should count $2\text{MHz} \times \frac{1}{40} sec = 50000$ times to ensure pin voltage is at 20Hz. The code can be written as below:

```
/* Name: main.c
 * Author: Yuchen Sun
 * Copyright: <insert your copyright message here>
 * License: <insert your license reference here>
 */

#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#define COMPAREVALUE 50000 //set comparevalue to 100

int main(void)
{
    set(DDRC, 6); // set PC6 as output
    teensy_clockdivide(0); //set the clock speed to 16MHz
    set(TCCR3B, WGM32); // set TIMER to mode 4
    set(TCCR3A, COM3A0); // set timer to toggle at OCR3A
    set(TCCR3B, CS31); // set the prescaler to /8
    OCR3A = COMPAREVALUE; // set timer count up to COMPAREVALUE
    /* insert the teensy initialiation here*/

    while(1); /* main loop*/
    return 0; /* never reached*/
}
```

The scope screen is shown as below:



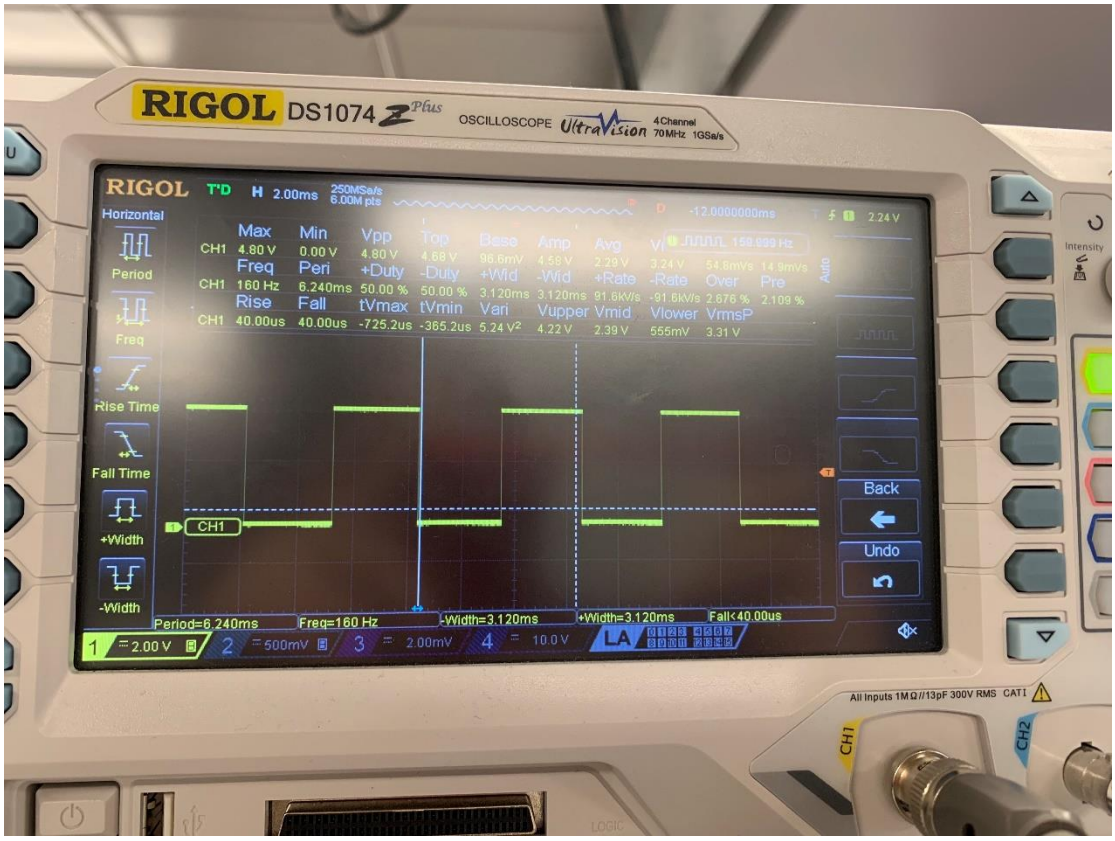
1.3.2 Using the timer register, adjust the system clock pre-scaler. Does the output change as you would expect, why or why not? What is the default system clock frequency?

