Arduino Course

WEIXIAN COLLEGE

Kun Lin

Email:lin-k23@mails.tsinghua.edu.cn

Part I

INTRODUCTION TO EMBEDDED SYSTEM

Embedded System

Definition

- Embedded means something that is attached to another thing.
- a computer hardware system having software embedded in it
- An embedded system can be an independent system or it can be a part of a large system.
- An embedded system is a microcontroller or microprocessor based system which is designed to perform a specific task.

Example

Embedded system Component

Embedded System Characteristics

- **Single Function**
- **Tightly constrained**

Size, Speed, Power ,Cost

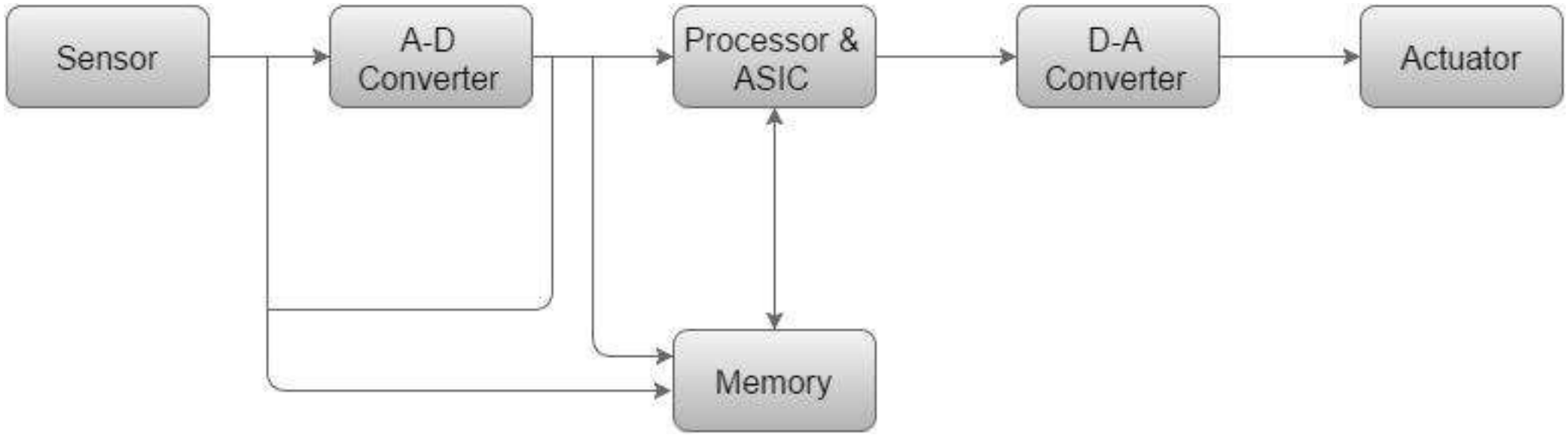
- **Reactive and Real time**
- **Microprocessors based**
- **Memory**
- **Connected**

Connected peripherals to connect input and output devices.

- **HW-SW systems**

Software is used for more features and flexibility. Hardware is used for performance and security.

Basic Structure of Embedded System



Basic Structure of Embedded system

- **Sensor** − It measures the physical quantity and converts it to an electrical signal which can be read by an observer or by any electronic instrument like an A2D converter. A sensor stores the measured quantity to the memory.
- **A-D Converter** − An analog-to-digital converter converts the analog signal sent by the sensor into a digital signal.
- **Processor & ASICs** − Processors process the data to measure the output and store it to the memory.
- **D-A Converter** − A digital-to-analog converter converts the digital data fed by the processor to analog data
- **Actuator** − An actuator compares the output given by the D-A Converter to the actual (expected) output stored in it and stores the approved output.

Part 2

ABOUT ARDUINO ENVIRONMENT

Arduino Environment

Arduino board

- 8-bit microcontroller
- USB programming interface
- I/O pins

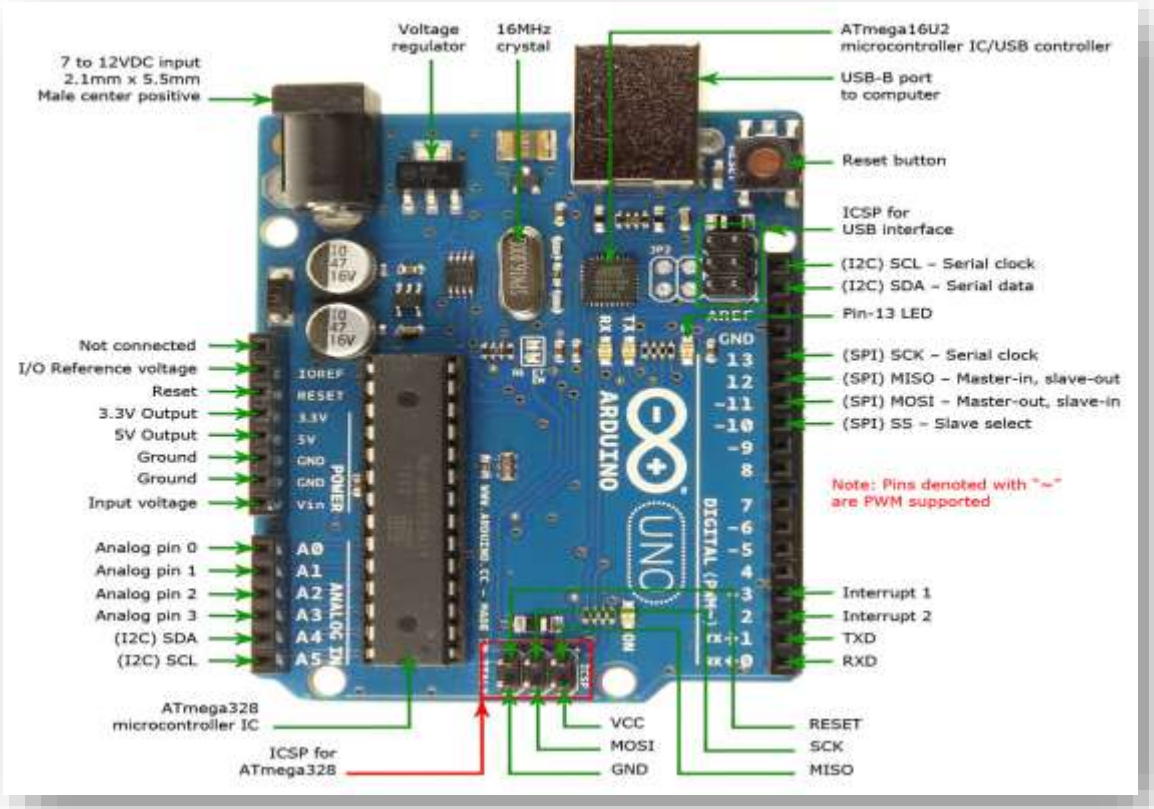
Software IDE

- Code Editor
- Compiler
- Library Manager
- Debugger

Shields

- Unique functionalities
- Easy to attach
- Good libraries provided

Arduino Board(UNO R3)



主要技术参数

微控制器	ATmega328P
工作电压	5伏特
输入电压(推荐)	7-12伏特
输入电压(极限)	6-20伏特
数字输入输出引脚	14个（其中有6个引脚可作为PWM引脚）
PWM引脚	6个
模拟输入引脚	6个
输入/输出引脚直流电流	20 毫安
3.3V引脚电流	50 毫安
Flash Memory(闪存)	32 KB (ATmega328P) 其中由 0.5 KB用于系统引导（bootloader）
SRAM（静态存储器）	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)

Microcontroller

ATmega328	• **Write your code into**
ATmega16U2	• **To communication with USB**

In this course we use **CH340** to replace ATmega16U2

- Operating voltage → 5V
- Input Voltage →7-12V
- Digital I/O pins → 14 pins → 0:13
- Analog input pins → 6 pins → A0:A5
- Flash Memory → 32KB
- Clock speed → 16MHz

Arduino Pins

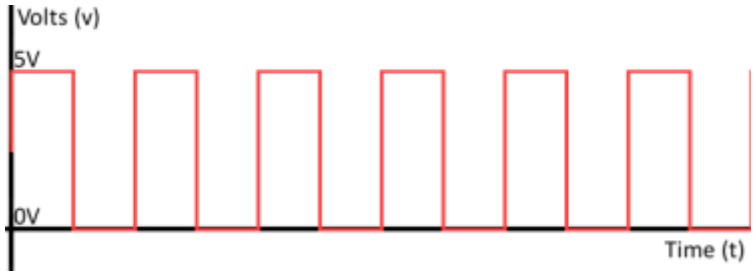
- Pins are wires connected to the micro controller
- Pins are the interface of micro controller
- Pins voltages are controlled by a sketch
- Pins voltages can be read by a sketch

- **Output pins**
  - Voltages is determined by sketch
  - Other components can be controlled through outputs
- **Input pins**
  - Input pins are controlled by other component
  - Arduino read the voltage on the pin allow it to respond to events

Analog Vs Digital

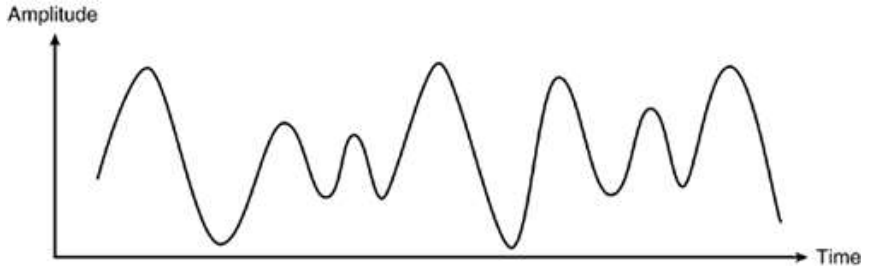
Digital

- Can be **only Two** Value 0volt or 5Volt
- Arduino Has 14 Digital **Input / Output** Pins



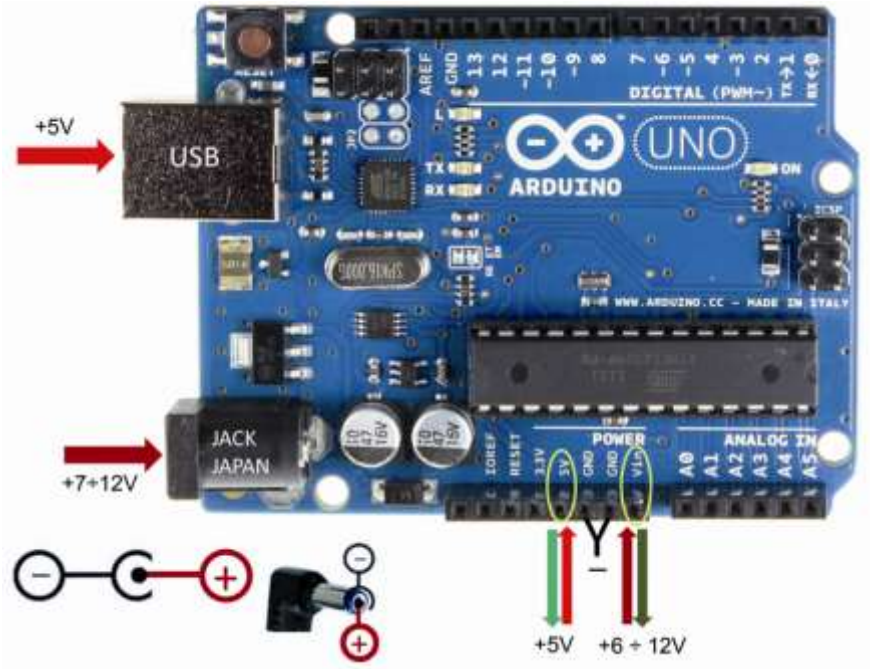
Analog

- Can be **any Value** from 0Volt to 5Volt
- Arduino Has 6 Analog Input Pins(**NO OUTPUT**)

