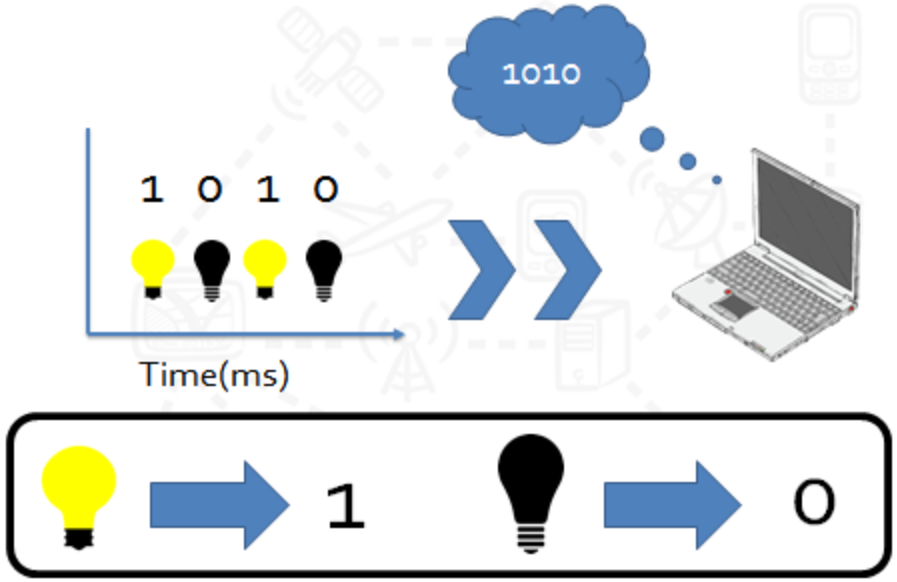
[2014-Fall] WNFA lab1 - CamCom

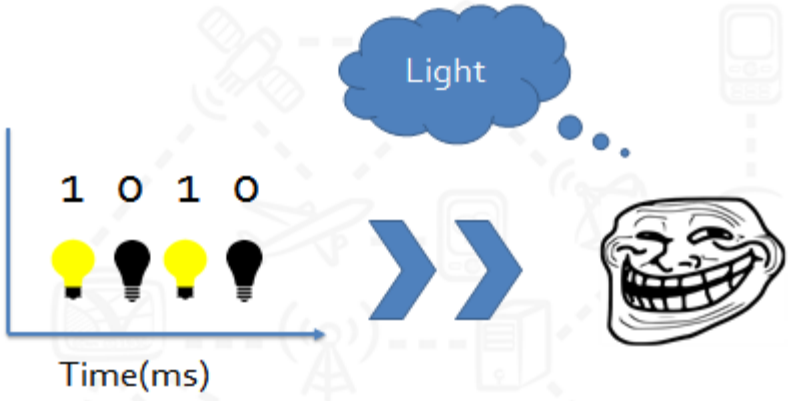
Camera Communication

2014/3/5

Visible Light Communication

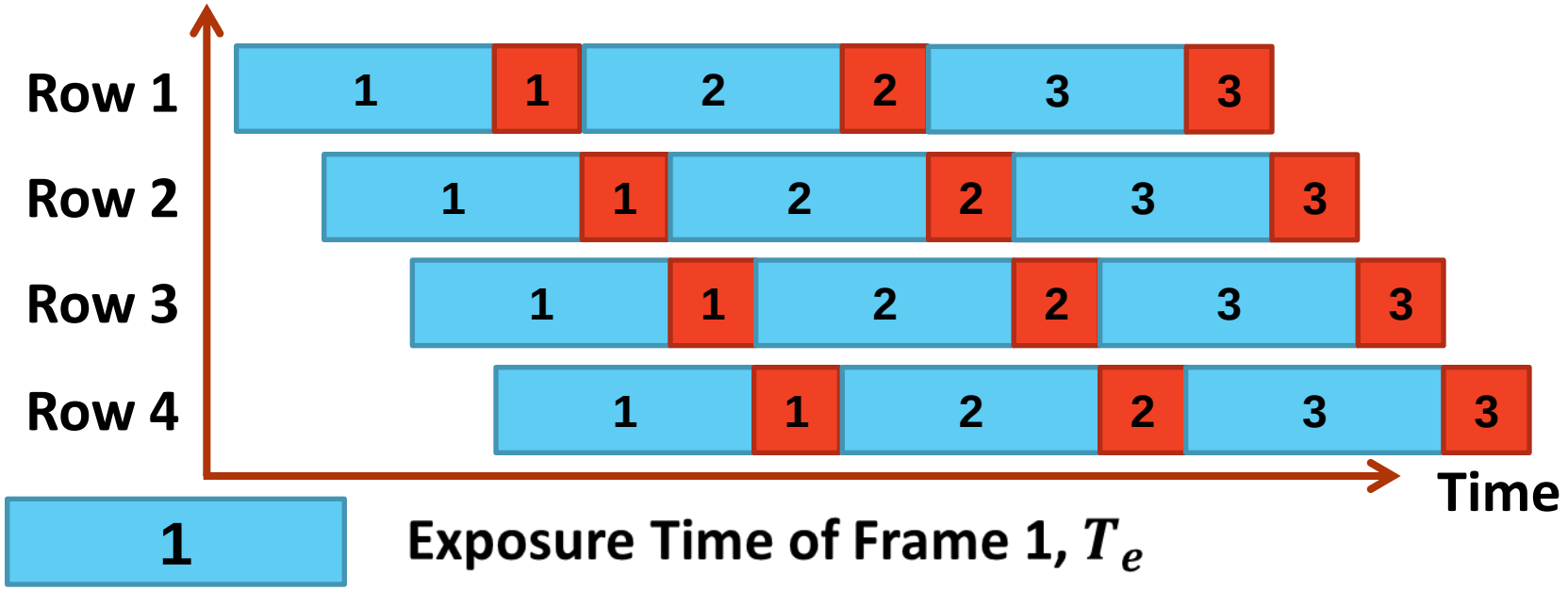
- Use visible light instead of electromagnetic wave for communications.
- Use cameras or light sensors as receivers.
  - Low sampling rate : typically 30 fps





