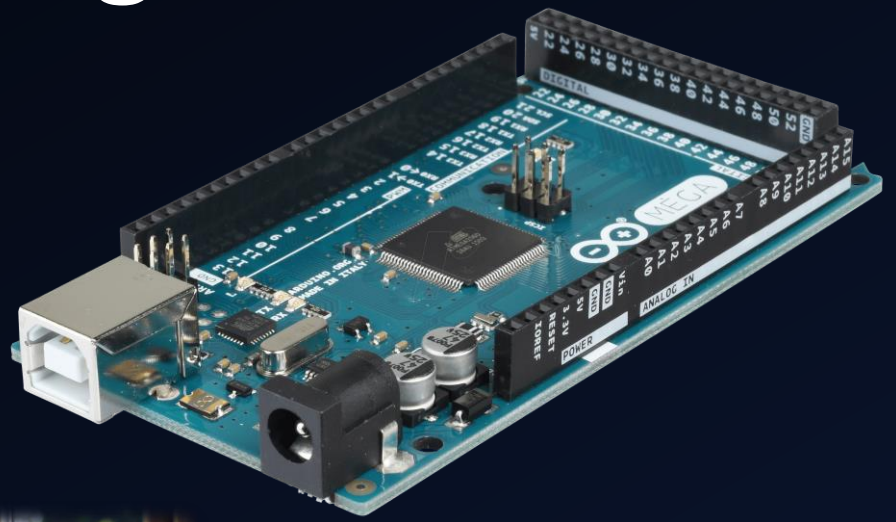
# Developing a simple burglar alarm using Arduino

## CHAPTER 1 – PIR MOTION SENSOR




# After studying this chapter, you will be able to:

1. Point out the rationale of a PIR motion sensor
2. State how a PIR motion sensor can help detecting suspicious physical behavior
3. Connect a PIR motion sensor to an Arduino Mega 2560 mainboard
4. Program the PIR motion to trigger other components

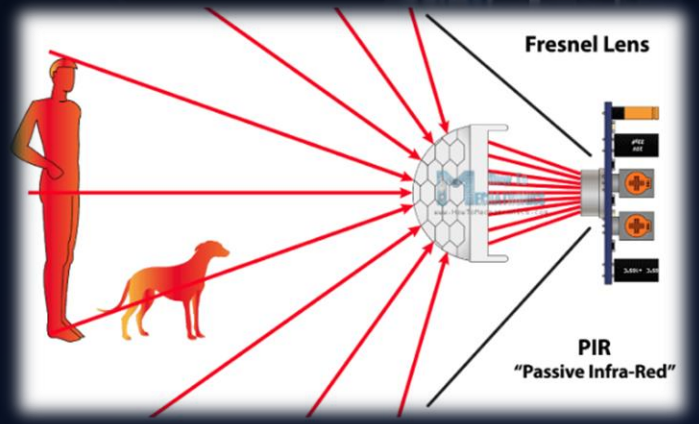
1. LED(s) module

- Supposed you are asked to set up an alarm, such that any approaching suspicions at night will trigger some sort of warnings (*e.g. flashing LEDs, activating loud siren(s)*), in what way can you detect such kind of behavior?

TIP

If you are trying to detect a suspicious person nearby, then **what will** a human or animal body **emit**?

- Since a human or animal body will emit heat energy (*in a form of **infrared radiation***) → using a **passive infra-red (PIR) sensor** may help detecting if there are any person in a certain range





A PIR sensor **will not**:

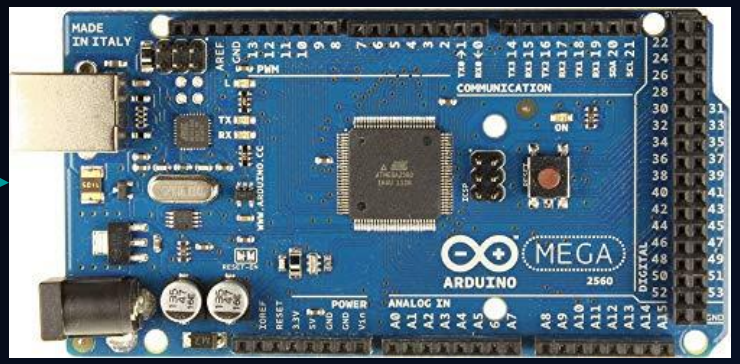
- Actively emit any kinds of energy for object detection

Instead, **it will**:

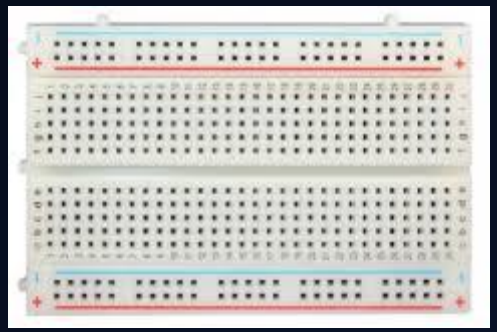
Wait for infra-red energy from objects for object detection

# Preparation

- 1x Arduino Mega 2560 Mainboard
- 1x PIR motion sensor
- 1x breadboard
- 3x female to male jumper cable
- 1x LED module



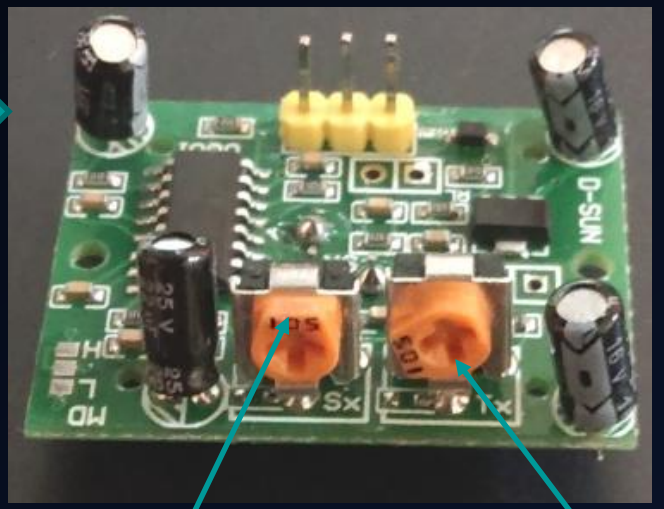




# Breakdown of PIR sensor


front

back




VCC(+5V)

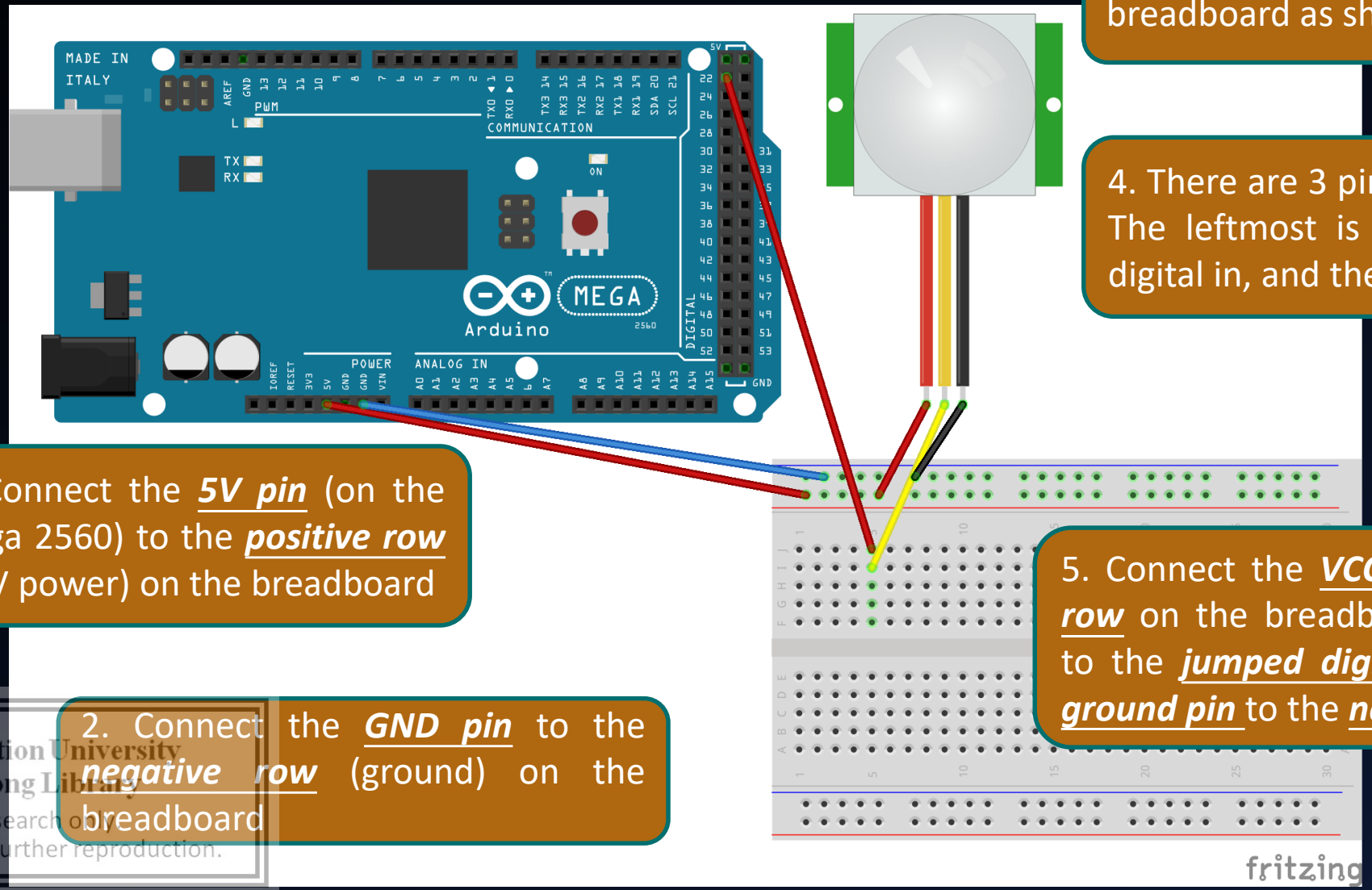
Digital OUT

GND

For adjusting sensitivity (clockwise → more sensitive)

For adjusting signal transmission delay (clockwise → more delay)

# Connection




3. Connect the ***digital pin 22*** to the breadboard as shown in the picture

4. There are 3 pins on the PIR sensor. The leftmost is VCC, the middle is digital in, and the rightmost is GND

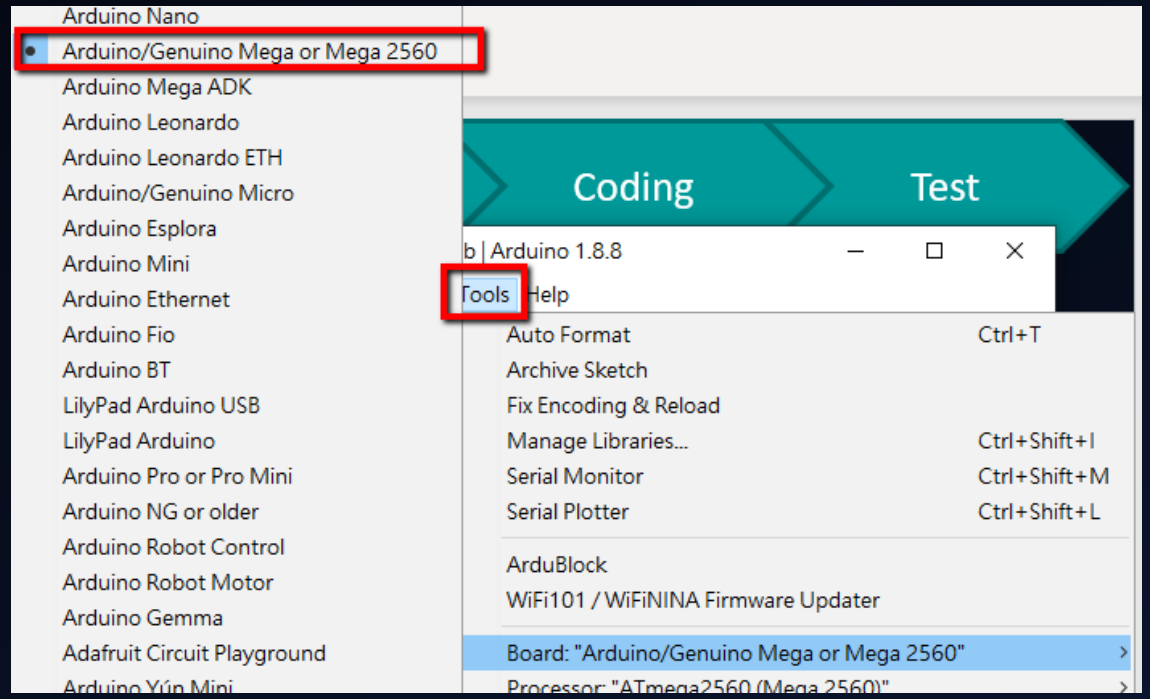
1. Connect the ***5V pin*** (on the mega 2560) to the ***positive row*** (+5V power) on the breadboard

5. Connect the ***VCC pin*** to the ***positive row*** on the breadboard, the digital pin to the ***jumped digital pin 22***, and the ***ground pin*** to the ***negative row***

2. Connect the ***GND pin*** to the ***negative row*** (ground) on the breadboard

# Practice

- Now open the Arduino IDE, and make sure you have selected the correct board type. (*In our example,* ***Arduino Mega 2560*** *should be selected*)

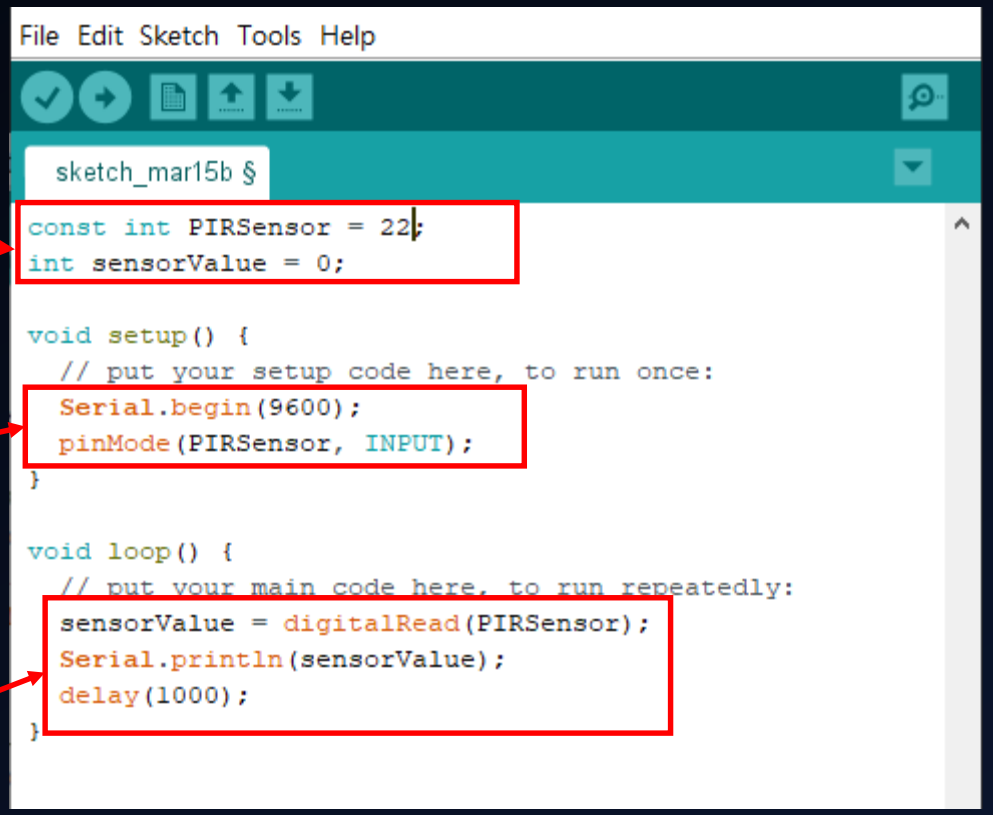


# Practice

1. Declare a constant integer for storing the ***pin no.*** for transmitting data / an integer for storing ***sensor value***

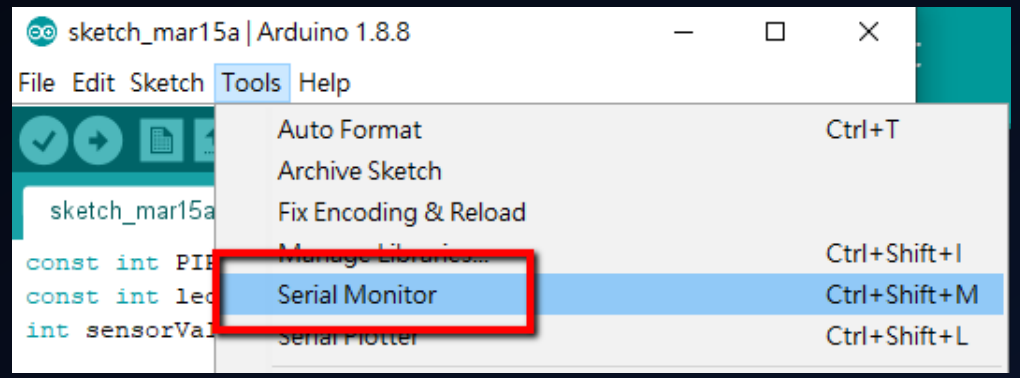
2. Set the baud rate (for serial monitoring the sensor data) & set the dedicated pin to ***input mode***

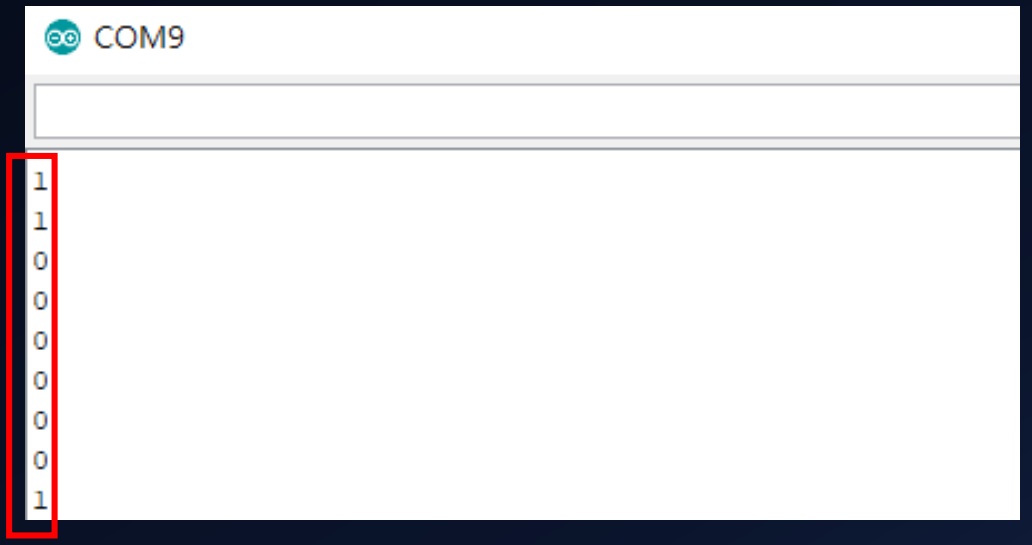
3. Read the PIR sensor data and print it to serial monitor for ***every second***



# Practice

- Now open the serial monitor
- If the cable connections and the coding are correct, you can see some similar values on the serial monitor as shown on the picture
  - "1" means the PIR sensor ***has detected*** movement from human/animal, and vice versa.



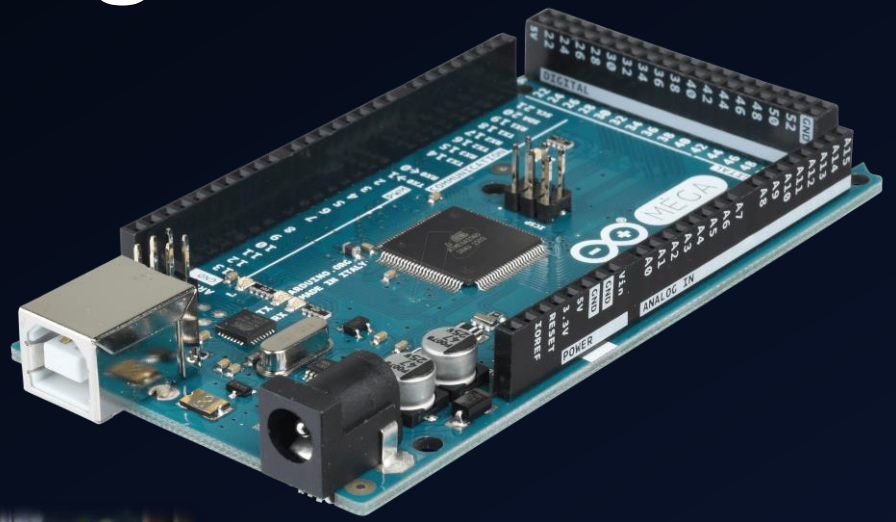
# Challenge

- If you want to use the PIR sensor to trigger an alert (i.*e. when the PIR sensor has detected motions,* ***lighten multiple LED units***), in what way should you connect and program the related modules?

- Is using only a PIR sensor can ensure all suspicious movements can be detected?

# Developing a simple burglar alarm using Arduino

## CHAPTER 2 – LED UNITS



# After studying this chapter, you will be able to:

1. Briefly describe the use of event-driven programming
2. Demonstrate the steps of connecting LED units to Arduino
3. Program a PIR sensor to trigger LED unit
4. Program multiple LED units to form advanced effects (*e.g. flickering*)

- In the last chapter, you have learned how to implement a PIR sensor.
- However, is that enough for a burglar alarm?

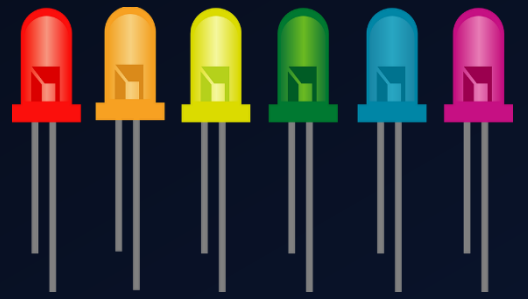
→ *A burglar alarm have to at least be able to **give warnings (signals)** upon detecting suspicious behavior*

TIP

In what way can a burglar alarm deliver warnings?

And what related modules should be used?

- A burglar alarm can deliver warnings in a variety of ways:
  - E.g. Emitting siren
  - Light blinking
  - Sending alert SMS
- In this chapter, **LED units** (*single color*) will be used as an example



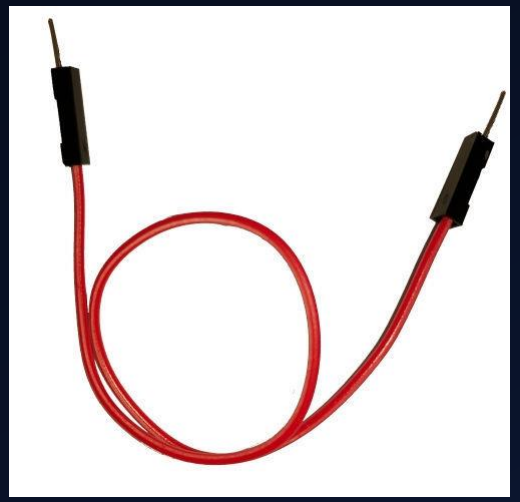
Remember!