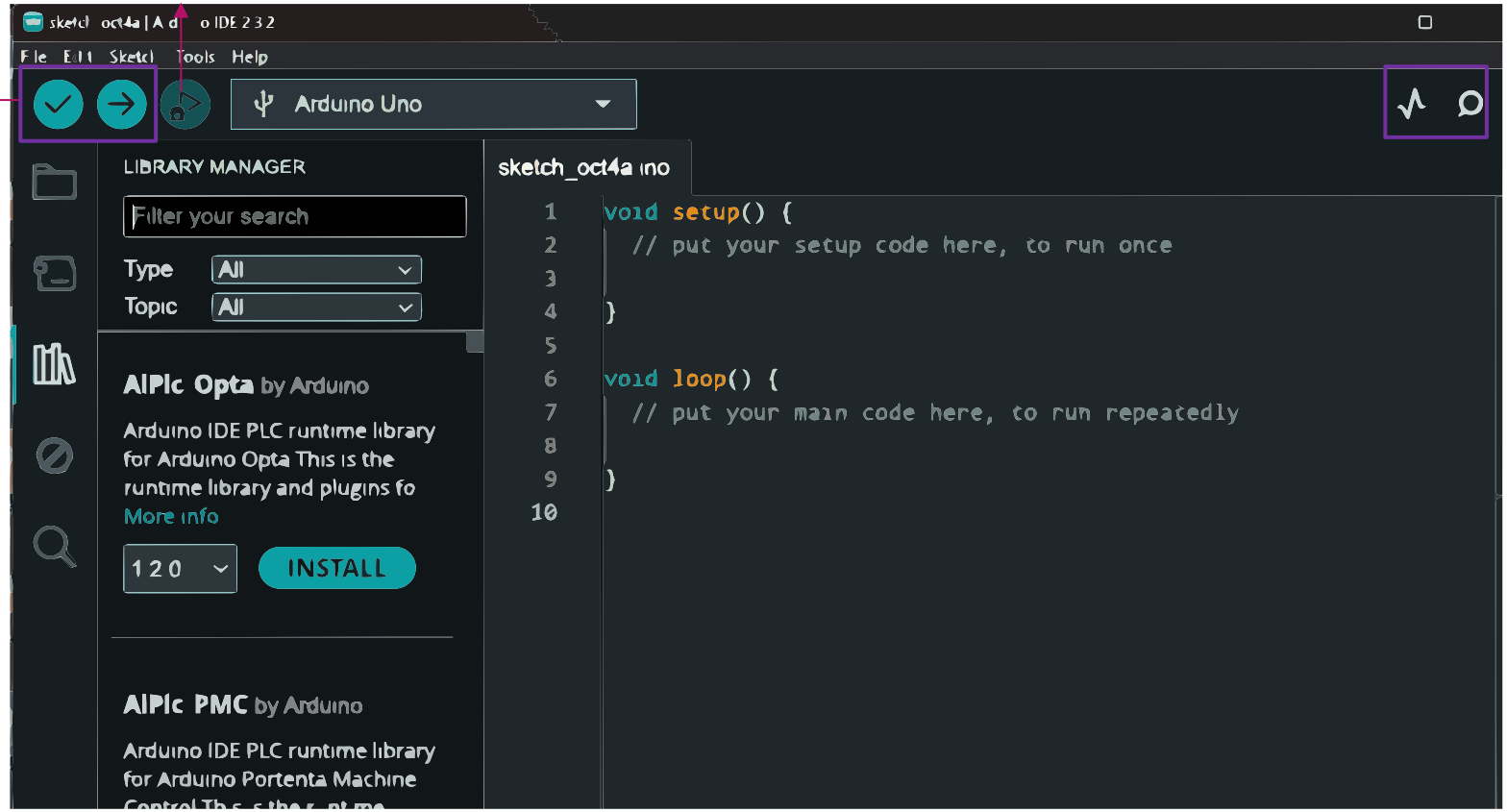


Power

- V_{in} to connect external voltage source
- GND the Ground(0V) of Arduino
- 5V Arduino supply you by 5V
- 3.3V Arduino supply you by 3.3V
- USB connector "to connect Arduino with laptop"
- Power jack "to connect external power source with Arduino
- Rest button "to restart the program"



Arduino IDE


Debug
Verify/Upload
Sketchbook
Board Manager
Library Manager
Debug
Search
Serial Plotter/Monitor
File Edit Sketch Tools Help
Arduino Uno
LIBRARY MANAGER
Filter your search
Type All
Topic All
AIPIc Opta by Arduino
Arduino IDE PLC runtime library for Arduino Opta This is the runtime library and plugins fo
More info
120
INSTALL
AIPIc PMC by Arduino
Arduino IDE PLC runtime library for Arduino Portenta Machine
sketch_oct4a.ino
1 void setup() {
2 // put your setup code here, to run once
3
4 }
5
6 void loop() {
7 // put your main code here, to run repeatedly
8
9 }
10

Setup the Arduino IDE

- Install Arduino app on your PC (Arduino)
- Connect the Board to your PC using USB cable
- Install USB and other drivers (CH340)

- Launch the Arduino app
- Select your Arduino board “Tools >>board>> Arduino UNO
- Select serial port “Tools >>port>>COM X” (**connect to UNO first**)
- Write your code !

Part 3

SKIP

INTRODUCTION TO COMPUTER PROGRAMMING

Basics symbol

Symbol	Operation
; “semi Colom”	Every line of code must ended by it
{ }	Group line of code Every function must be start and ended by it
/*…….*/ or //…	To write comments between it

Variables

Variables Name Conditions

- A sequence of visible character
- Must start with a non-numerical character
- No C language keywords

Data Types Size

Type	Keyword	Value range which can be represented by this data type
Character	char	-128 to 127 or 0 to 255
Number	int	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
Small Number	short	-32,768 to 32,767
Long Number	long	-2,147,483,648 to 2,147,483,647
Decimal Number	float	1.2E-38 to 3.4E+38 till 6 decimal places

Variable Scope

Local Variable

Variables that are declared inside a function or block are local variables. They can be used only by the statements that are inside that function or block of code.

```
Void setup () {

}

Void loop () {
   int x , y ;
   int z ; Local variable declaration
   x = 0;
   y = 0; actual initialization
   z = 10;
}
```

Formal Parameter

can be declared In the definition of function parameters

Global Variables

Global variables are defined outside of all the functions, usually at the top of the program.

```
Int T , S ;
float c = 0 ; Global variable declaration

Void setup () {

}

Void loop () {
   int x , y ;
   int z ; Local variable declaration
   x = 0;
   y = 0; actual initialization
   z = 10;
}
```

C – operator

- Arithmetic Operators
- Comparison Operators
- Boolean Operators
- Bitwise Operators
- Compound Operators

Arithmetic Operators

Operator name	Operator simple	Description
assignment operator	=	Stores the value to the right of the equal sign in the variable to the left of the equal sign.
addition	+	Adds two operands
subtraction	-	Subtracts second operand from the first
multiplication	*	Multiply both operands
division	/	Divide numerator by denominator
modulo	%	Modulus Operator and remainder of after an integer division

Comparison Operators

Operator name	Operator simple	Description
equal to	==	Checks if the value of two operands is equal or not, if yes then condition becomes true.
not equal to	!=	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.
less than	<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.
greater than	>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.
less than or equal to	<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.
greater than or equal to	>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.

Boolean Operators

Operator name	Operator simple	Description
and	&&	Called Logical AND operator. If both the operands are non-zero then then condition becomes true.
or	||	Called Logical OR Operator. If any of the two operands is non-zero then then condition becomes true.
not	!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.

Bitwise Operators

Operator name	Operator simple	Description
and	&	Binary AND Operator copies a bit to the result if it exists in both operands.
or	|	Binary OR Operator copies a bit if it exists in either operand
xor	^	Binary XOR Operator copies the bit if it is set in one operand but not both.
not	~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.
shift left	<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.
shift right	>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.

Compound Operators

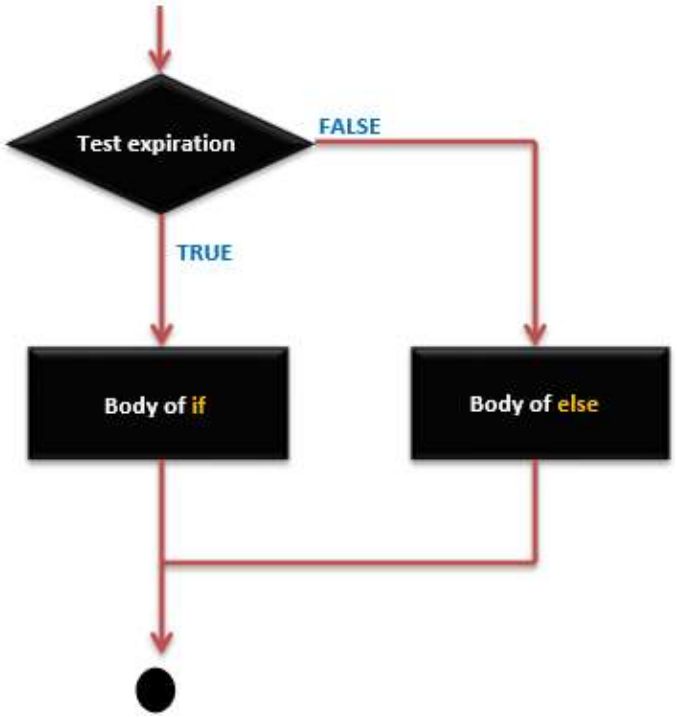
Operator name	Operator simple	Description	Example
increment	++	Increment operator, increases integer value by one	A++ will give 11
decrement	--	Decrement operator, decreases integer value by one	A-- will give 9
compound addition	+=	Add AND assignment operator. It adds right operand to the left operand and assign the result to left operand	B += A is equivalent to B = B+ A
compound subtraction	-=	Subtract AND assignment operator. It subtracts right operand from the left operand and assign the result to left operand	B -= A is equivalent to B = B - A
compound multiplication	*=	Multiply AND assignment operator. It multiplies right operand with the left operand and assign the result to left operand	B*= A is equivalent to B = B* A
compound division	/=	Divide AND assignment operator. It divides left operand with the right operand and assign the result to left operand	B /= A is equivalent to B = B / A

Control Statements

- If statements
- If-else Statements
- If-else if-else statement
- Switch statement

If statement

```
if( expression )
{
Statement 1
}
else
{
statement 2
}
```



If –else if - else

if(expression1)

{

Statement 1

}

else if (expression2)

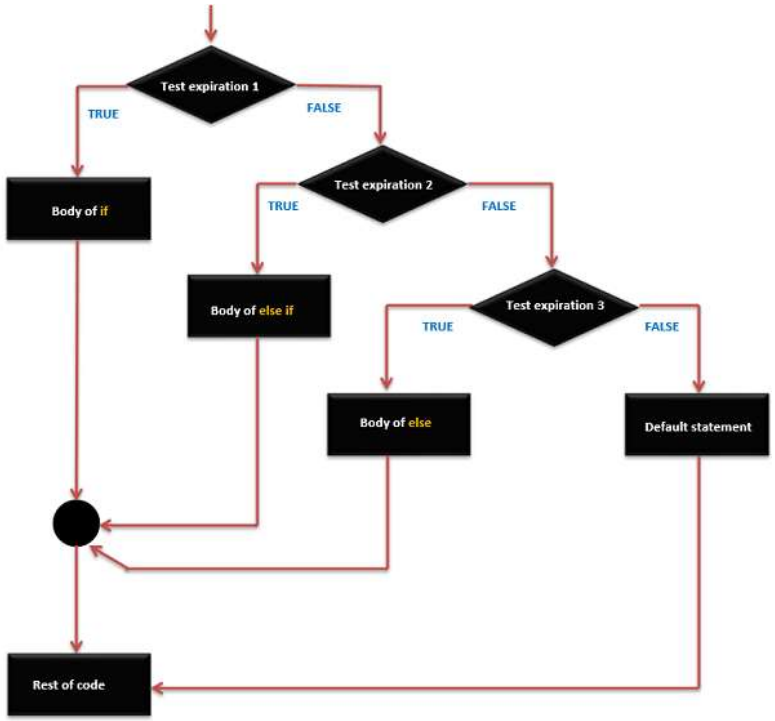
{

statement 2

}

else

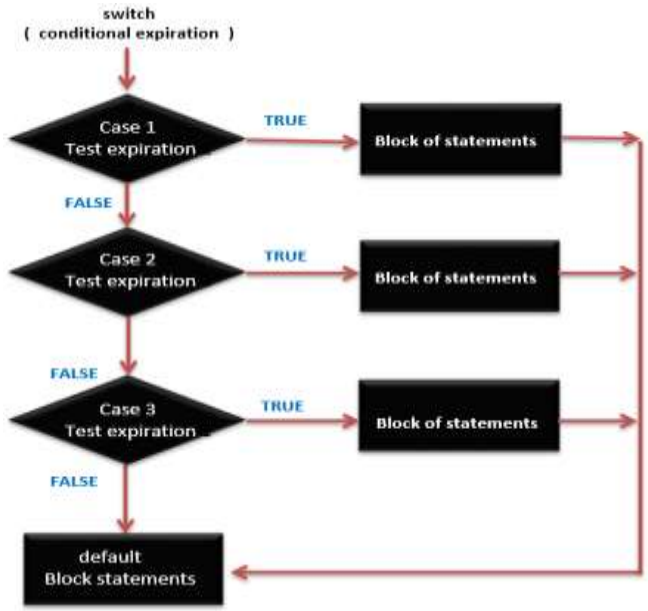
{ statement3 }



Switch statement

```
switch (expression )
{
 case ‘const_expr1’ : stat1
 break ;
 case ‘const_expr2’ : stat2
 break ;
 default : stat3
}
```