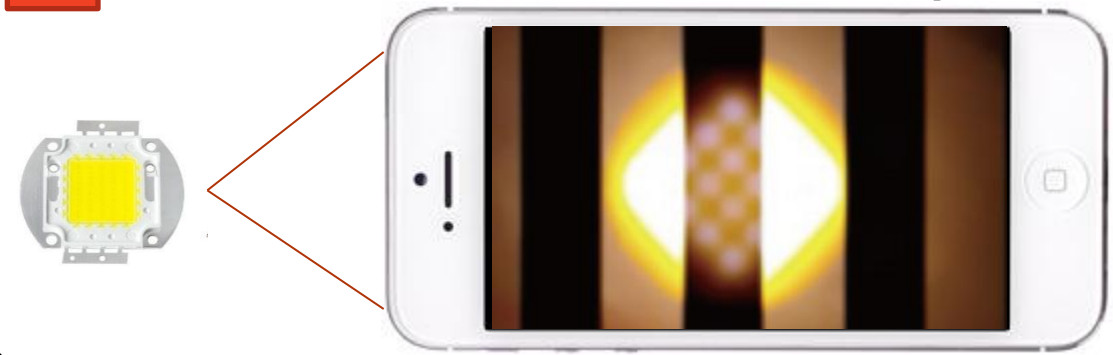
Human eyes can't perceive LED's high frequency flickering
(persistence of vision)