Be **extra careful** when connecting the LED units

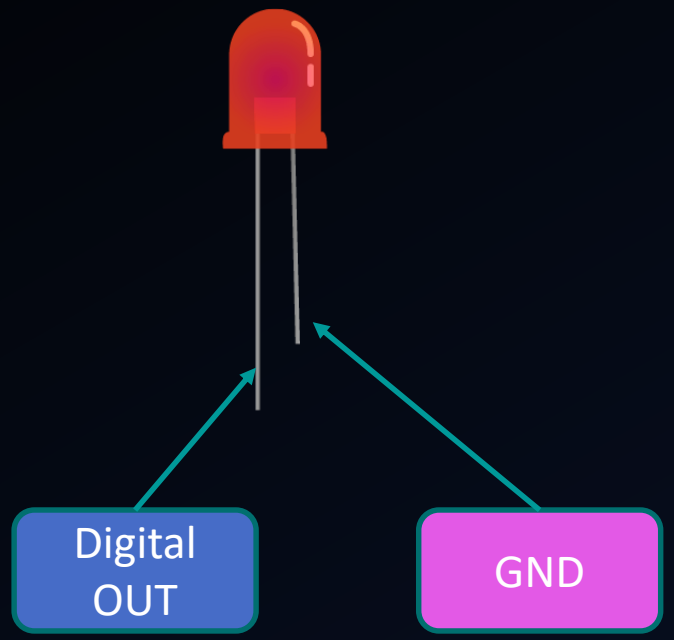
- Those pins under the LED units are rather weak, using force to handle them may cause them to break apart

# Preparation

- 1x Arduino Mega 2560 Mainboard
- 8x male to male jumper cable
- 4x LED module
- 4x 220ohm resistor
- *---outputs from chapter 1---*

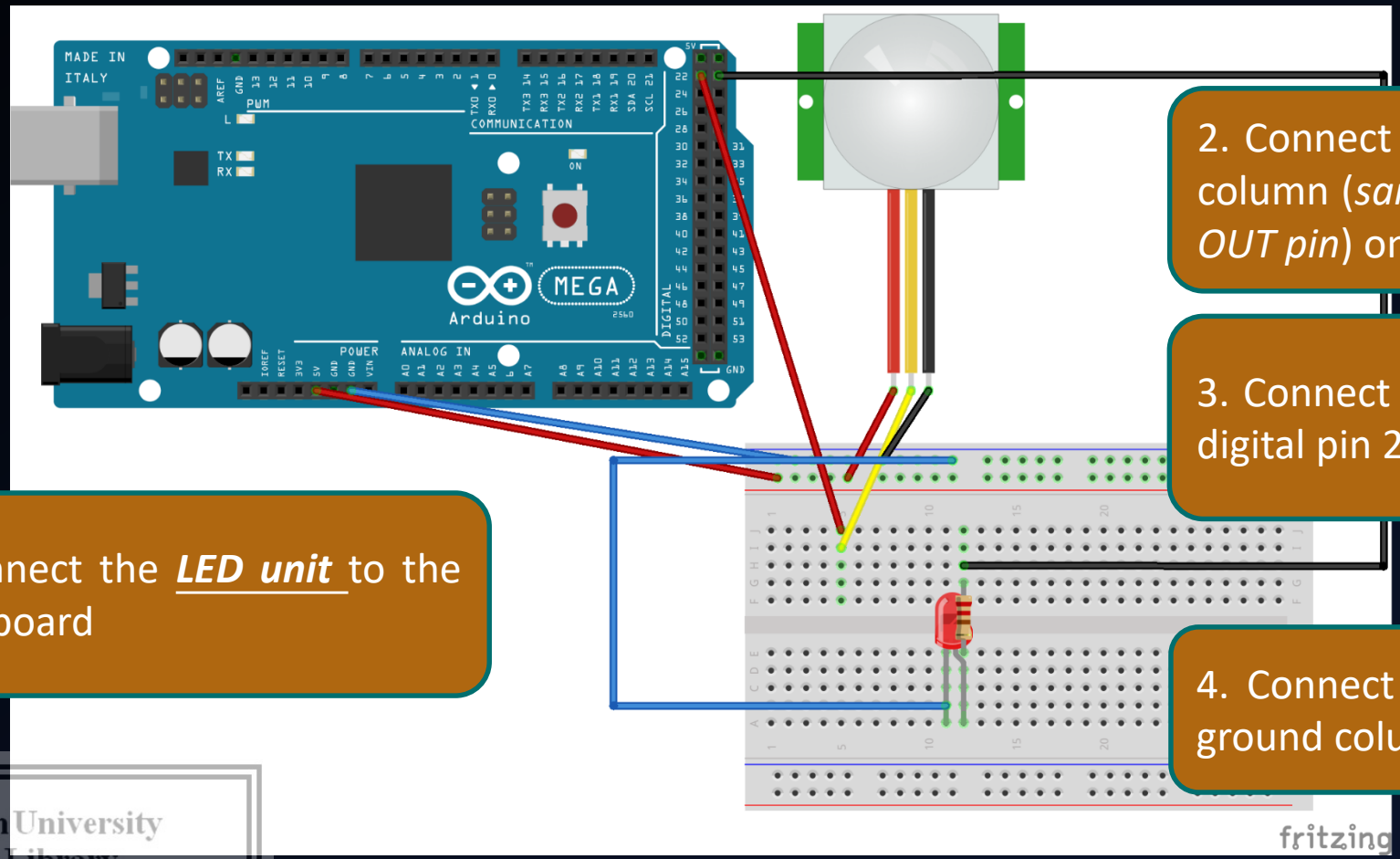



# Breakdown of LED module




Do not mix up the **digital out pin (longer one)** with the **GND pin (shorter one)**, as it may cause the LED malfunction

# Connection (based on chapter 1 output)




1. Connect the ***LED unit*** to the breadboard

2. Connect a 220ohm resistor to the column (*same column to the DIGITAL OUT pin*) on the breadboard

3. Connect the LED digital pin to the digital pin 23 on Arduino MEGA 2560

4. Connect the LED GND pin to the ground column on the breadboard

# Practice (based on chapter 1 output)

1. Declare a constant integer for storing the ***pin no.*** for LED digital pin

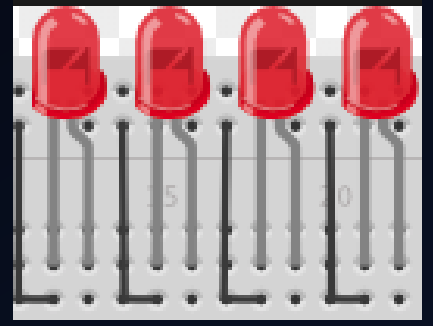
2. Set the dedicated pin to OUTPUT mode

3. If PIR sensor detected motion, illuminate the LED. Otherwise, turn it off.

sketch_mar15a §

```
const int PIRSensor = 24;
const int ledPin =  23;
int sensorValue = 0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(PIRSensor, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  sensorValue = digitalRead(PIRSensor);
  Serial.println(sensorValue);
  delay(1000);
  if (sensorValue == HIGH) {
    digitalWrite(ledPin, HIGH);
  }
  else {
    digitalWrite(ledPin, LOW);
  }

}
```

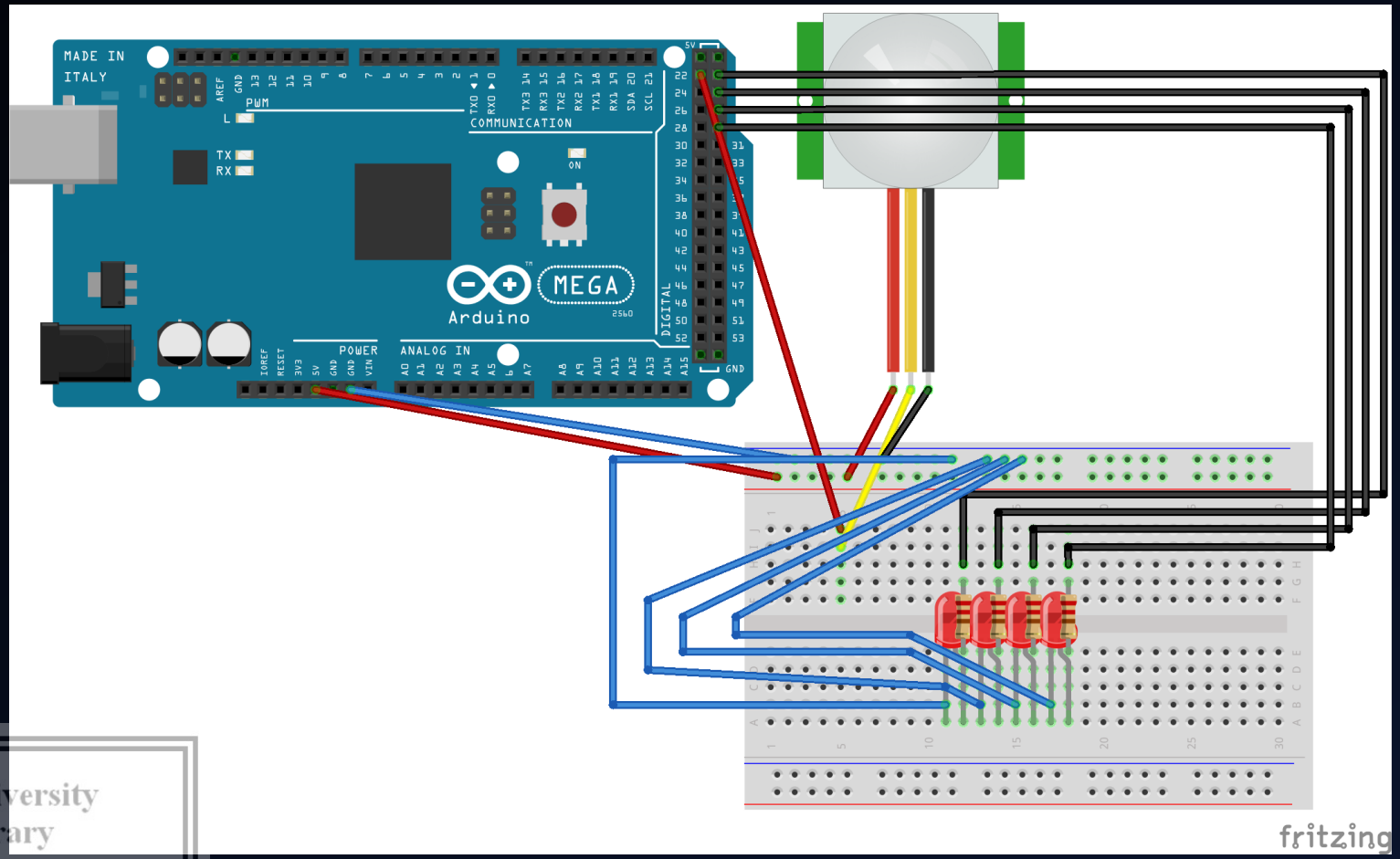
# Challenge



- How about if you want to strengthen the alert effect by using multiple LED units, and adding some effects to them ?
  - **<u>Blinking</u>**

    (*LED1 →LED2 →LED3 →LED4 →LED1 →……*)

# Challenge (TIPS 1)



# Challenge (TIPS 2)

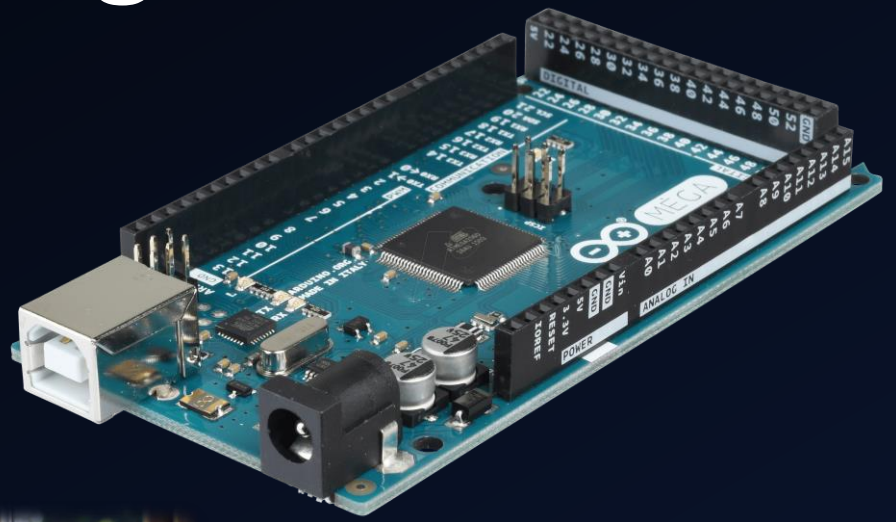
```
const int ledPin =  13;
const int ledPin2 =  12;
const int ledPin3 =  11;
const int ledPin4 =  10;
```

What does while(1) mean?

```
void ledLoop(){
  while(1){
    digitalWrite(ledPin, HIGH);
    delay(50);
    digitalWrite(ledPin, LOW);
    delay(50);
    digitalWrite(ledPin2, HIGH);
    delay(50);
    digitalWrite(ledPin2, LOW);
    delay(50);
    digitalWrite(ledPin3, HIGH);
    delay(50);
    digitalWrite(ledPin3, LOW);
    delay(50);
    digitalWrite(ledPin4, HIGH);
    delay(50);
    digitalWrite(ledPin4, LOW);
  }
```

# Developing a simple burglar alarm using Arduino

## CHAPTER 3 – ULTRASONIC SENSOR

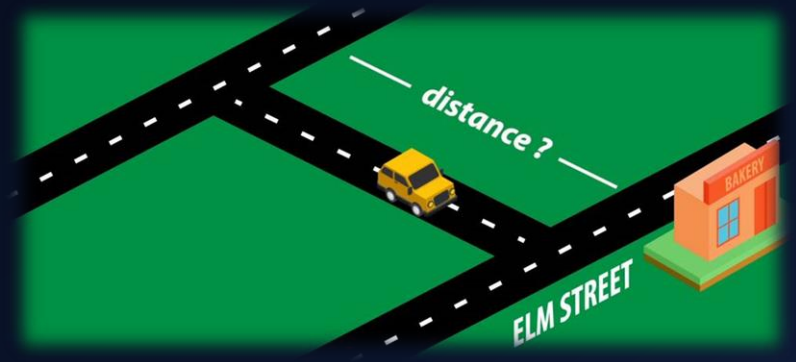



# After studying this chapter, you will be able to:

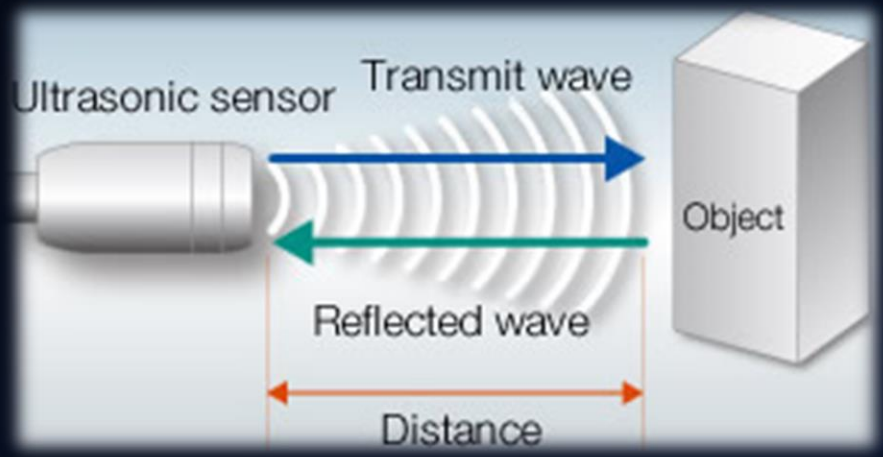
1. Describe the drawbacks of detecting motions using a single PIR sensor
2. Implement a ultrasonic sensor on Arduino
3. Combine the use of PIR sensor and ultrasonic sensor for motion detection
4. Use an ultrasonic sensor to trigger LED units

- In chapter 1, you have already learn how to detect motions using PIR sensor.
- However, do you think using a single PIR sensor can provide accurate result in terms of movement detection?

→ *What if I want to know the exact distance between an object and the sensor?*



- An ultrasonic sensor can be used to measure the ***distance*** between an object and the sensor




1. The sensor will first emit ultrasound signal
2. The signal will reflect when hitting an object
3. The time difference between the transmit wave and the reflected wave will be used to calculate the distance

TIP

*The effective detection distance in 2cm - 400cm. For the detection of object out of this range, another sensor will be used (introduced in later chapter)*

- We already know using ultrasonic sensor can detect distance, so why we have to ***combine the use of PIR sensor***, is that redundant?


Is that good if the ultrasonic sensor are activated all the time for motion detection?

- Using a PIR sensor to detect if an object is in range → in range → activate ultrasound sensor to detect distance → in distance range → trigger alert → (*reduce battery consumption*)

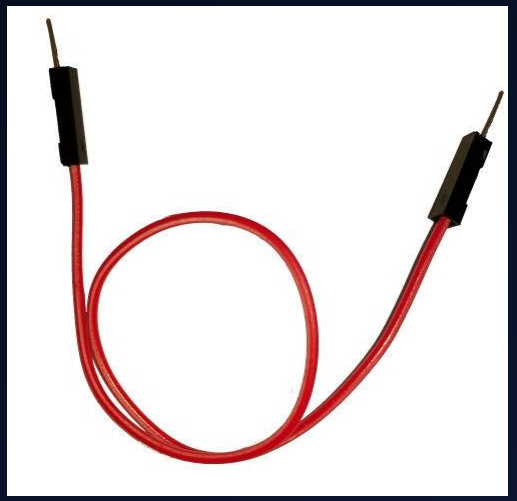

The ultrasonic sensor have to use at least **5V power**

- The 3.3V pin on Arduino will not work.

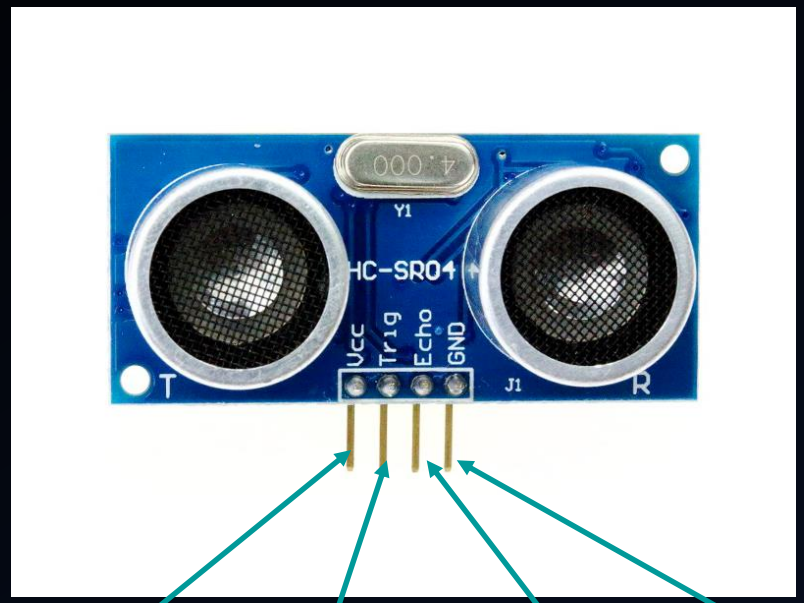
# Preparation

- 1x Arduino Mega 2560 Mainboard
- 4x male to male jumper cable
- *---outputs from chapter 1 & 2---*





# Breakdown of ultrasound sensor




VCC (+5V)

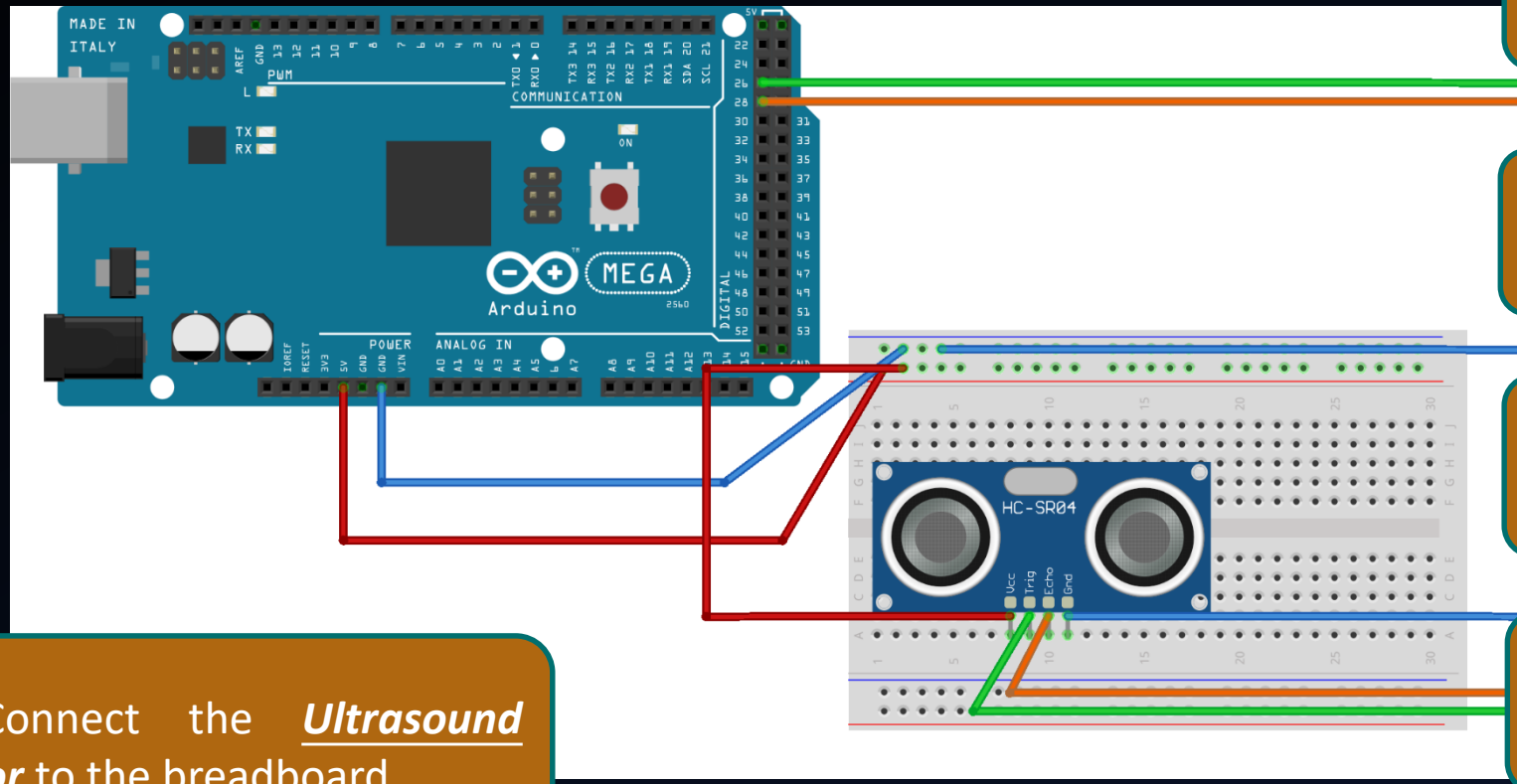
Trig (signal transmit)