- Compere each case with expression and operate the case equal to expression
- **Default** → operating when no case is matching
- **Break** → use to stop operation



Loop Statements

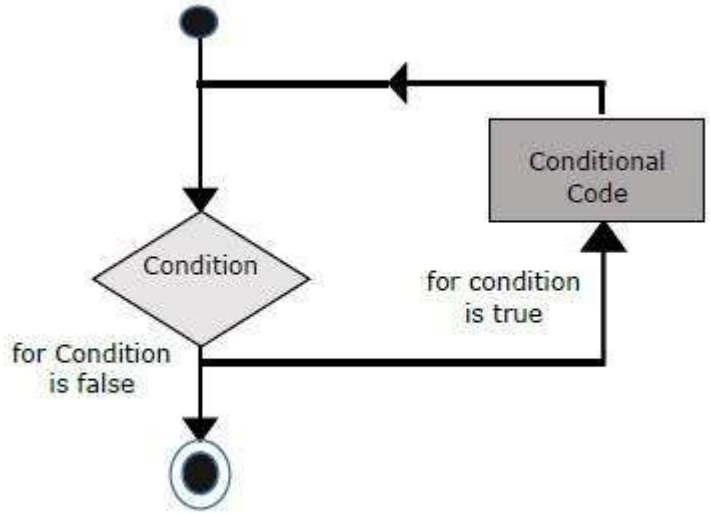
- For Loop
- While Loop
- Do-while loop
- Nested Loop
- Infinite Loop

For statement

for (**expr1; expr2; expr3;**)

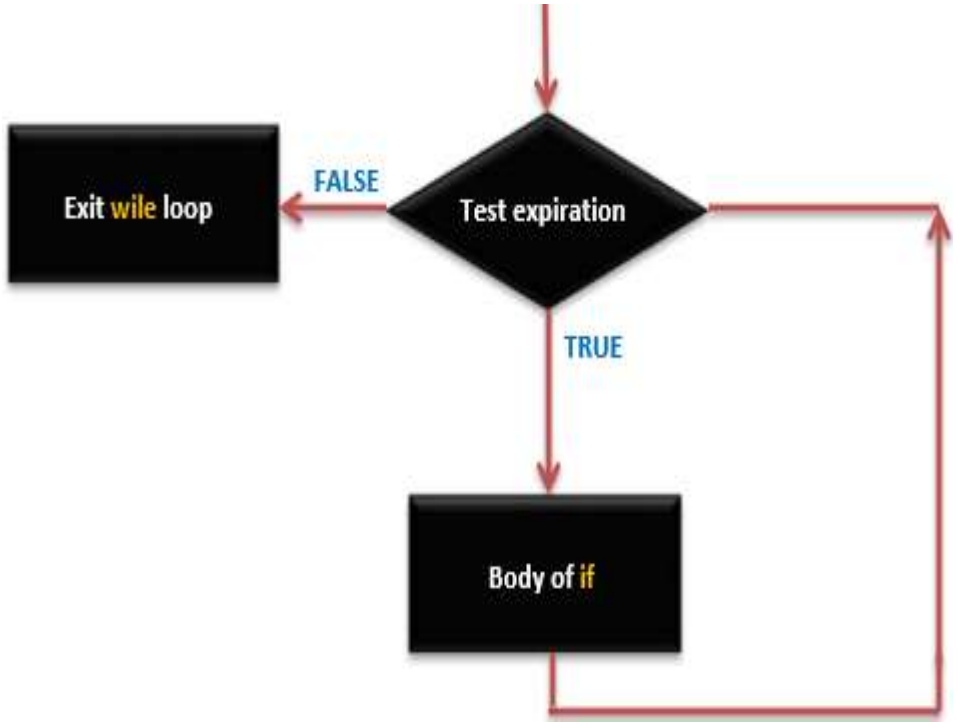
{ statement }

- expr1→ initialization
- expr2→ termination
- expr3→ step



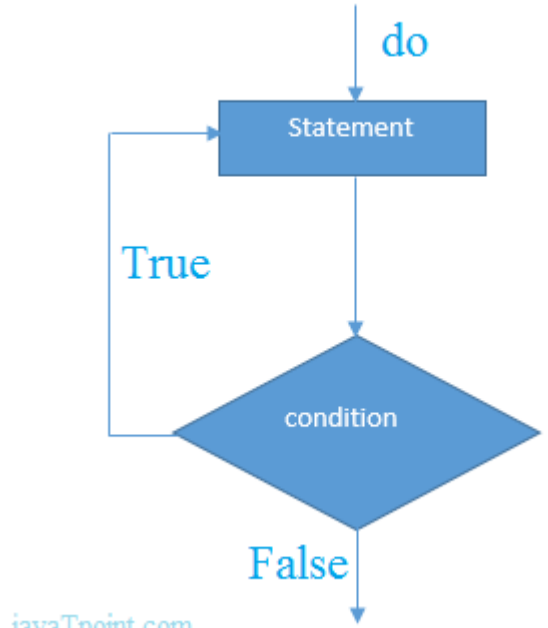
While Loop

```
while (expression)
{
Statements
}
```



Do-while Loop

```
do {
    Block of statements;
}
while (expression);
```



Nested Loop

```
for ( initialize ;control; increment or decrement) {
   // statement block
   for ( initialize ;control; increment or decrement) {
      // statement block
   }
}
```

- use one loop inside another loop.

Infinite Loop

Using for loop

```
for (;;) {
   // statement block
}
```

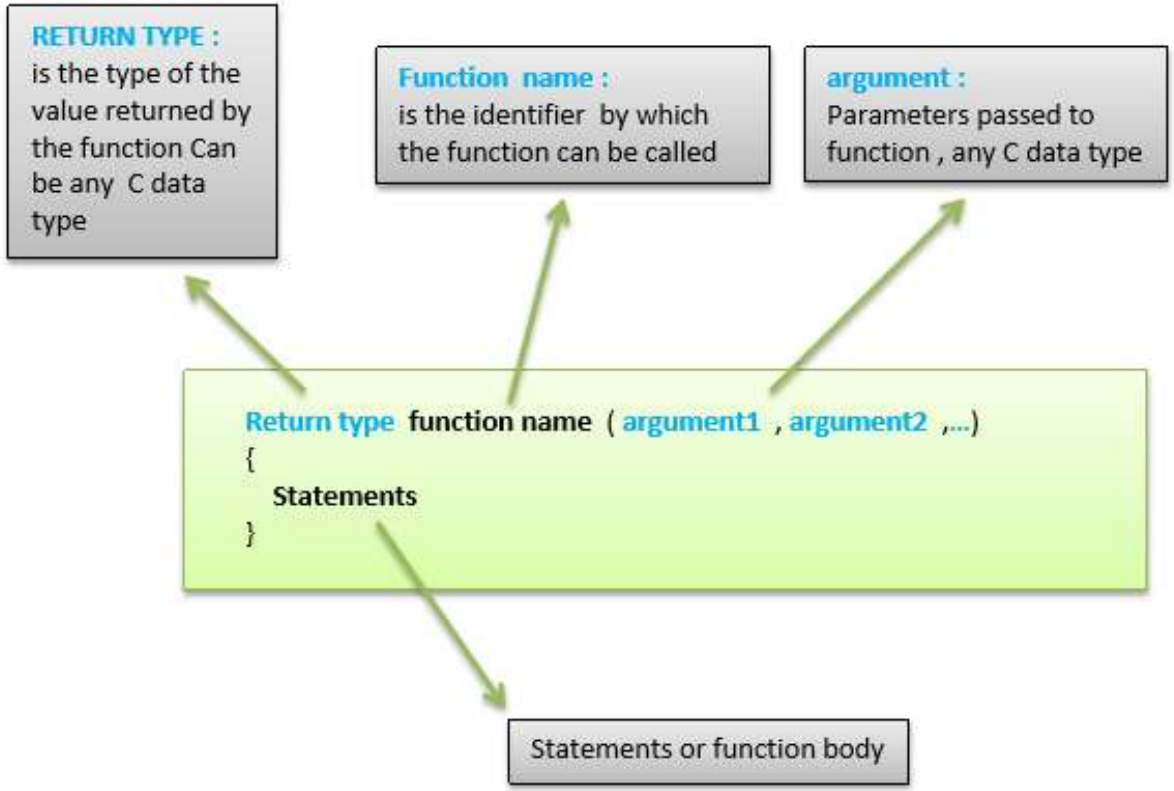
Using while loop

```
while(1) {
   // statement block
}
```

Using do...while loop

```
do {
   Block of statements;
}
while(1);
```

Functions


RETURN TYPE :
is the type of the value returned by the function Can be any C data type
Function name :
is the identifier by which the function can be called
argument :
Parameters passed to function , any C data type
Return type function name (argument1 , argument2 ,...)
{
Statements
}
Statements or function body

Part 4

ARDUINO C

Setup () Function

- Use for primary order
- Start by powering Arduino

```
void setup ()
{
arguments
}
```

Loop() function

- Work as long as Arduino work
- Start after setup function end
- The main code written in it

```
void loop()
{
Arguments
}
```

Pin mode function

- These function allow access to the pins
- Set a pin to act as an INPUT or OUTOUT

pinMode(pin , mode)

Pin

- Is the number of the pin **0:13** for **digital**
- analog **A0:A5** for **analog** pins

Mode

- Is the I/O mode the pin is set to "**INPUT**" "**OUTPUT**"

Digital output function

- Assign the state of an output pin
- Assign either LOW or HIGH

digitalWrite(pin , Value)

Value

- HIGH
- LOW

Digital input function

- Returns the state of an input pin
- Returns either LOW or HIGH

digitalRead(pin)

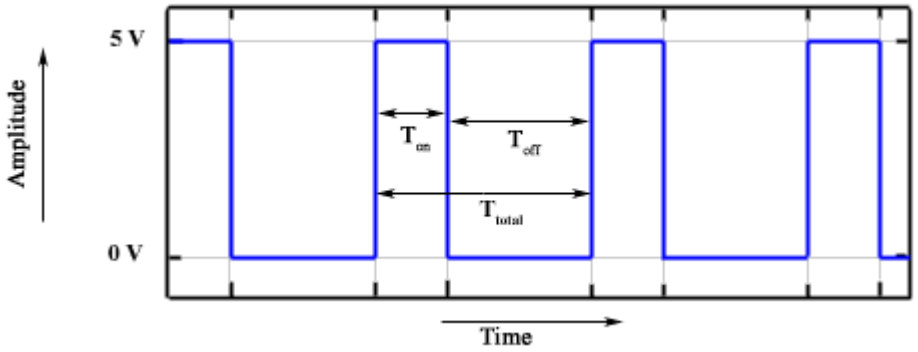
Analog input function

- Returns the state of an analog pin
- Returns an integer number from 0 to 1023
- 0 for 0Volt 1023 for 5Volts

analogRead(pin)

Pulse Width Modulation

- **On-Time** − Duration of time signal is high.
- **Off-Time** − Duration of time signal is low.
- **Period** − It is represented as the sum of on-time and off-time of PWM signal.
- **Duty Cycle** − It is represented as the percentage of time signal that remains on during the period of the PWM signal.



