EE3070 Project Design

Basis for Arduino Programming

Dr. Wallace Tang

Semester A 2021/22

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Introduction

- Arduino: an open-source electronics platform based on easy-to-use hardware and software.
- An open-source Arduino software (IDE) is available. https://www.arduino.cc/en/Main/Software
  - To write code and upload to the board
- There are many examples, coding …

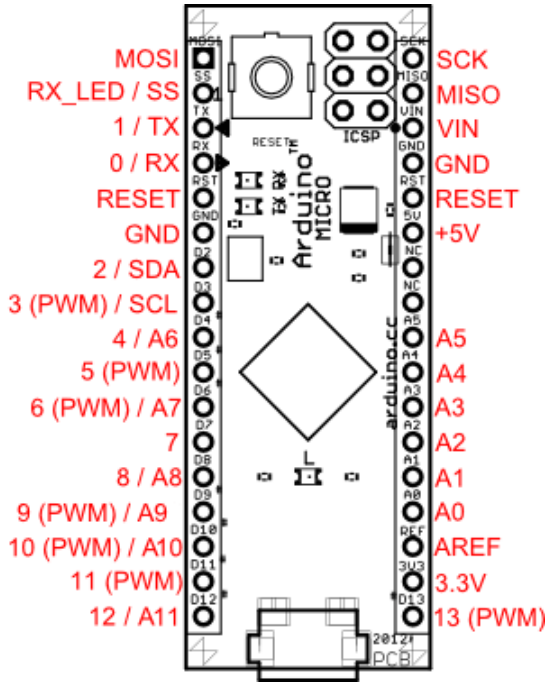
  https://www.arduino.cc/en/Tutorial/HomePage
- There are many different Arduino boards

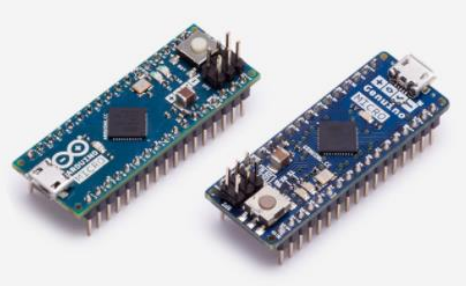
https://www.arduino.cc/en/Products/Compare

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Example: Arduino Micro

- It is a microcontroller board based on ATmega32U4





Technical specs

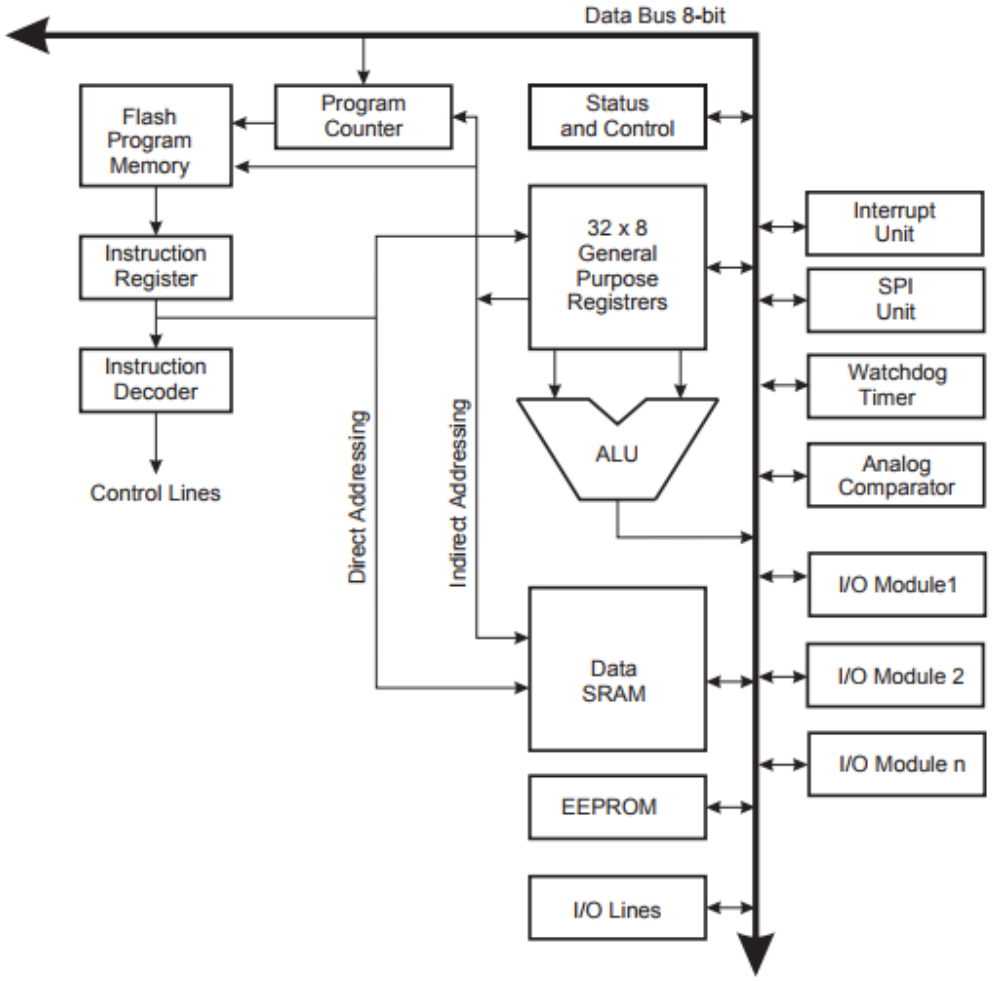
Microcontroller	ATmega32U4
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega32U4) of which 4 KB used by bootloader
SRAM	2.5 KB (ATmega32U4)
EEPROM	1 KB (ATmega32U4)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	48 mm
Width	18 mm
Weight	13 g