Echo (receive reflect signal)

GND

Please remember ***which pin numbers*** (on Arduino) you are mapping to ***Trig pin*** & ***Echo pin***. Otherwise, you will get incorrect distance calculation result

# Connection




1. Connect the ***Ultrasound sensor*** to the breadboard

2. Connect the GND pin to the negative row of the breadboard

3. Connect the +5V pin to the positive row of the breadboard

4. Connect the VCC pin on the sensor to the positive row of the breadboard

5. Connect the Trig pin on the sensor to the digital pin 26 on the Arduino MEGA 2560

6. Connect the Echo pin on the sensor to the digital pin 28 on the Arduino MEGA 2560

7. Connect the GND pin on the sensor to the negative row on the breadboard

# Practice

1. Declare variables to store the pin number and distance information

2. Set Trig pin to output mode, and echo pin to input mode

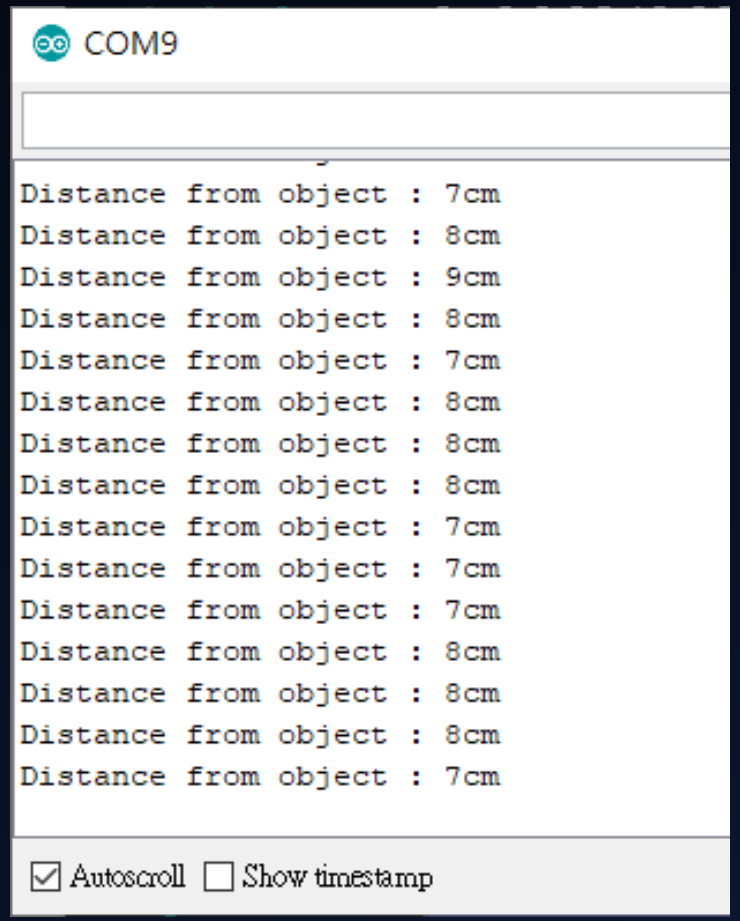
3. Tell sensor to transmit ultrasound signal for 10ms

4. Calculate the time of receiving the bounce back signal, and calculate the distance

```
sketch_mar16a

int trigPin = 26;
int echoPin = 28;
long duration, cm;

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);

  cm = (duration/2) / 29.1;

  Serial.print("Distance from object : ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(250);
}
```

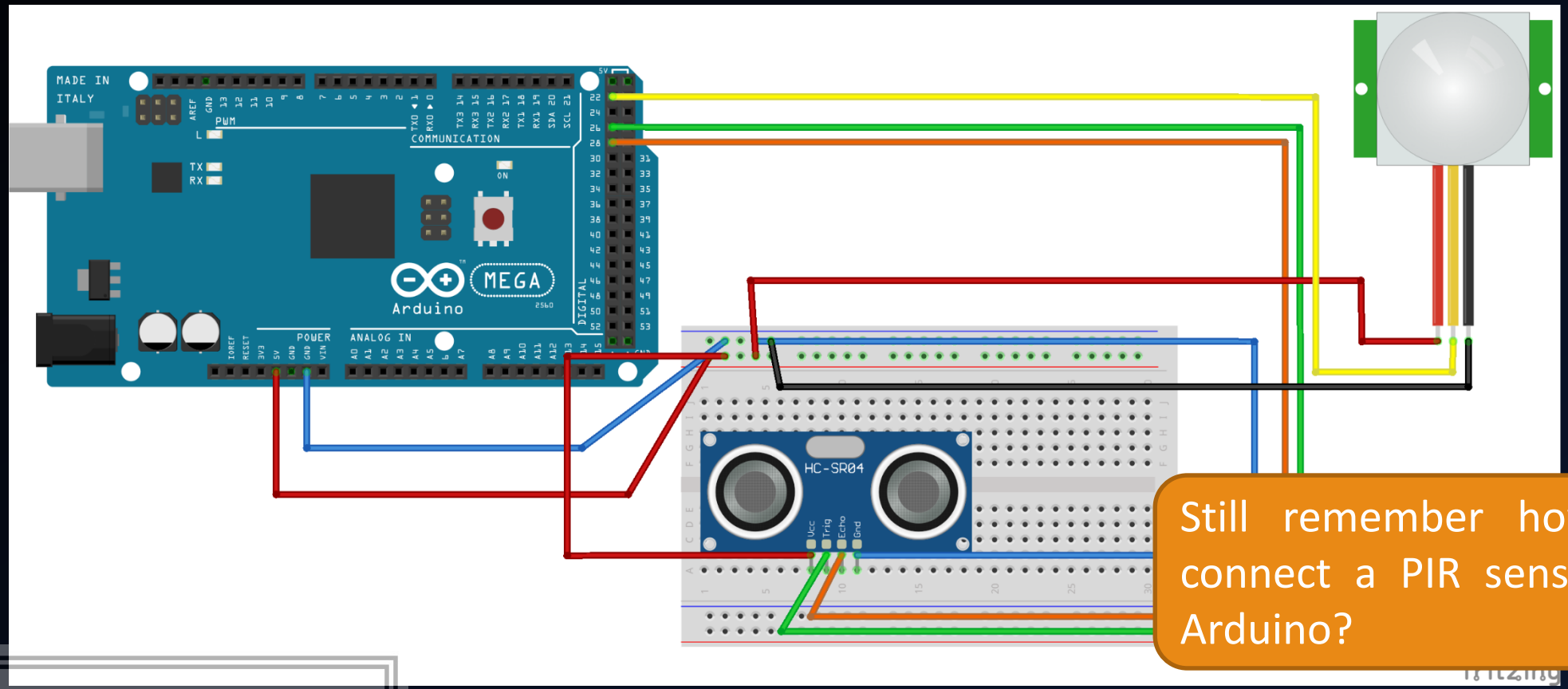
# Practice

If successful, you should see some similar data in the serial monitor




*How about combining the use of PIR sensor?*

# Practice (2)




Still remember how to connect a PIR sensor to Arduino?

# Practice (2 – based on chapter 1 & 2)

Remember the program we developed in chapter 1 & 2? Now we need to modify **that program** a bit to make to PIR sensor will work together with ultrasonic sensor

1. We first copy the code we written in the loop() part before into a new method → activateUltrasound()

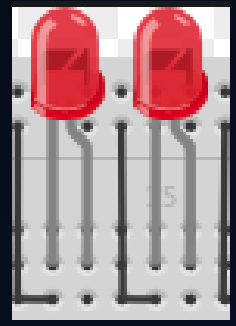
2. Now put the method under the **if statement**, so that the ultrasound sensor will be activated **if the PIR sensor has detected motion in its detection range**. Otherwise, **turn off the ultrasound sensor**.

```
void activateUltrasound(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);

  cm = (duration/2) / 29.1;

  Serial.print("Distance from object : ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(250);
}
```

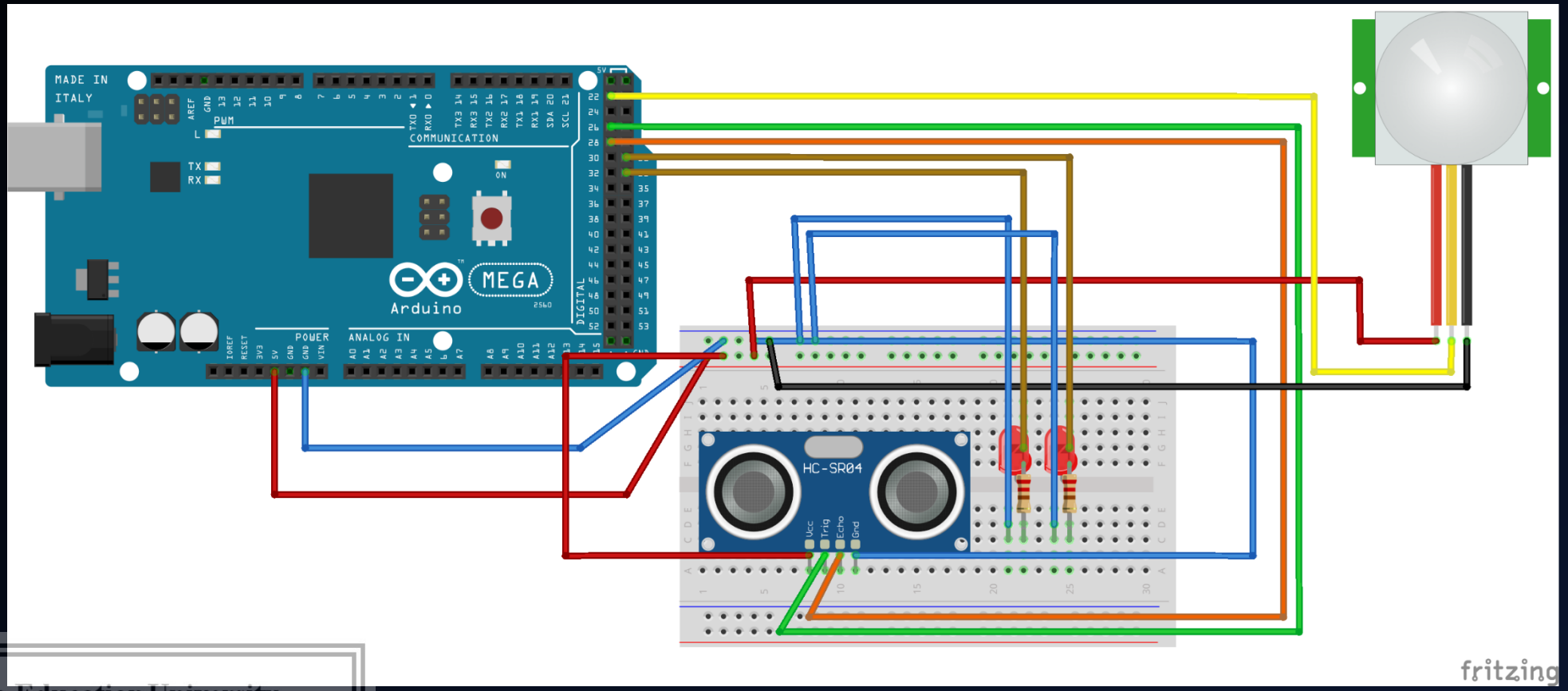
```
void loop() {
  // put your main code here, to run repeatedly:
  sensorValue = digitalRead(PIRSensor);
  Serial.println(sensorValue);
  delay(250);
  if (sensorValue == HIGH) {

    activateUltrasound();
  }
  else {

    digitalWrite(trigPin, LOW);
  }

}
```

# Challenge

- How about if you want to trigger an alert by making two LED units to blink interchangeability, when the ultrasound sensor has detected an object **within 60 cm**?



# Challenge (TIPS 1)



# Challenge (TIPS 2)

```
const int ledPin =  13;
const int ledPin2 =  12;
```

Still remember what to do if you want to implement LED units?

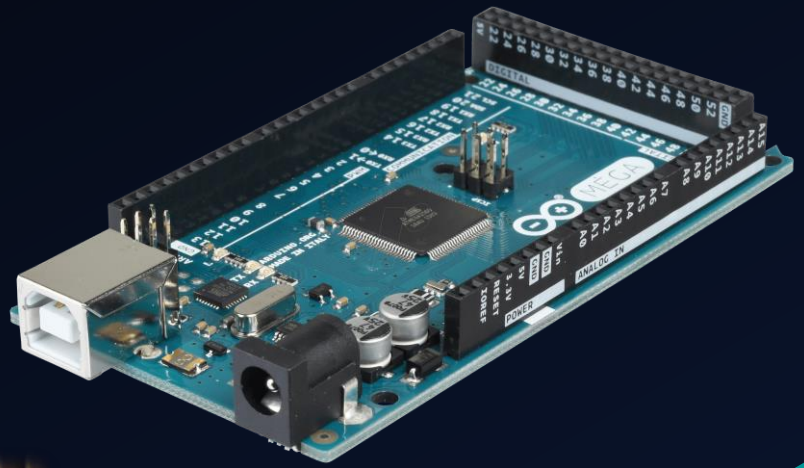
By reviewing Chapter 2, do you still remember how to make an multiple LEDs to blink interchangeability?

```
void ledBlink(){
   while(1){
    digitalWrite(ledPin, HIGH);
    delay(50);
    digitalWrite(ledPin, LOW);
    delay(50);
    digitalWrite(ledPin2, HIGH);
    delay(50);
    digitalWrite(ledPin2, LOW);
    delay(50);
}
```

If you want to trigger an alert by **making LEDs to blink**, only when an object is detected **within 60cm**, how should you construct **the if-then-else statement**?

# Developing a simple burglar alarm using Arduino

## CHAPTER 4 – ACTIVE BUZZER



# After studying this chapter, you will be able to:

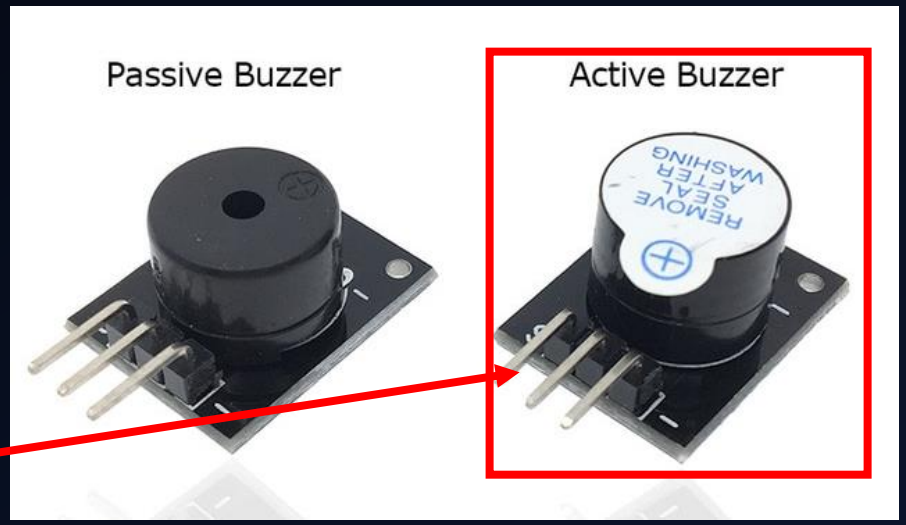
1. Point out the difference between active and passive buzzer
2. Implement a active buzzer on Arduino
3. Use a ultrasonic sensor to trigger an active buzzer

- In chapter 3, you have already learn how to trigger an alert by making LEDs to blink using ultrasonic sensor.
- However, do you think only using blinking LEDs can scare off suspicious person/possible intrusion?

→ *In what way can I strengthen the effect of alerts delivered by the burglar alarm?*

- A buzzer can be used to emit sound at specific pitch
- There are 2 types of buzzer
  - **Active buzzer** → Can only emit sound at **fixed pitch**
  - **Passive buzzer** → Can emit sound at **varied pitch**

We will focus on active buzzer this lesson

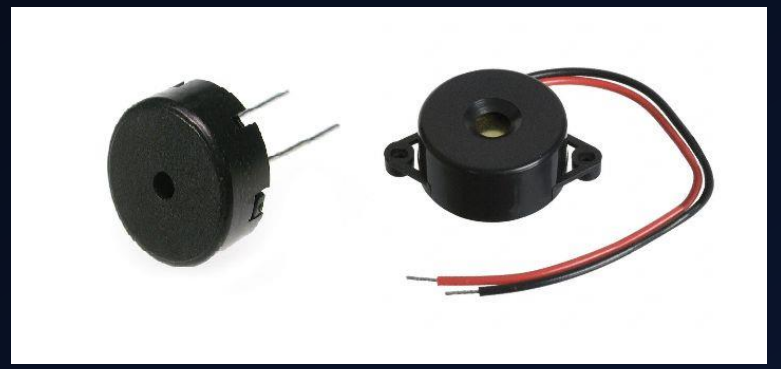



TIP

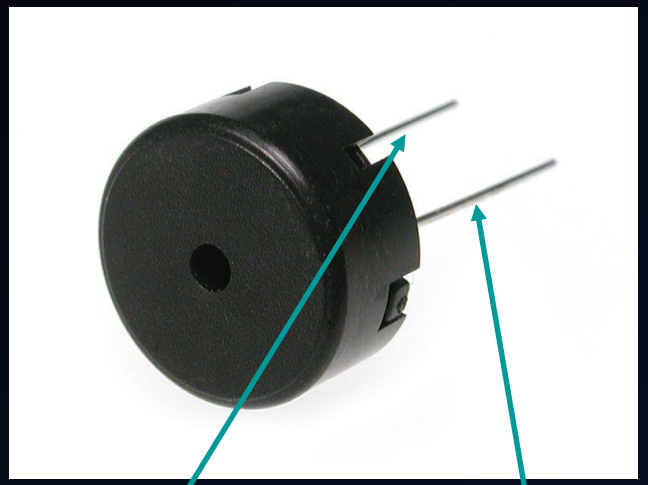
*For active buzzer, since it can only emit fixed pitch sound, you **need not input the pulse value** for it.*

# Preparation

- 1x Arduino Mega 2560 Mainboard
- 1x active buzzer
- 2x male to female jumper cable
- *---outputs from chapter 3---*

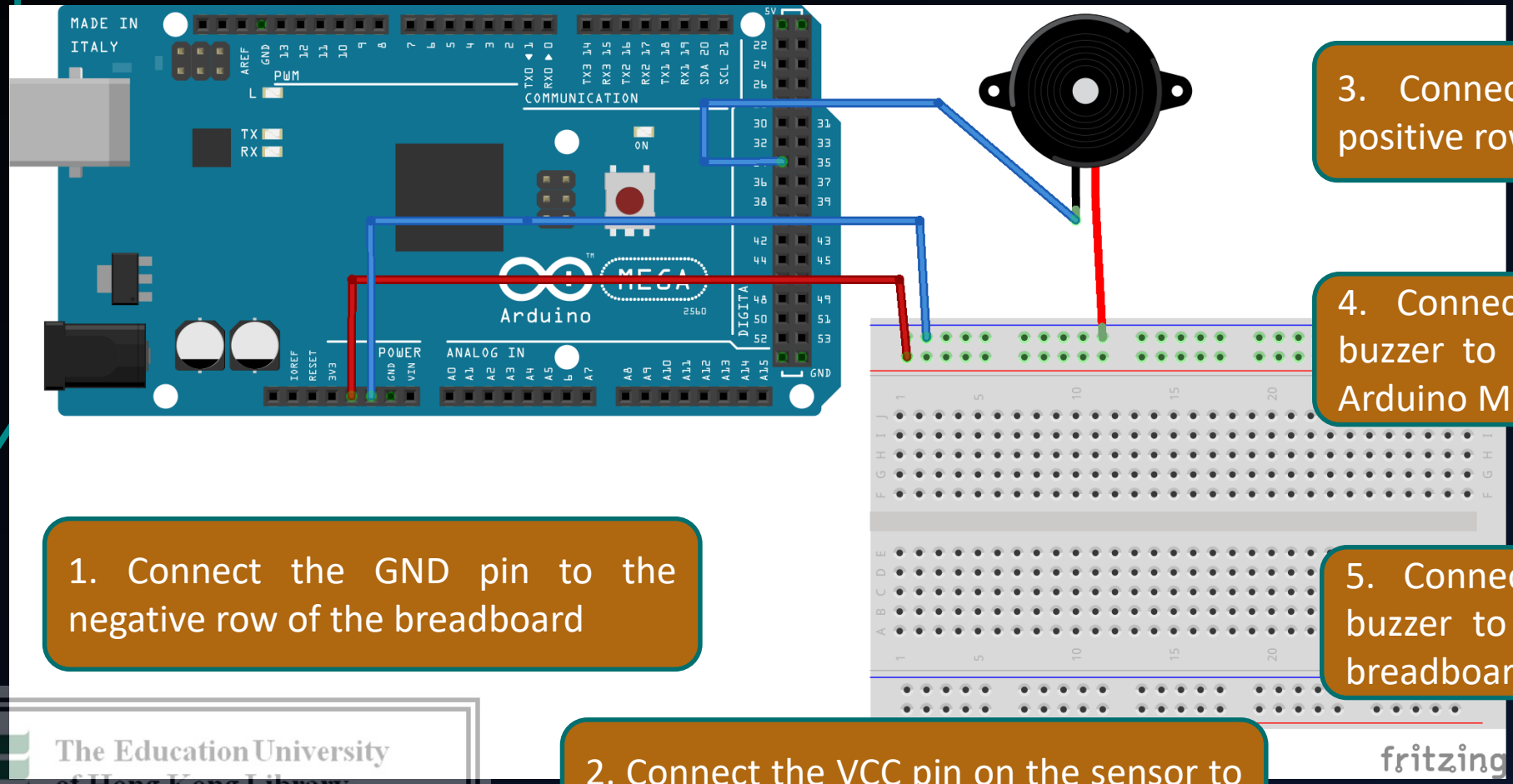



# Breakdown of active buzzer




Remember! *If you want to generate sounds **at varied pitch**, consider using a **passive buzzer***

# Connection (Active buzzer)




1. Connect the GND pin to the negative row of the breadboard

2. Connect the VCC pin on the sensor to the positive row of the breadboard

3. Connect the +5V pin to the positive row of the breadboard

4. Connect the data pin on the buzzer to the digital pin 34 on the Arduino MEGA 2560

5. Connect the GND pin on the buzzer to the negative row of the breadboard

# Practice 1 (Active buzzer)

1. Declare a variable to store the pin number for the buzzer

2. Set the buzzer for emitting a tone for 50ms

3. Tell the buzzer to emit another tone for 50ms

4. If successful, you should hear a tone in a format like "***beep↑ -beep↓ -beep↑ -beep↓***" interchangeability.

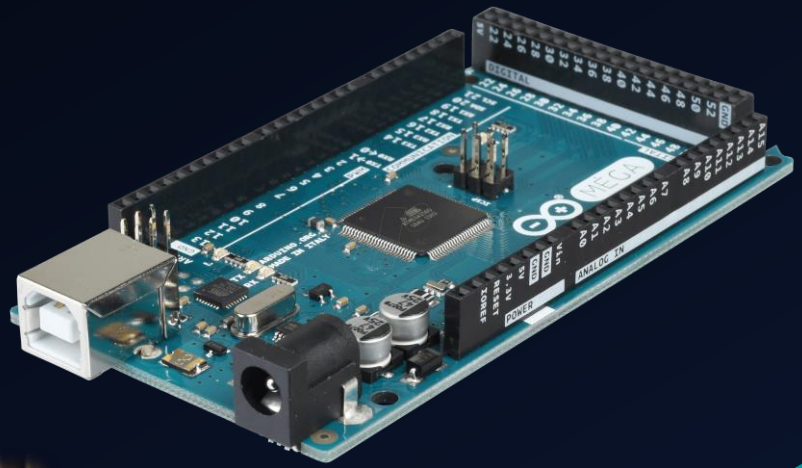
```
sketch_mar17a §
const int buzzerPin = 34;

void setup() {
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  // pulse the buzzer on for a short time
  for (int x = 0; x < 50; x++){
  digitalWrite(buzzerPin, HIGH);
  delay(1);
  digitalWrite(buzzerPin, LOW);
  delay(1);
  }
 for (int y = 0; y < 50; y++){
  digitalWrite(buzzerPin, HIGH);
  delay(2);
  digitalWrite(buzzerPin, LOW);
  delay(2);
 }
}
```

# Challenge

- How about if you want to trigger an alert by generating a tone, when the ultrasound sensor has detected an object **within 50 cm**?
  - And, how to make the siren sound **5 seconds for every detected object** within the specified criteria?

