陽明交大
NYCU

物聯網裝置與平台
IoT Devices and Platforms

曾煜棋、吳昆儒

National Yang Ming Chiao Tung University

Ch 5, Sensors (1) - Summary

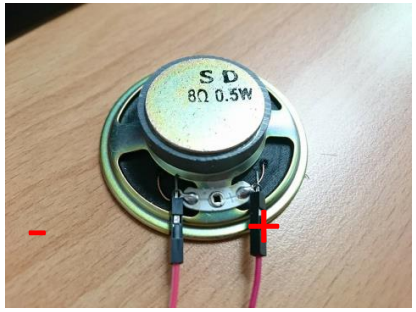
- Arduino IDE and how it interacts with the external world
  - Sensors
- Understanding this course:
  - Discussion: upload to e3
  - Quiz: show your code to TA
    - For remote-access, use discord to interact with TA

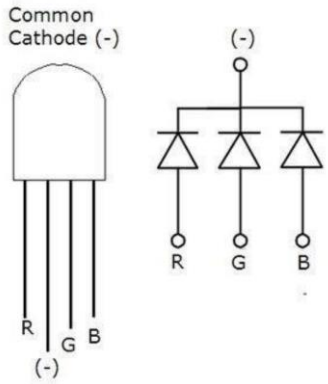
Labs (Last week)

1. **AnalogInOutSerial**: Read an analog input pin, map the result, and then use that data to dim or brighten an LED.
2. **AnalogInput**: Use a potentiometer to control the blinking of an LED.
3. **Calibration**: Define a maximum and a minimum for expected analog sensor values.
4. **Fading**: Use an analog output (PWM pin) to fade an LED.
5. **Smoothing**: Smooth multiple readings of an analog input.

Labs (This week)

1. **toneMelody: play a melody with a piezo speaker**
2. **tonePitchFollower: play a pitch on a piezo speaker depending on an analog input.**
3. **ReadASCIIString (RGB LED): Parse a comma-separated string of ints to fade an LED.**
4. **Ultrasonic sensor**
   - Measure distance





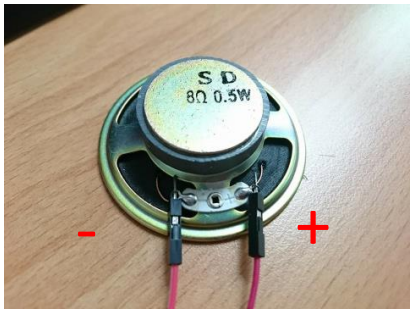
Ultrasonic

Tone Melody

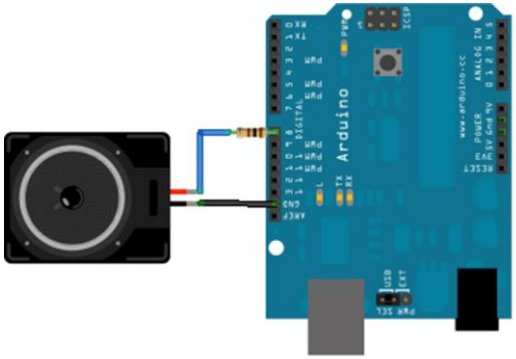
play a melody with a Piezo

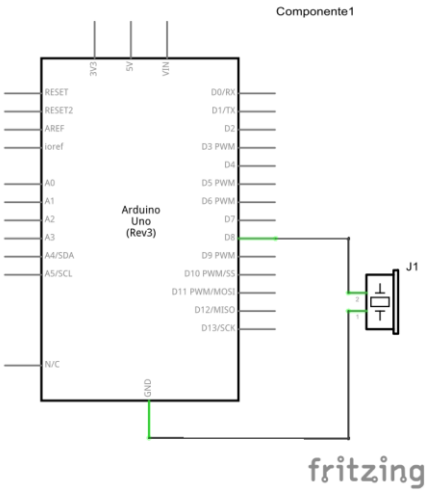
Lab1. Tone Melody

- Goal: shows how to use the tone() command to generate notes. It plays a little melody you may have heard before.
- Hardware Required
  - Arduino board
  - piezo buzzer or a speak with 100 ohm resistor
  - hook-up wire



P=0.5 (W); V=5 (V)
$P=V^2/R$
R=50 ohm

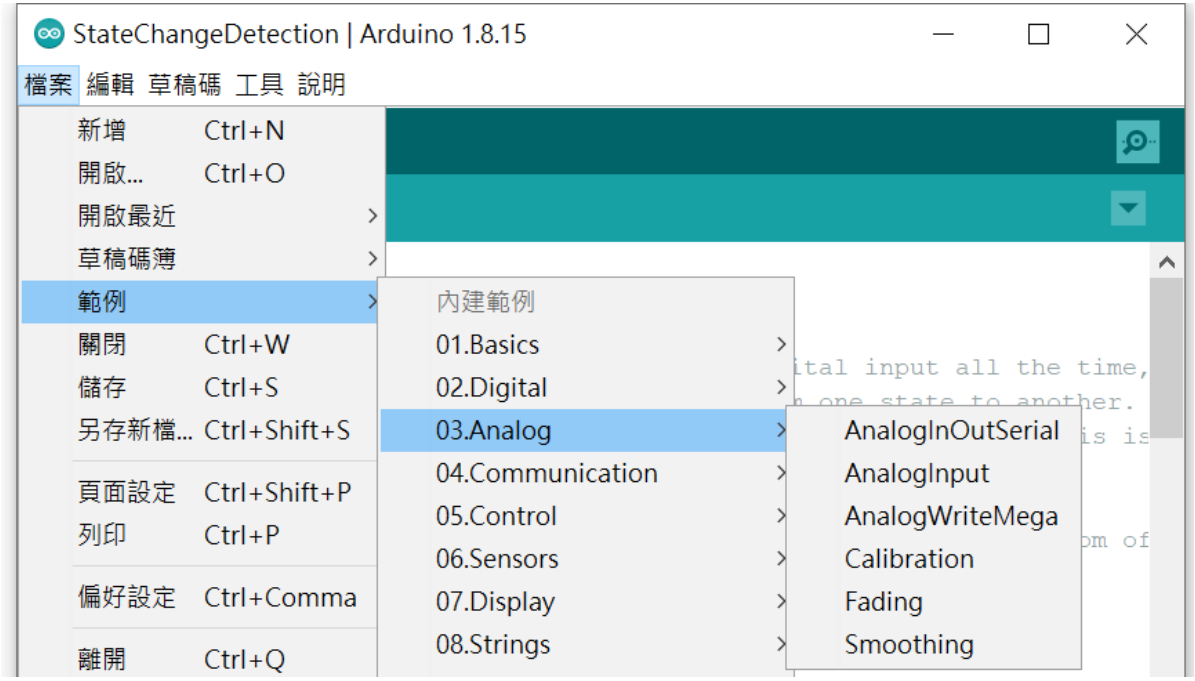




Lab1. Tone Melody

Arduino IDE

Open--->File--->Examples--->02.Digital-> ToneMelody



Lab1. Tone Melody

```
#include "pitches.h"

// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {
```

```
    // to calculate the note duration, take one second divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(8);
  }
}

void loop() {
  // no need to repeat the melody.
}
```