For teensy 2.0, Teensy defaults to only 2MHz(from https://www.pjrc.com/teensy/prescaler.html). First we set the timer speed at 16MHz, using `teensy_clockdivide(0)`. Adjust the prescaler of Timer3 by modifying CS30, CS31 and CS32.

Clock Source - The default clock source for Timer 3 is the system clock. You can set the prescaler by modifying CS30, CS31, and CS32 in TCCR3B:

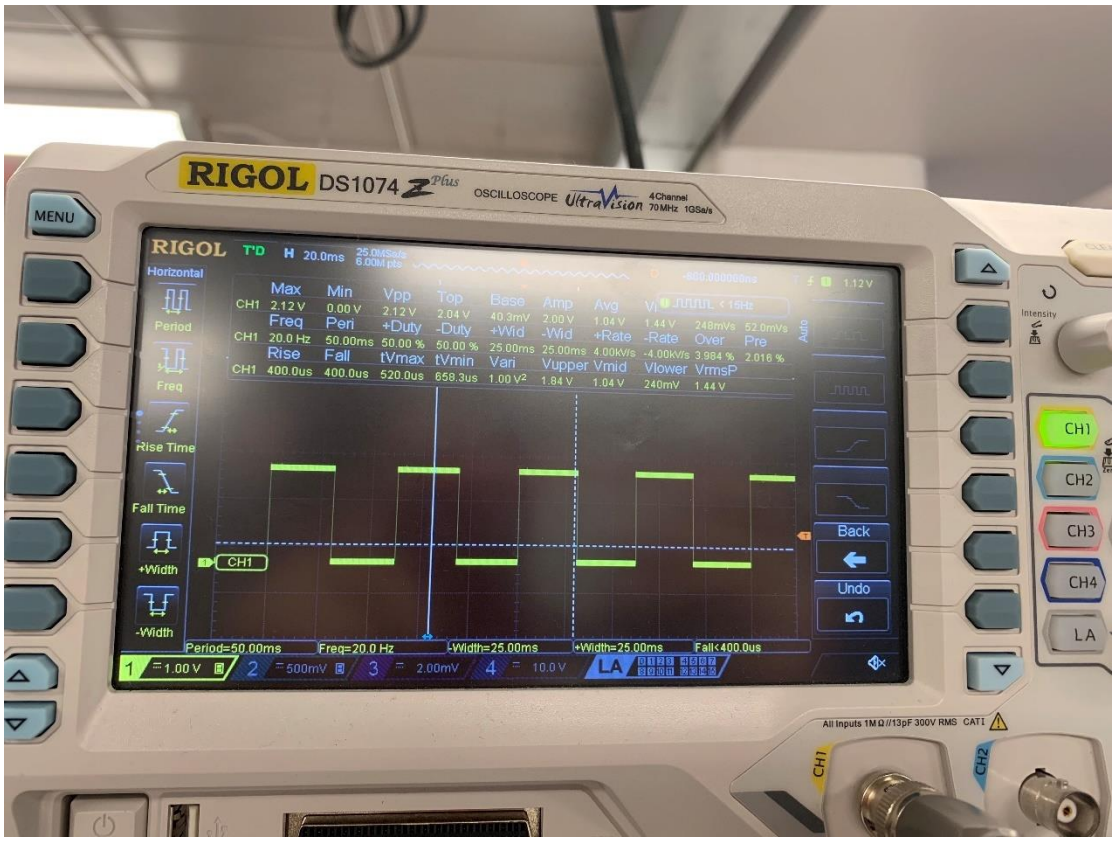
TCCR3B: CS32	TCCR3B: CS31	TCCR3B: CS30	Prescaler
0	0	0	Clock Source Off
0	0	1	/1
0	1	0	/8
0	1	1	/64
1	0	0	/256
1	0	1	/1024