# Developing a simple burglar alarm using Arduino

## CHAPTER 5 – PASSIVE BUZZER

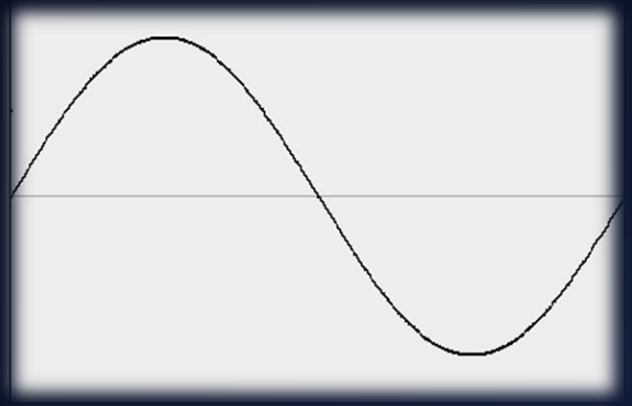


# After studying this chapter, you will be able to:

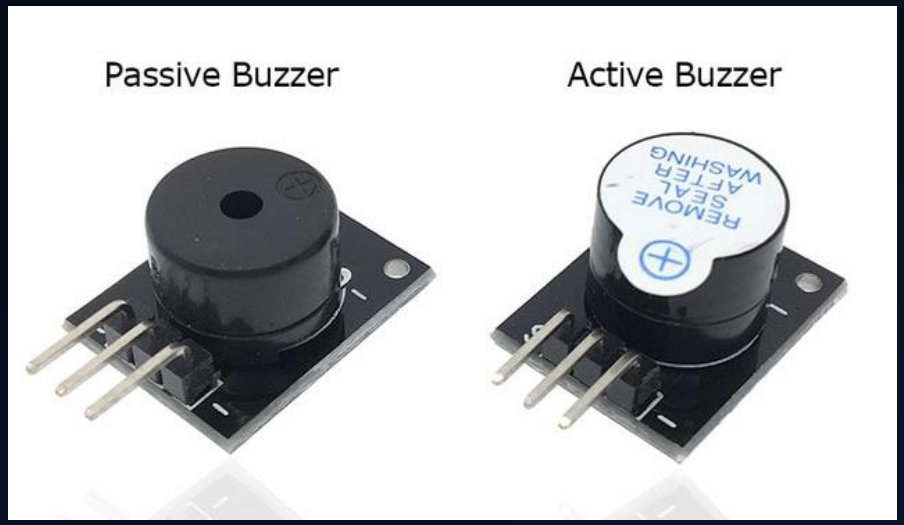
1. Point out the difference between active and passive buzzer
2. Implement a passive buzzer on Arduino
3. Program a passive buzzer to emit a sound with varied tones
4. Use a ultrasonic sensor to trigger an passive buzzer

- In chapter 4, you have already learn how to make a sound at **fixed pitch** using active buzzer.
- However, sometimes you may want to make the siren effect **more prominent**, in which an active buzzer may not fulfill your requirement.

→ *What type of buzzer should I adopt to emit a sound with* ***varied pitches****?*



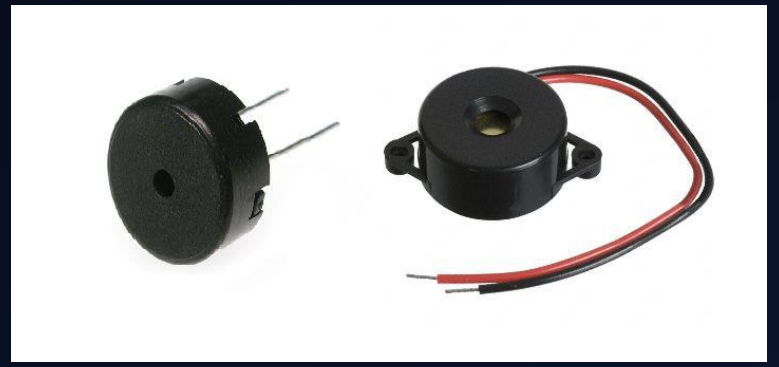
- Still remember the characteristics of the type of buzzer we have implemented in chapter 4?
  - **Active buzzer** → Can only emit sound at **fixed pitch**
- In this chapter, we will focus on **passive buzzer**
  - **Passive buzzer** → Can emit sound at **varied pitch**



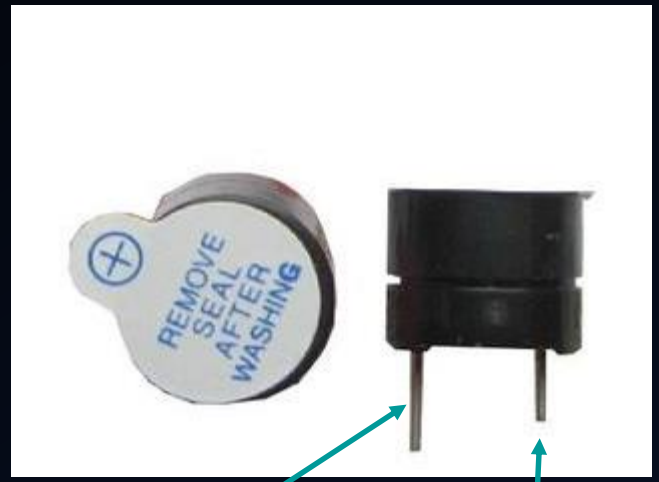

TIP *For passive buzzer, you have to **input pulse value manually**.*

# Preparation

- 1x Arduino Mega 2560 Mainboard
- 1x passive buzzer
- 2x male to female jumper cable
- *---outputs from chapter 3 & 4---*

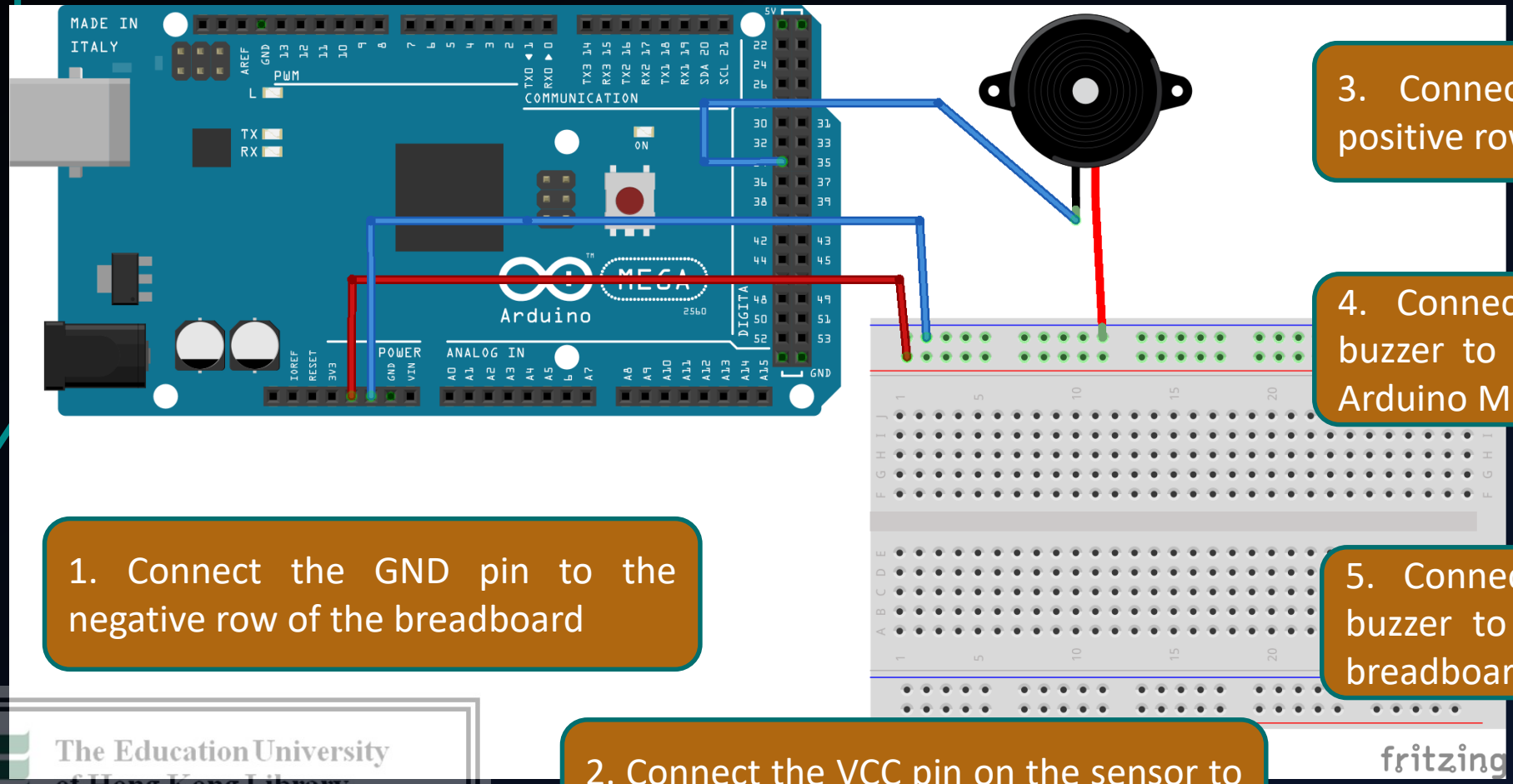
# Breakdown of active & passive buzzers




Remember!

*Unlike the active buzzer, the digital pin of a passive buzzer is **longer than** its GND pin, don't mix it up.*

# Connection (Passive buzzer)




1. Connect the GND pin to the negative row of the breadboard

2. Connect the VCC pin on the sensor to the positive row of the breadboard

3. Connect the +5V pin to the positive row of the breadboard

4. Connect the data pin on the buzzer to the digital pin 34 on the Arduino MEGA 2560

5. Connect the GND pin on the buzzer to the negative row of the breadboard

# Practice 1 (Passive buzzer)

*How about if I want to generate a sound with varied tone and frequency?* → ***passive buzzer***

Same connection as active buzzer, only need to **replace the buzzer unit to a passive one**

1. Create a method called intrusionSiren()

2. A sound will increase in hertz from **100hz to 1500hz** gradually, each hertz will last for 10ms

3. A sound will decrease in hertz from **1500hz to 100hz** gradually, each hertz will last for 10ms

4. Upon successful, you will hear a sound with increasing tone, and a sound with decreasing tone intersectionally

sketch_mar17c

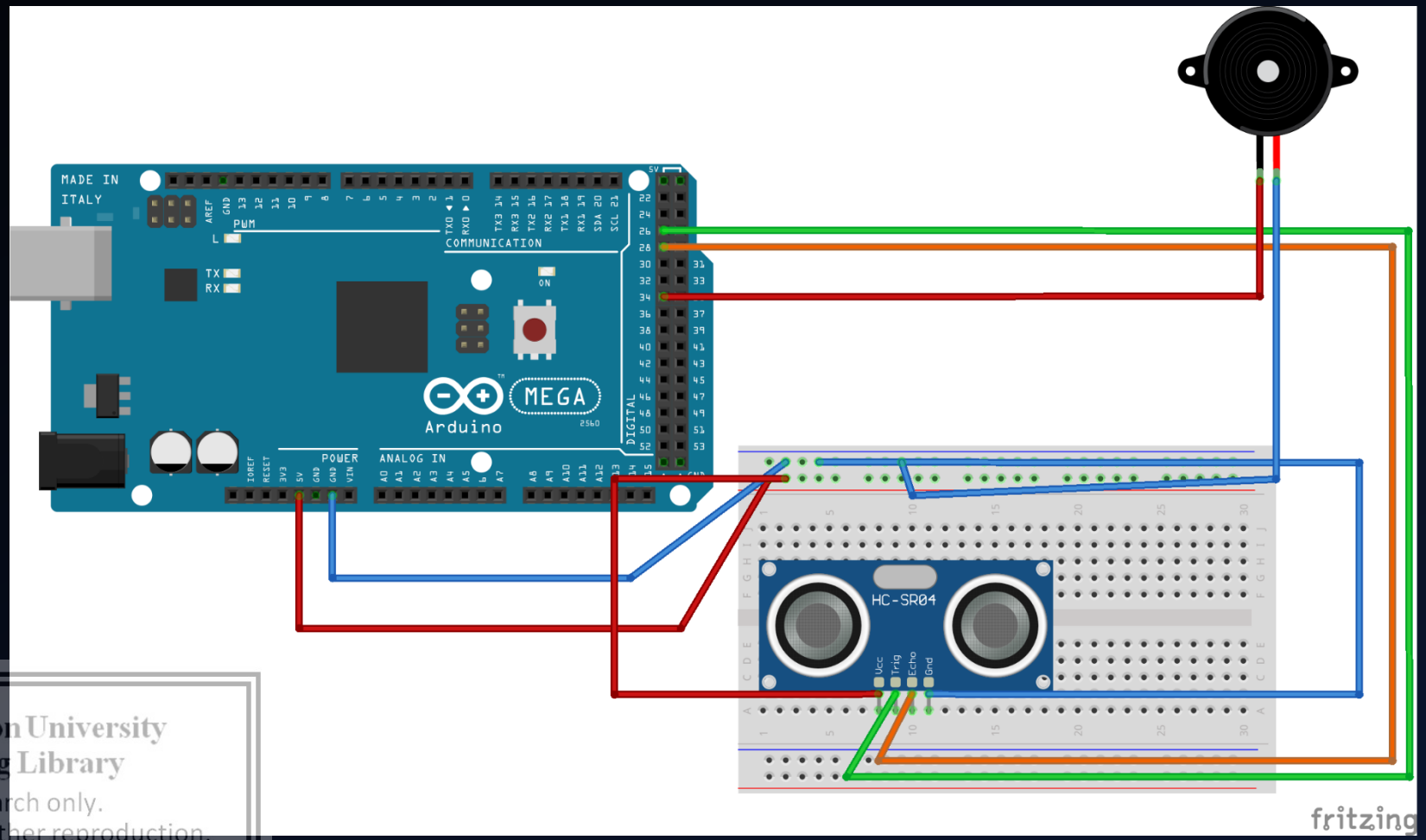
```
int buzzerPin=30;

void setup() {
 pinMode(buzzerPin, OUTPUT);
 }

void loop() {
  intrusionSiren(buzzerPin);
  }

void intrusionSiren(int pin) {
  for (int x=100; x<1500; x++) {
    tone(pin, x, 10);
    delay(1);
    }
  for (int y=1500; y>100; y--) {
    tone(pin, y, 10);
    delay(1);
    }
}
```

# Challenge

- How about if you want to trigger an alert by generating a tone you have done **in practice 1**, when the ultrasound sensor has detected an object **within 50 cm**?
  - And, how to make the siren sound 5 seconds for every detected object within the specified criteria?

# Challenge (TIPS 1)

Still remember how to connect an ultrasound sensor to Arduino? (you may review chapter 3)



# Challenge (TIPS 2)

Remember the ultrasound program we developed in **chapter 3**? We now need to use it again, and combine it with the program **in practice 2** (with a bit of modification)

1. Declare an unsigned 32-bit integer variable for storing the duration of the siren sound in timestamp format (***2000L = 2 seconds***)

2. Declare a method ***intrusionSiren()*** as in practice 2. Then declare a condition so that if an object is within 50cm detection range, the ***intrusionSiren()*** method will be called to make the buzzer sound for 5 seconds

*** To be more specific, this line means to start the countdown once the siren start making sound, and it will compare with the time stored in the ***"period" variable.*** *If tStart exceeds the pre-defined time, the buzzer will stop making sound****

```
empt
int trigPin = 26;
int echoPin = 28;
long duration, cm;
int buzzerPin=30;
uint32_t period = 2000L;

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(buzzerPin, OUTPUT);
}

void loop()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = (duration/2) / 29.1;
  while (cm <= 50){
    intrusionSiren(buzzerPin);
    resetUltraSound();
  }
}

void intrusionSiren(int pin) {
  for( uint32_t tStart = millis();  (millis()-tStart) < period;){
    for (int x=2000; x<3500; x++) {
      tone(pin, x, 10);
    }
    for (int y=3500; y>2000; y--) {
      tone(pin, y, 10);
    }
  }
}

void resetUltraSound(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = (duration/2) / 29.1;
```

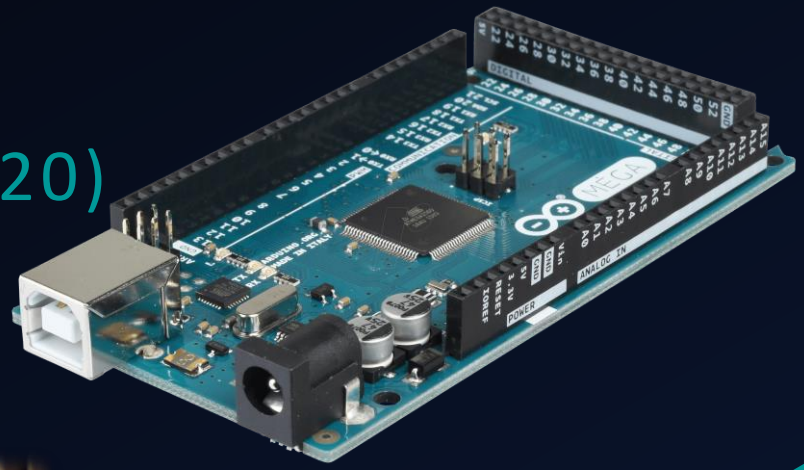
3. We need to construct a method to **reset the ultrasound sensor** once the buzzer has made a sound. **Otherwise, the system will stick to the last object distance value**, in which will make the buzzer fail to stop properly.

# Challenge

- Minnie claims using an ultrasound sensor together with a PIR sensor can almost cover all the possible intrusion scenarios, do you agree with her claim?

# Developing a simple burglar alarm using Arduino

## CHAPTER 6 – VIBRATION SENSOR (SW420)



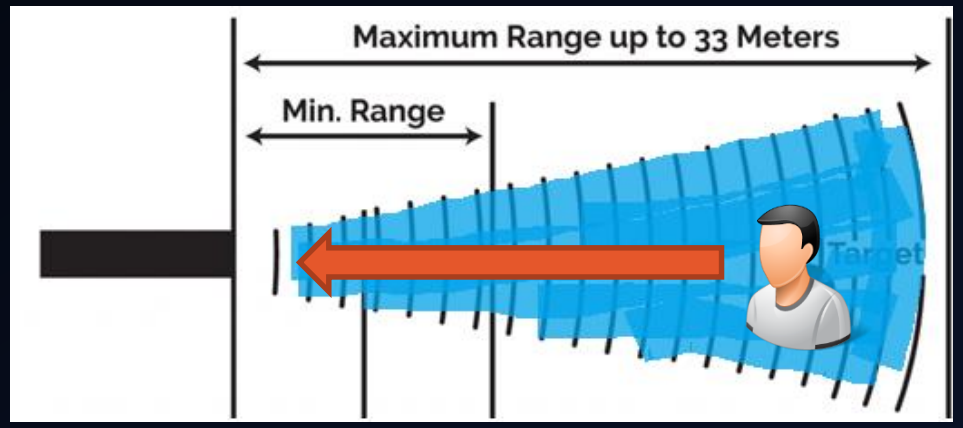
# After studying this chapter, you will be able to:

1. Point out the limitation of ultrasonic sensor
2. State & explain how an vibration sensor can improve the detection of irregular movements
3. Implement a vibration sensor on Arduino
4. Use a vibration sensor to trigger alerts (e.g. buzzer, LEDs blinking)

- In chapter 3, you have already learnt an ultrasonic sensor can tell the **object distance**, which cannot be done by PIR sensor accurately, thus enhancing the object detection.
- However, does that mean **all the suspicious motions** (possible intrusion scenarios) can be covered by them?

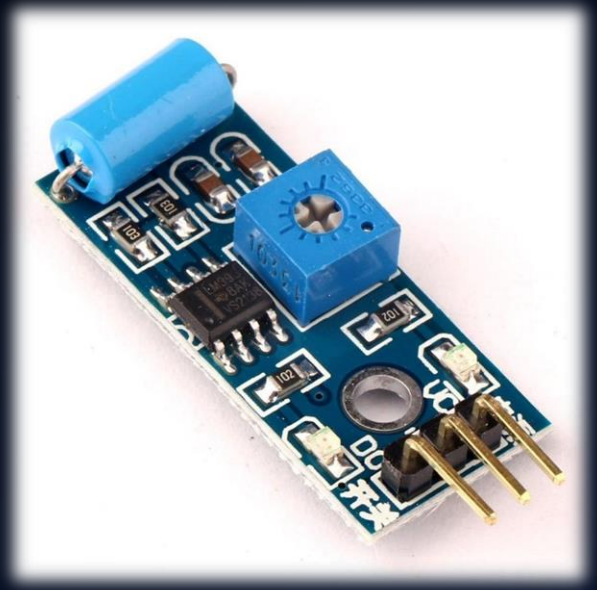
→ *What are the possible intrusion scenarios that may not be able to covered by either PIR sensor or ultrasonic sensor (or combining both of them)?*

Let's say there is a suspicious person (or movement) approaching, as it is **inside the detection area of the sensor** (blue area), he can be detected by the sensor




How about if a person is walking towards the sensor, but in the **green area**? As he is out of the detection area of the sensor, it is possible he **may not be detected** by the sensor.