Lab1. Syntax

- Syntax
  - tone(pin, frequency, duration)
- Description
  - Generates a square wave of the specified frequency.
  - A duration can be specified, otherwise the wave continues until a call to noTone(). // see next page
  - Only one tone can be generated at a time.
    - If a tone is already playing on a different pin, the call to tone() will have no effect. (Need to call noTone())
    - If the tone is playing on the same pin, the call will set its frequency.
- Example
  - tone(8, melody[thisNote], noteDuration);

https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/

Lab1. Syntax

- Syntax
  - noTone(pin)
- Description
  - Stops the generation of a square wave triggered by tone().
  - Has no effect if no tone is being generated.
- Parameters
  - pin: the pin on which to stop generating the tone

Discussion 1

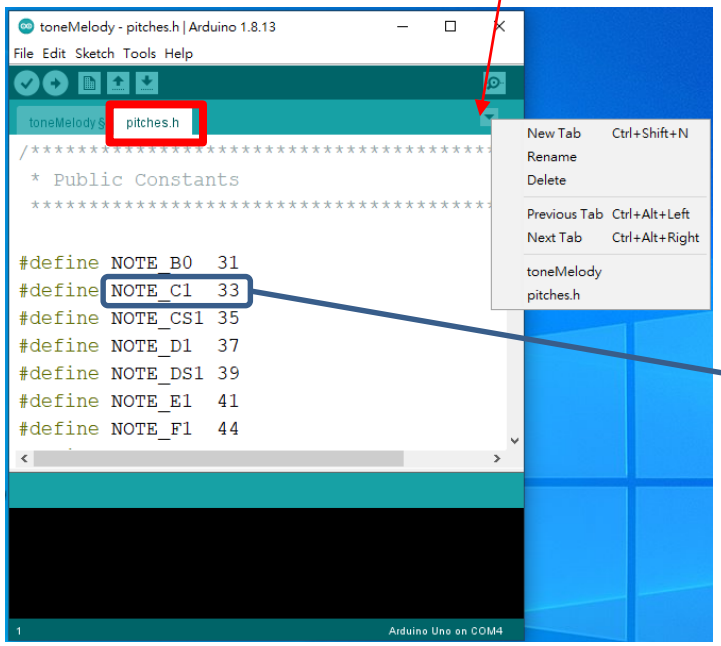
- Why do we need pauseBetweenotes?
- Try to change the multiplier in "noteDuration*1.30" to 1, 0.5, and 1.5.

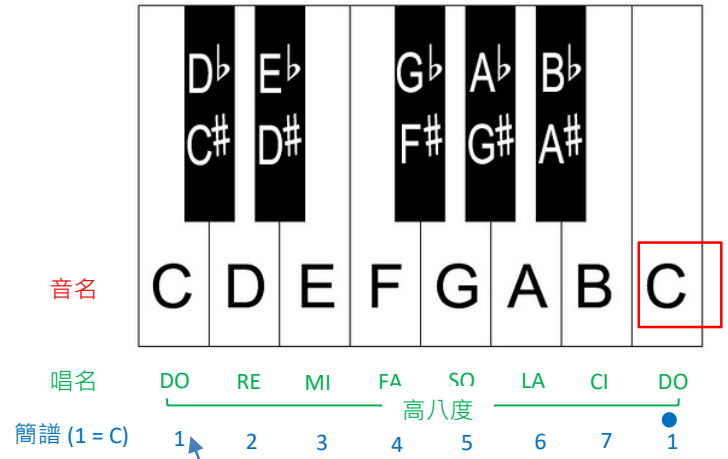
```
for (int thisNote = 0; thisNote < 8; thisNote++) {
  // to calculate note duration, take one second divided by note type
  int noteDuration = 1000/noteDurations[thisNote];  // 1 sec = 4 ♩ = 8 ♪
  tone(8, melody[thisNote], noteDuration);
  int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
  noTone(8);                    // stop the tone playing
}
```