$$T_{total} = T_{on} + T_{off}$$

$$D = \frac{T_{on}}{T_{on} + T_{off}} = \frac{T_{on}}{T_{total}}$$

Analog Output Function

- writes an analog value (PWM wave) to a pin
- this function works on pins 3, 5, 6, 9, 10, and 11.
- Values from 0 to 255
- 0 mean Zero Volt
- 255 mean 5 Volt

analogWrite(pin,value)

Delay function

- Pauses the program for milliseconds
- Useful for human interaction
  - **delay(m sec)**

Delay Microseconds()

- The same function of normal “delay” but
- Normal delay measure in mille second
- This function measure in micro second

delayMicroseconds(time);

pulseIn()

- We use this function to calculate time from Arduino call it to end of code
- Frist argument is the pin number
- Second argument is the pulse level we want to detect

pulseIn (Pin, State, TimeOut(optional));

Serial Communication

- Used for communication between the Arduino board and a computer or other devices.
- All Arduino boards have at least one serial port
- It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB.
- Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.
- SoftwareSerial can avoid using pin0 and pin1

Serial functions

- **Serial . begin(Baud Rate)**
- To start communication with other devices
- Sets the data rate in bits per second (baud) for serial data transmission
- Used in **setup()**
- **Serial . print(Value, Format)**
- To show data received from Arduino on serial monitor on your PC .
- Format is optional Parameter [BIN, OCT, HEX, DEC]
- Printed Data shown in the same line
- Used in **loop()**

Serial Communication

- **Serial.available()**
- Get the number of bytes available for reading from the serial port.
- Data Returned stored in Serial Buffer Register

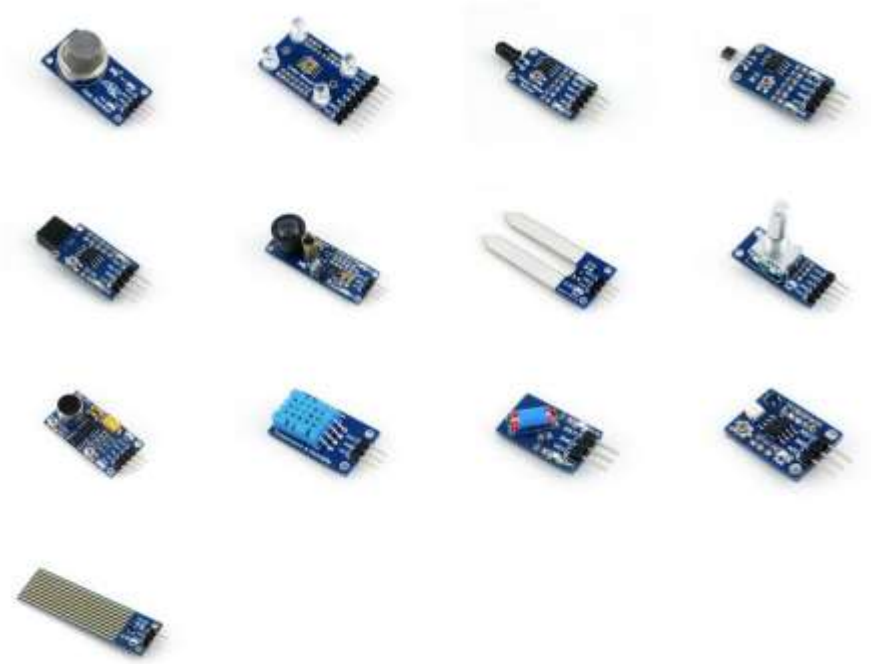
- **Serial.read()**
- Reads incoming serial data
- Return Data

Part 5

SENSORS & ACTUATORS
AND EXPERIMENT!

Sensors

- How bright is it ?
- How loud is it ?
- How humid is it ?
- How distance is it ?
- Is the button being pressed ?
- Is there motion ?

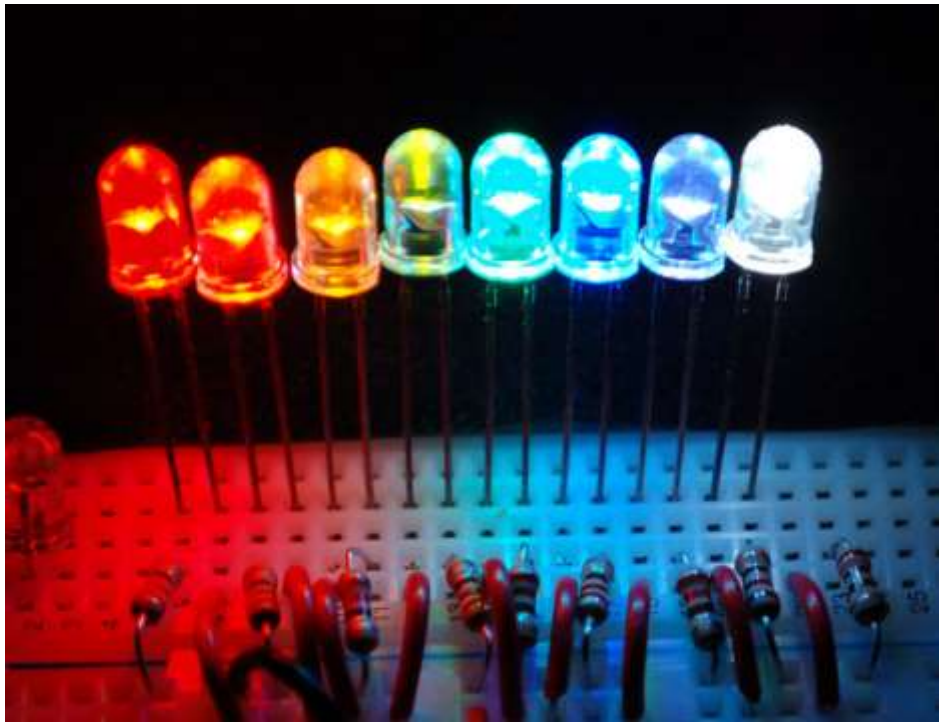


Sensors

- Allow the microcontroller **receive** info. From the environments
- Perform operation **based** on the state of environment
- Microcontroller sense **voltage only**
- Sensors logic must **convert** an environmental effect into voltage

Actuators

- Devices that cause something to happen in the physical world
- Visual → LEDs , LCDs
- Audio → Buzzer , Speaker
- Motion → Motors

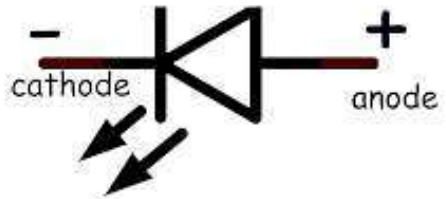


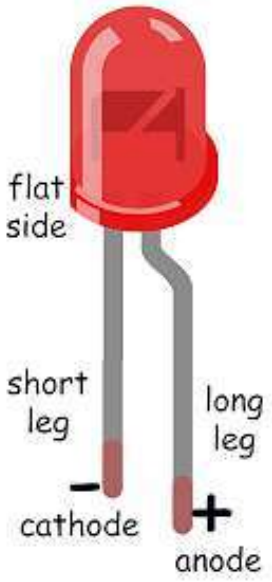
Basic Electronic Component

- **LED**
- **Button**
- LDR
- Buzzer
- …

LED

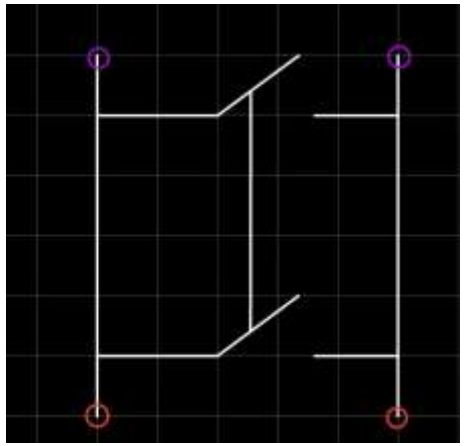
- Two terminal **anode +** the **long** terminal
  cathode – the **short** terminal
- Current only flow in one direction **+ to –**
- Anode – cathode voltage must be above threshold (e.g. RED: 1.8V)
- LEDs have a **maximum current limit**
- Don't connect LEDs directly with power supply (Use **resistor**)

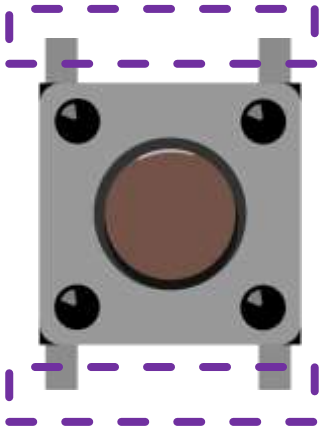




Push button

- Closing switch complete circuit
- Voltage on both terminal is identical when switch is closed