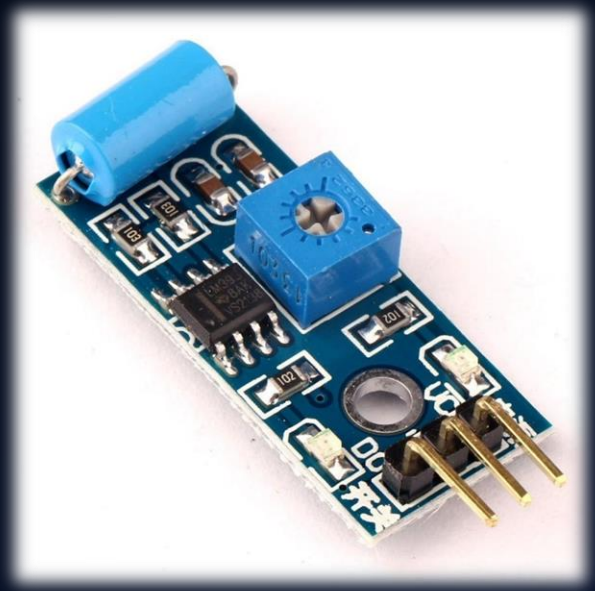
- A **vibration sensor** may help addressing to that issue
- In case a thief can bypass the PIR sensor & ultrasonic sensor, if he/she is trying to break into the room, **the vibration caused** can still be detected by the vibration sensor.
- Vibration sensors are very common in the anti-theft alarms for cars, luxury paintings exhibitions, etc.



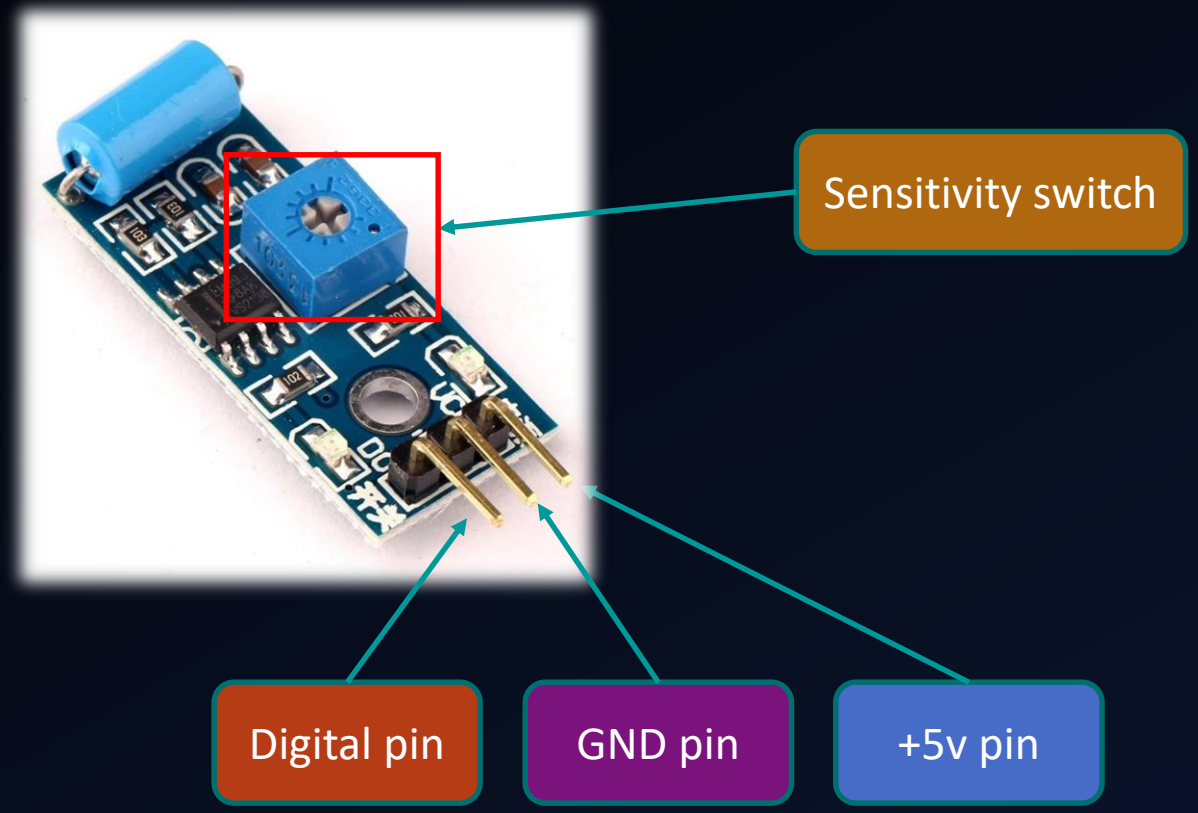
TIP *For vibration sensors on Arduino, we may see a grey switch on its back. By **turning it clockwise** using a screwdriver, you can increase the sensitivity of vibration detection*

# Preparation

- 1x Arduino Mega 2560 Mainboard
- 1x SW420 vibration sensor
- 3x male to female jumper cable
- *---outputs from chapter 3 & 4---*

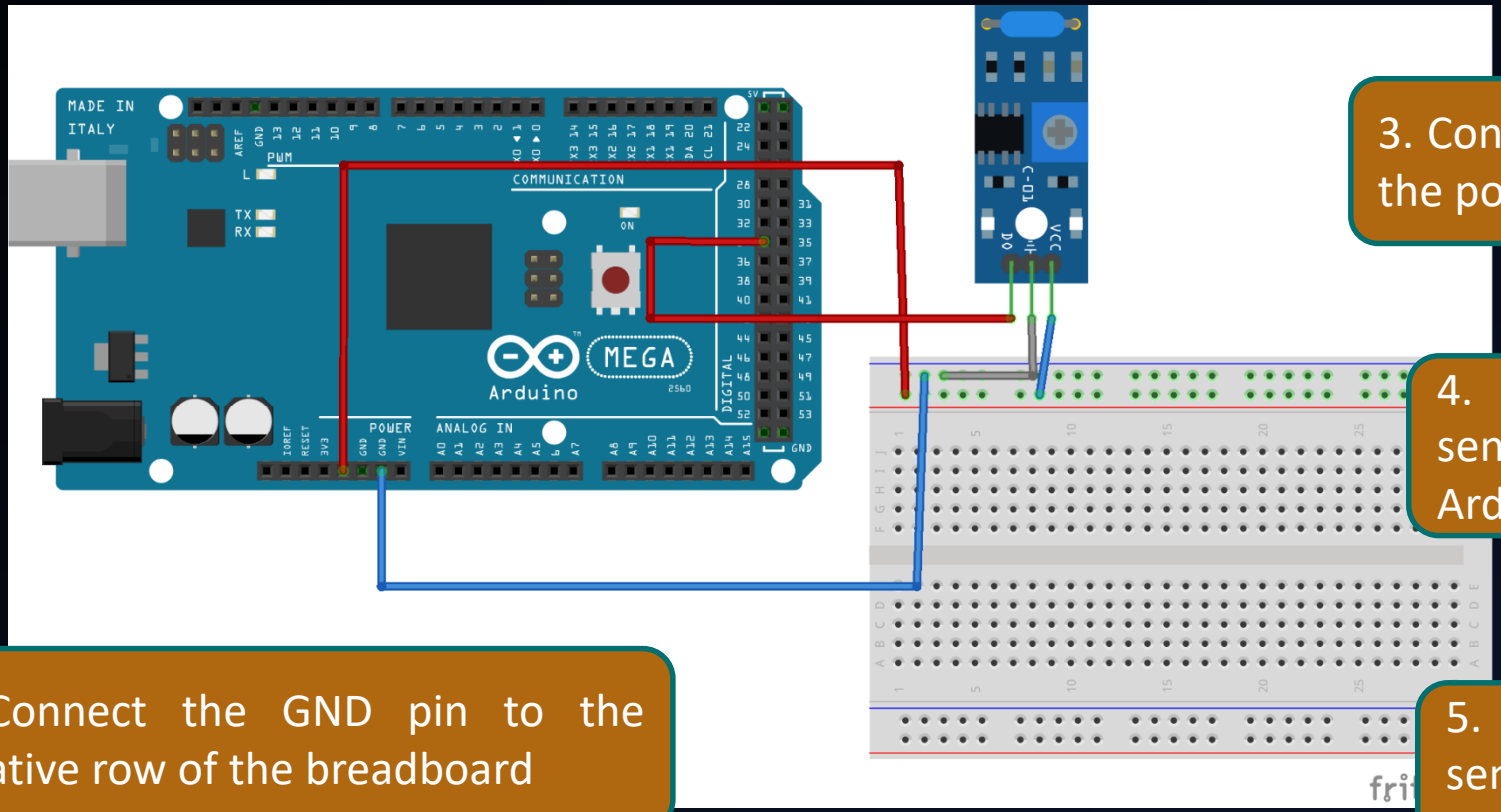
# Breakdown of SW420 vibration sensor




Remember!

*Do not turn the sensitivity switch to a **too high/low value**, as it may cause **false alarm** more easily!*

# Connection (vibration sensor)




3. Connect the +5V pin on the sensor to the positive row of the breadboard

4. Connect the data pin on the sensor to the digital pin 34 on the Arduino MEGA 2560

1. Connect the GND pin to the negative row of the breadboard

5. Connect the GND pin on the sensor to the negative row of the breadboard

2. Connect the +5v pin on the MEGA 2560 to the positive row of the breadboard

# Practice 1 (Vibration sensor)

1. Declare a variable to store the pin number for the vibration sensor

2. Setup a method with return value of the measured (pulse) value from the vibration sensor

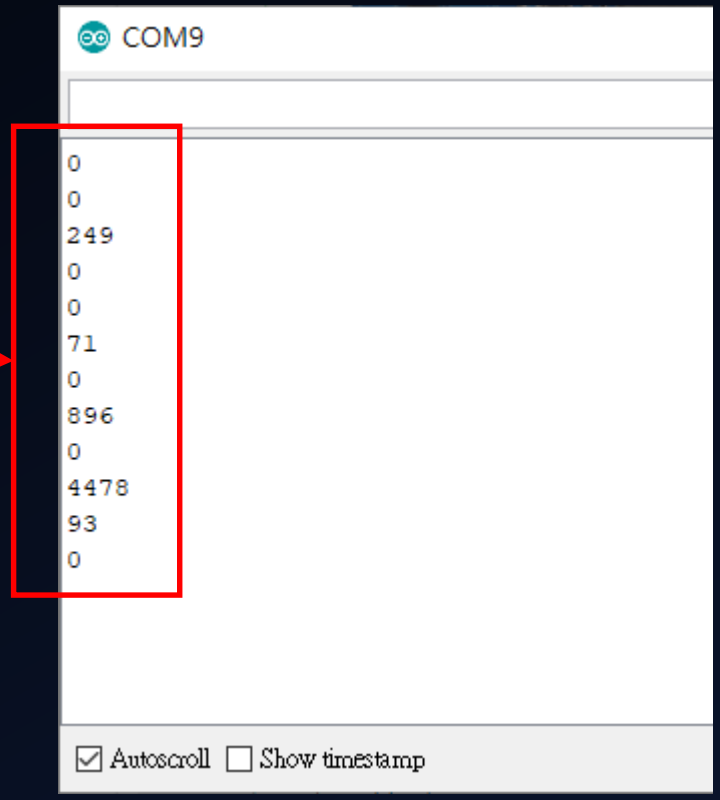
3. Declare a method TP_init() you have just set up, so that the program will fetch the value from the sensor for every 50ms

4. If successful, you should see the value from the vibration sensor in the serial monitor for every 50ms

```
sketch_mar20a
int LED_Pin = 13;
int vibration_Pin = 34;

void setup(){
  pinMode(LED_Pin, OUTPUT);
  pinMode(vibration_Pin, INPUT);
  Serial.begin(9600);
}
void loop(){
  long measurement = TP_init();
  delay(50);
  Serial.println(measurement);
  if (measurement > 1000){
    digitalWrite(LED_Pin, HIGH);
  }
  else{
    digitalWrite(LED_Pin, LOW);
  }
}

long TP_init(){
  delay(10);
  long measurement = pulseIn (vibration_Pin, HIGH);
  return measurement;
}
```

# Practice 1 (Vibration sensor)

When successful, you should see some similar values as shown on the picture on the right. (The higher the value, the more intense the vibration in which the sensor has detected)

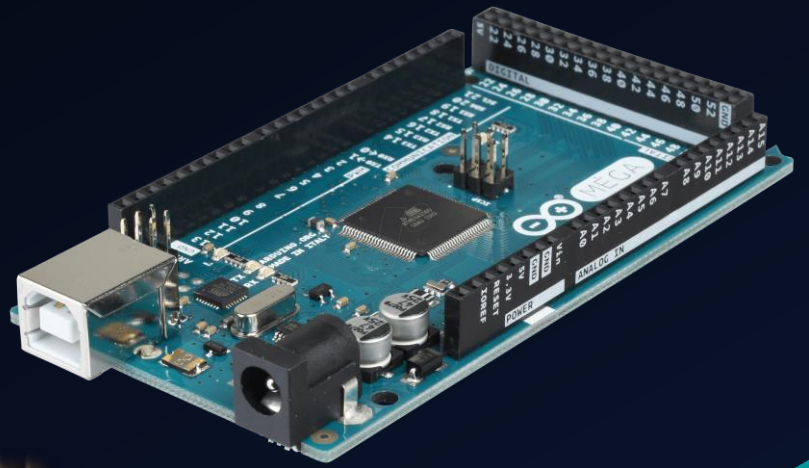


# Challenge

- How about if you want to trigger an alert by making multiple LEDs to blink, when the vibration sensor has detected a vibration with **pulse value >= 1000** ?
  - And, how to make the siren sound 10 seconds, when an **ultrasound sensor has detected an object within 50cm** & vibration sensor has detected a **vibration with pulse value >= 500**?

# Developing a simple burglar alarm using Arduino

## CHAPTER 7 – LCD DISPLAY MODULE (1602A I2C) – **PART 1**

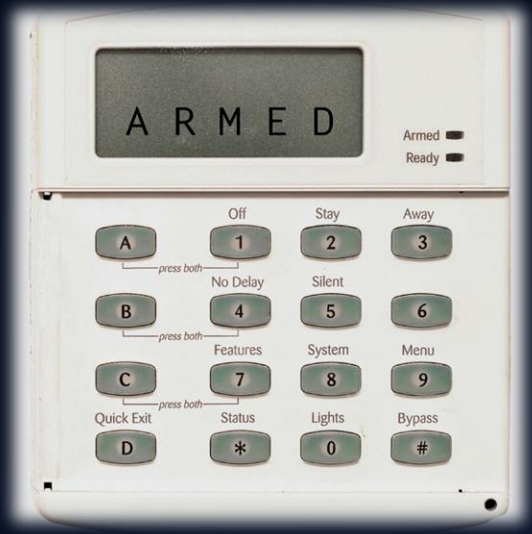


# After studying this chapter, you will be able to:

1. State & explain the usage of an LCD display in a burglar alarm
2. Implement a LCD display on Arduino
3. Outline the text display mechanisms on a 1602a LCD display
4. Display text on a 1602a LCD display

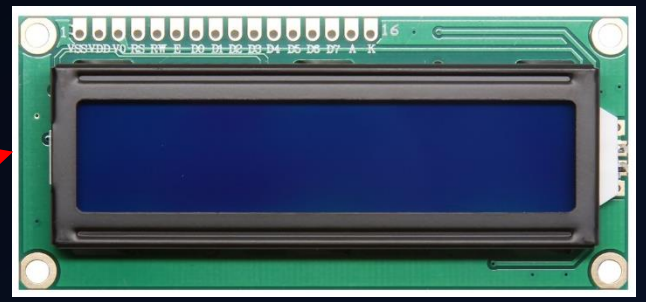
- In the past chapters, you may already know how the use different kind of sensors to detect motions/irregular movements, and trigger alerts.
- However, it may be difficult for us to read the **actual system status (how the alarm system is preforming) in detail**
  - E.g. the actual **sensor values**
  - System status message (and so on)

→ *Which module should we use to display text messages, such that the readability of the system status can be enhanced?*



- A **1602a LCD module** may help displaying text messages

1602a non-i2c version



- With a LCD display module, the people, or the one who are administering the burglar alarm system, may have a clearer understanding of the system status

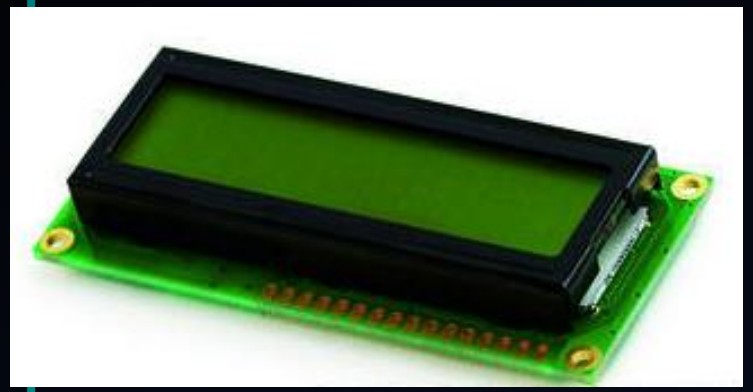
1602a i2c version

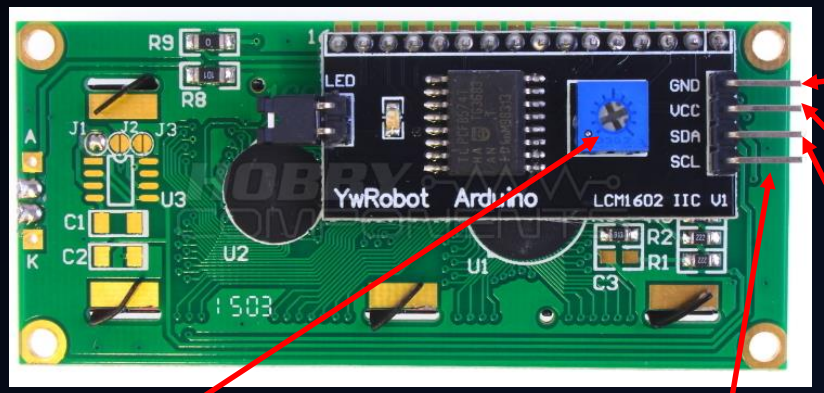

TIP *For simplifying the cable connection, it is recommend the choose the **I2C version** of 1602a LCD module, instead of the non-I2C one.*

# Preparation

- 1x Arduino Mega 2560 Mainboard
- 1x I2C 1602a LCD display
- 4x female to male jumper cable


# Breakdown of I2C 1602a LCD display module






GND pin

+5V pin

SDA pin

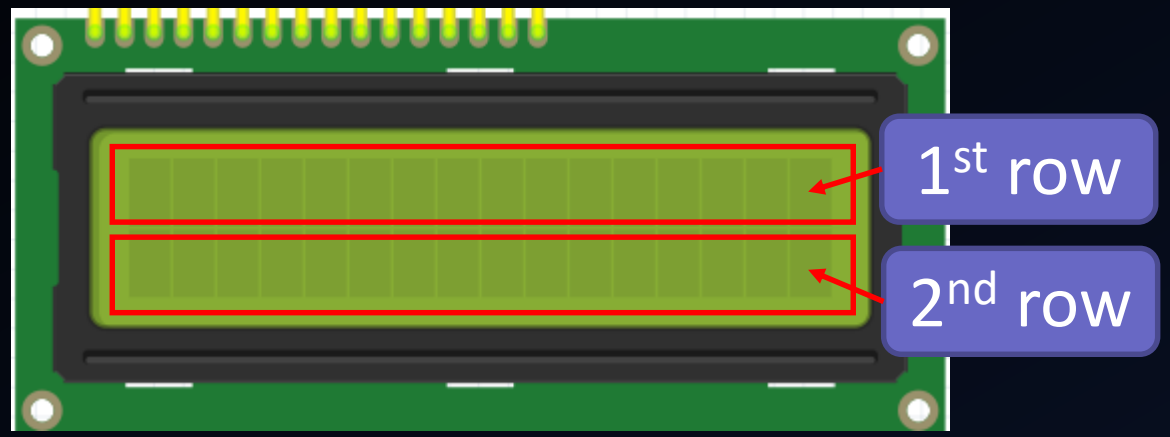
Backlight adjustment switch

SCL pin


*Do not turn the backlight adjustment switch to a **too high/low value.***

Otherwise, you may hardly see the text on the LCD display!