Rolling Shutter



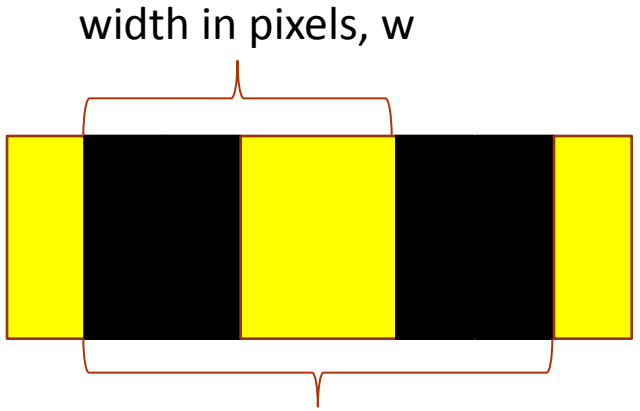
1 Exposure Time of Frame 1, T_e

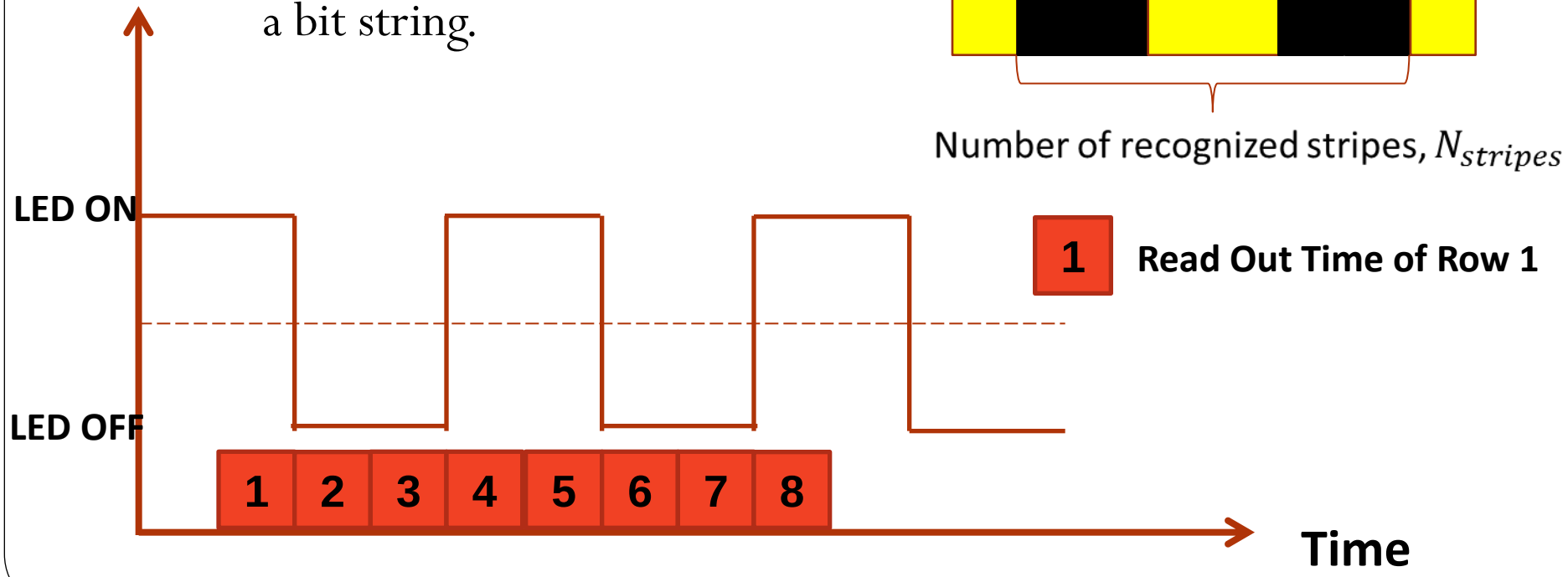
1 Read Out Time of Frame 1, T_r



Rolling Shutter

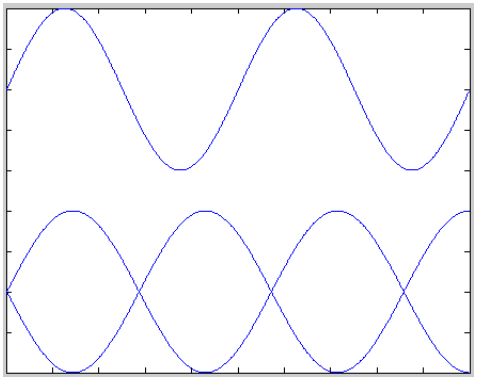
- Frequency-shift on-off keying
  - $f = \frac{N_{stripes}}{2wT_r}$
  - Each frequency represents a bit string.

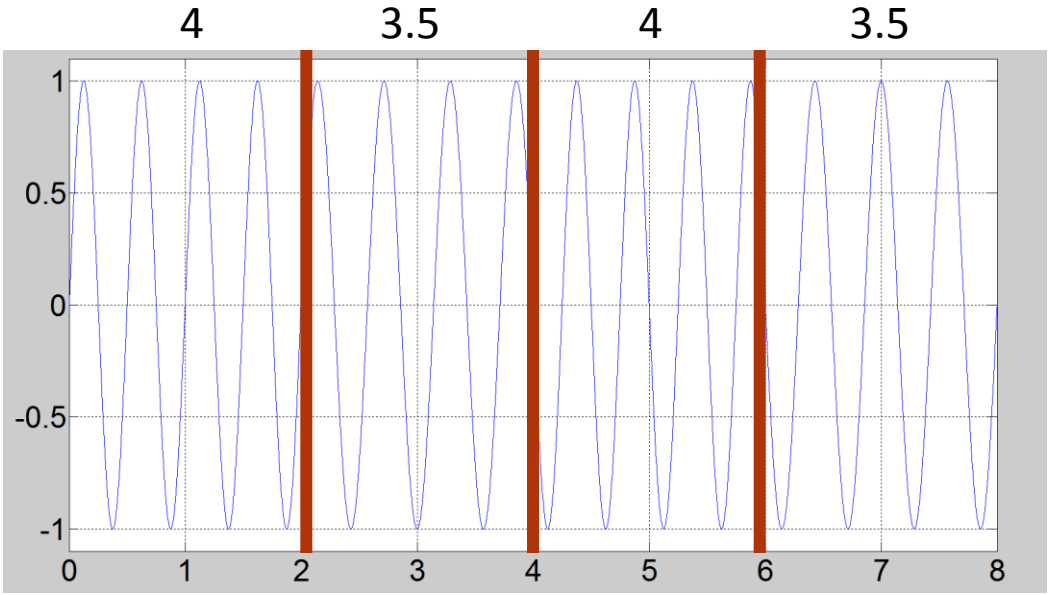




Another method – ufsook

- Only use 2 frequencies, 30*n and 30*(n - 0.5)
- What's the frame difference ?
  - 30*n : 0, since same phase
  - 30*(n-0.5) : 255 * #pixels