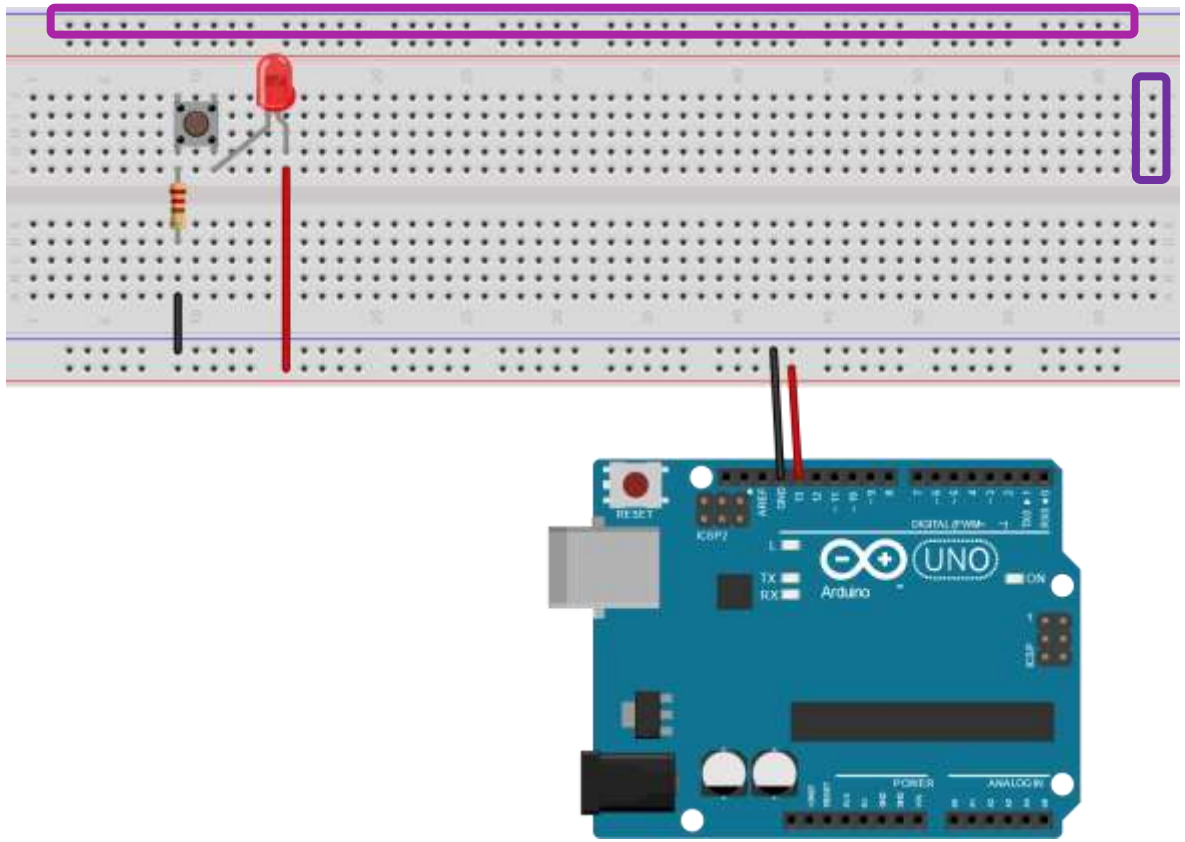




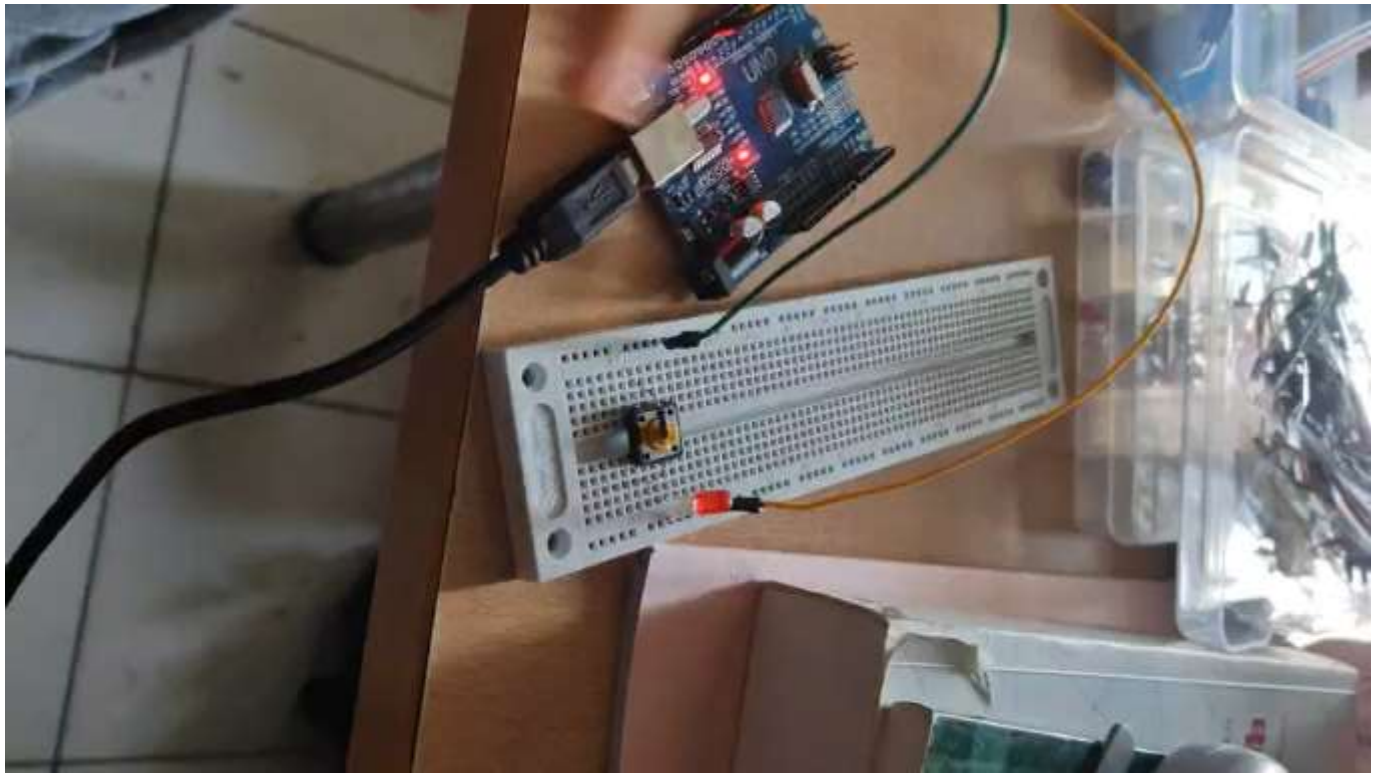
EXP. 1: Light up your LED!

1_LED_blink.ino

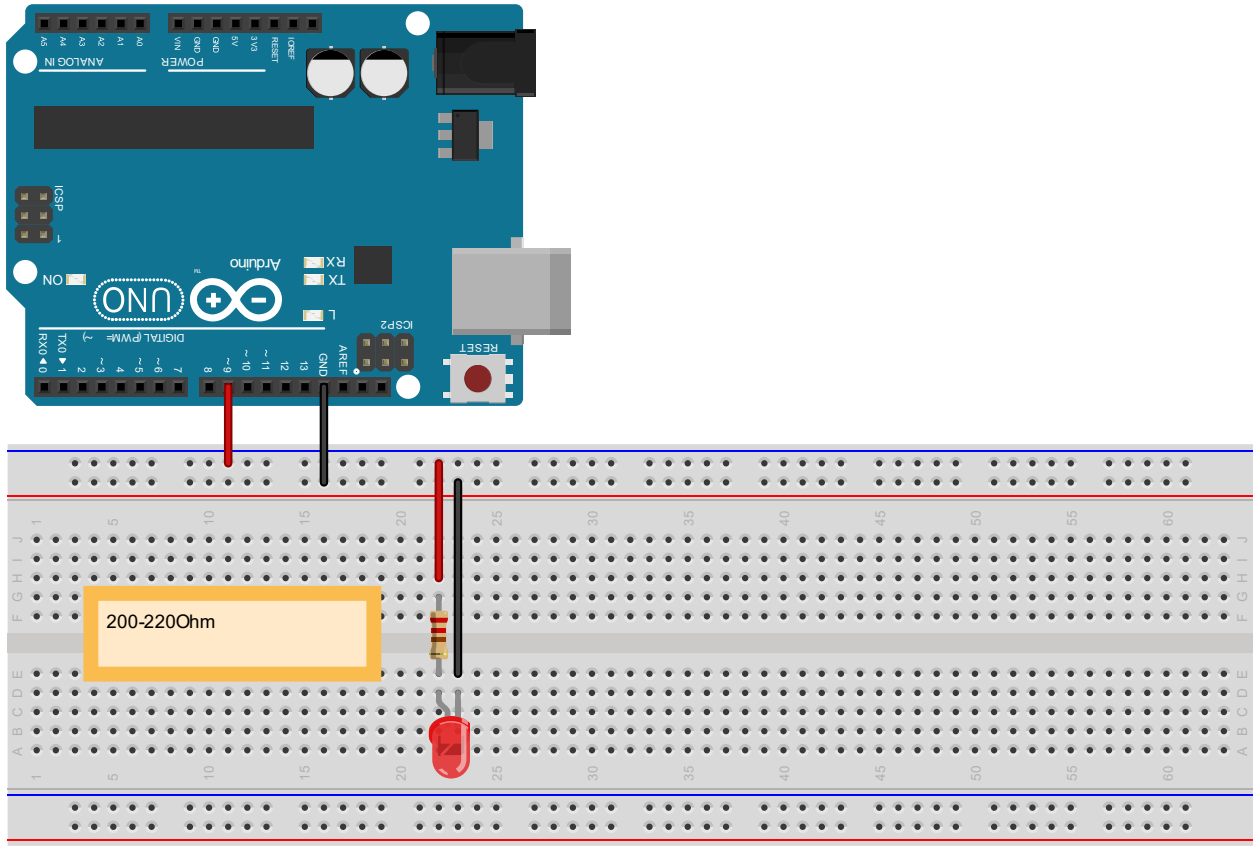
```
1   void setup() {
2     pinMode(13, OUTPUT);
3     // '13' mark the pin you choose
4     //'OUTPUT' is the pin mode
5   }
6   // the loop function runs over and over again forever
7   void loop() {
8     // set LED blink at a certain frequency
9     digitalWrite(13, HIGH);
10    delay(250);
11    digitalWrite(13, LOW);
12    delay(250);
13  }
14
```



EXP. 1: Light up your LED!



EXP. 2: Breathing LED (PWM)

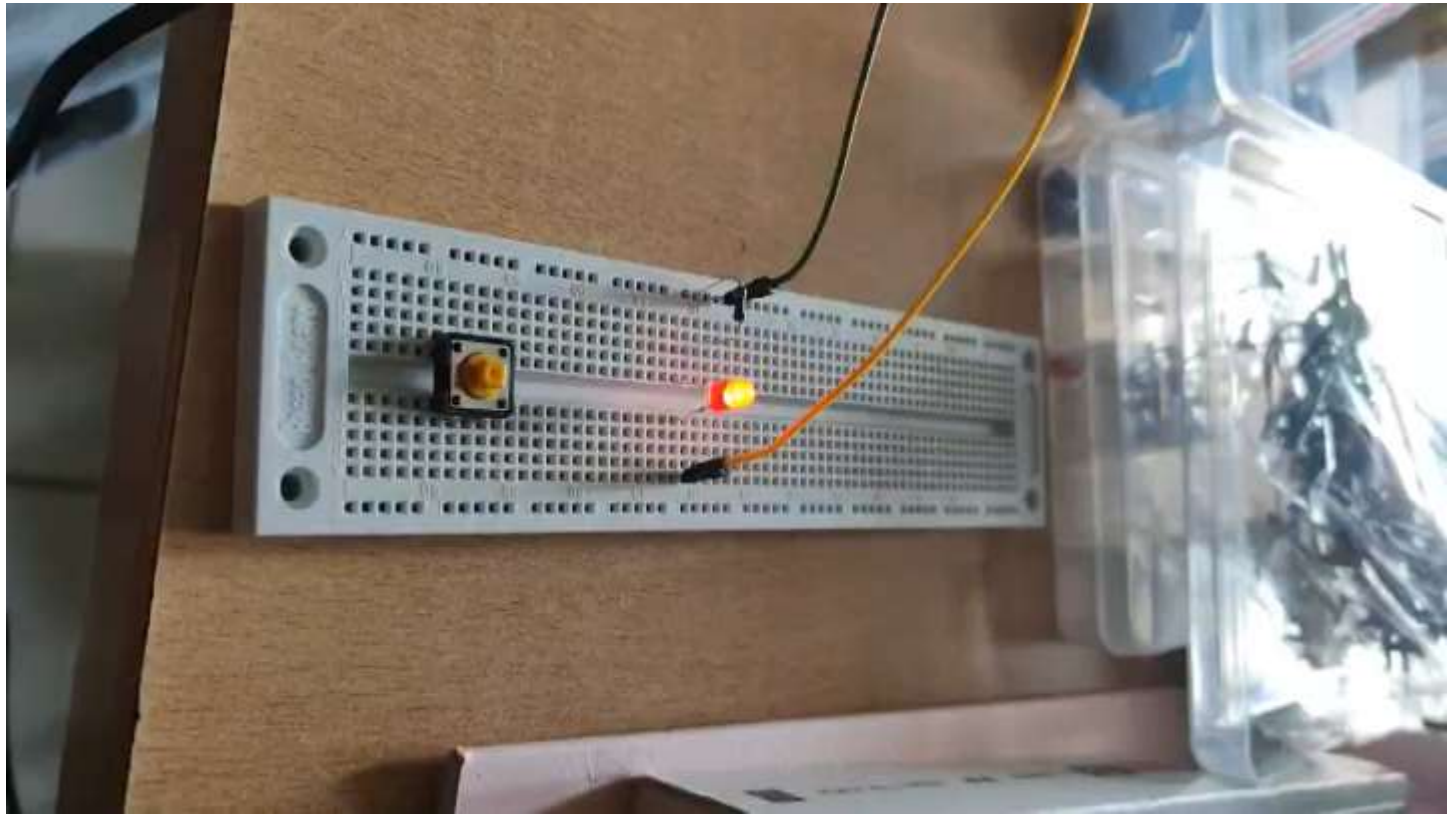


2_breathing_LED.ino

```
1  // 定义LED连接的引脚
2  const int ledPin = 9;  // 使用PWM的引脚
3  // 定义亮度和变化量
4  int brightness = 0;    // LED的当前亮度
5  int fadeAmount = 3;    // 每次循环改变的亮度量
6  void setup() {
7    Serial.begin(9600);
8    // 初始化 LED 引脚为输出
9    pinMode(ledPin, OUTPUT);
10 }
11 void loop() {
12   // 设置LED的亮度
13   analogWrite(ledPin, brightness);
14   // 改变亮度
15   brightness = brightness + fadeAmount;
16   // 当亮度到达极限时反转亮度变化方向
17   if (brightness <= 0 || brightness >= 255) {
18     fadeAmount = -fadeAmount;  // 反转变化方向
19   }
20   Serial.println(brightness);
21   // 控制呼吸的速度
22   delay(30);  // 调整这个值来改变呼吸的速度
23 }
24
```

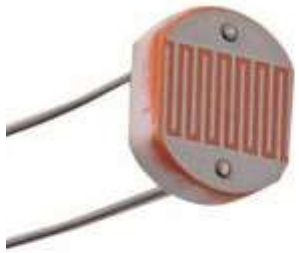
PWM Output

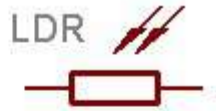
EXP. 2: Breathing LED (PWM)