# Breakdown of I2C 1602a LCD display module



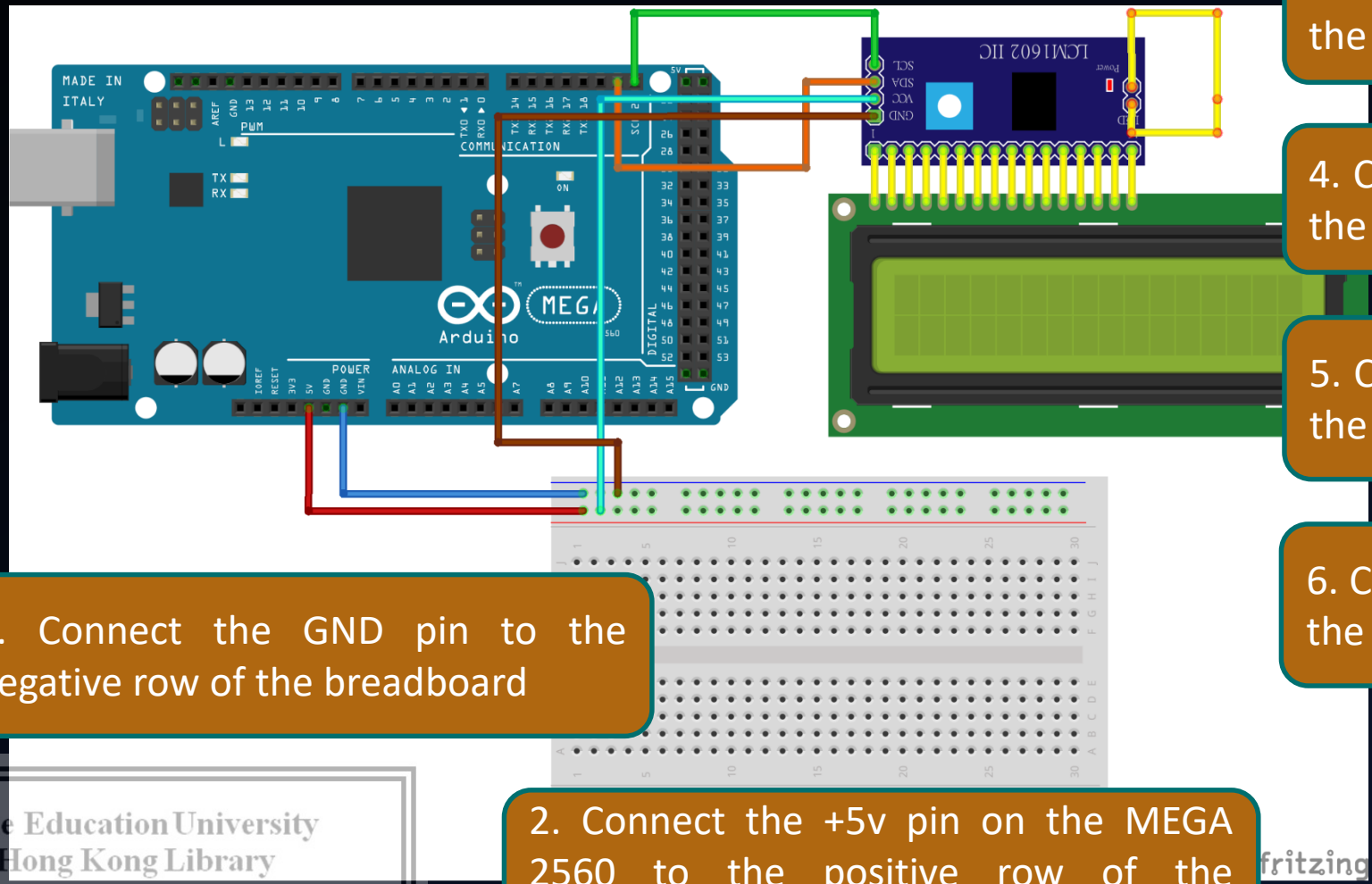

For a 1602a LCD display, it consists of 2 rows, each of them can display up to 16 **ASCII characters**

If we want to display the text (“HELLO”) at the start of the first row, we need to set the **cursor position to (0, 0)**

Similarly, if we decided to display some text at the start of the second row, then the **cursor position will be (0, 1)**

Remember! *Do not set the 1602a LCD to display non-ASCII characters. Otherwise, garbled text will be displayed.*

# Connection (I2C 1602a LCD display)




1. Connect the GND pin to the negative row of the breadboard

2. Connect the +5v pin on the MEGA 2560 to the positive row of the breadboard

3. Connect the SCL pin on the 1602a to the SCL pin 22 of Arduino

4. Connect the SDA pin on the 1602a to the SDA pin 21 of Arduino

5. Connect the VCC pin on the 1602a to the positive row of the breadboard

6. Connect the GND pin on the 1602a to the negative row of the breadboard

# Practice 1 (I2C 1602a LCD display)

1. Include two libraries for communicating with I2C 1602a LCD

2. Declare the type of LCD you are using (i.e. 2 rows, 16 characters each)

3. Set the LCD to display text "**Hello world**" on the 1st row

4. Set the LCD to display text "**I love ICT**" on the 2nd row

LCD1

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Hello world");

  lcd.setCursor(0,1);
  lcd.print("I love ICT");
}

void loop(){

}
```

# Practice 1 (I2C 1602a LCD display)

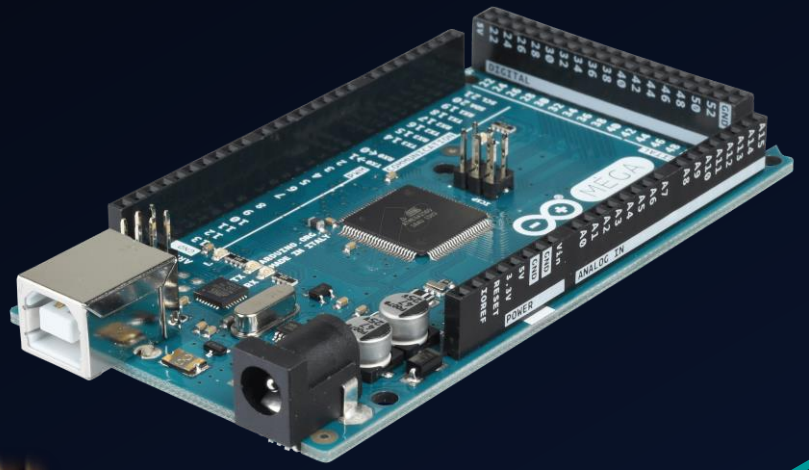
When successful, you should see the text as shown on the picture

# Challenge

- How about if you want show the **distance value from the ultrasound sensor** for every detected object?
  - And, how to make the LCD display to show a message, when an object is detected **within 50cm?**

# Developing a simple burglar alarm using Arduino

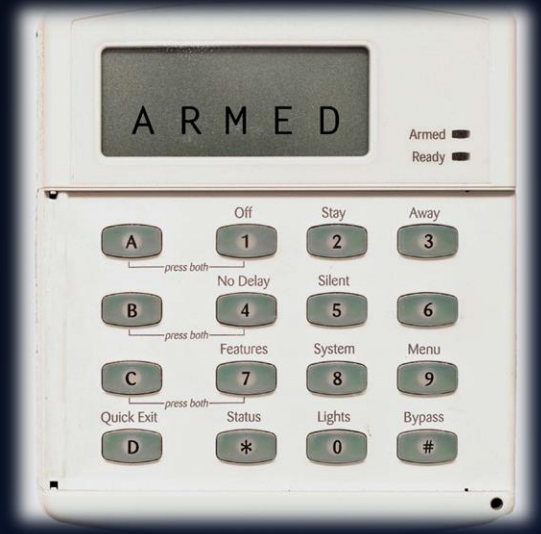
## CHAPTER 8 – LCD DISPLAY MODULE (1602A I2C) – **PART 2**



# After studying this chapter, you will be able to:

1. State & explain the usage of an LCD display in a burglar alarm
2. Display sensor values on 1602a I2C LCD
    1. E.g. **distance values** from ultrasound sensor
3. Update the text on the LCD display

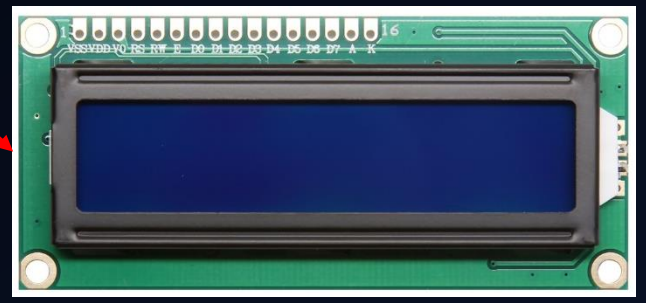
- In chapter 7, you may already know how to display text on a 1602a I2C LCD display.
- To make the text display **more meaningful**, we can combine the use of LCD display with different sensors in a burglar alarm:
  - E.g. Displaying the actual **sensor values**
  - System status message (and so on)



***REVISION:***

- A **1602a LCD module** may help displaying text messages
- With a LCD display module, the people, or the one who are administering the burglar alarm system, may have a clearer understanding of the system status

1602a non-i2c version



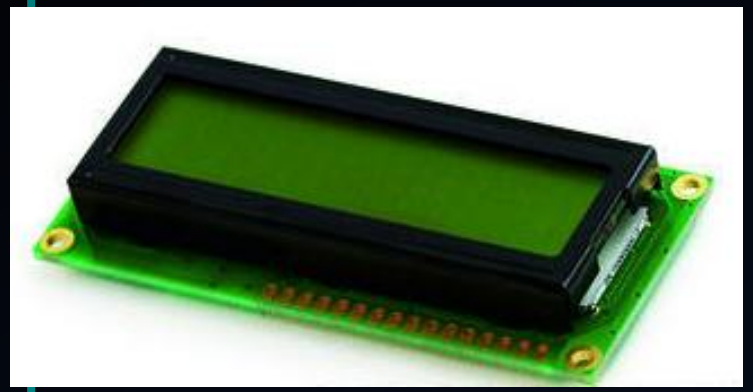
1602a i2c version

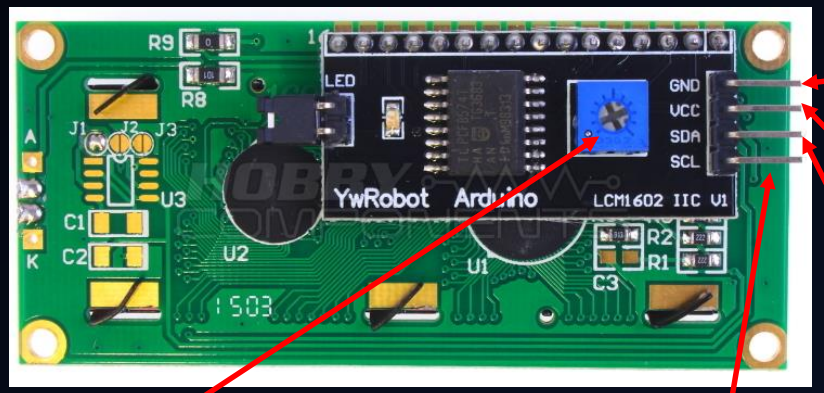

TIP *For displaying **real-time sensor values**, we need to use extra method from the I2C 1602a LCD library, which will be introduced later.*

# Preparation

- 1x Arduino Mega 2560 Mainboard
- 1x I2C 1602a LCD display
- 4x female to male jumper cable
- 1x ultrasound sensor


# *(REVISION)*Breakdown of I2C 1602a LCD display module






GND pin

+5V pin

SDA pin

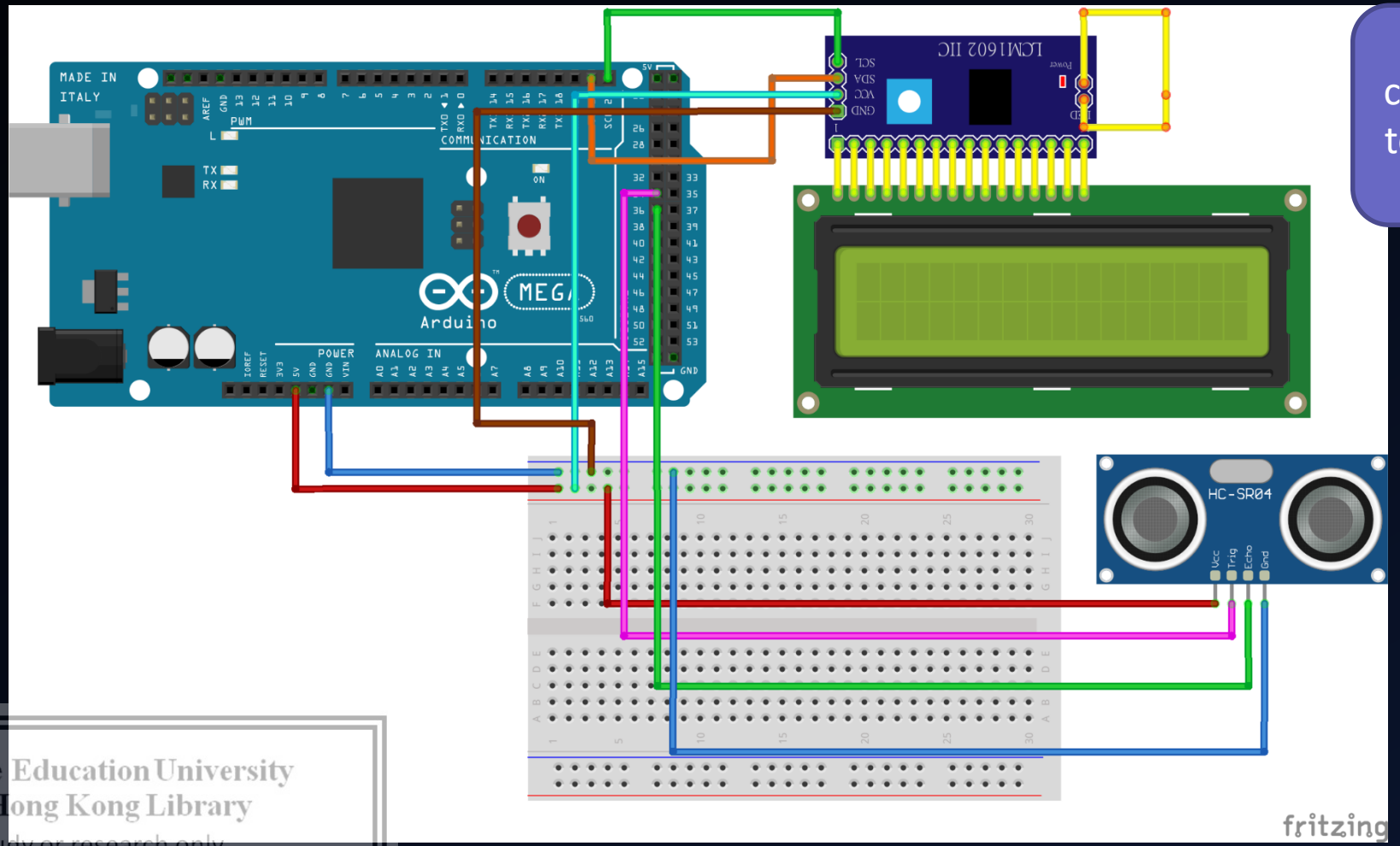
Backlight adjustment switch

SCL pin


*Do not turn the backlight adjustment switch to a **too high/low value.***

Otherwise, you may hardly see the text on the LCD display!

# Connection (I2C 1602a LCD display)




Still remember how to connect an ultrasound sensor to Arduino? (*you may refer to chapter 2 for revision*)

# Practice 1 (I2C 1602a LCD display)

1. Include two libraries for communicating with I2C 1602a LCD

2. Set the LCD to display the text "***Object distance***" in the 1st row

3. Set the LCD to display the sensor value from the ultrasound sensor

4. Set the LCD to display the updated value for every 500ms

Reuse the code in **chapter 3**, with a bit modification

LCD2 §

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
int trigPin = 26;
int echoPin = 28;
long duration, cm;

LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Object Distance:");
}

void loop()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = (duration/2) / 29.1;

  lcd.setCursor(0,1);
  lcd.print(cm);
  lcd.print("cm");
  delay(500);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Object Distance:");
}
```

# Practice 1 (I2C 1602a LCD display)

When successful, you should see the sensor value displayed on the LCD, and it will updated for every 0.5s (in case the distance value has changed)

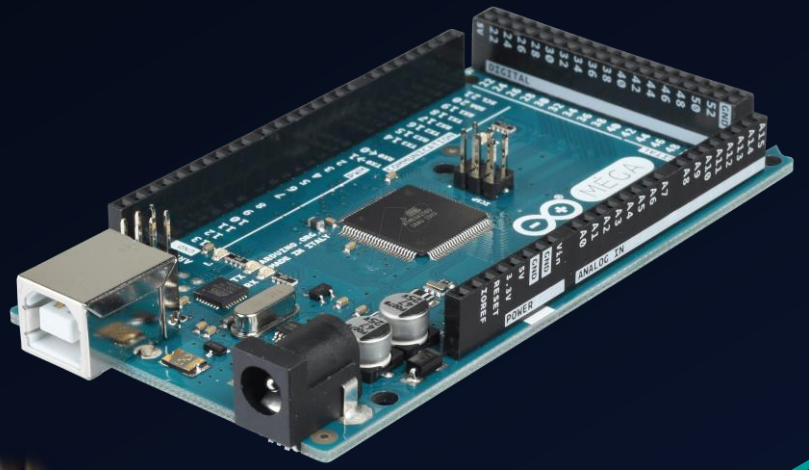
# Challenge

- How about if you want to make the LCD display to show a message "*POSSIBLE INTRUSION DETECTED*" for 3 seconds, when an object is detected **within 50cm?**
  - Afterwards, the LCD should display the value from the ultrasound sensor again

# Developing a simple burglar alarm using Arduino

## CHAPTER 9 – RC522 RFID MODULE

## PART 1



# After studying this chapter, you will be able to:

1. State & explain the usage of RFID card in a burglar alarm system
2. Implement a RC522 RFID reader on Arduino
3. Demonstrate the steps of getting the UID of a RFID card using RC522
4. Display the UID on LCD display

- In the previous chapters, you may already learnt about how to implement different kinds of sensors for intrusion detection, as well as delivering alerts using a variety kind of modules (e.g. buzzers, LEDs)
- However, you may discover we seems focusing on triggering alerts/alarms using sensors, ***but how to manually stop an activated alarms triggered by sensors?***


→ *For example, a burglar alarm has been triggered. After investigation, the authorized security guard need to stop the buzzer alerts, in what way can he do so?*

- A **RFID reader, combining with RFID cards** may help achieving this

RC522 RFID Reader


- With a RFID reader, the security guard can tap a designated RFID card to the reader on the burglar alarm system, to temporarily deactivate the buzzer alerts manually.

RFID card & tag

TIP *There are 3 major types of RFID cards in terms of **sensing distance**. Pay attention when choosing which type of RFID cards to use*

## Type of RFID card:

Will be implemented in this & next chapter

| Type | Sensing distance |
|---|---|
| Close-coupled | < 1 cm |
| Proximity | > 1 cm and < 10 cm |
| Vicinity | Within 50 cm |

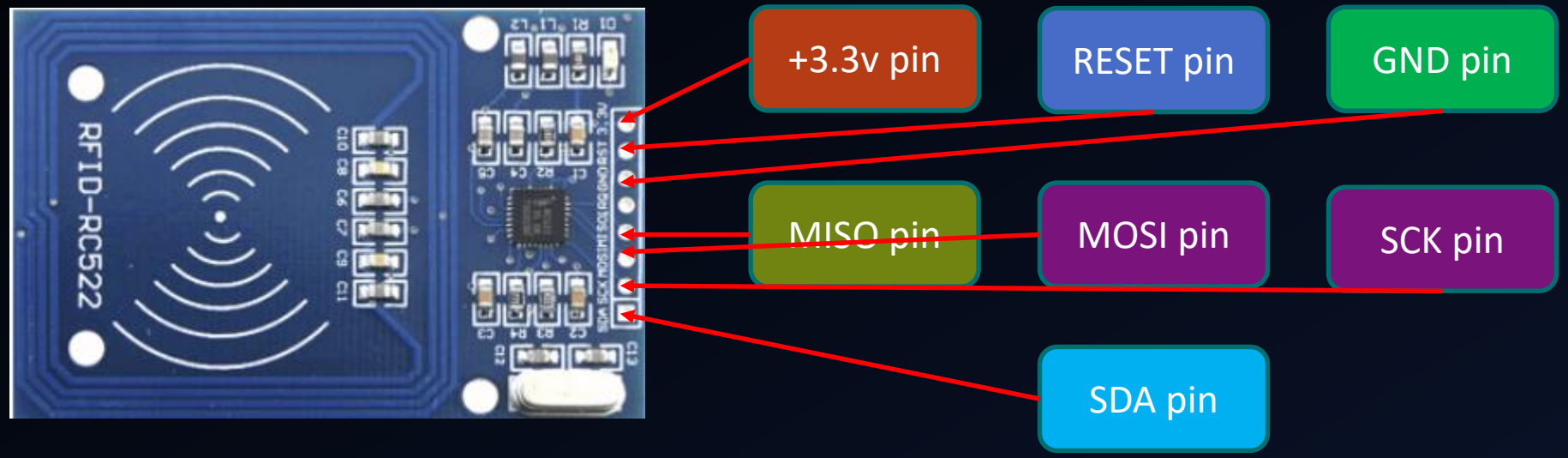
In this chapter, we will first introduce **how to read the unique UID of a RFID tag**


*For the close-coupled type, you need to tap the card **VERY CLOSE** to the reader for the card to be successfully read*

# Preparation

- 1x Arduino Mega 2560 Mainboard
- 1x I2C 1602a LCD display
- 4x female to male jumper cable
- 4x male to female jumper cable
- 1x RC522 card reader
- 1x RFID card

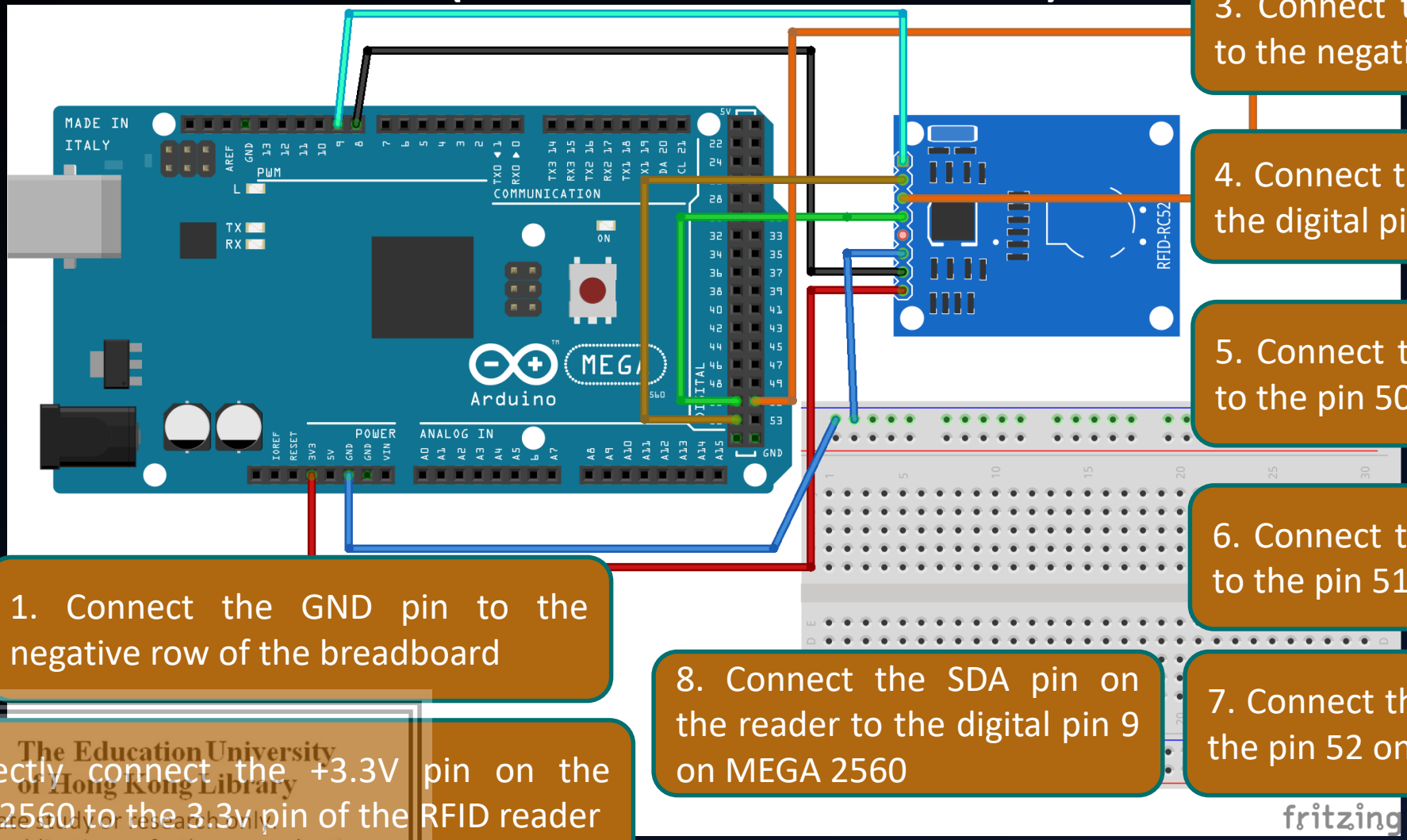

# Breakdown of RC522 RFID reader




Remember!