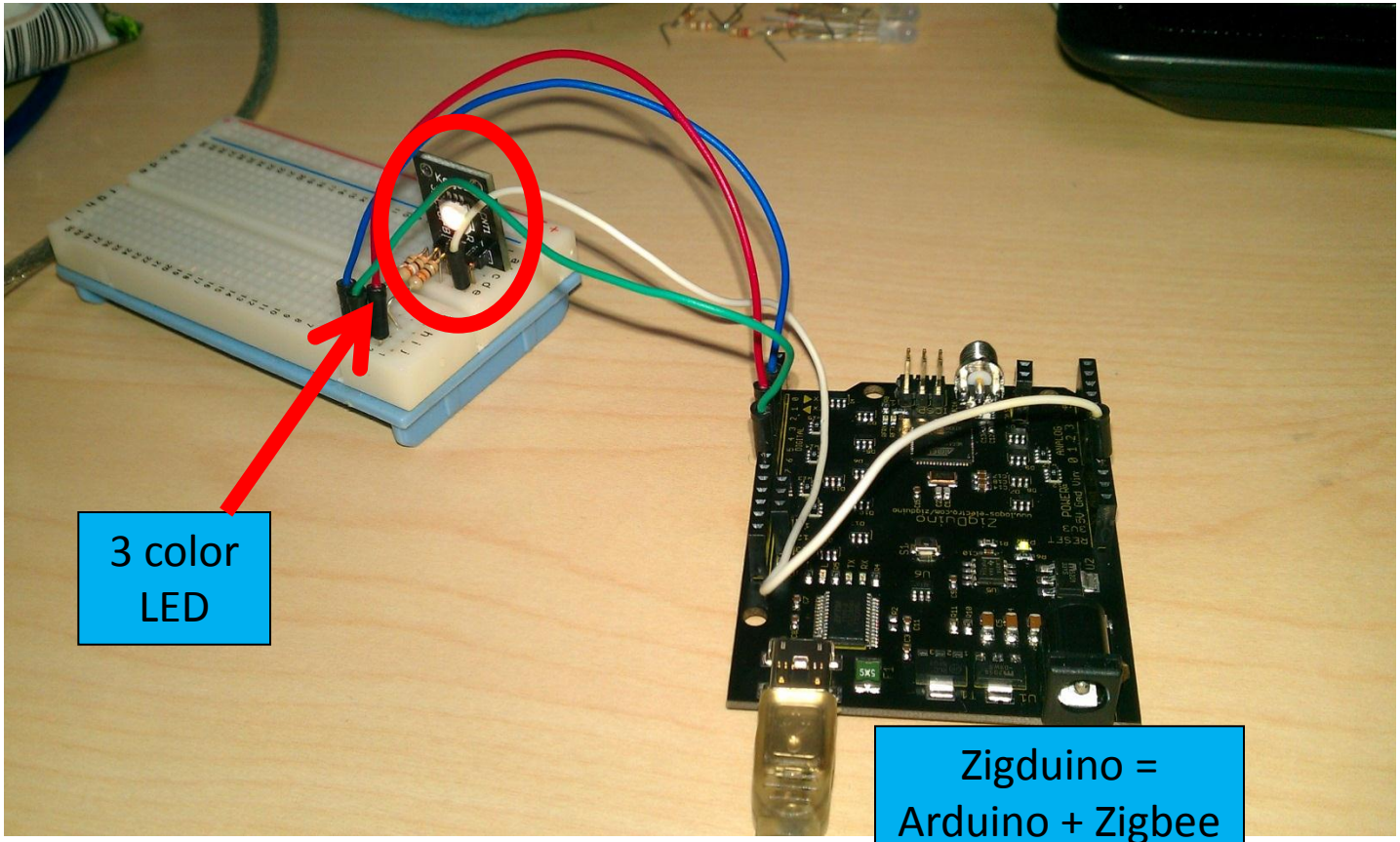




http://w.csie.org/~r01922104/hw1.3gp

Our Tx, Rx hardware

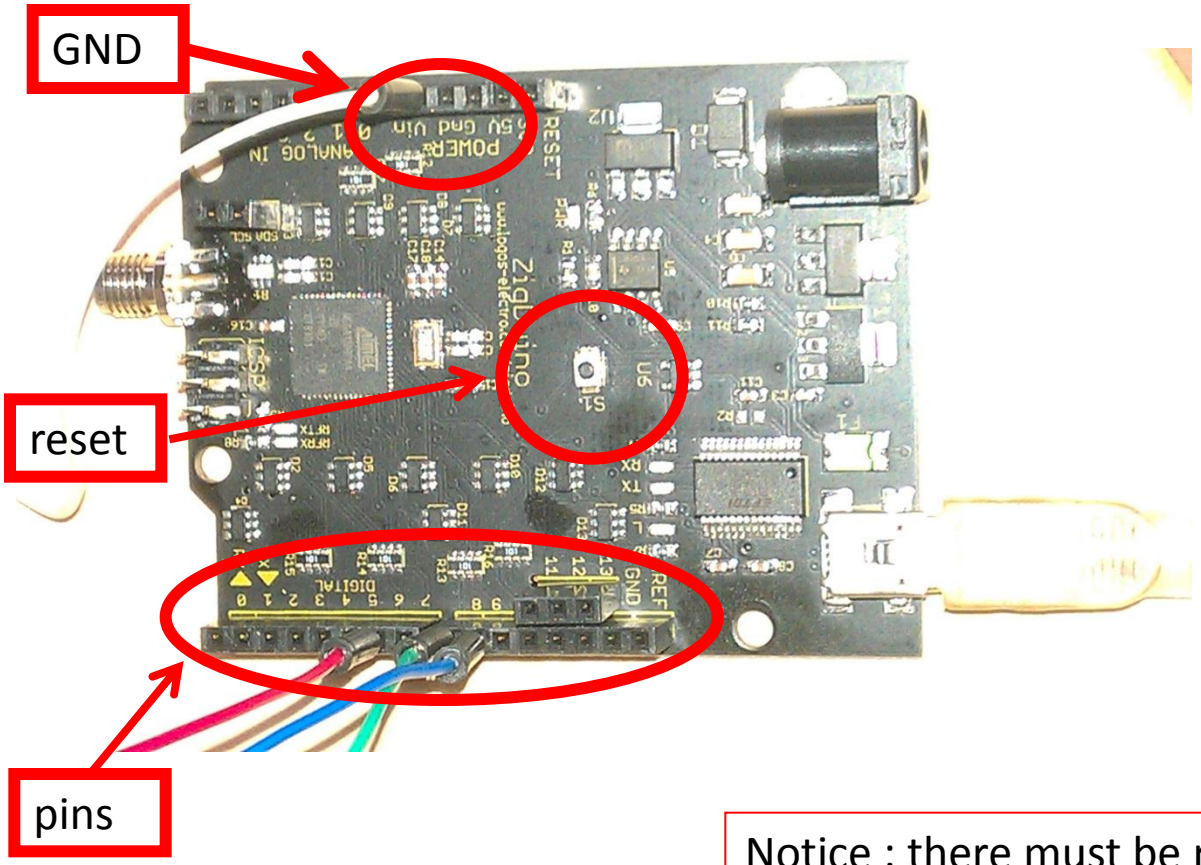
Transmitter :
Zigduino + LED