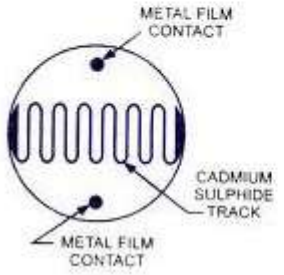


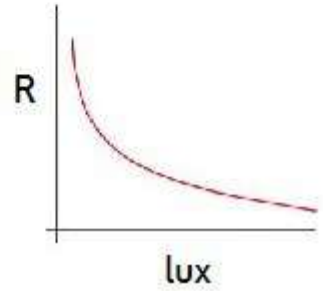
Light Depended Resistor “ LDR”

- Two terminal device
- Analog sensor
- In darkness resistance increase
- In brightness resistance decrease



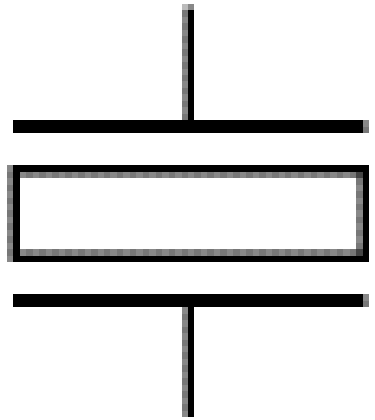






Buzzer

- Two input : signal / ground
- Produces a sound when applying voltage

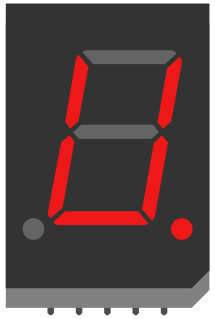


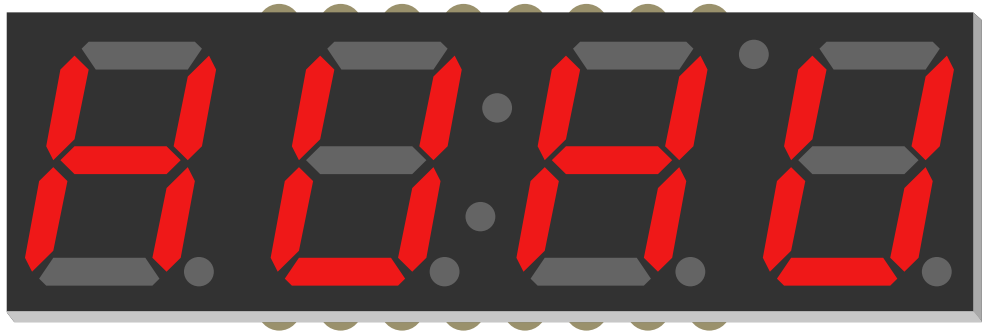
Magic Modules

- **Display Module**
- **Ultrasonic HC-SR04**
- **Bluetooth Module HC-05**
- Relay Module
- Motors
- …

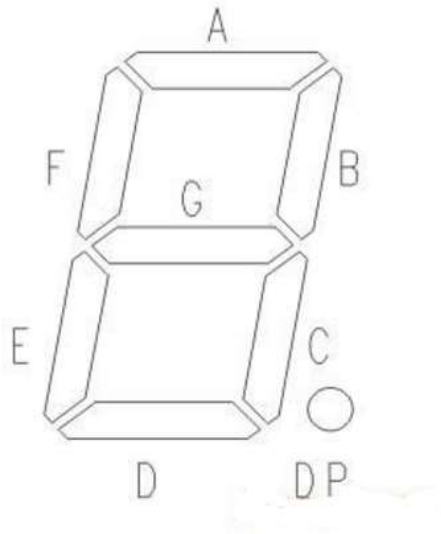
Digital Tube

- Has it's own Library to use with Arduino (SevSeg)
- Multiplexing
- Persistence of Vision



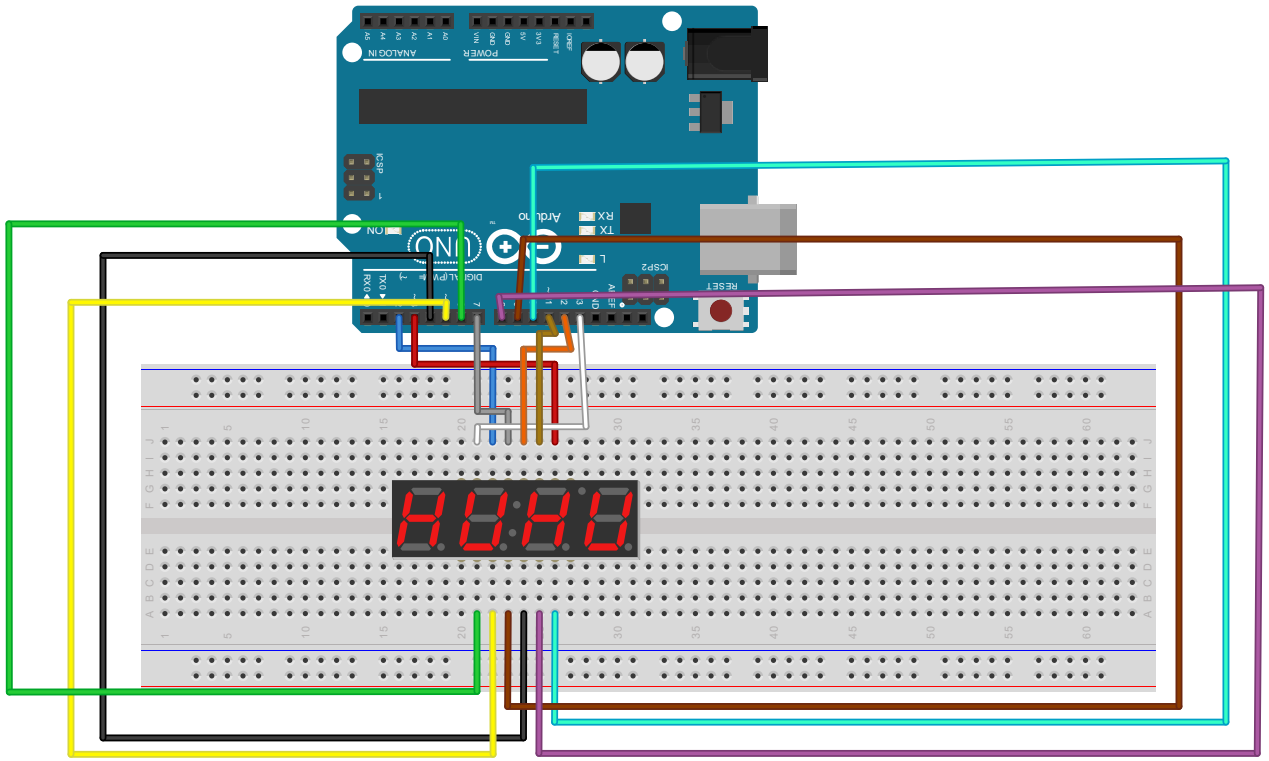


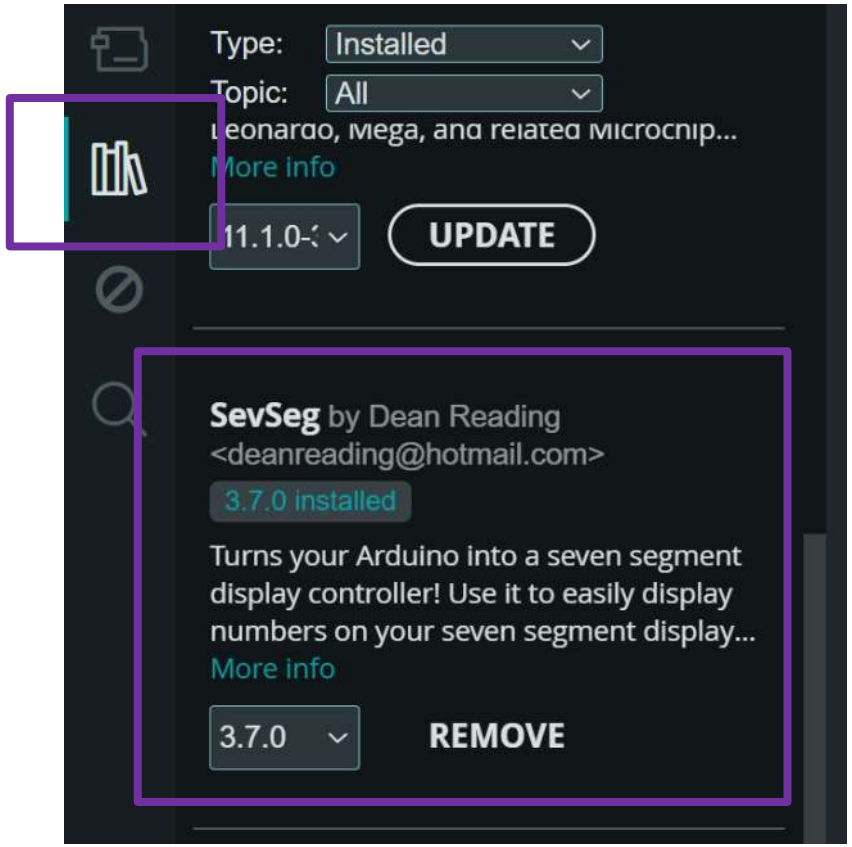
EXP. 3: Shining Moments: 4-digit Timer



EXP. 3: Shining Moments: 4-digit Timer

⚠ It is best to use **protection resistor**.





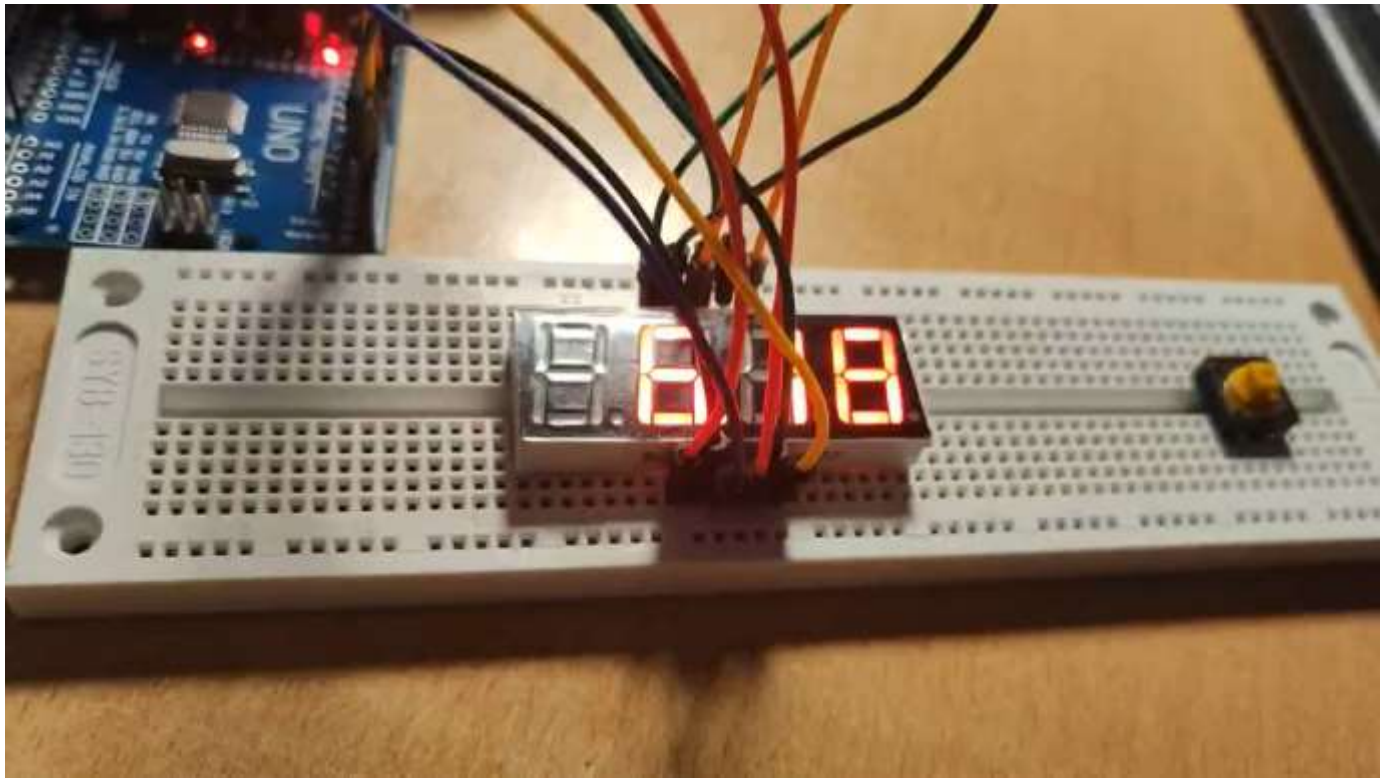
EXP. 3: Shining Moments: 4-digit Timer

5_digital_tube.ino

```ino
#include "SevSeg.h"
//Instantiate a seven segment controller object
SevSeg sevseg;
void setup() {
  byte numDigits = 4;
  byte digitPins[] = { 13, 12, 11, 10 };
  byte segmentPins[] = { 2, 3, 4, 5, 6, 7, 8, 9 };
  byte hardwareConfig = COMMON_ANODE;  //共阳极
  bool leadingZeros = true;
  sevseg.begin(hardwareConfig, numDigits,
               digitPins, segmentPins, leadingZeros);
  sevseg.setBrightness(0);
}
```