*The pin number mapping is different between Arduino UNO & Arduino Mega 2560, pay extra attention during implementation*

# Connection (RC522 RFID reader)




1. Connect the GND pin to the negative row of the breadboard

2. Directly connect the +3.3V pin on the MEGA 2560 to the 3.3v pin of the RFID reader

3. Connect the GND pin on the reader to the negative row on the breadboard

4. Connect the RST pin on the reader to the digital pin 8 on MEGA 2560

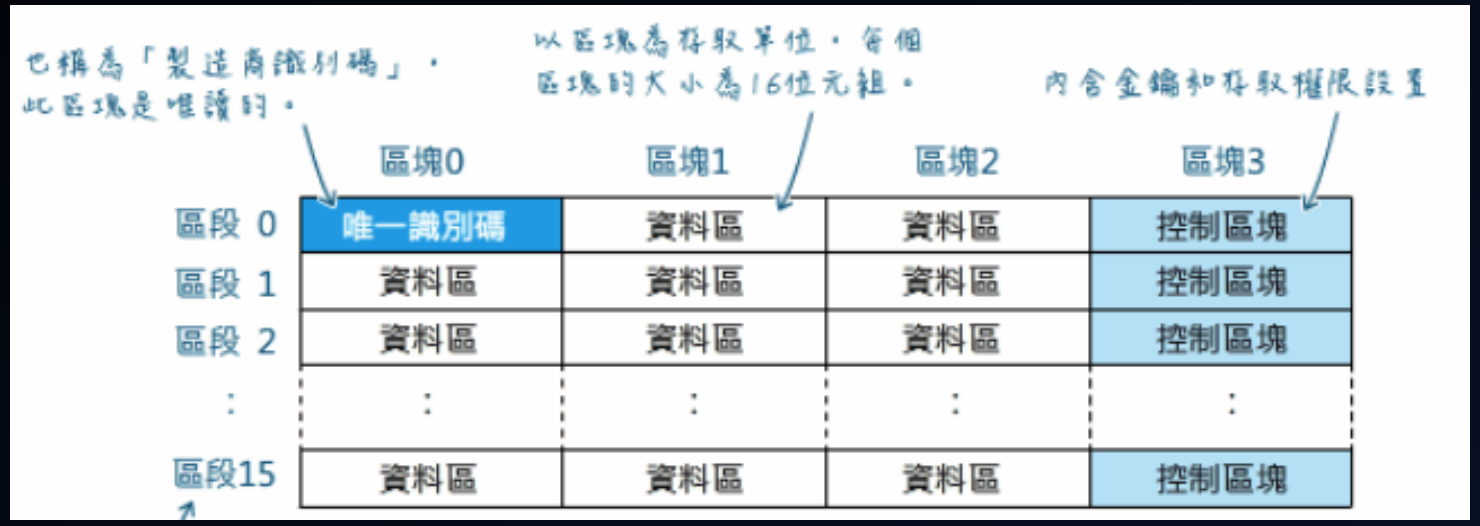
5. Connect the MISO pin on the reader to the pin 50 on MEGA 2560

6. Connect the MOSI pin on the reader to the pin 51 on MEGA 2560

7. Connect the SCK pin on the reader to the pin 52 on MEGA 2560

8. Connect the SDA pin on the reader to the digital pin 9 on MEGA 2560

# Practice 1 (RC522 RFID reader)




Source: swf.com.tw

In general, a RFID card consists of 1KB EEPROM memory, and it is divided into **16 sectors**. Each sector consists of **4 blocks**. The unique UID we need to read is stored in **block 0 in sector 0 (0, 0)**

# Practice 1 (RC522 RFID reader)

1. Include two libraries for communicating with RC522 RFID reader

2. Define the SDA & RST pin number

3. Initialize the RC522 RFID reader

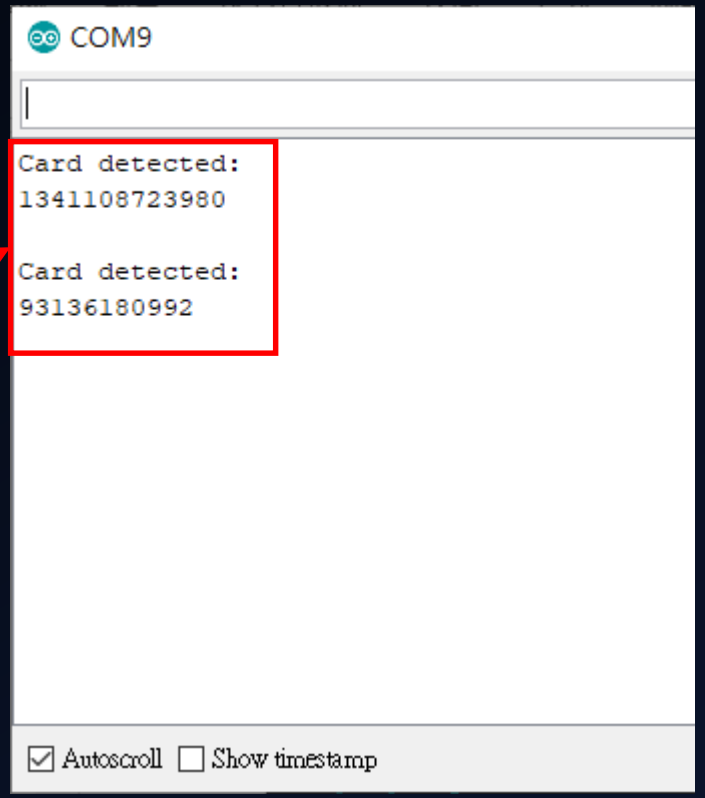
4. Detect RFID card for every second. If a card is detected, read the UID stored in (0, 0) and print it in decimal format to serial monitor

rfid

```
#include <SPI.h>
#include <RFID.h>

#define SDA 9
#define RESET 8

RFID RC522(SDA, RESET);

void setup()
{
  Serial.begin(9600);
  SPI.begin();
  RC522.init();
}

void loop()
{
  if (RC522.isCard())
  {
    RC522.readCardSerial();
    Serial.println("Card detected:");
    for(int i=0;i<5;i++)
    {
    Serial.print(RC522.serNum[i],DEC);
    }
    Serial.println();
    Serial.println();
  }
  delay(1000);
}
```

# Practice 1 (I2C 1602a LCD display)

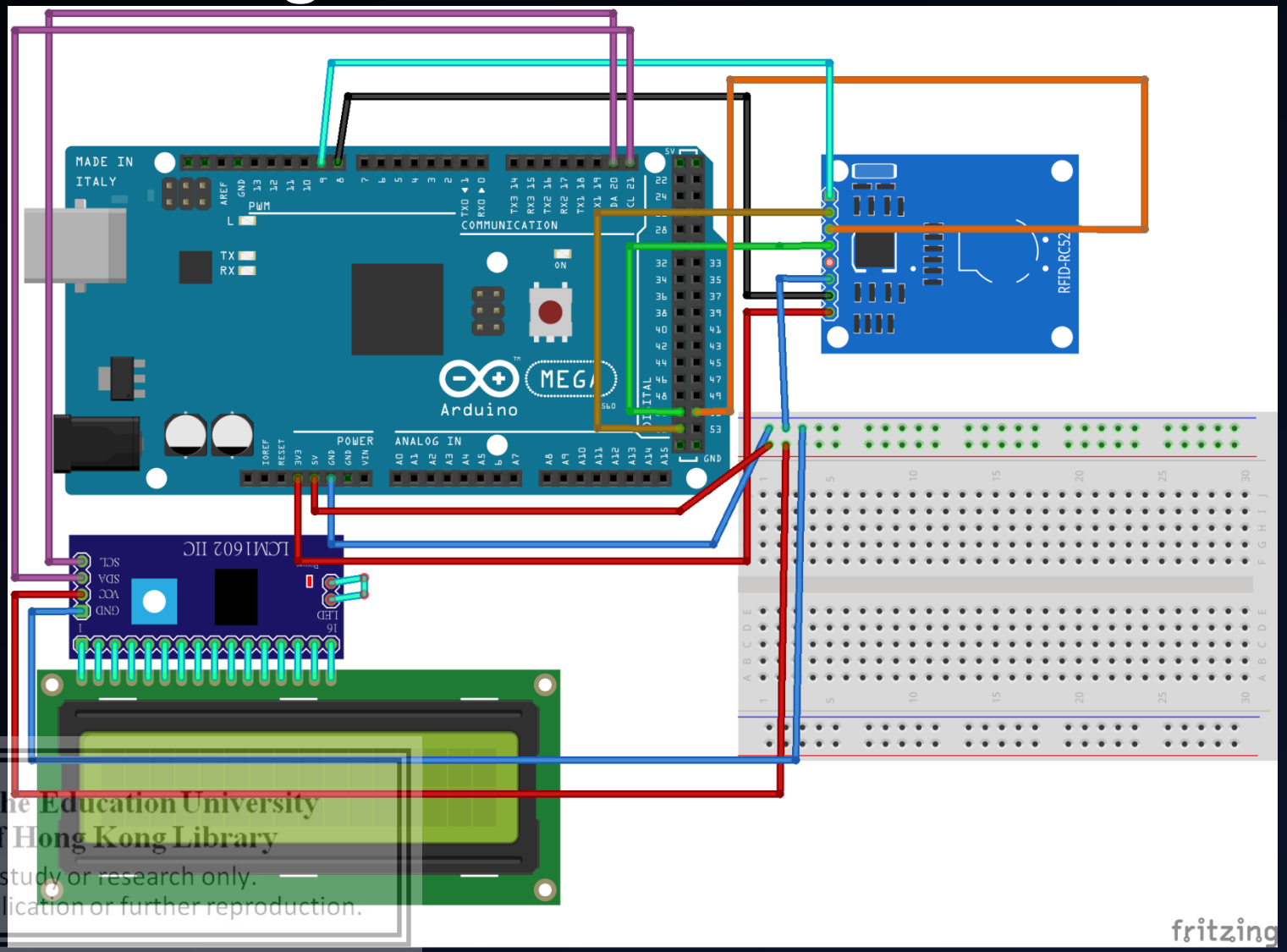
Now tap the RFID cards to the reader. If successful, you should see the card UID in the serial monitor



# Challenge

- How about if you want show the **card UID** on the 1602a LCD once a compatible RFID card is detected?
  - And, how to make a buzzer to sound, if a unauthorized card is detected?
  - ***And how to disarm the intrusion siren with a authorized RFID card?***

# Challenge




Still remember the 1602a LCD connection? You may refer to the notes of chapter 7

# Challenge

1. Remember to include two libraries for 1602a LCD

2. 1602a LCD initialization

3. Initialize the RC522 RFID reader

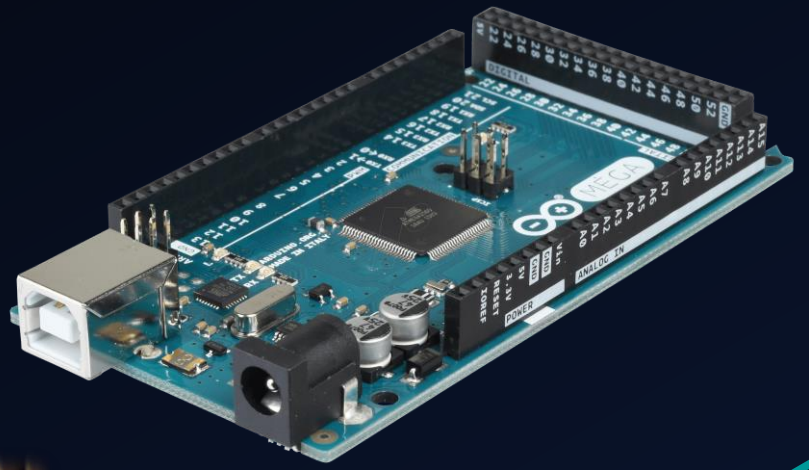
4. Show a message for detected RFID card and its UID

```
#include <SPI.h>
#include <RFID.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

LiquidCrystal_I2C lcd(0x27,16,2);
#define SDA 9
#define RESET 8

RFID RC522(SDA, RESET);

void setup()
{
  lcd.init();
  lcd.backlight();
  Serial.begin(9600);
  SPI.begin();
  RC522.init();
}

void loop()
{
  if (RC522.isCard())
  {
    RC522.readCardSerial();
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Card detected:");
    lcd.setCursor(0,1);
    for(int i=0;i<5;i++)
    {
    lcd.print(RC522.serNum[i],DEC);
    }
  }
  delay(1000);
}
```

# Developing a simple burglar alarm using Arduino

## CHAPTER 10 – RC522 RFID MODULE

## PART 2



# After studying this chapter, you will be able to:

1. Recall the possible use of RFID card in an burglar alarm system
2. Disable the triggered alerts using an dedicated RFID card

- In chapter 9, you already know how to read the unique UID of an RFID card.
- Then, how about stopping an activated alarms triggered by sensors **using a dedicated RFID card**?


→ *For example, how to stop an alarm using a RFID card* ***with UID "XXXXXXXX"?***

## *RFID card & sensor revision*

- A **RFID reader, combining with RFID cards** may help achieving this

RC522 RFID Reader


- With a RFID reader, the security guard can tap a designated RFID card to the reader on the burglar alarm system, to temporarily deactivate the buzzer alerts manually.

RFID card & tag

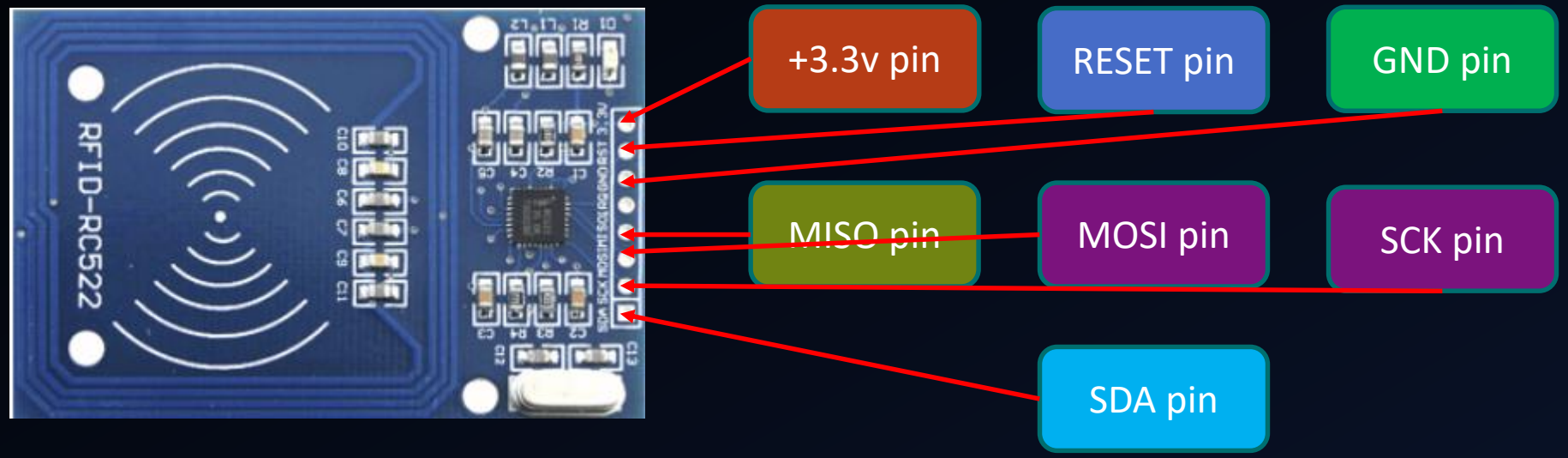
TIP *The (0, 0) sector of a RFID card, which contains the UID of itself, is **read-only**.*

# Preparation

- 1x Arduino Mega 2560 Mainboard
- 1x I2C 1602a LCD display
- 4x female to male jumper cable
- 1x RC522 card reader
- 1x RFID card
- 1x LED unit
- 1x 220ohm resistor


## *RFID card & sensor revision*

# Breakdown of RC522 RFID reader

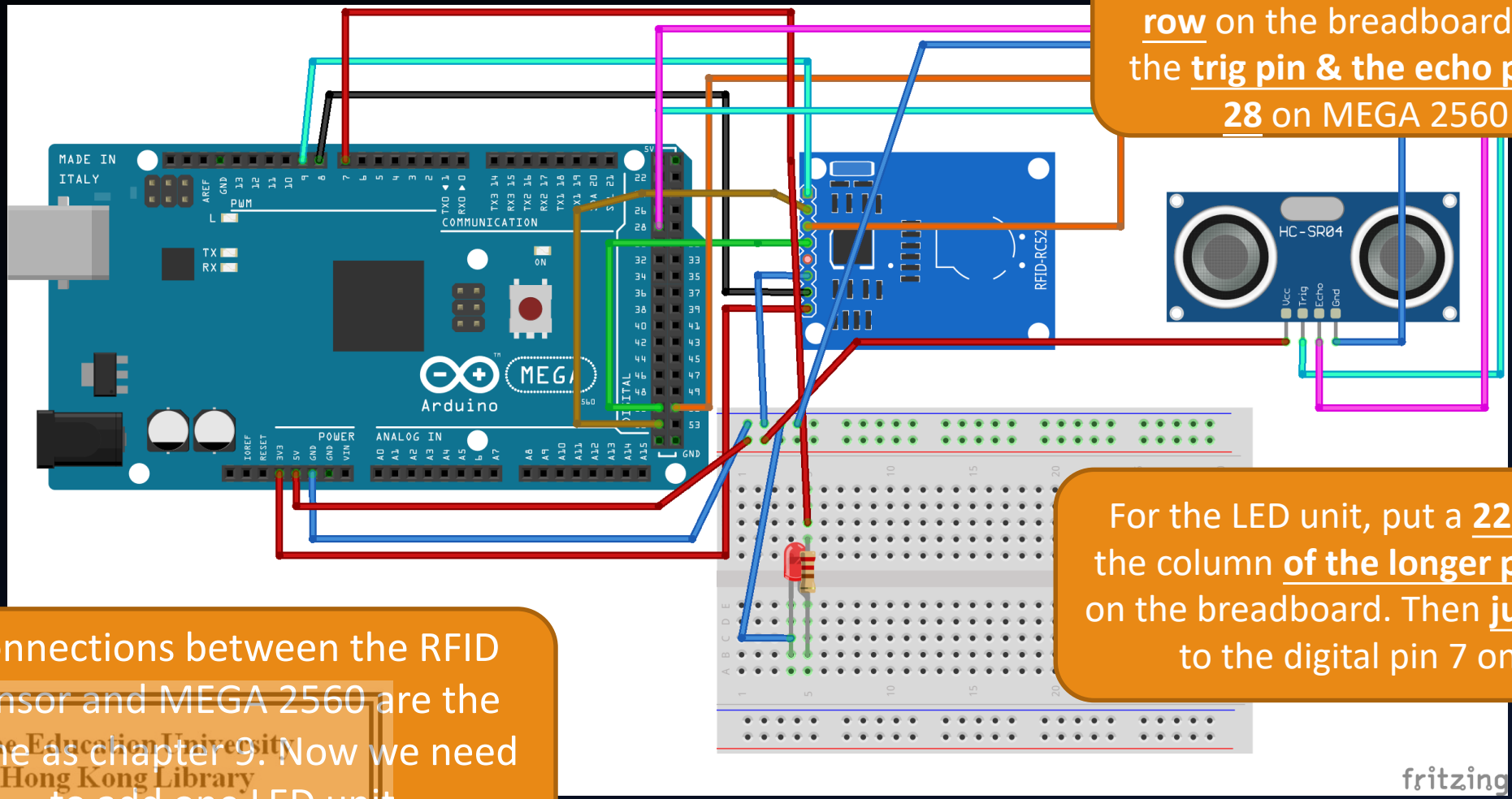



**(YOUR TASK)**

*Assume if the ultrasound sensor has detected an object within 50cm, a LED unit will blink until you tap a designated RFID card on the reader.*

# Connection (RC522 RFID reader)

For the ultrasound sensor, connect the **VCC pin** & **GND pin** to the **positive** & **negative row** on the breadboard respectively, and the **trig pin & the echo pin** to the **pin 26 & 28** on MEGA 2560 respectively




For the LED unit, put a **220ohm resistor** on the column **of the longer pin of the LED unit** on the breadboard. Then **jump the longer pin** to the digital pin 7 on MEGA 2560

Connections between the RFID sensor and MEGA 2560 are the same as chapter 9. Now we need to add one LED unit.

# Practice 1 (RC522 RFID reader)

1. Include two libraries for RC522 RFID reader

2. Define pin number for the LED and ultrasound sensor, and variable for storing distance data

3. Define variables for storing the UID read by the reader, and the UID of the designated card

4. Detect RFID card for every second. If a card is detected, read the UID stored in (0, 0) and print it in decimal format to serial monitor

5. Read the distance data and store it into variable **cm**

6. If object is detected within 50cm, the LED will blink continuously until a designated card is tapped on the reader

ultrasoundLEDRFID §

```
#include <SPI.h>
#include <RFID.h>
#define SDA 9
#define RESET 8
int ledPin =  7;
int trigPin = 26;
int echoPin = 28;
long duration, cm;
String uid = "";
String dUid = "1341108723980";

RFID RC522(SDA, RESET);

void setup() {
  Serial.begin (9600);
  pinMode(ledPin, OUTP
  pinMode(trigPin, OUT
  pinMode(echoPin, INP
  SPI.begin();
  RC522.init();
  uid = dUid;

}

void loop() {
  cardDetect();
  if (uid.equals(dUid) == false){
  blinkLED();
  }
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = (duration/2) / 29.1;
  while (cm <= 50){
    uid = "";
    resetUltraSound();
  }
}
```

# Practice 1 (RC522 RFID reader)

7. A method for making a LED unit to blink

8. A method for detecting RFID cards

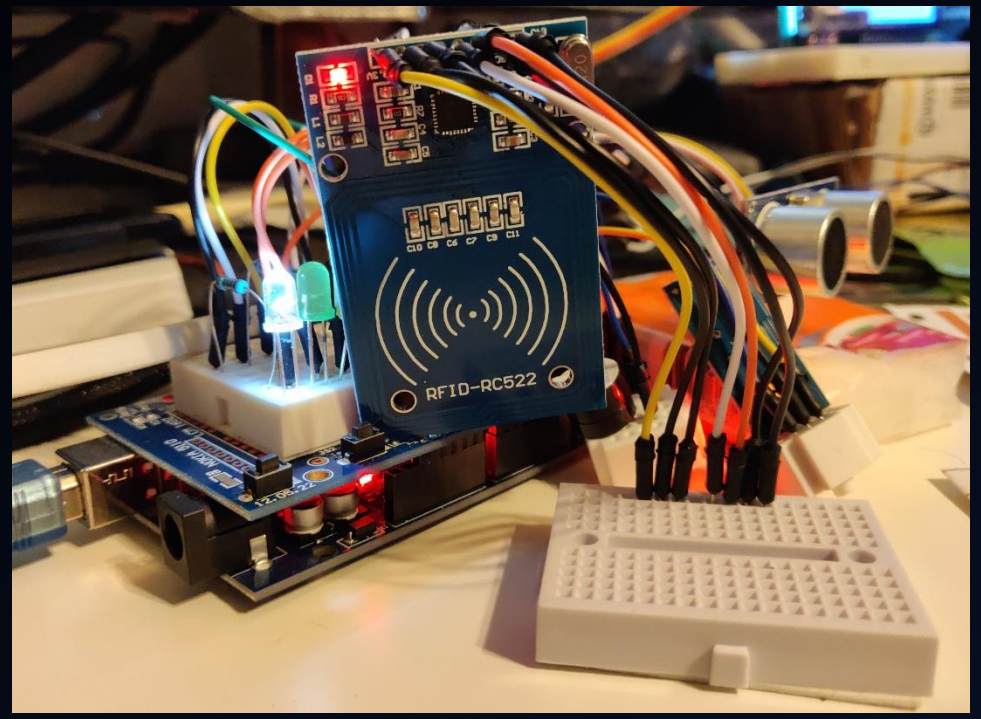
9. A method for resetting the ultrasound sensor after each detection of an object