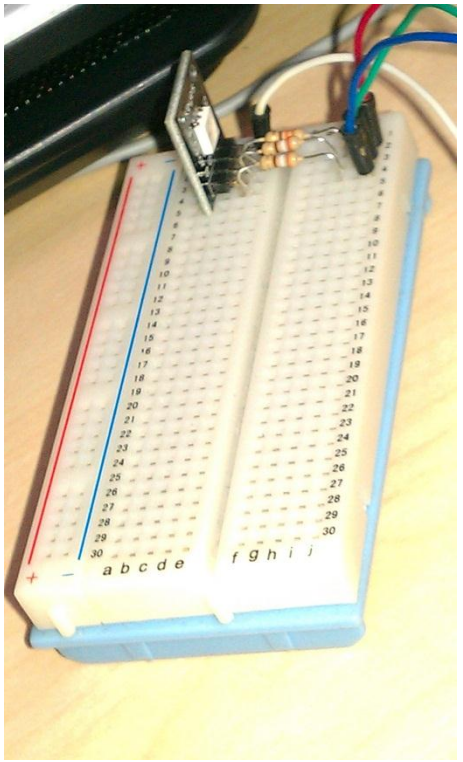


Receiver :
Camera

Tx part : Zigduino + LEDs

- http://www.csie.ntu.edu.tw/~hsinmu/courses/lib/exe/fetch.php?media=arduino_ins.pdf