Lab1. Tone Melody

pitches.h

Create new tab





高八度

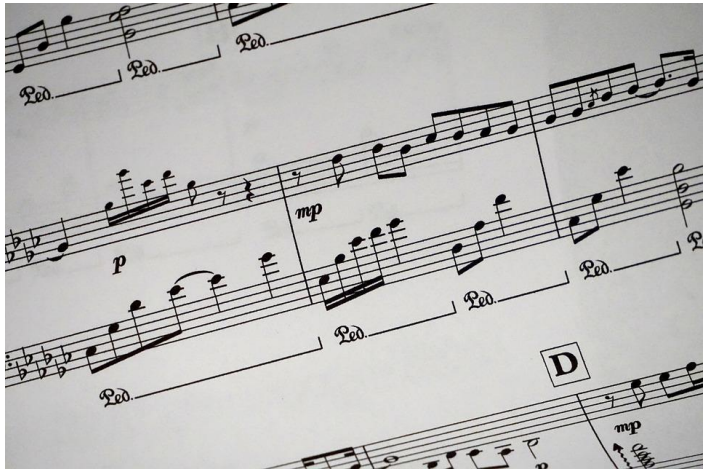
頻率，單位為赫茲。括號內為距離中央C（261.63赫茲）的半音距離。

八度→ / 音名↓	0	1	2	3	4	5	6	7	8	9
C	16.352 (−48)	32.703 (−36)	65.406 (−24)	130.81 (−12)	261.63 (0)	523.25 (+12)	1046.5 (+24)	2093.0 (+36)	4186.0 (+48)	8372.0 (+60)
C♯/D♭	17.324 (−47)	34.648 (−35)	69.296 (−23)	138.59 (−11)	277.18 (+1)	554.37 (+13)	1108.7 (+25)	2217.5 (+37)	4434.9 (+49)	8869.8 (+61)
D	18.354 (−46)	36.708 (−34)	73.416 (−22)	146.83 (−10)	293.66 (+2)	587.33 (+14)	1174.7 (+26)	2349.3 (+38)	4698.6 (+50)	9397.3 (+62)
D♯/E♭	19.445 (−45)	38.891 (−33)	77.782 (−21)	155.56 (−9)	311.13 (+3)	622.25 (+15)	1244.5 (+27)	2489.0 (+39)	4978.0 (+51)	9956.1 (+63)
E	20.602 (−44)	41.203 (−32)	82.407 (−20)	164.81 (−8)	329.63 (+4)	659.26 (+16)	1318.5 (+28)	2637.0 (+40)	5274.0 (+52)	10548 (+64)
F	21.827 (−43)	43.654 (−31)	87.307 (−19)	174.61 (−7)	349.23 (+5)	698.46 (+17)	1396.9 (+29)	2793.8 (+41)	5587.7 (+53)	11175 (+65)
F♯/G♭	23.125 (−42)	46.249 (−30)	92.499 (−18)	185.00 (−6)	369.99 (+6)	739.99 (+18)	1480.0 (+30)	2960.0 (+42)	5919.9 (+54)	11840 (+66)
G	24.500 (−41)	48.999 (−29)	97.999 (−17)	196.00 (−5)	392.00 (+7)	783.99 (+19)	1568.0 (+31)	3136.0 (+43)	6271.9 (+55)	12544 (+67)
G♯/A♭	25.957 (−40)	51.913 (−28)	103.83 (−16)	207.65 (−4)	415.30 (+8)	830.61 (+20)	1661.2 (+32)	3322.4 (+44)	6644.9 (+56)	13290 (+68)
A	27.500 (−39)	55.000 (−27)	110.00 (−15)	220.00 (−3)	440.00 (+9)	880.00 (+21)	1760.0 (+33)	3520.0 (+45)	7040.0 (+57)	14080 (+69)
A♯/B♭	29.135 (−38)	58.270 (−26)	116.54 (−14)	233.08 (−2)	466.16 (+10)	932.33 (+22)	1864.7 (+34)	3729.3 (+46)	7458.6 (+58)	14917 (+70)
B	30.868 (−37)	61.735 (−25)	123.47 (−13)	246.94 (−1)	493.88 (+11)	987.77 (+23)	1975.5 (+35)	3951.1 (+47)	7902.1 (+59)	15804 (+71)

https://zh.wikipedia.org/wiki/%E9%9F%B3%E9%AB%98

Quiz 1

- Design a short song and play!!



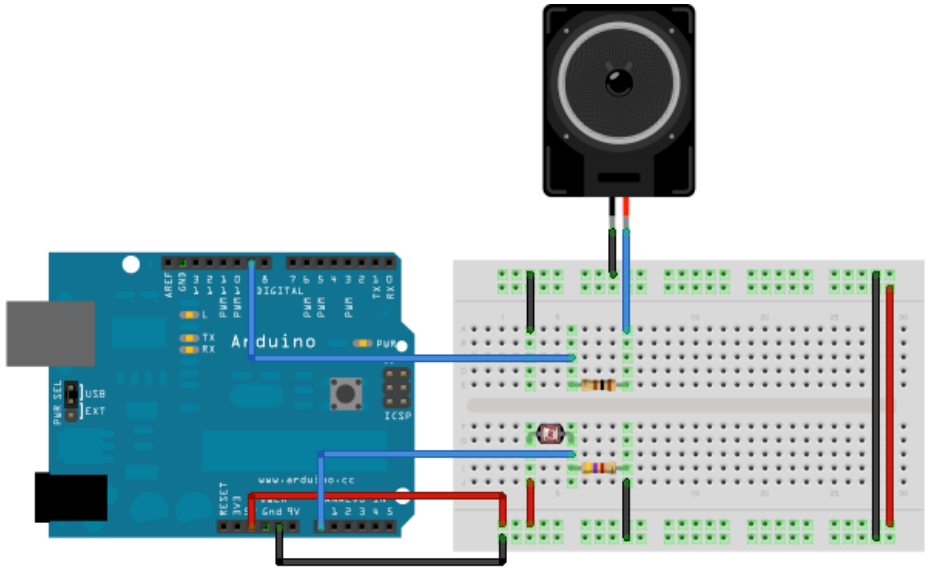
melody[] = {?????????};

Lab 2. Pitch follower

play a pitch on a piezo speaker depending on an analog input

Lab 2. Pitch follower

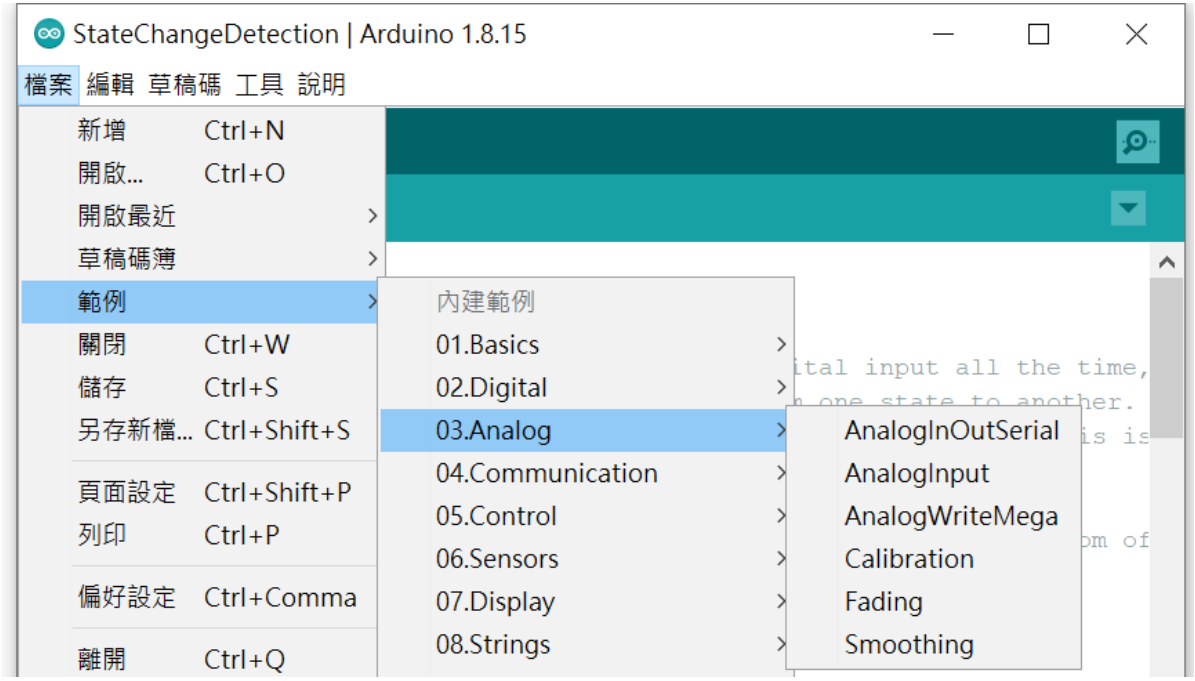
- Goal: This example shows how to use the tone() command to generate a pitch that follows the values of an analog input
- Hardware Required
  - 8-ohm speaker
  - 1 photocell
  - 4.7K ohm resistor
  - 100 ohm resistor
  - breadboard
  - hook up wire