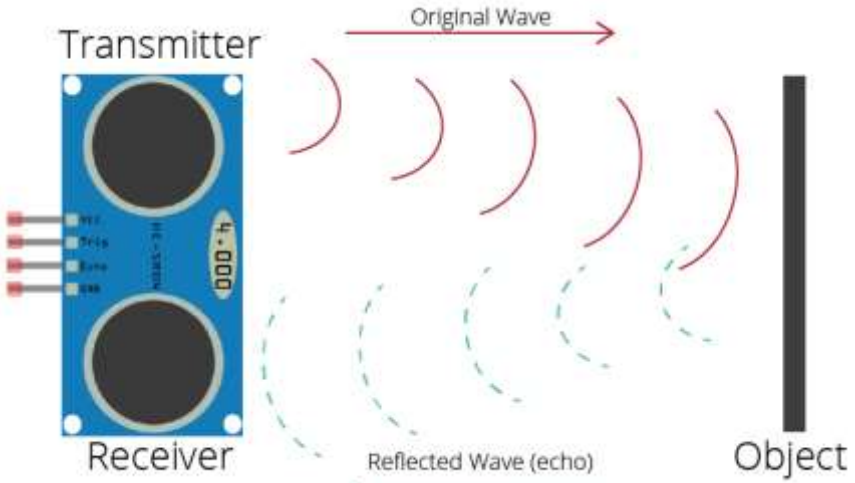
```ino
void loop() {
  static unsigned long timer = millis();
  static int deciSeconds = 0;
  if (millis() - timer >= 10) {
    timer += 10;
    deciSeconds++;
    if (deciSeconds == 2000) {
      // Reset to 0 after counting for 20 seconds.
      deciSeconds = 0;
    }
    sevseg.setNumber(deciSeconds, 2);
  }
  // Must run repeatedly
  sevseg.refreshDisplay();
}
```

EXP. 3: Shining Moments: 4-digit Timer



Ultrasonic

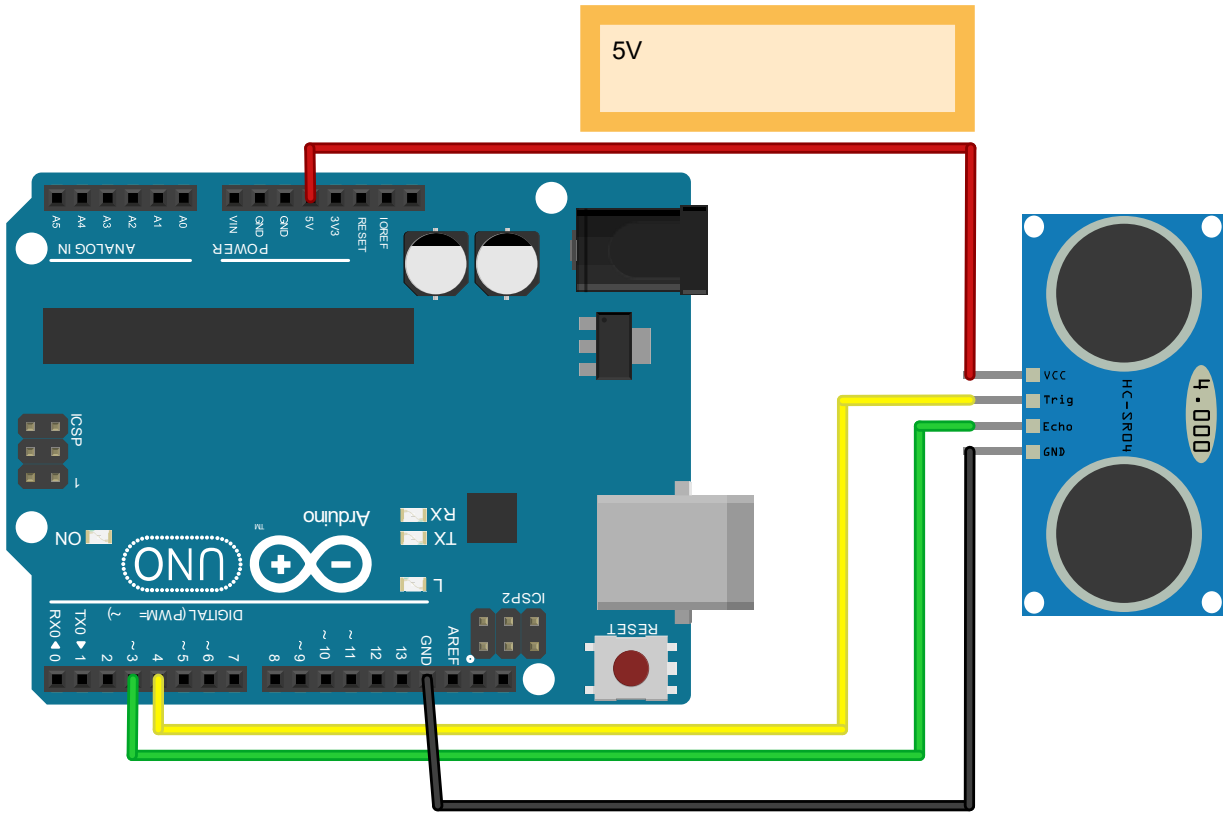
- 4 terminal device
- Vcc → 5V
- GND → 0V
- Trig → output
- Echo→ input



- To calculate distance in cm use the following equation

Time in micro second /58 = distance in cm

EXP. 4: Exploring Distance with HCSR04



EXP. 4: Exploring Distance with HCSR04

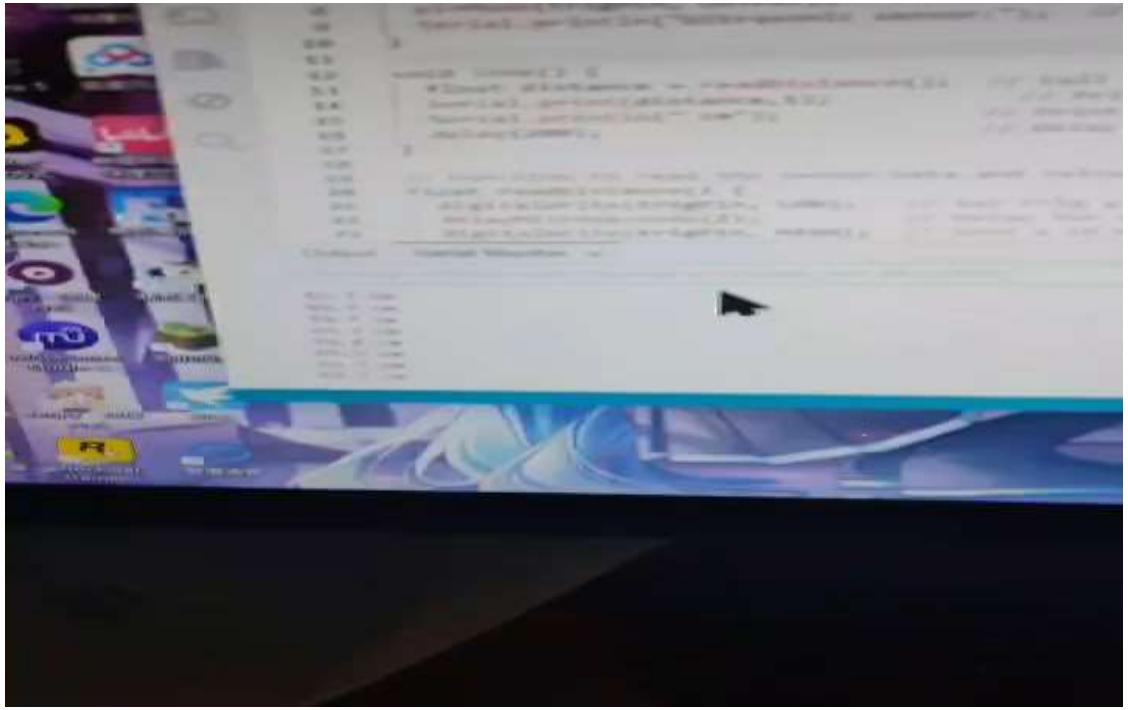
3_ultrasonic_sensor.ino

```
1   // Define the pin numbers for the ultrasonic sensor
2   const int echoPin = 3;
3   const int trigPin = 4;
4
5   void setup() {
6     Serial.begin(9600);                    // Start serial communication with a baud rate of 9600
7     pinMode(echoPin, INPUT);               // Set echo pin as input
8     pinMode(trigPin, OUTPUT);              // Set trig pin as output
9     Serial.println("Ultrasonic sensor:");  // Print a message indicating the ultrasonic sensor is ready
10  }
11
12  void loop() {
13    float distance = readDistance();  // Call the function to read the sensor data and get the distance
14    Serial.print(distance,1);           // Print the distance value
15    Serial.println(" cm");             // Print " cm" to indicate the unit of measurement
16    delay(200);                        // Delay for 400 milliseconds before repeating the loop
17  }
18
```

EXP. 4: Exploring Distance with HCSR04

```
18
19  // Function to read the sensor data and calculate the distance
20  float readDistance() {
21    digitalWrite(trigPin, LOW);   // Set trig pin to low to ensure a clean pulse
22    delayMicroseconds(2);         // Delay for 2 microseconds
23    digitalWrite(trigPin, HIGH);  // Send a 10 microsecond pulse by setting trig pin to high
24    delayMicroseconds(10);
25    digitalWrite(trigPin, LOW);  // Set trig pin back to low
26
27    // Measure the pulse width of the echo pin and calculate the distance value
28    float distance = pulseIn(echoPin, HIGH) / 58.00;  // Formula: (340m/s * 1us) / 2
29    return distance;
30  }
31
```