Fix COMPAREVALUE at 50,000. When pre-scaler is /1, the LED should blink at $2 \times 50{,}000 \times \frac{1}{16MHz} = 160Hz$, the screen of the scope and the code is shown as below




```c
/* Name: main.c
 * Author: Yuchen Sun
 * Copyright: <insert your copyright message here>
 * License: <insert your license reference here>
 */

#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#define COMPAREVALUE 50000 //set comparevalue to 100

int main(void)
{
    set(DDRC, 6); // set PC6 as output
    teensy_clockdivide(0); //set the clock speed to 16MHz
    set(TCCR3B, WGM32); // set TIMER to mode 4
    set(TCCR3A, COM3A0); // set timer to toggle at OCR3A
    set(TCCR3B, CS30); // set the prescaler to /1
    OCR3A = COMPAREVALUE; // set timer count up to COMPAREVALUE
    /* insert the teensy initialiation here*/

    while(1); /* main loop*/
    return 0; /* never reached*/
}
```

When pre-scaler is /8, the LED should blink at $2 \times 50{,}000 \times \frac{1}{2MHz} = 20Hz$, the screen of the scope and the code is shown as below.




```c
/* Name: main.c
 * Author: Yuchen Sun
 * Copyright: <insert your copyright message here>
 * License: <insert your license reference here>
 */

#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#define COMPAREVALUE 50000 //set comparevalue to 100

int main(void)
{
    set(DDRC, 6); // set PC6 as output
    teensy_clockdivide(0); //set the clock speed to 16MHz
    set(TCCR3B, WGM32); // set TIMER to mode 4
    set(TCCR3A, COM3A0); // set timer to toggle at OCR3A
    set(TCCR3B, CS31); // set the prescaler to /8
    OCR3A = COMPAREVALUE; // set timer count up to COMPAREVALUE
    /* insert the teensy initialiation here*/

    while(1); /* main loop*/
```

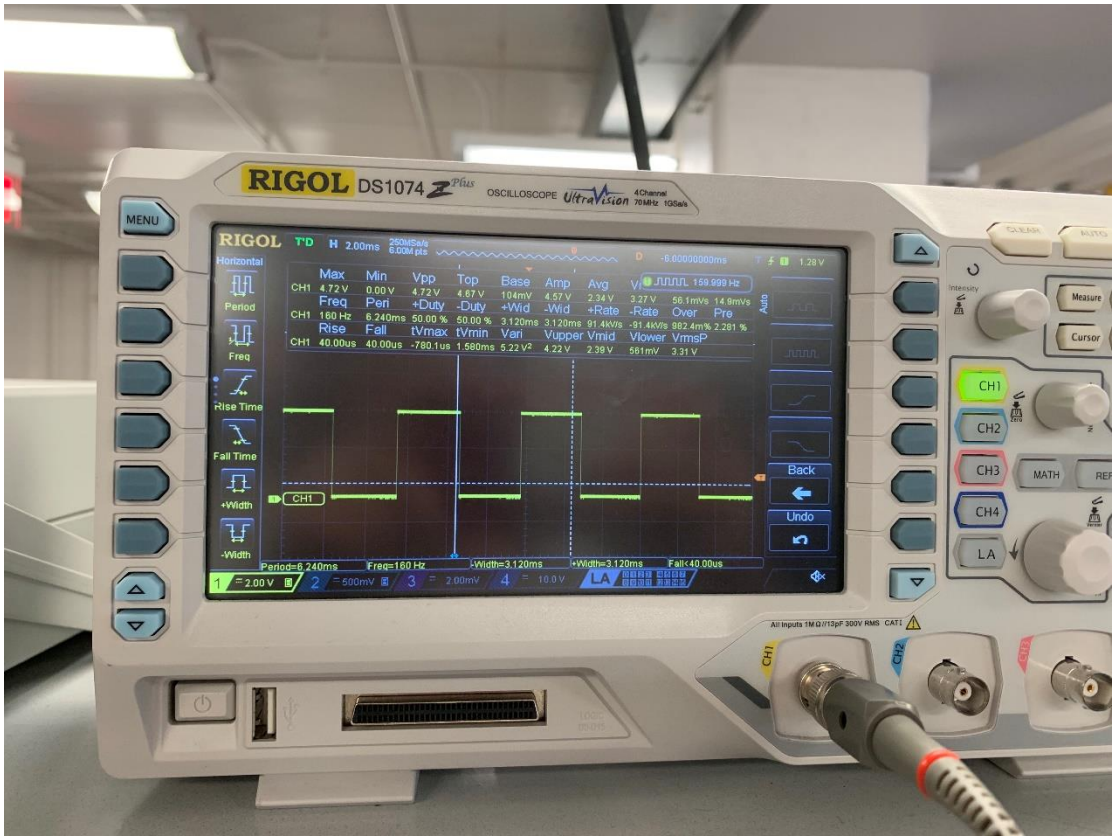
```
    return 0; /* never reached*/
}
```