Lab 2. Pitch follower

Arduino IDE

Open--->File--->Examples--->02.Digital->tonePitchFollower



Built-in sample code

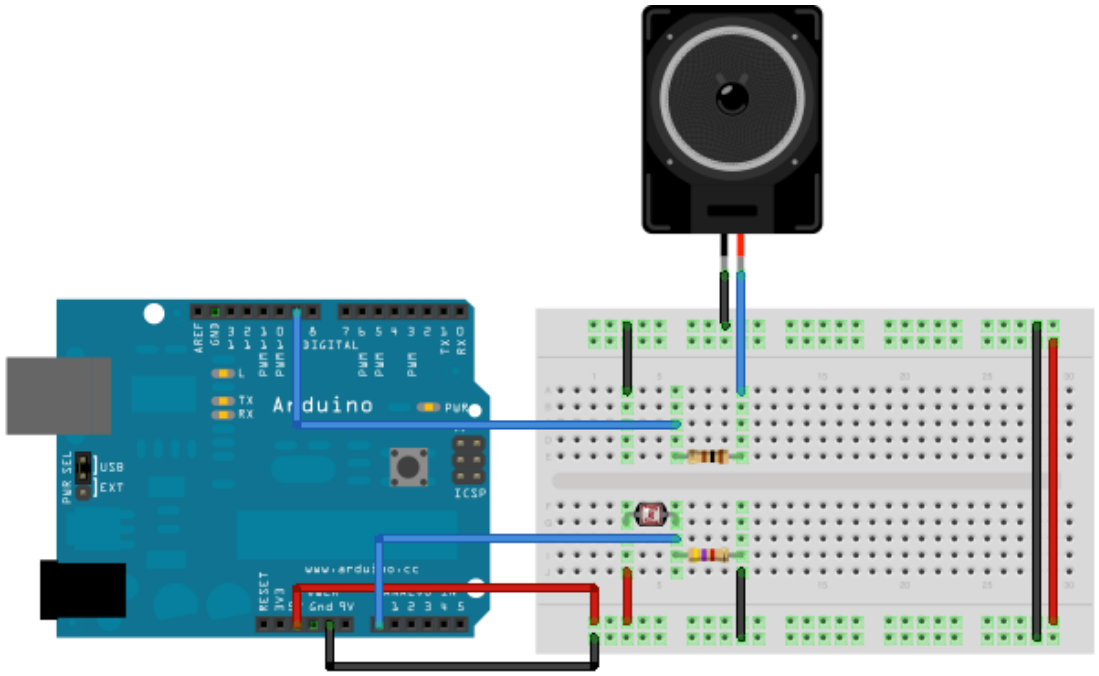
```
void setup() {
  // initialize serial communications (for debugging only):
  Serial.begin(9600);
}

void loop() {
  // read the sensor:
  int sensorReading = analogRead(A0);
  // print the sensor reading so you know its range
  Serial.println(sensorReading);
  // map the analog input range (in this case, 400 - 1000 from the photoresistor)
  // to the output pitch range (120 - 1500Hz)
  // change the minimum and maximum input numbers below depending on the range
  // your sensor's giving:
  int thisPitch = map(sensorReading, 400, 1000, 120, 1500);

  // play the pitch:
  tone(9, thisPitch, 10);
  delay(1);        // delay in between reads for stability
}
```

Lab2. syntax

- Syntax
  - map(value, fromLow, fromHigh, toLow, toHigh)
- Description
  - Re-maps a number from one range to another.
  - Does not constrain values to within the range.
  - May use constrain() function either before or after
- Example
  - /* Map an analog value (10bit) to 8 bits */
  - val = map(val, 0, 1023, 0, 255);

Discussion 2

- In addition to use photocell, what else sensors can we use to control a speaker?
  - Think of any other sensors with analog readings.



Quiz 2

- Compose a melody by yourself
  1. Use a potentiometer to change the frequency of tone (pin, frequency, duration)
  2. Use a button to record the value of potentiometer as frequency (max 10 values)
  3. Press the button for 5 sec to replay the melody you composed