Notice : there must be resistors (電阻) in the circuit

First program – Blink.ino

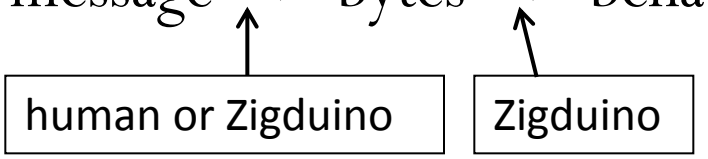
```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
} /* note: you can use analogWrite(led, value) instead */
```

Tx part

- message => bytes => behavior of LEDs

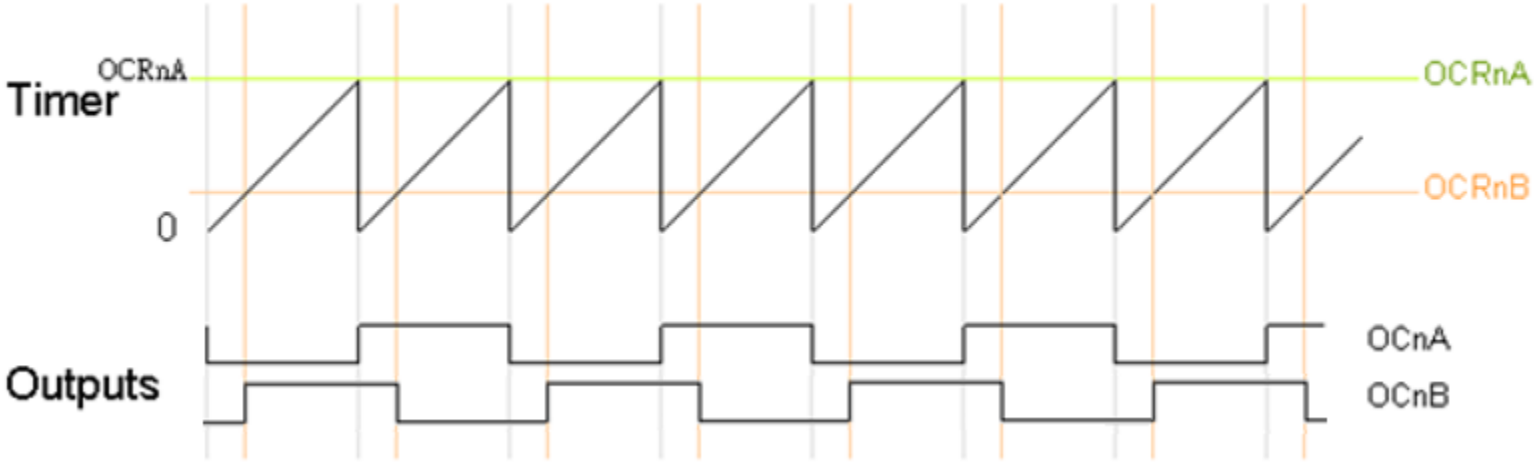


- You can use anything to represent bits:
  - color, frequency, luminance, waveform, duplication, ……
  - Don't forget the low sampling rate of camera (30fps)

- Get extra score if people cannot notice the behavior of LEDs
- Range your score with data rate

CTC mode

- Clear Timer on Compare (timer == OCRnA)
- Generate accurate square waves
  - all done by hardware, won't affected by interrupt
  - Can simultaneously use all 3 timers easily.



- Set the corresponding bits in TCCRnX(n=1,2,3 X=A,B) registers for mode selection and timer prescaler
- Set OCRnX for "frequency"(OCRnA) or "delay"(OCRnB,OCRnC)
- Frequency = 16M / prescaler / OCRnA / 2
- Additional support : interrupt handler (see document)

CTC mode – sample code

```
void setup() {
  // use Timer1, set control registers here
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
 TCCR1A = _BV(COM1A0) | _BV(COM1B0) | _BV(COM1C0);
 TCCR1B = _BV(WGM12) | _BV(CS12) | _BV(CS10);
  OCR1A = 32767;
  OCR1B = 16383;
  OCR1C = 8191;
}
void loop(){
    // change OCRnX values to manipulate frequency
    OCR1A = 32767, OCR1B = 16383, OCR1C = 8191;
    delay(10000);  // this may not be so accurate
    OCR1A = 16383,OCR1B = 8191,OCR1C = 4095;
    delay(10000);
}
```