When pre-scaler is /64, the LED should blink at $2 \times 50{,}000 \times \frac{1}{250KHz} = 2.5Hz$, the screen of the scope and the code is shown as below.


```
/* Name: main.c
 * Author: Yuchen Sun
 * Copyright: <insert your copyright message here>
 * License: <insert your license reference here>
 */

#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#define COMPAREVALUE 50000 //set comparevalue to 100

int main(void)
{
    set(DDRC, 6); // set PC6 as output
    teensy_clockdivide(0); //set the clock speed to 16MHz
    set(TCCR3B, WGM32); // set TIMER to mode 4
    set(TCCR3A, COM3A0); // set timer to toggle at OCR3A
    set(TCCR3B, CS30); // set the prescaler to /64
    set(TCCR3B, CS31); //set the prescaler to /64
```

```
    OCR3A = COMPAREVALUE; // set timer count up to COMPAREVALUE
    /* insert the teensy initialiation here*/

    while(1); /* main loop*/
    return 0; /* never reached*/
}
```

Now turn on the timer without prescaler. The screen of the scope and the code is shown as below.




```
/* Name: main.c
 * Author: Yuchen Sun
 * Copyright: <insert your copyright message here>
 * License: <insert your license reference here>
 */

#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#define COMPAREVALUE 50000 //set comparevalue to 100

int main(void)
{
    set(DDRC, 6); // set PC6 as output
    teensy_clockdivide(0); //set the clock speed to 16MHz
    TCCR3B = 0x01; // Turn on counter (no prescale)
    set(TCCR3B, WGM32); // set TIMER to mode 4
```

```
    set(TCCR3A, COM3A0); // set timer to toggle at OCR3A
    OCR3A = COMPAREVALUE; // set timer count up to COMPAREVALUE
    /* insert the teensy initialiation here*/

    while(1); /* main loop*/
    return 0; /* never reached*/
}
```

Reading from the scope we know that LED blinks at 160Hz, which is equals to the value when the prescaler is /1. Therefore, the default prescaler is /1.

Above all, LED blinks as what I expect. It is because when I set prescaler to /1, /8 and /256 respectively, the timer speed is divided by /1, /8 and /256 respectively. Since I set the system clock speed to 16MHz originally, the timer speed become 16MHz, 2MHz, and 250KHz respectively. With COMPAREVALUE setting at 50,000, the LED should blink at 160Hz, 20Hz and 2.5Hz respectively. From oscilloscope we know that the theoretical values and the actual values match perfectly, proving the function of prescaler.