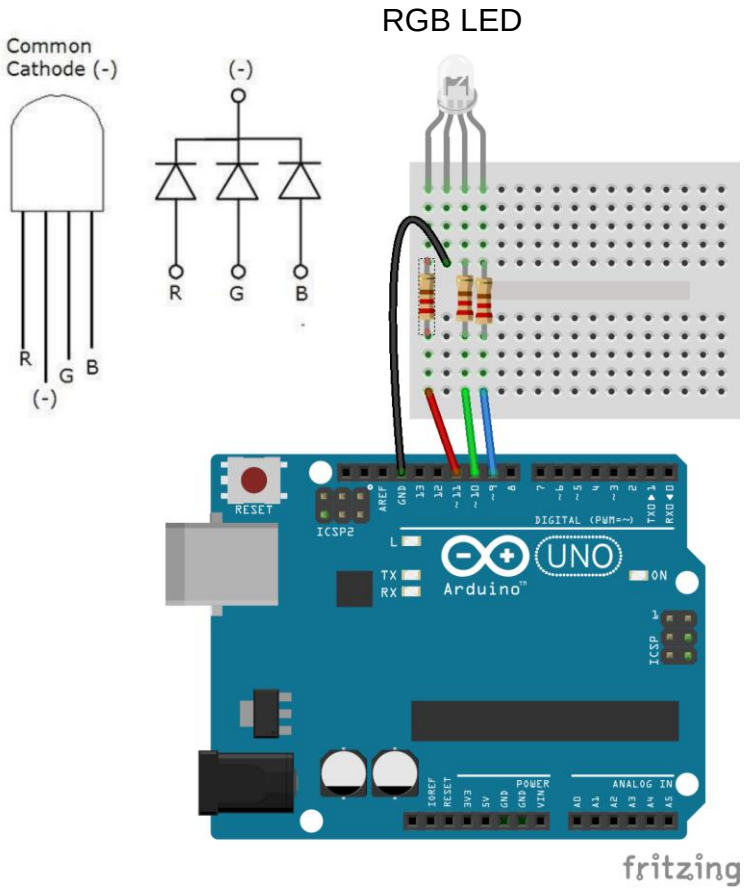
- Hint:
  - When a button is pressed, the value of potentiometer is added into melody[] = {frequency1, frequency2, …., frequency 10}.
  - In other words, 10 values are added into melody[] after pressing the button for 10 times.
  - Then press the button for 5 sec to playback the melody you recorded.

Lab3. ReadASCIIString (RGB LED)

Parse a comma-separated string of ints to fade an LED.

Lab3. ReadASCIIString

- Goal: use the Serial.parseInt() function to locate values separated by a non-alphanumeric character.
- Hardware Required
  - Arduino Board
  - Breadboard
  - Hookup wire
  - Common anode RGB LED
  - Three 220-ohm resistors



Lab3. ReadASCIIString

Arduino IDE

Open--->File--->Examples--->04. Communication--->ReadASCIIString

- Step 1: Open example code --->04. Communication--->ReadASCIIString
- Step 2: Open your Serial monitor.
- Step 3: Enter values between 0-255 for the lights in the following format : “Number1, Number2, Number3 \n”. (\n = press enter)
- Once you have sent the values to the Arduino, the attached LED will turn the color you specified, and you will receive the HEX values in the serial monitor.

Lab3. ReadASCIIString

```
// pins for the LEDs:
const int redPin = 3;
const int greenPin = 5;
const int bluePin = 6;

void setup() {
  // initialize serial:
  Serial.begin(9600);
  // make the pins outputs:
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}
```

Lab3. ReadASCIIString

```
void loop() {
  // if there's any serial available, read it:
  while (Serial.available() > 0) {

    // look for the next valid integer in the incoming serial stream:
    int red = Serial.parseInt();
    // do it again:
    int green = Serial.parseInt();
    // do it again:
    int blue = Serial.parseInt();

    // look for the newline. That's the end of your sentence:
    if (Serial.read() == '\n') {
```

Lab3. ReadASCIIString

```
    // constrain the values to 0 - 255 and invert
    // if you're using a common-cathode LED, just use "constrain(color, 0, 255);"
    red = 255 - constrain(red, 0, 255);
    green = 255 - constrain(green, 0, 255);
    blue = 255 - constrain(blue, 0, 255);

    // fade the red, green, and blue legs of the LED:
    analogWrite(redPin, red);
    analogWrite(greenPin, green);
    analogWrite(bluePin, blue);

    // print the three numbers in one string as hexadecimal:
    Serial.print(red, HEX);
    Serial.print(green, HEX);
    Serial.println(blue, HEX);
  }
 }
}
```

Lab3. syntax

- Syntax
  - Serial.available()
- Description
  - **Get the number of bytes (characters) available** for reading from the serial port.
  - This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes).
- Returns
  - The number of bytes available to read.
- Example
  - if (Serial.available() > 0) { // reply only when you receive data:
    }

https://www.arduino.cc/reference/en/language/functions/communication/serial/available/

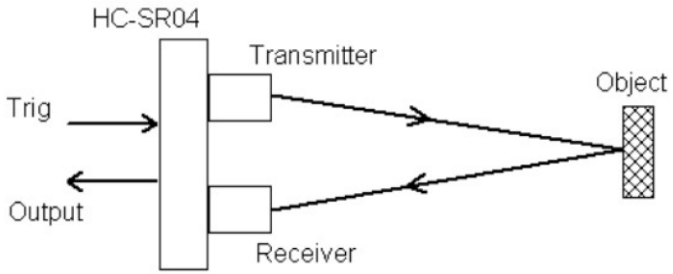
Lab3. syntax

- Syntax
  - Serial.parseInt()
- Description
  - **Looks for the next valid integer** in the incoming serial stream.parseInt() inherits from the Stream utility class.
- Returns
  - the next valid integer (long)
- Example
  - int red = Serial.parseInt();

https://www.arduino.cc/reference/en/language/functions/communication/serial/parseint/

Lab4. Ultrasonic:

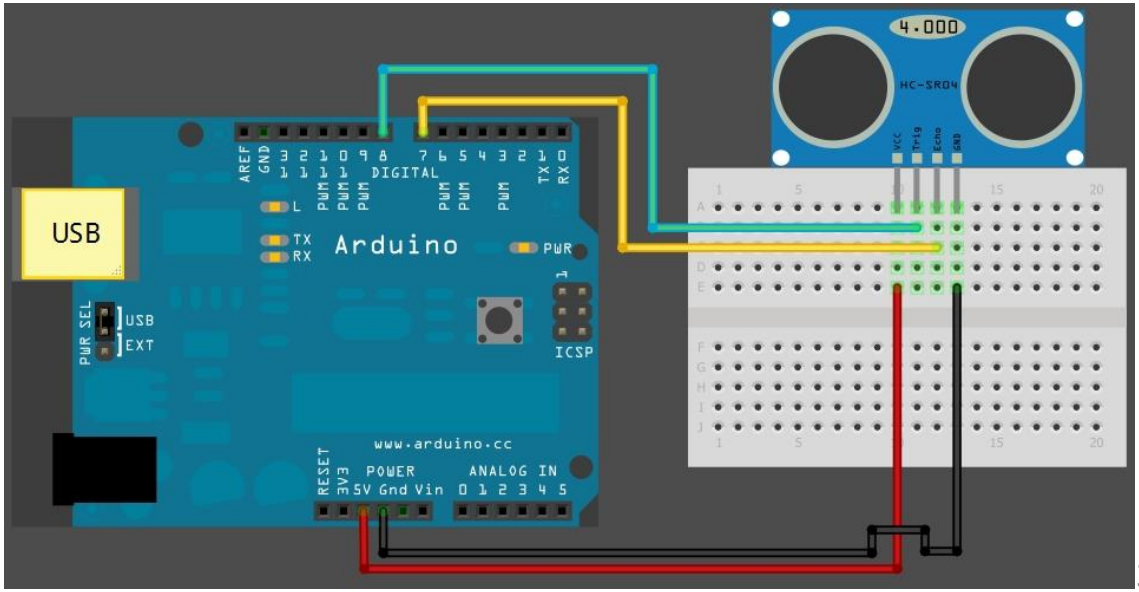
Measure distance



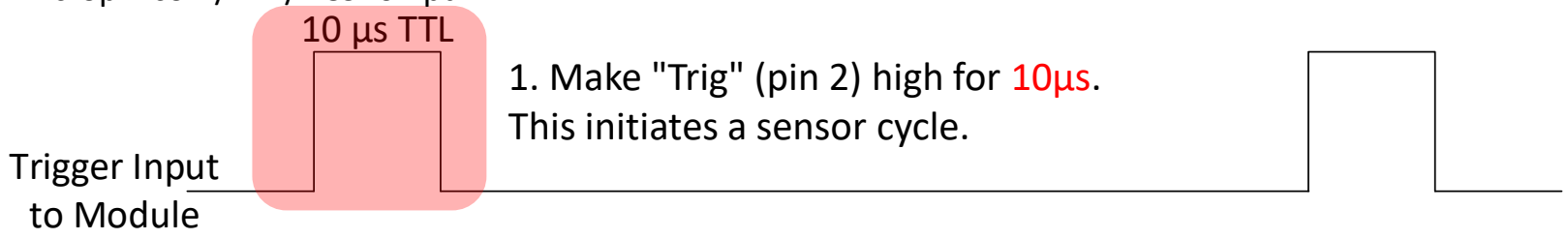
Lab4. Ultrasonic

- Goal: Measure distance by using ultrasonic sensor. Demonstrates use of the pulseIn() function.
- Hardware Required
  - Arduino Board
  - HC-SR04 ultrasonic sensor
  - Breadboard
  - hook-up wire

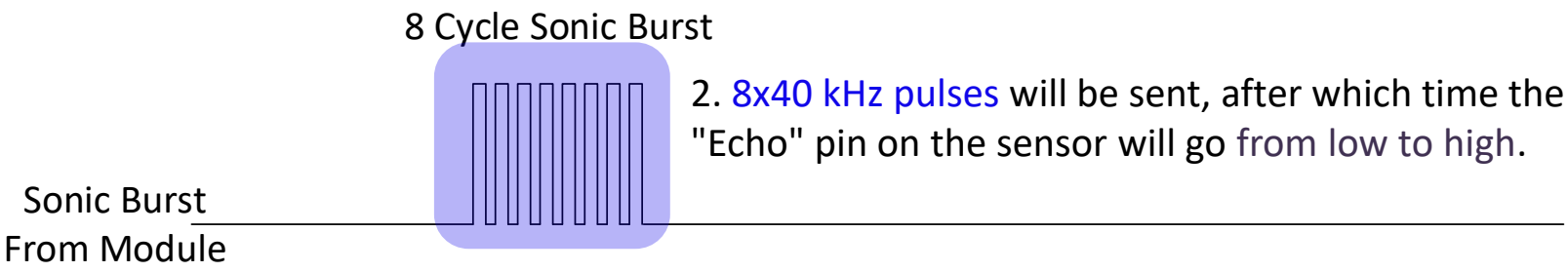
VCC, Trigger, Echo, GND



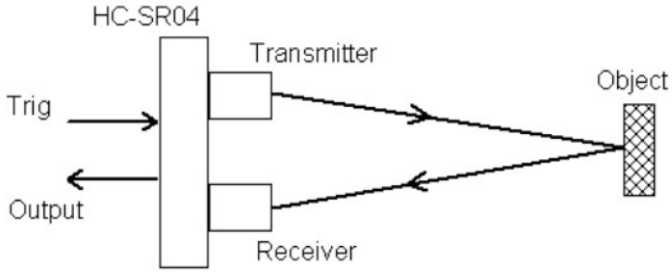
http://www.micropik.com/PDF/HCSR04.pdf



1. Make "Trig" (pin 2) high for 10µs.
This initiates a sensor cycle.

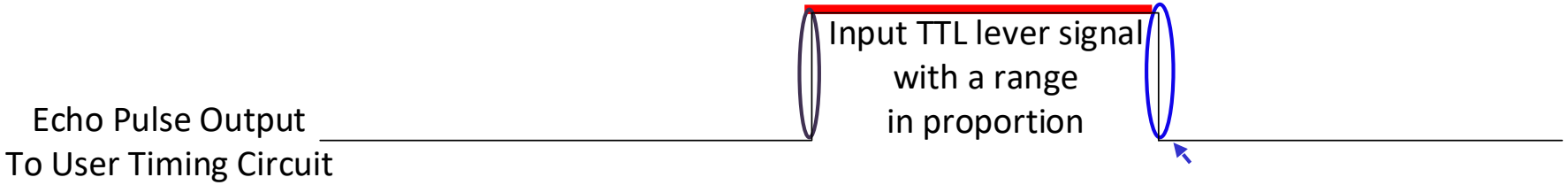


2. 8x40 kHz pulses will be sent, after which time the "Echo" pin on the sensor will go from low to high.



3. The 40kHz sound wave will bounce off the nearest object and return to the sensor.

VCC, Trigger, Echo, GND



4. When the sensor detects the reflected sound wave, the Echo pin will go low again.

5. The **distance** between the sensor and the detected object can be calculated based on **the length of time the Echo pin is high**.