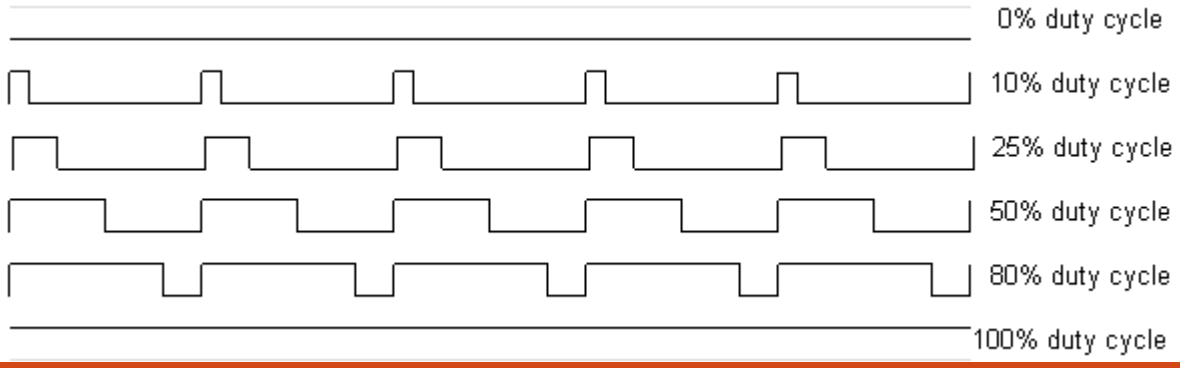
CTC mode

Timer name	sizeof registers	Control value (fill TCCRnX)	Prescaler	Output pin name <-> pin in Zigduino
Timer1	16-bit	WGM = 4 COM1A/B/C = 1 CS = 1/2/3/4/5	1 / 8 / 64/ 256 / 1024	OC1A <-> pin11 OC1B <-> pin10 OC1C <-> pin9 (pin11 need a jumper)
Timer3	16-bit	WGM = 4 COM3A/B/C = 1 CS = 1/2/3/4/5	1 / 8 / 64/ 256 / 1024	OC3A <-> pin5 OC3B <-> pin6 OC3C <-> pin3
Timer2	8-bit	WGM = 2 COM2A = 1 CS = 1~7	1/8/32/64 /128/256/ 1024	OC2A <-> pin8

- For more details
  - http://www.atmel.com/Images/doc8266.pdf (register definition)
  - https://static.squarespace.com/static/511f4f0de4b09463c7605f13/t/5275c478e4b07e72f74c7442/1383449720133/zigduino-v6c-schematic.pdf

Fast PWM (with OCRnA top)

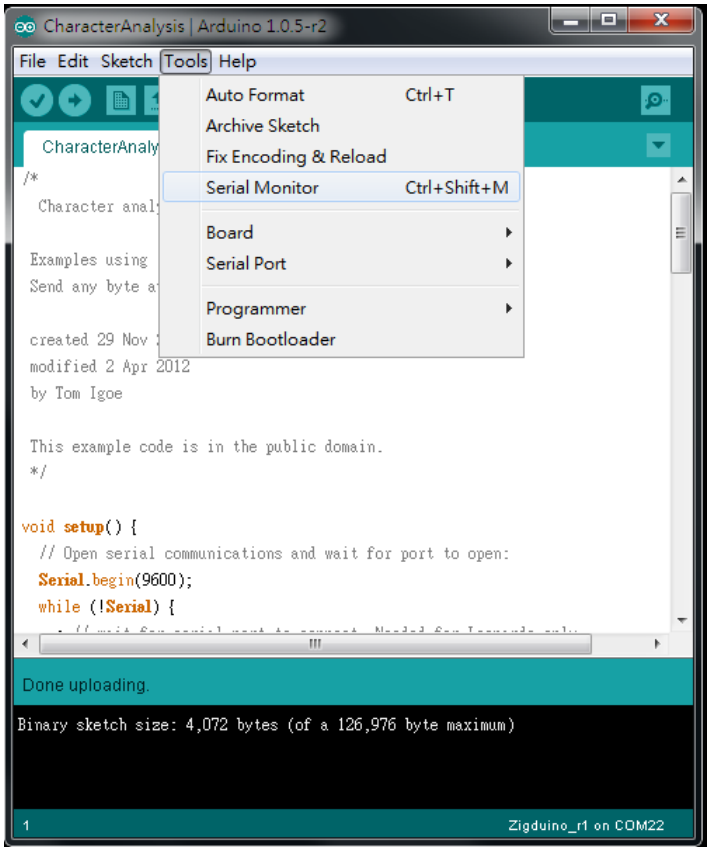
- Pulse-width modulation
  - Can be used to control duty cycle
  - http://arduino.cc/en/Tutorial/SecretsOfArduinoPWM (pins are different)
  - Differences in luminance and ratio of light/dark stripe width