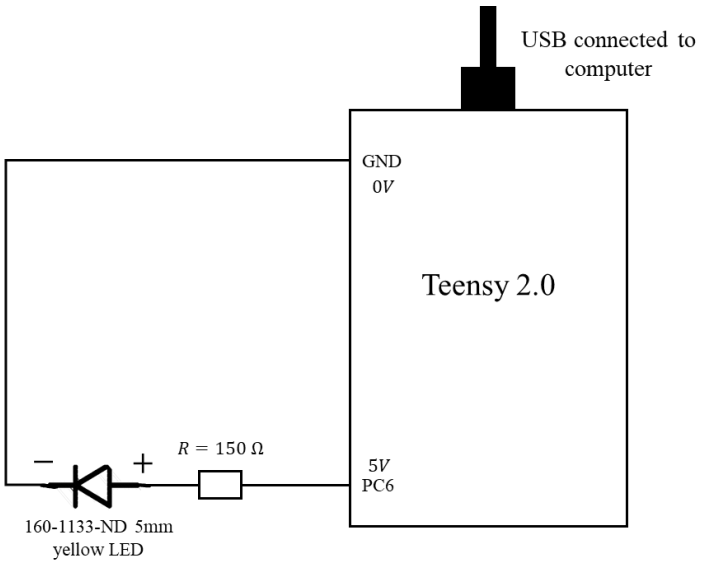
Let the teensy runs at its default speed. Delete `teensy_clocckdivide(0)` and turn on the timer without prescaler. The code and the screen of scope are shown as below,

```
/* Name: main.c
 * Author: Yuchen Sun
 * Copyright: <insert your copyright message here>
 * License: <insert your license reference here>
 */

#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#define COMPAREVALUE 50000 //set comparevalue to 100

int main(void)
{
    set(DDRC, 6); // set PC6 as output
    TCCR3B = 0x01; // Turn on counter (no prescale)
    set(TCCR3B, WGM32); // set TIMER to mode 4
    set(TCCR3A, COM3A0); // set timer to toggle at OCR3A
    OCR3A = COMPAREVALUE; // set timer count up to COMPAREVALUE
    /* insert the teensy initialiation here*/

    while(1); /* main loop*/
    return 0; /* never reached*/
}
```

The LED blinks at 80Hz, which means the LED is 1/160 sec on and 1/160 sec off. Therefore, the default system clock is $50{,}000 \times \frac{1}{160}\text{sec} = 8MHz$, other than 2MHz or 16MHz.

1.3.3 Use PWM functions of the ATEMGAu4 timer ro do the same thing you did in 1.2.5 Explain which timer options you use for generating the PWM. Show that you can generate 0% and 100% duty cycle. Submit the code and circuit diagram for this part. Submit a short video of the system working.

The circuit diagram is shown as below



I choose mode 14 to generate PWM. Timer Modes controls how the timer will count, what the maximum value will be, and whether to drive the output compare pins.

TCCR3B: WGM33	TCCR3B: WGM32	TCCR3A: WGM31	TCCR3A: WGM30	Function
Normal: Timer UP to a value, reset to 0x0000:				
0	0	0	0	(mode 0) UP to 0xFFFF (16-bit)
0	1	0	0	(mode 4) UP to OCR3A
1	1	0	0	(mode 12) UP to ICR3
Single-Slope: Timer UP to a value, reset to 0x0000 (set/reset PWM):				
0	1	0	1	(mode 5) UP to 0x00FF (8-bit), PWM mode
0	1	1	1	(mode 7) UP to 0x03FF (10-bit), PWM mode
1	1	1	0	(mode 14) UP to ICR3, PWM mode
Dual-Slope: Timer UP to a value, DOWN to 0x0000 (set/reset PWM):				
0	0	0	1	(mode 1) UP to 0x00FF, DOWN to 0x0000, PWM mode
0	0	1	1	(mode 3) UP to 0x03FF, DOWN to 0x0000, PWM mode
1	0	1	0	(mode 10) UP to ICR3, DOWN to 0x0000, PWM mode

(from https://alliance.seas.upenn.edu/~medesign/wiki/index.php/Guides/MaEvArM-timer3)
From the table above, we know that when the timer set to mode 14. The timer could generate PWM, counting up to ICR3. Therefore, by setting the value of ICR3 and OCR3A, we could easily adjust the maximum number of counting and at what value the timer will reset to 0x0000. When timer is set to mode 14, we could adjust in TCCR3A to control the output behavior.