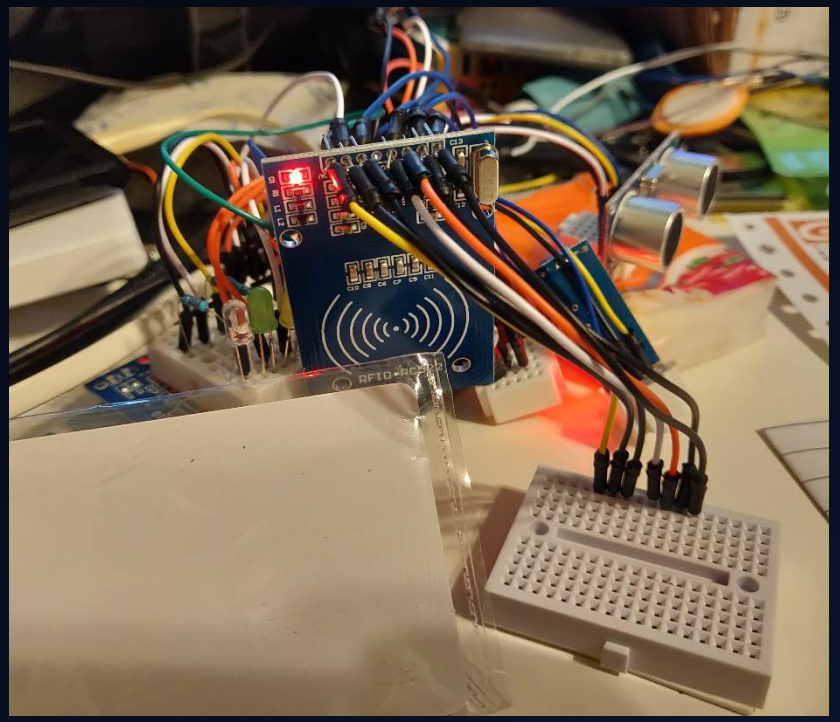
```
void blinkLED(){
   digitalWrite(ledPin, HIGH);
   delay(20);
   digitalWrite(ledPin, LOW);
   delay(20);
}
```

```
void cardDetect(){
  if (RC522.isCard())
  {
    uid = "";
    RC522.readCardSerial();
    for(int i=0;i<5;i++)
    {
    uid = uid + String(RC522.serNum[i],DEC);
    }
    Serial.print(uid);
    Serial.println();
    Serial.println();
  }
    delay(200);
}
```

```
void resetUltraSound(){
   digitalWrite(trigPin, LOW);
   delayMicroseconds(5);
   digitalWrite(trigPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(trigPin, LOW);
   pinMode(echoPin, INPUT);
   duration = pulseIn(echoPin, HIGH);
   cm = (duration/2) / 29.1;
}
```

# Practice 1 (I2C 1602a LCD display)



If successful, you should see the LED will blink when a object is detected within 50cm, and the blinking will stop only when the designated RFID card is tapped

# Challenge

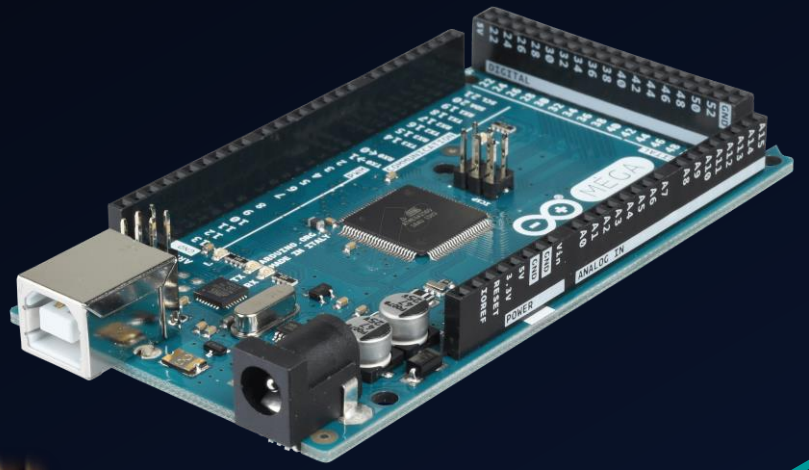
- Try placing an object within 50cm of the sensor, and **not moving it away**. Discover if there's any problem with the coding. If so, what's wrong?

- Try adding a **passive buzzer**, so that the **LED will blink and the buzzer will sound** until a designated card is tapped, when an object is detected.

# Developing a simple burglar alarm using Arduino

## CHAPTER 11 – RC522 RFID MODULE

## PART 3



# After studying this chapter, you will be able to:

1. Recall the skills of disabling the triggered alerts using an dedicated RFID card
2. Debug & correct the possible bugs of triggered alerts
3. Trigger multiple kind of alerts using RFID card

- In chapter 10, you already know how to disable alerts triggered by the ultrasound sensor using RFID card
- However, is it the same with the detection of **static** objects and **moving** objects?

→ *For example, a person walk pass the sensor*

***V.S.***

*A person walk in front of the sensor and stay still*

# Review

- Try to look the part (in red) of code you developed last chapter, have you noticed a problem?

- No matter an object is detected or not, the sensor always ***keep running***.
- If an object approached the sensor then ***stay still***, the sensor will keep ***detecting the same object*** again & again



***So, what will happen due to these issues?***

ultrasoundLEDRFID §

```
#include <SPI.h>
#include <RFID.h>
#define SDA 9
#define RESET 8
int ledPin =  7;
int trigPin = 26;
int echoPin = 28;
long duration, cm;
String uid = "";
String dUid = "1341108723980";

RFID RC522(SDA, RESET);

void setup() {
  Serial.begin (9600);
  pinMode(ledPin, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  SPI.begin();
  RC522.init();
  uid = dUid;

}

void loop() {
  cardDetect();
  if (uid.equals(dUid) == false){
  blinkLED();
  }
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = (duration/2) / 29.1;
  while (cm <= 50){
    uid = "";
    resetUltraSound();
  }
}
```

# Review

***So, what will happen due to these issues?***

- Since you haven't design a way to temporarily stop the sensor from running after each object detection, if a **still object** is in front of the sensor:

Object detected

Make LED to blink *x1*

Make LED to blink *x3*

Make LED to blink *x2*

Make LED to blink ***x(n)……***

→ ***The blinkLED() method will be called infinitely, which will cause the LED not to blink***



ultrasoundLEDRFID §

```
#include <SPI.h>
#include <RFID.h>
#define SDA 9
#define RESET 8
int ledPin =  7;
int trigPin = 26;
int echoPin = 28;
long duration, cm;
String uid = "";
String dUid = "1341108723980";

RFID RC522(SDA, RESET);

void setup() {
  Serial.begin (9600);
  pinMode(ledPin, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  SPI.begin();
  RC522.init();
  uid = dUid;

}

void loop() {
  cardDetect();
  if (uid.equals(dUid) == false){
  blinkLED();
  }
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = (duration/2) / 29.1;
  while (cm <= 50){
    uid = "";
    resetUltraSound();
  }
}
```

***To avoid mistakes, the flow of delivering alerts should be as follows:***

1. Sensor start running
2. If an object **is detected**
    1. Trigger alerts (LED, buzzer, etc)
    2. **Temporarily stop sensor** from running
    3. If a **dedicated RFID card** is tapped on the reader
        1. Resume the sensor
    4. Else keeping the sensor in “STOP” status
3. Else keep the sensor from running



ultrasoundLEDRFID §

```
#include <SPI.h>
#include <RFID.h>
#define SDA 9
#define RESET 8
int ledPin =  7;
int trigPin = 26;
int echoPin = 28;
long duration, cm;
String uid = "";
String dUid = "1341108723980";

RFID RC522(SDA, RESET);

void setup() {
  Serial.begin (9600);
  pinMode(ledPin, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  SPI.begin();
  RC522.init();
  uid = dUid;

}

void loop() {
  cardDetect();
  if (uid.equals(dUid) == false){
  blinkLED();
  }
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = (duration/2) / 29.1;
  while (cm <= 50){
    uid = "";
    resetUltraSound();
  }
}
```

## *So how to implement these in coding?*

- We need **a flag** to store the status of object detection
- i.e. object detected → **true**
- No object detected → **false**
- Hence, if **flag = false** → sensor running
- If **flag = true** → stop sensor temporarily

```
ultrasoundLEDRFID §
#include <SPI.h>
#include <RFID.h>
#define SDA 9
#define RESET 8
int ledPin =  7;
int trigPin = 26;
int echoPin = 28;
long duration, cm;
String uid = "";
String dUid = "1341108723980";

RFID RC522(SDA, RESET);

void setup() {
  Serial.begin (9600);
  pinMode(ledPin, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  SPI.begin();
  RC522.init();
  uid = dUid;

}

void loop() {
  cardDetect();
  if (uid.equals(dUid) == false){
  blinkLED();
  }
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = (duration/2) / 29.1;
  while (cm <= 50){
    uid = "";
    resetUltraSound();
  }
}
```

# Practice 1

## Using the same program in ***chapter 10***:

1. Add a Boolean variable for storing the object detection status

```
ultrasoundLEDRFID

#include <SPI.h>
#include <RFID.h>
#define SDA 9
#define RESET 8
int ledPin =  7;
int trigPin = 26;
int echoPin = 28;
long duration, cm;
String uid = "";
String dUid = "1341108723980";
bool isDetected = false;
```

2. Add a if-statement, so that only if no objects are detected, the sensor will run.

```
void loop() {
  cardDetect();
  if (uid.equals(dUid) == false){
  blinkLED();
  }
  if (isDetected == false){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = (duration/2) / 29.1;
  while (cm <= 50){
    isDetected = true;
    uid = "";
    resetUltraSound();
  }
  }
}
```

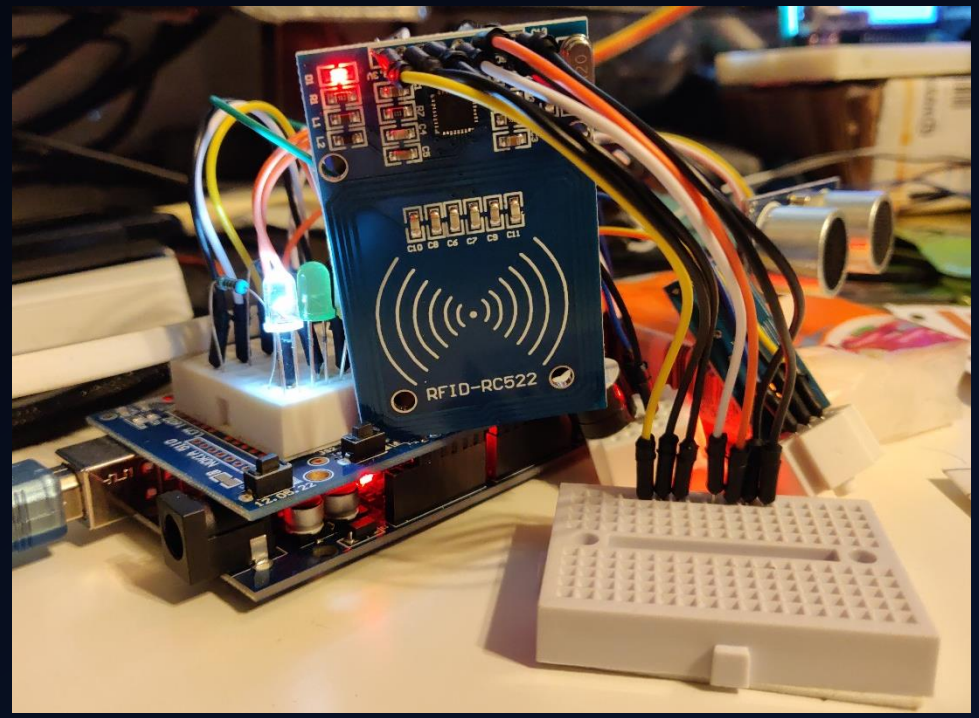
3. If an object is detected, the Boolean value stored in isDetected will become “true”, which the running of sensor(s) will be paused

4. Add another if-statement, so that if the triggered alerts are disabled by a RFID card, the value stored in isDetected will become false, which will make the sensor(s) resume operation(s) again

```
void cardDetect(){
  if (RC522.isCard())
  {
    uid = "";
    RC522.readCardSerial();
    for(int i=0;i<5;i++)
    {
    uid = uid + String(RC522.serNum[i],DE
    }
    if (uid.equals(dUid) == false){
      isDetected = false;
    }
    Serial.print(uid);
    Serial.println();
    Serial.println();
  }
  delay(200);
}
```

# Practice 1

If successful, you should see the LED blinks even an object is approaching the sensor(s) and stay still in the front, unlike the logic error as discovered in chapter 10



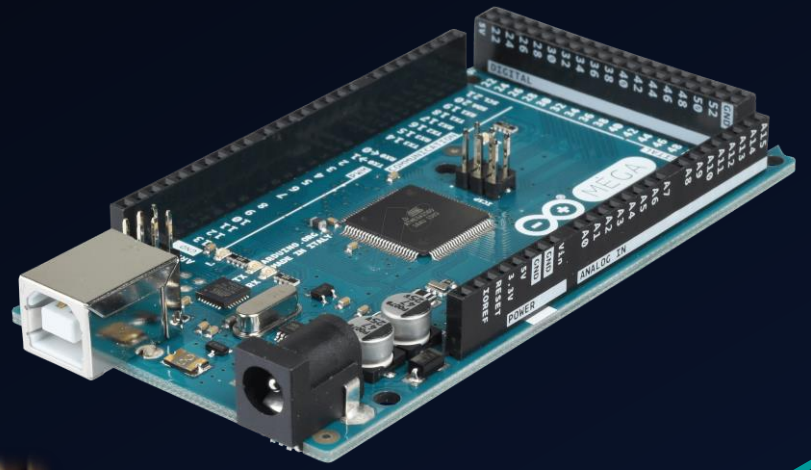
# Challenge

- Instead of just the LEDs, add a buzzer unit, so that if an object is detected (including ***still & moving objects***), LED blinking & buzzer siren alerts can be delivered properly

- Try adding a **1602a LCD,** so that not just alerts will be delivered, but also the **related system messages** will be shown on the LCD (which sensor is triggered/distance information/authorized RFID card or not)

# Developing a simple burglar alarm using Arduino

## CHAPTER 12 – CONSOLIDATION AND CONCLUSION



# After studying this chapter, you will be able to:

1. Recall the characteristics of sensors in a burglar alarm system
2. Select appropriate sensors & integrate it to form a burglar alarm system with different functions, based on the given scenarios

# Review of sensors (1)

- If you want to know if a person is in the detection range of the system, you should use _____ sensors
- If you want to tell the **particular object distance**, then _____ sensors should be used
- May claims she can use an **active buzzer** to emit a sound with varied pitch, is her claim correct? (YES/NO)

# Review of sensors (2)

- Sam decided to display a warning message on a **1602a LCD**, and he wants the message to be displayed starting from the **1st character on the 2nd line**. He claims the code "**setCursor(1, 1)**" can help him to do so, is his claim correct? (YES/NO)

- Mary said a **RFID card** can only be used for **disabling something**, but **not triggering something** in a burglar alarm system, do you agree with her?

# Sensors integration

- In the previous chapters, you may know how to use different kind of sensors to trigger alerts
- However, in reality, the **sensors integration** is usually much more complicated. There is **no best integration** that can provide the best level of security.
- Instead, we need to carefully investigate **the requirements by customers or companies**, so as to come up with an integration / design which can **best suit their needs**.

# Scenarios

When the system has detected shock (*intensity >= 1000*), I want the system to emit sound at *varied pitch*, and *2 LED units* will blink interchangeability.

***Client A***

***Client B***

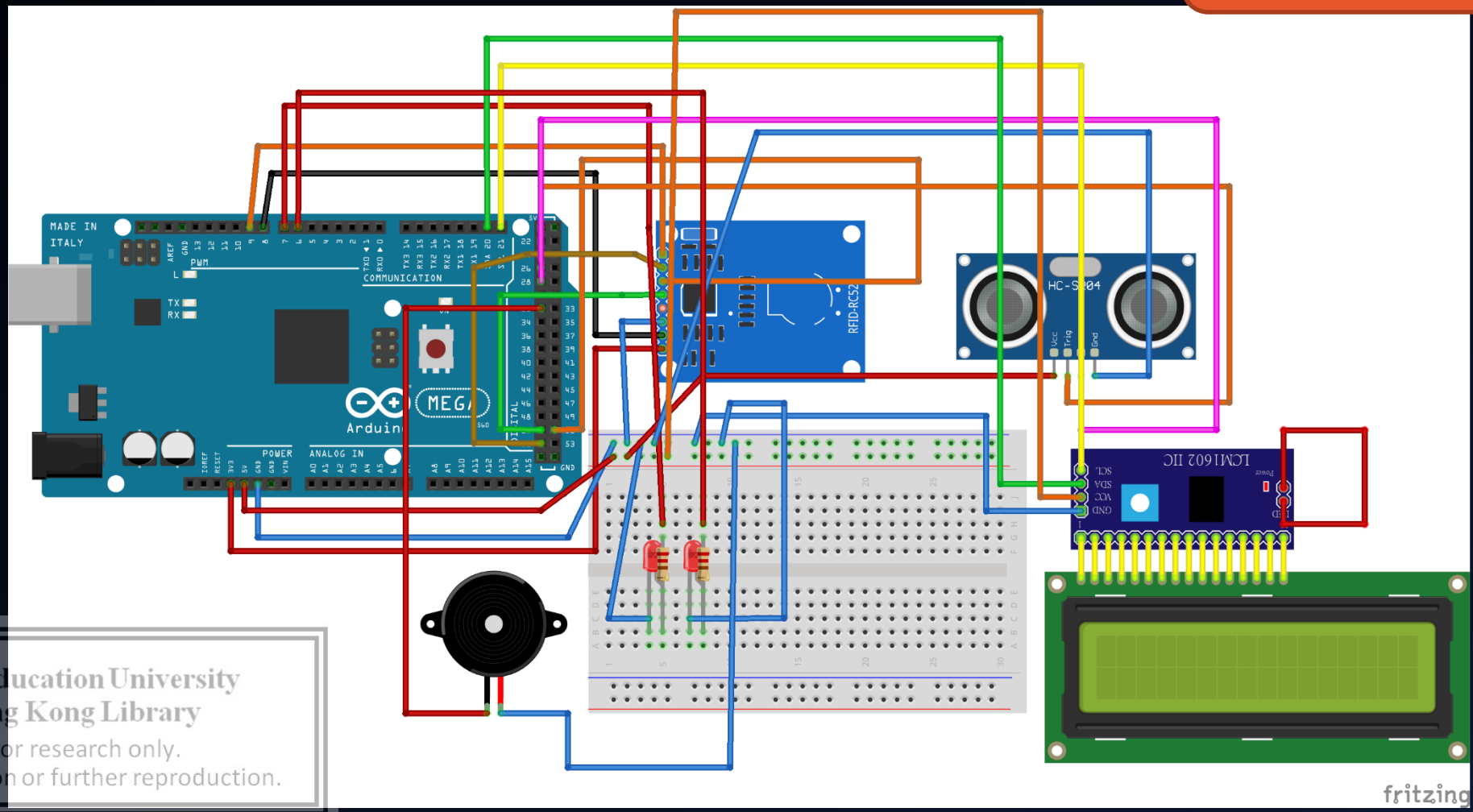
When the system has detected an object (*object distance <= 100cm*), I want the system to emit sound at *fixed pitch*, until a *registered RFID card* is tapped on the RFID reader

I want the object distance of all sensed objects to be *displayed on a LCD*. When the system has detected an object (*object distance <= 100cm*), I want the system to emit sound *at fixed pitch*, until a *registered RFID card* is tapped on the RFID reader

***Client C***

***In the following, we will take "client C" as the example of implementation***

An integration of connections we have done in previous chapter



# Client C example (1)

We use the **same programs** developed in **previous chapters**, with a bit modification to make the example "Client C" to work

```
#include <SPI.h>
#include <RFID.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#define SDA 9
#define RESET 8
int ledPin =  7;
int ledPin2 =  6;
int trigPin = 26;
int echoPin = 28;
long duration, cm;
String uid = "";
String dUid = "1341108723980";
const int buzzerPin = 30;
bool isDetected = false;
```

**Variable initialization** for buzzer, RFID, LEDs, ultrasound sensor and object detection status

```
RFID RC522(SDA, RESET);
LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {
  Serial.begin (9600);
  pinMode(ledPin, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  SPI.begin();
  RC522.init();
  uid = dUid;
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Object Distance:");
}
```

**Initializing installed modules**

# Client C example (2)

We use the **same programs** developed in **previous chapters**, with a bit modification to make the example "Client C" to work

```
void loop() {
  cardDetect();
  if (uid.equals(dUid) == false){
  blinkLED();
  buzzerSiren();
  }
  if (isDetected == false){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  cm = (duration/2) / 29.1;

  lcd.setCursor(0,1);
  lcd.print(cm);
  lcd.print("cm");
  while (cm <= 50){
    int cmDetected = cm;
    isDetected = true;
    uid = "";
    resetUltraSound();
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Possible");
    lcd.setCursor(0,1);
    lcd.print("Intrusion (");  lcd.print(cmDetected);  lcd.
  if (isDetected == false){
  delay(400);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Object Distance:");
  }
```

**To configure** the loop part, so that the system will make LEDs to blink, and to make the buzzer to sound, when **object distance <= 50cm is detected**

Display **distance information** on LCD

Reset the message on LCD, once the alarm system is resetted with RFID card

# Client C example (3)

We use the **same programs** developed in **previous chapters**, with a bit modification to make the example "Client C" to work

```
void blinkLED(){
   digitalWrite(ledPin, HIGH);
   delay(20);
   digitalWrite(ledPin, LOW);
   delay(20);
   digitalWrite(ledPin2, HIGH);
   delay(20);
   digitalWrite(ledPin2, LOW);
   delay(20);
}
```

The method for making 2 LED units to blink

```
void buzzerSiren(){
   // pulse the buzzer on for a short time
  for (int x = 0; x < 50; x++){
  digitalWrite(buzzerPin, HIGH);
  delay(2);
  digitalWrite(buzzerPin, LOW);
  delay(0);
 }
 for (int y = 0; y < 50; y++){
  digitalWrite(buzzerPin, HIGH);
  delay(2);
  digitalWrite(buzzerPin, LOW);
  delay(0);
 }
}
```

The method for making the active buzzer to sound

# Client C example (4)

We use the **same programs** developed in **previous chapters**, with a bit modification to make the example "Client C" to work

```
void cardDetect(){
  if (RC522.isCard())
  {
    uid = "";
    RC522.readCardSerial();
    for(int i=0;i<5;i++)
    {
    uid = uid + String(RC522.serNum[i],DEC);
    }
    if (uid.equals(dUid) == true){
      isDetected = false;
    }
    Serial.print(uid);
    Serial.println();
    Serial.println();
  }
    delay(200);
}
```

The method for detecting & identifying RFID card

```
void resetUltraSound(){
   digitalWrite(trigPin, LOW);
   delayMicroseconds(5);
   digitalWrite(trigPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(trigPin, LOW);
   pinMode(echoPin, INPUT);
   duration = pulseIn(echoPin, HIGH);
   cm = (duration/2) / 29.1;
}
```

The method for resetting the sensor (ultrasound in this example)

# Round-up

- Using the programs you have developed in previous chapters, can you help **customer A & B** building the burglar alarm according to their requirements?

- Apart from the type of sensors you have learnt in these 12 chapters, can you think of some type of sensors which can **further enhance the functionality & security** level of the burglar alarm?