EXP. 4: Exploring Distance with HCSR04

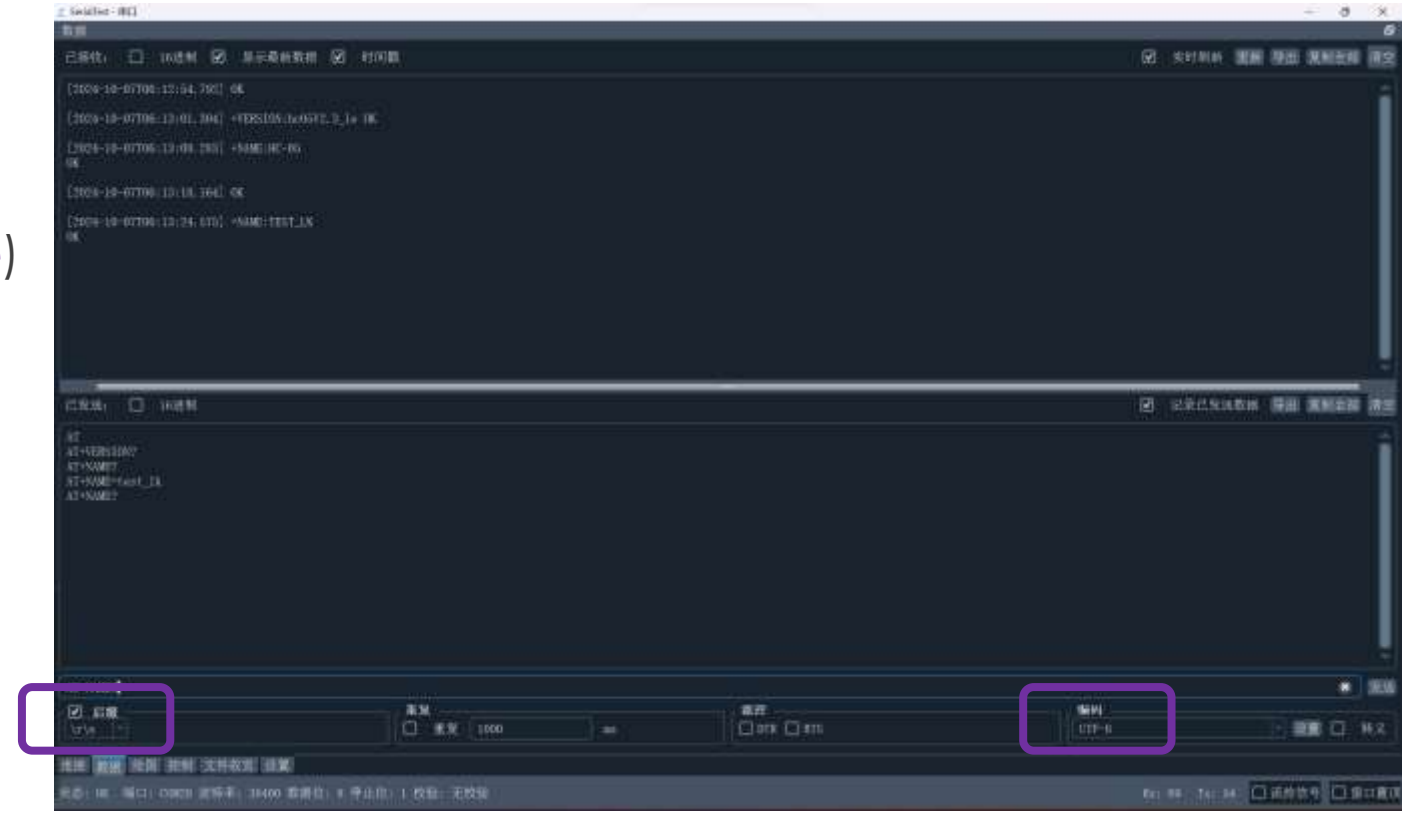


Bluetooth Module

- Use Serial Communication
- **TX** in Bluetooth connected to **RX** with **Arduino** or **USB to TTL** and Vise Versa
- Two Modes
- **Commands Mode**: use AT instructions to set parameters
- **Data Mode**: used for data transmission

Set Your HC-05 First !

- Download **Serial Test** for your PC
- Connect your HC-05 with **USB to TTL**
- Hold down the black button **while** powering on it(Enter **Command Mode**)
- Set your HC-05 with **AT instructions**

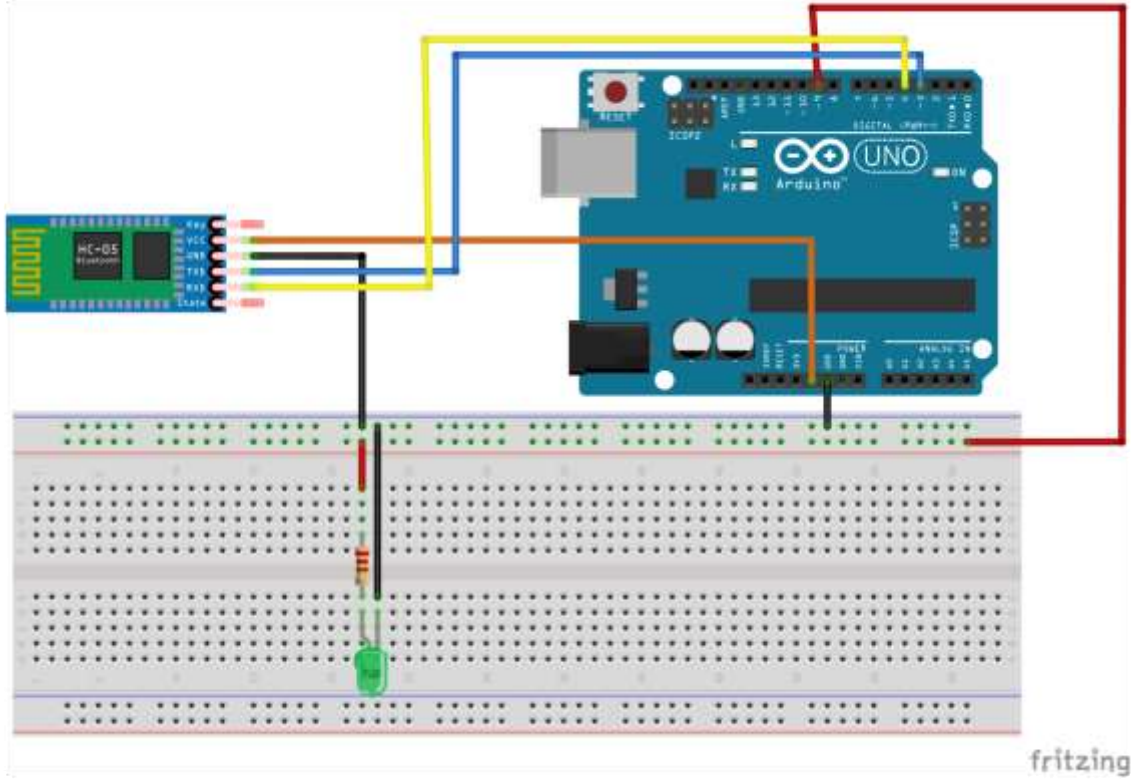


AT Commands

AT Command	
AT+XXX?	AT+XXX=<param>
AT+NAME?	AT+NAME=TEST
AT+VERSION?	
AT+PSWD?	AT+PSWD=<param>

- For further more settings please refer to HC-05 Instructions

EXP. 5: Control Your LED with BT



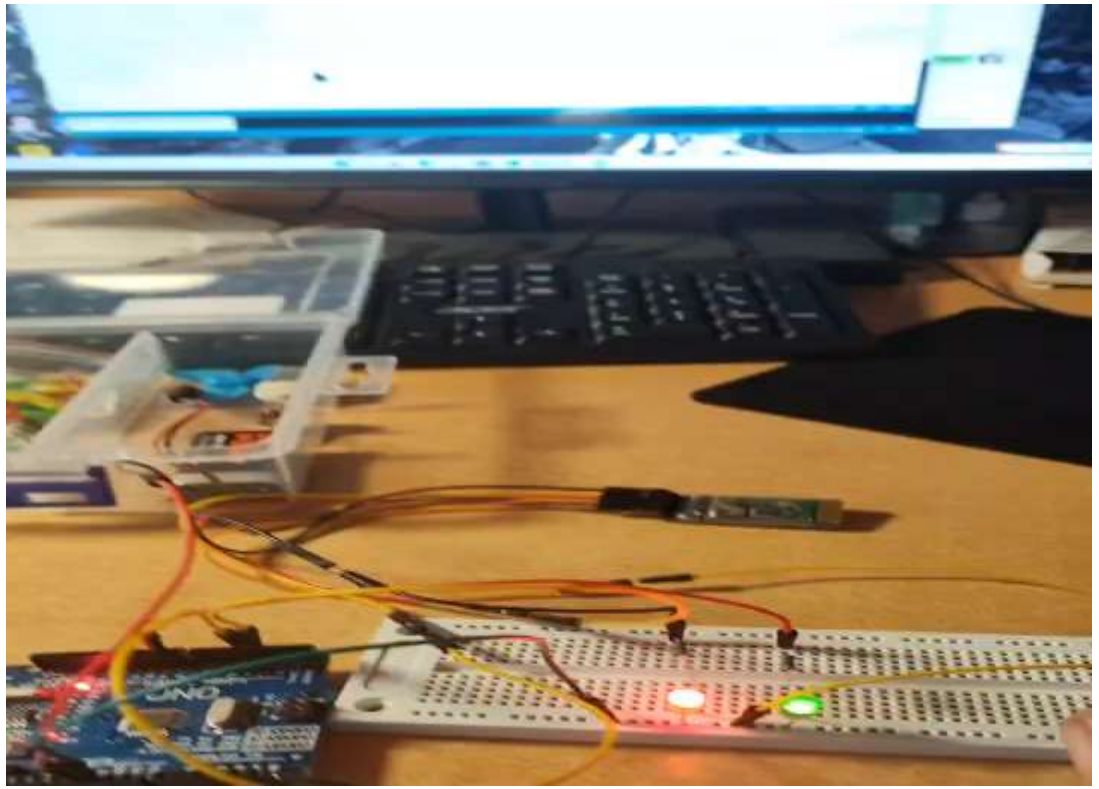
EXP. 5: Control Your LED with BT

4_ble_hc05.ino

```ino
#include <SoftwareSerial.h>

// 创建SoftwareSerial对象，参数为RX和TX引脚
SoftwareSerial BTSerial(3, 4);  // RX, TX

const int ledPin = 9;  // LED连接的引脚（支持PWM）
int brightness = 255;    // 初始亮度为0

void setup() {
  // 初始化软串口
  BTSerial.begin(9600);
  // 初始化LED引脚
  pinMode(ledPin, OUTPUT);
  // 初始化串口监视器
  Serial.begin(9600);
  Serial.println("蓝牙LED亮度控制已启动，等待指令...");
}
```

```ino
void loop() {
  if(flag==0){
    analogWrite(ledPin, brightness);
    flag++;
  }
  // 检查是否有蓝牙数据可读
  if (BTSerial.available()) {
    // 读取蓝牙模块发送的数据
    String input = BTSerial.readString();
    // 转换为整数
    brightness = input.toInt();
    // 限制亮度范围在0到255之间
    brightness = constrain(brightness, 0, 255);
    // 调节LED亮度
    analogWrite(ledPin, brightness);
    // 在串口监视器上输出当前亮度值
    Serial.print("接收到的亮度值: ");
    Serial.println(brightness);
  }
}
```

EXP. 5: Control Your LED with BT



Relay Module

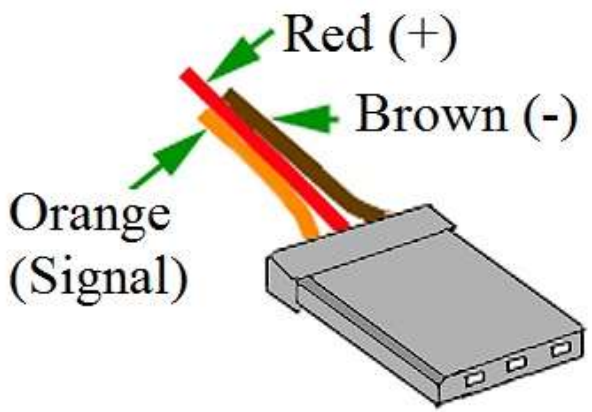
- Use to Control High Power Devices
- Digital Control
- NC → Normally Open
- NO → Normally Close

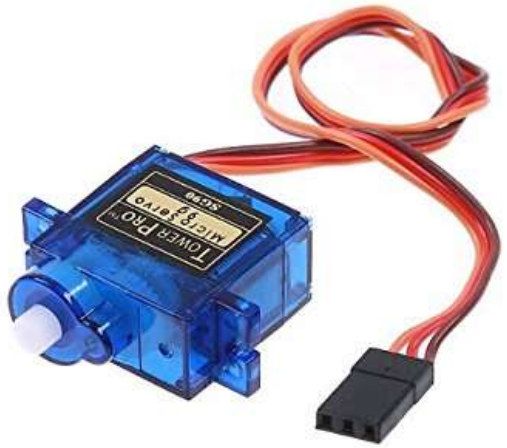
DC-Motors

- Need H-bridge to control
- Never Connect directly with Arduino

Servo Motor

- You can control in it's angle
- Use PWM to control
- Only 180 degree rotate





Part 6

QUESTIONS ?

Summary

Part 1.
Intro to embedded sys

Part2.
Intro to Arduino IDE

Part3.
Basic C language

Part4.
Basic Arduino C

Part5.
Sensors & Actuators
Our Experiments!

The END

THANK YOU!