When operating in modes 5, 7, or 14, a match between TCNT3 and OCR3A will yield the following:

TCCR3A: COM3A1	TCCR3A: COM3A0	Function
0	0	no change
1	0	clear at OCR3A, set at rollover
1	1	set at OCR3A, clear at rollover

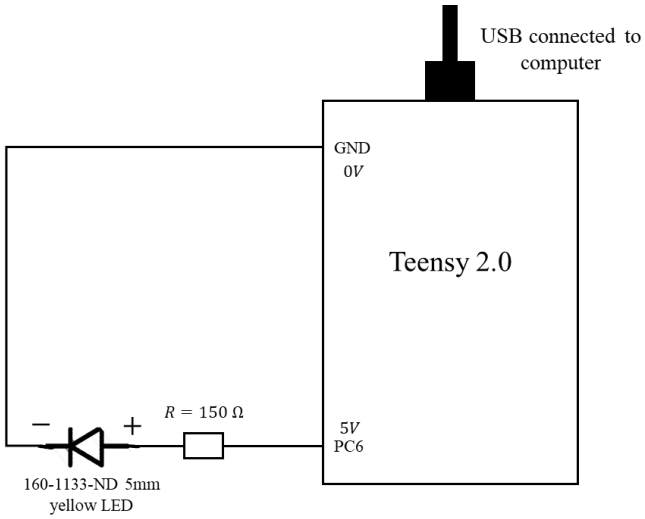
```
/* Name: main.c
 * Author: Yuchen Sun
 * Copyright: <insert your copyright message here>
 * License: <insert your license reference here>
 */

#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#define COMPAREVALUE 100 //set comparevalue to 100

int main(void)
{
    set(DDRC,6); // set PC6 as output
    teensy_clockdivide(0); // set the teensyclock to 16MHz
    set(TCCR3B, CS30); // set the prescaler to /1
    set(TCCR3B, WGM33); // set timer to mode 14
    set(TCCR3B, WGM32); // set timer to mode 14
```

```
    set(TCCR3A, WGM31); // set timer to mode 14
    set(TCCR3A, COM3A1); // clear at OCR3A and set at rollover
    ICR3 = COMPAREVALUE; // set PWM up to COMPAREVALUE
    float duty_circle; // create a duty circle variable
    duty_circle = 0; // set duty circle to 1;
    OCR3A = duty_circle*COMPAREVALUE; // set OCR3A equals to duty_circle*COMPAREVALUE
    /* insert the initialization here*/

    while(1); // main loop
    return 0; /* never reach*/
}
```

1.4 Practice with Loops

1.4.1 Create code to make the external LED pulse. The pulse should start at 0 intensity, take 1 second to increase in intensity until it is full brightness, then 1 second to decrease in brightness and repeat.

Take that code and change it so that time to increase and the time to decrease is variable. Submit a video of repeating asymmetric pulses (0.3 second to full intensity and 0.7 seconds to 0 intensity) along with well commented code and circuit diagram.