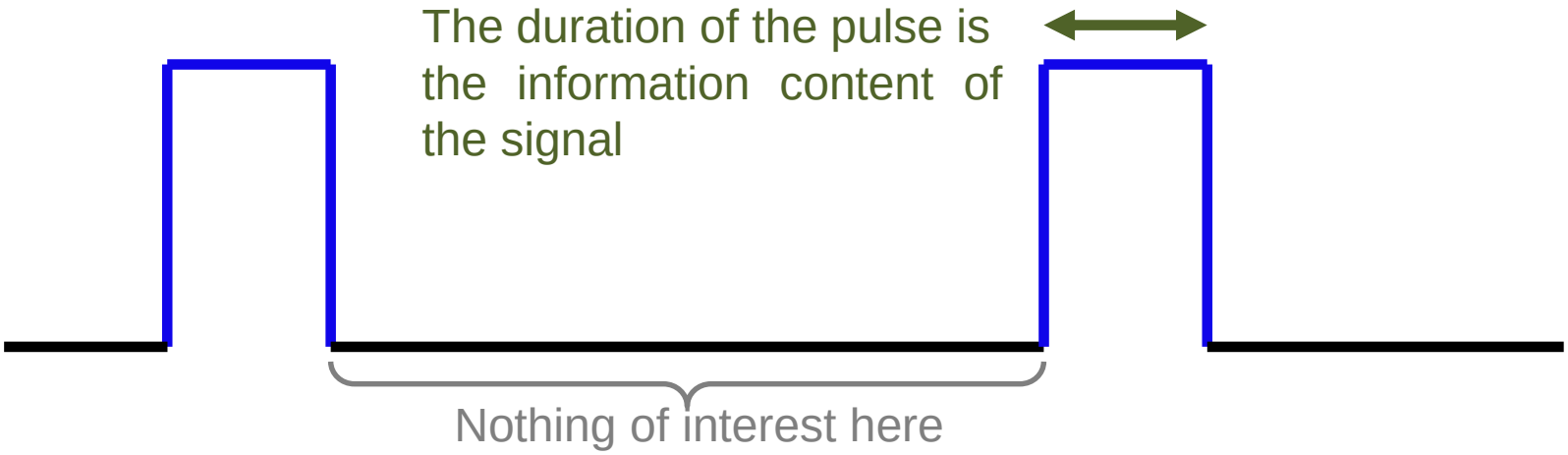
HC-SR04 operation

1. Make "Trig" (pin 2) high for 10μs. This initiates a sensor cycle.
2. 8x40 kHz pulses will be sent, after which time the "Echo" pin on the sensor will go from low to high.
3. The 40kHz sound wave will bounce off the nearest object and return to the sensor.
4. When the sensor detects the reflected sound wave, the Echo pin will go low again.
5. The distance between the sensor and the detected object can be calculated based on the length of time the Echo pin is high.
6. If no object is detected, the Echo pin will stay high for 38ms and then go low.

Datasheet: http://www.micropik.com/PDF/HCSR04.pdf

Lab4. Ultrasonic

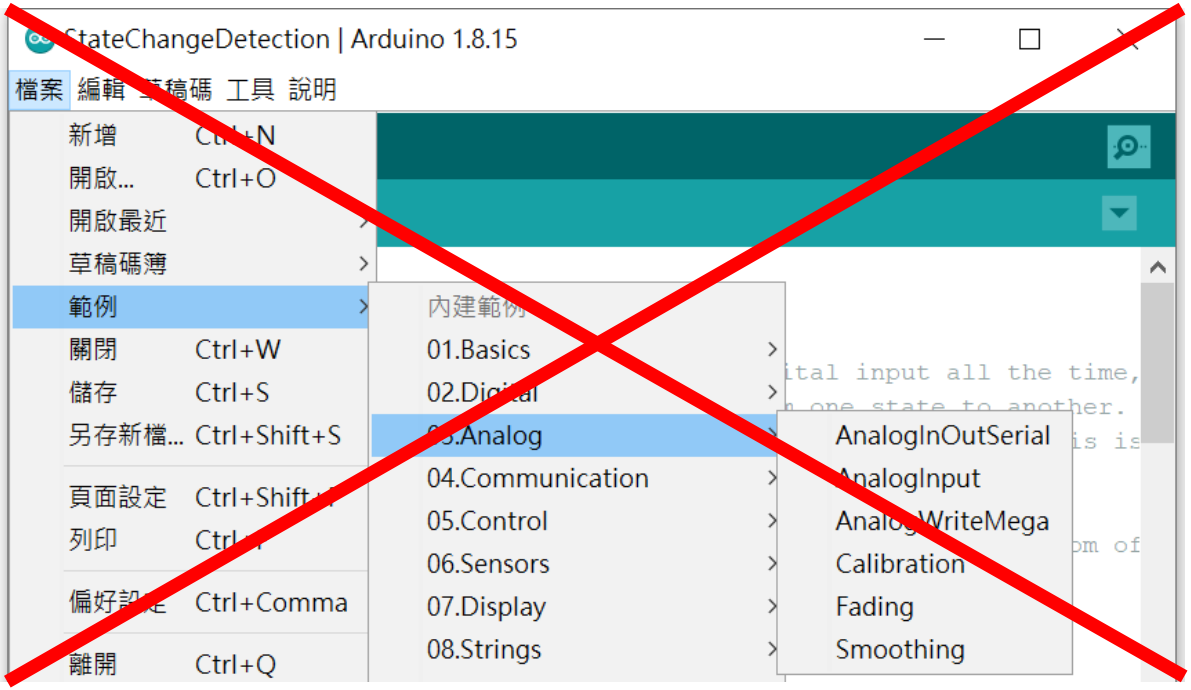
- Function: pulseIn
- Description
  - Reads a pulse (either HIGH or LOW) on a pin.
  - For example, if **value** is **HIGH**, pulseIn() waits for the pin to go **HIGH**, starts timing, then waits for the pin to go **LOW** and stops timing.
  - **Returns the length of the pulse in microseconds**.
    Gives up and returns 0 if no pulse starts within a specified time out.