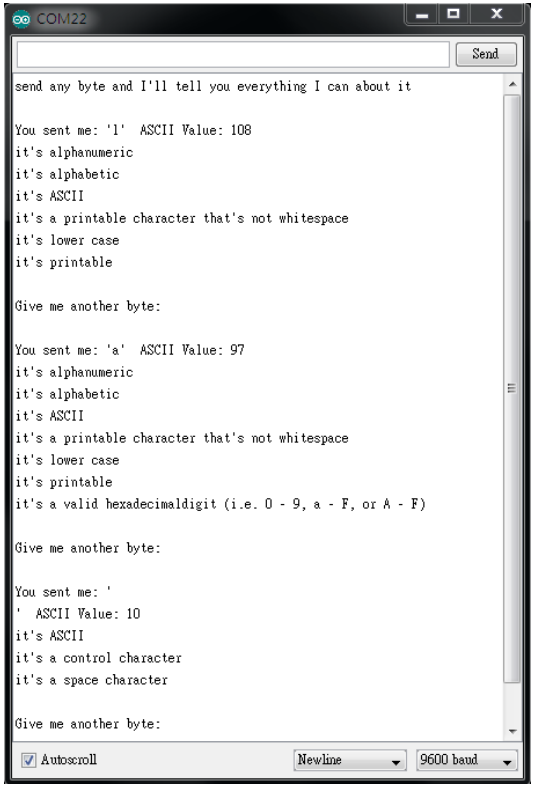



```
pinMode(6, OUTPUT);
 TCCR3A = _BV(COM3B1) | _BV(WGM31) | _BV(WGM30);
 TCCR3B = _BV(WGM32) | _BV(WGM31) | _BV(CS32) | _BV(CS30); // TOP=0xFFFF
 OCR3A = 8191;
 OCR3B = 2047; // 25% duty cycle
```

serial port monitor – text I/O

- see Examples 08. Strings
- Use the serial monitor to communicate





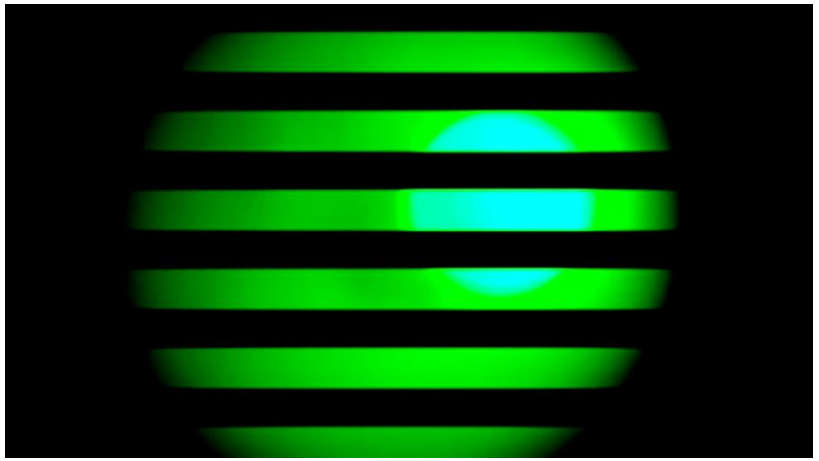
Rx part : video processing

- video (-> image) -> get LEDs behavior-> bytes -> message
- Video to image:
  - ffmpeg –i input.avi ouput%d.jpg
  - supported by windows, linux, and OS X

- Counting Stripes : need some DIP
  - find LED position, image diff, color histogram, edge detection, Fourier transform, auto-correlation, ……
  - 3 basic ways : DIP, FFT, auto-correlation

Rx part : an example pseduo code

```
for all images
{
        read image;
        find bounding box // where are the stripes
        calculate row power
        count stripes to get freq // through DIP or FFT
        get output bit by freq
}
```



Demo – 3/27 (Thu.)

- Please register the demo slot. And come to CSIE R424 at that time.
  - https://docs.google.com/spreadsheet/ccc?key=0AgYjnPH1HzgydFA3VmJxcDVncEtQYjQySk9sRF9sa0E&usp=drive_web#gid=0
- All you should do are :
  1. Transmit the character string given by TA
  2. Record the videos
  3. Decode and get the message
- grade :
  - Tx (20% + 10% for not noticed by human eyes + 10% data rate)
  - Rx : Video Decode (30% + 10% for accuracy)
  - Report : describe what you have done, and problems encountered (20%)
  - We will choose 2 groups to demo in the next class, and their grade will be 100%.

Any questions ?

- Contact to TAs :
  - facebook https://www.facebook.com/groups/wn14spring/
  - Email : wn@csie.ntu.edu.tw
  - Office hour : Mon 14:00 ~ 16:00 @ CSIE R424