The code is as shown as below. The video is in the .zip file, named 141.mov.

```
/* Name: main.c
 * Author: Yuchen Sun
 * Copyright: <insert your copyright message here>
 * License: <insert your license reference here>
 */

#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#define COMPAREVALUE 100 //set comparevalue to 100
```

```
int main(void)
{
    set(DDRC,6); // set PC6 as output
    teensy_clockdivide(0); // set the teensyclock to 16MHz
    set(TCCR3B, CS30); // set the prescaler to /1
    set(TCCR3B, WGM33); // set timer to mode 14
    set(TCCR3B, WGM32); // set timer to mode 14
    set(TCCR3A, WGM31); // set timer to mode 14
    set(TCCR3A, COM3A1); // clear at OCR3A and set at rollover
    ICR3 = COMPAREVALUE; // set PWM up to COMPAREVALUE
    int i; // set the step of PWM
    int upTime; // create a time variable for LED to be fully ON
    upTime = 300; // set upTime to 0.3 sec
    int downTime; // create a time variable for LED to be fully OFF
    downTime = 700; // set downTime to 0.7 sec
    float intensityIncrease; // create waiting time for intensity increase
    intensityIncrease = upTime/COMPAREVALUE; // intensityIncrease should be 3 ms;
    float intensityDecrease; // create waiting time for intensity decrease
    intensityDecrease = downTime / COMPAREVALUE; // intensityDecrease should be 7 ms;
    /* insert your initialization here*/

    for(;;) // the main loop
    {
        for(i = 0; i < ICR3; i++) // loop for the intensity increase
        {
            OCR3A = i; // set the duty circle of LED
            teensy_wait(intensityIncrease); // teensy wait at a certain time at a certain duty circl
e
        }
        for(i = COMPAREVALUE; i >0 ; i--) // loop for the intensity decrease
        {
            OCR3A = i; // set the duty circle of LED
            teensy_wait(intensityDecrease); // teensy wait at a certain time at a certain duty circl
e
        }
    }
    return 0; /* never reach*/
}
```

1.4.2 Use subroutine so that the pulsing of the code with variable increasing and decreasing intensity is easy to call from another routine. Modify the code above so that the maximum intensity can also be changed. Create a routine that causes the LED to blink as if it were a

heartbeat (e.g. lub dub). That is the LED percentage intensity i should follow the pattern below at time t seconds but smoothly interpolated intensities between each value:

t = 0 i = 0
t = 0.1 i = 100
t = 0.5 i = 0
t = 0.6 i = 50
t = 1.0 i = 0
t = 3.0 i = 0
t = 3.1 i = 100
t = 3.5 i = 0
t = 3.6 i = 50
t = 4.0 i = 0

The code is shown as below. The video is in the .zip file, named 142.mov.

```
/* Name: main.c
 * Author: Yuchen Sun
 * Copyright: <insert your copyright message here>
 * License: <insert your license reference here>
 */

#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#define COMPAREVALUE 100 //set comparevalue to 100

// declare the subroutine
int lighting(float); //Create a subroutine that control the intensity of LED

int main(void)
{
    set(DDRC,6); // set PC6 as output
    teensy_clockdivide(0); // set the teensyclock to 16MHz
    set(TCCR3B, CS30); // set the prescaler to /1
    set(TCCR3B, WGM33); // set timer to mode 14
    set(TCCR3B, WGM32); // set timer to mode 14
    set(TCCR3A, WGM31); // set timer to mode 14
    set(TCCR3A, COM3A1); // clear at OCR3A and set at rollover
    ICR3 = COMPAREVALUE; // set PWM up to COMPAREVALUE
    float intensity_ls[2] = {1, 0.5}; // create a variable for LED intensity
    int j; // create a variable for the postion in intensity_ls
    float LEDintensity; // create a variable for intensity for LED
    /* insert the teensy initialization here */

    for(;;) /* insert the main loop*/
    {
        for (j = 0; j < 2; j++) // the pulsing loop
```

```
        {
            LEDintensity =intensity_ls[j]; // set LED intensity
            lighting(LEDintensity); // call the subroutine to control the LED intensity
        }
        teensy_wait(2000); // Let LED be off for 2 sec
    }
    return 0;
}

int lighting(float intensity)
{
    int i; // set the step of PWM
    int upTime = 100; // create a time variable for LED to be fully ON
    int downTime = 400; // create a time vatiable for LED to be full OFF
    int intensityIncrease; // create waiting time variable for intensity increase
    intensityIncrease = upTime / COMPAREVALUE; // waiting time for intensity fully increase should b
e 1 ms
    int intensityDecrease; //  create waiting time variable for intensity decrease
    intensityDecrease = downTime / COMPAREVALUE; // waiting time for intensity fully decrease should
 be 4 ms

    for ( i = 0; i < intensity*COMPAREVALUE; i++) // loop for intensity increase
    {
        OCR3A = i; // set the duty circle of LED
        teensy_wait(intensityIncrease*(1/intensity)); // teensy wait at a certain time at a certain
duty circle
    }
    for( i = intensity*COMPAREVALUE; i > 0; i--) // loop for intensity decrease
    {
        OCR3A = i; // set the duty circle of LED
        teensy_wait(intensityDecrease*(1/intensity)); // teensy wait at a ceertain time at a certain
 duty circle
    }

    return 0;
}
```