Lab4. Ultrasonic

- No build-in example
- Try to write the code by yourself
  - Refer to the slides

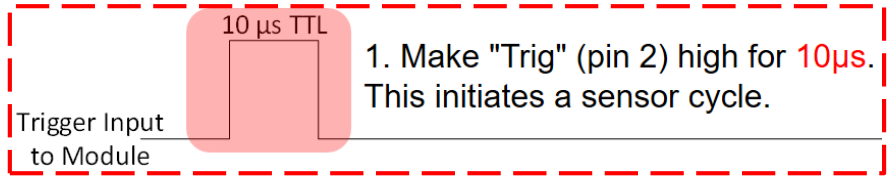
Arduino IDE



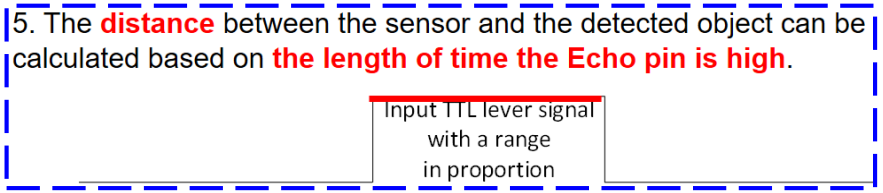
Sample Code

```
#define echoPin 7       // Echo Pin
#define trigPin 8       // Trigger Pin
#define LEDPin 13       // Onboard LED

int maximumRange = 200;          // Maximum range needed
int minimumRange = 0;            // Minimum range needed
long duration, distance;         // Duration used to calculate distance

void setup() {
 Serial.begin (9600);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 pinMode(LEDPin, OUTPUT);        // Use LED indicator (if required)
}
```

```
void loop() {
/* The following trigPin/echoPin cycle is used to determine the
 distance of the nearest object by bouncing soundwaves off of it. */
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
```




```
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
```

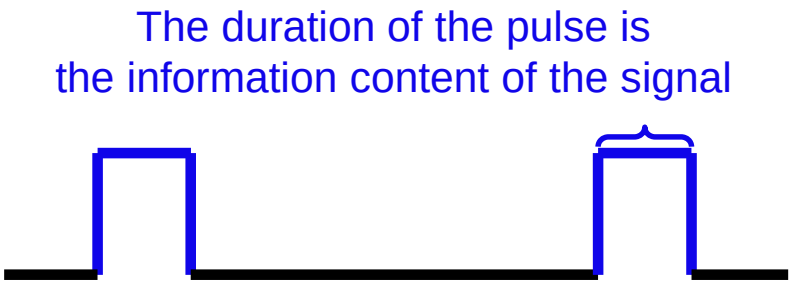



```
 duration = pulseIn(echoPin, HIGH);

 //Calculate the distance (in cm) based on the speed of sound.
 distance = duration/58.2;
```

```
if (distance >= maximumRange || distance <= minimumRange){
/* Send a negative number to computer and Turn LED ON  to indicate "out of range"
*/
        Serial.println("-1");
        digitalWrite(LEDPin, HIGH);
}
else {
/* Send the distance to the computer using Serial protocol, and
turn LED OFF to indicate successful reading. */
        Serial.println(distance);
        digitalWrite(LEDPin, LOW);  // turn LED OFF to indicate successful reading
}
//Delay 50ms before next reading.
delay(50);
}
```

Lab 4. Syntax

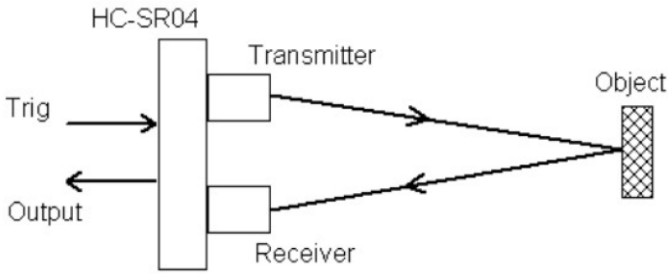
- Syntax
  - pulseIn(pin, value)
  - pulseIn(pin, value, timeout)



- Description
  - Reads a pulse (either HIGH or LOW) on a pin.
    For example, if value is HIGH, pulseIn() waits for the pin to go HIGH, starts timing, then waits for the pin to go LOW and stops timing.
    Returns the length of the pulse in microseconds.
    Gives up and returns 0 if no pulse starts within a specified time out (default: 1s).
- Example
  - duration = pulseIn(pin, HIGH); // in microseconds

Discussion 3

- Explain Ultrasonic sensor, what is trigger and echo?
- Why do we use 1/58.2 in the CODE?
  - Hint: sound speed: 340m/s, or 1cm per 29.1 us
- What is the difference between delayMicroseconds(us) and delay(ms)?
- Do some experiments. What is the max distance that your ultrasonic sensor can detect?

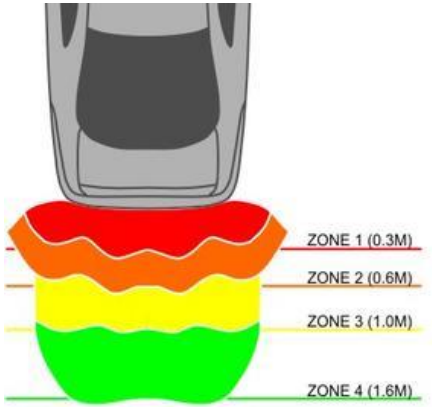


max distance??

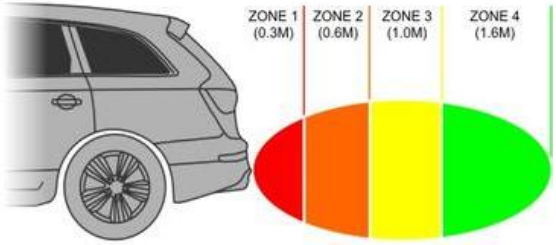
VCC, **Trigger, Echo**, GND

Quiz 3

- Design a Parking Assist System (倒車雷達)
  - use ultrasonic sensor, RGB LED and speaker
  - Divide the detecting distance into three parts: a) Safe; b) Be careful; and c) Dangerous.



  - Use the speaker to reminder the driver.
    - Safe: green + no sound (> 1m)
    - Be careful: yellow + beeping(0.3 to 1m)
    - Dangerous: red + fast beeping (<0.3 m)



  - Please keep this quiz code, we will extend this in next week.

Summary

Summary

- Practice Labs by yourself
- **Write Answers for Discussion 1 to 3**
  - Upload to e3 before next class
- **Quiz: Write code for quiz, then demonstrate to TAs**
  - Design a short song and play
  - Compose a melody by yourself
  - Design a Parking Assist System