1.4.3 Modify the above code so that the heartbeat stays at a constant frequency, but the maximum intensity slowly fades as if the heartbeat is getting weaker. Have it take 20 beats before it is reduced to 0 intensity. Submit a short video and the commented code.

The code is shown as below. The video is in the .zip file, naming 143.mov.

```
/* Name: main.c
```

```
 * Author: Yuchen Sun
 * Copyright: <insert your copyright message here>
 * License: <insert your license reference here>
 */

#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#define COMPAREVALUE 100 //set comparevalue to 100

// declare the subroutine
int lighting(float); // create a subroutine that control the LED intensity

int main(void)
{
    set(DDRC,6); // set PC6 as output
    teensy_clockdivide(0); // set the teensyclock to 16MHz
    set(TCCR3B, CS30); // set the prescaler to /1
    set(TCCR3B, WGM33); // set timer to mode 14
    set(TCCR3B, WGM32); // set timer to mode 14
    set(TCCR3A, WGM31); // set timer to mode 14
    set(TCCR3A, COM3A1); // clear at OCR3A and set at rollover
    ICR3 = COMPAREVALUE; // set PWM up to COMPAREVALUE
    int j ; // create a variable mark for the position of LED intensity
    float LEDintensity; // create a variable for intensity for LED
    /* insert the teensy initialization here */

    for(;;) /* insert the main loop*/
    {
        for (j = 0; j < 20; j++) // the pulsing loop
        {
            LEDintensity = 1- j * 0.05; // set LED intensity
            lighting(LEDintensity); // call the subroutine to control the LED intensity
        }
        teensy_wait(500); //Let LED be off for 0.5 sec
    }
    return 0; /* never reach*/
}

int lighting(float intensity) //subroutine to control the LED intensity
{
    int i; // set the step of PWM
    int upTime = 100; // create a time variable for LED to be fully ON
    int downTime = 400; // create a time vatiable for LED to be full OFF
    float intensityIncrease; // create waiting time variable for intensity increase
    intensityIncrease = upTime / COMPAREVALUE; // intensityIncrease should be 1ms
```

```
    float intensityDecrease; //  create waiting time variable for intensity decrease
    intensityDecrease = downTime / COMPAREVALUE; // intensityDecrease should be 4ms

    for ( i = 0; i < intensity*COMPAREVALUE; i++) // loop for intensity increase
    {
        OCR3A = i; //set the duty circle of LED
        teensy_wait(intensityIncrease*(1.0/intensity)); // teensy wait at a certain time at a certai
n duty circle
    }
    for( i = intensity*COMPAREVALUE; i > 0; i--) // loop for intensity decrease
    {
        OCR3A = i; // set the duty circle of LED
        teensy_wait(intensityDecrease*(1.0/intensity)); // teensy wait at a certain time at a ceerta
in duty circle
    }

    return 0; /*never reach*/
}
```