About Arduino Micro: https://store.arduino.cc/usa/arduino-micro

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Microcontroller ATmega32U4

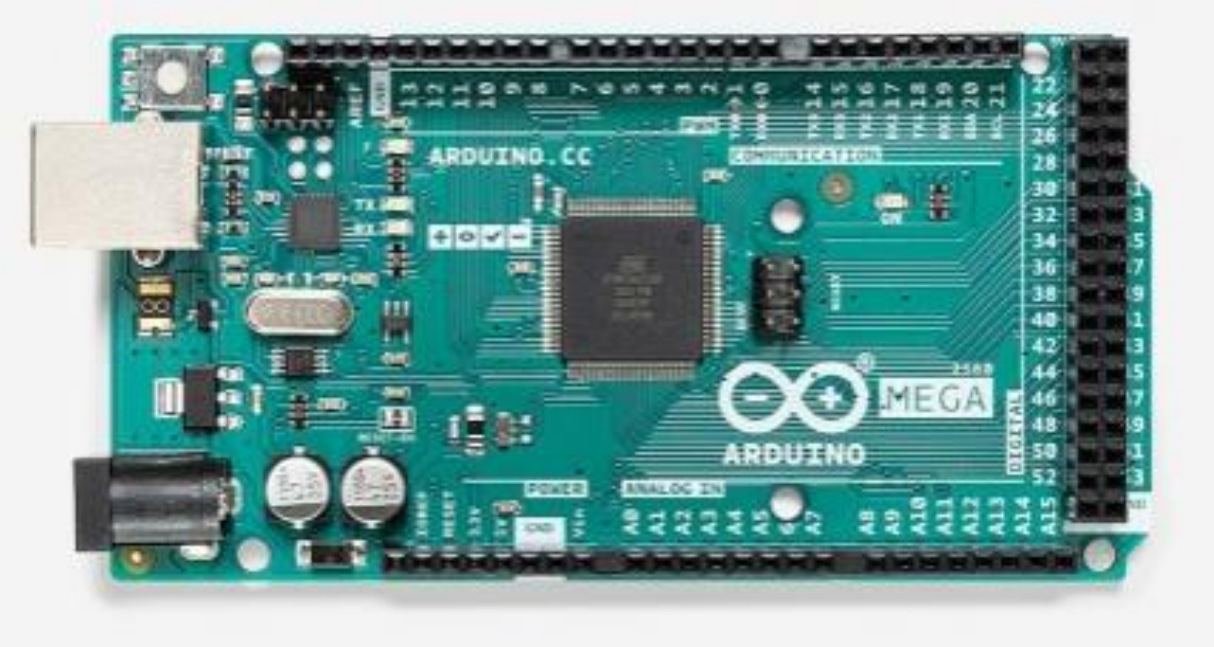
- 8-bit microcontroller
  - Memory (Flash program memory; Data SRAM; EEPROM)
  - I/O: I/O lines; I/O modules
  - ALU (arithmetic logic unit)
  - General purpose registers
  - Others: Interrupt; SPI; Watchdog timer;
- Hold all the compiled code (software) and execute the commands
- Can access the microcontroller peripherals



Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Example: Arduino Mega 2560

- It is a microcontroller board based on ATmega2560

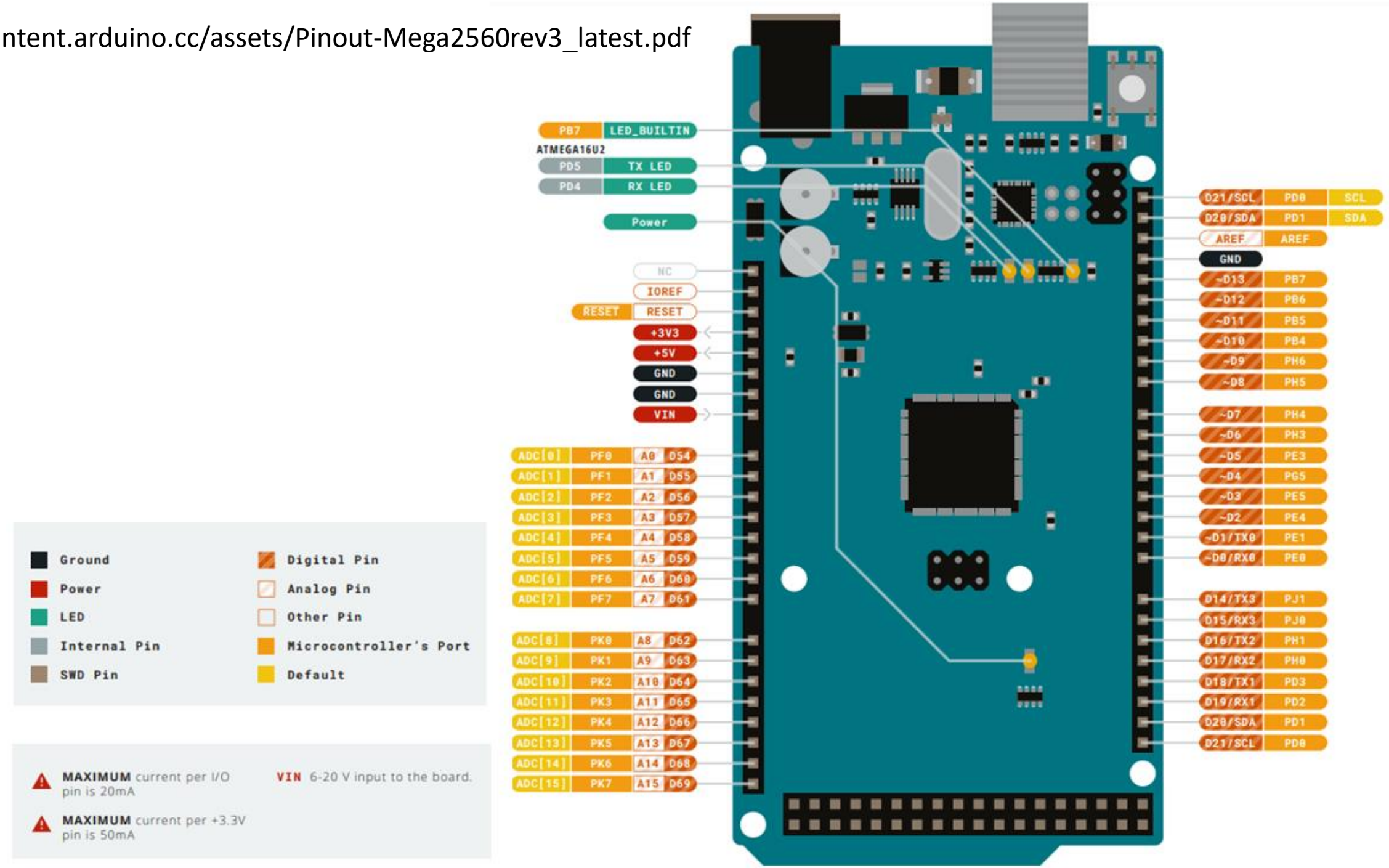


Technical specs

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

About Arduino Mega 2560 v3: https://store.arduino.cc/usa/mega-2560-r3

https://content.arduino.cc/assets/Pinout-Mega2560rev3_latest.pdf



Ground
Power
LED
Internal Pin
SWD Pin

Digital Pin
Analog Pin
Other Pin
Microcontroller's Port
Default

MAXIMUM current per I/O pin is 20mA

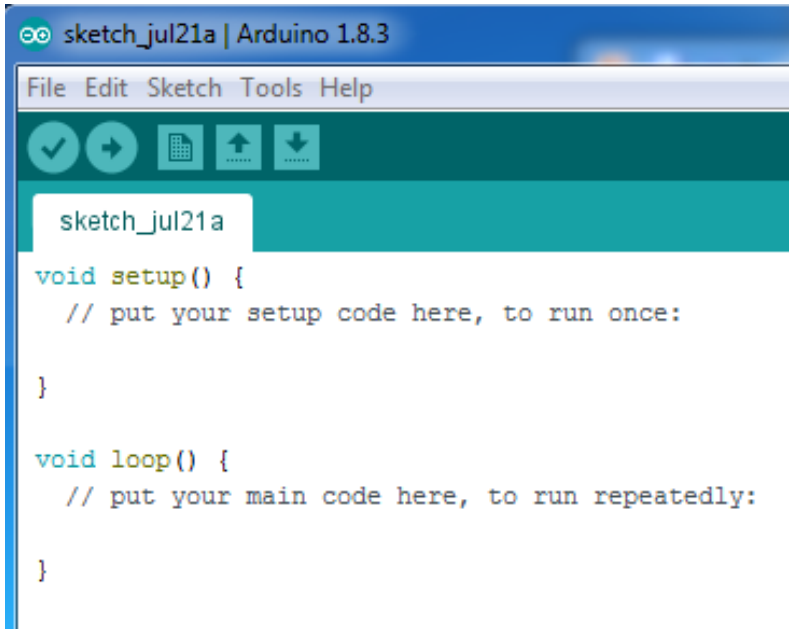
MAXIMUM current per +3.3V pin is 50mA

VIN 6-20 V input to the board.

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Integrated Development Environment (IDE)

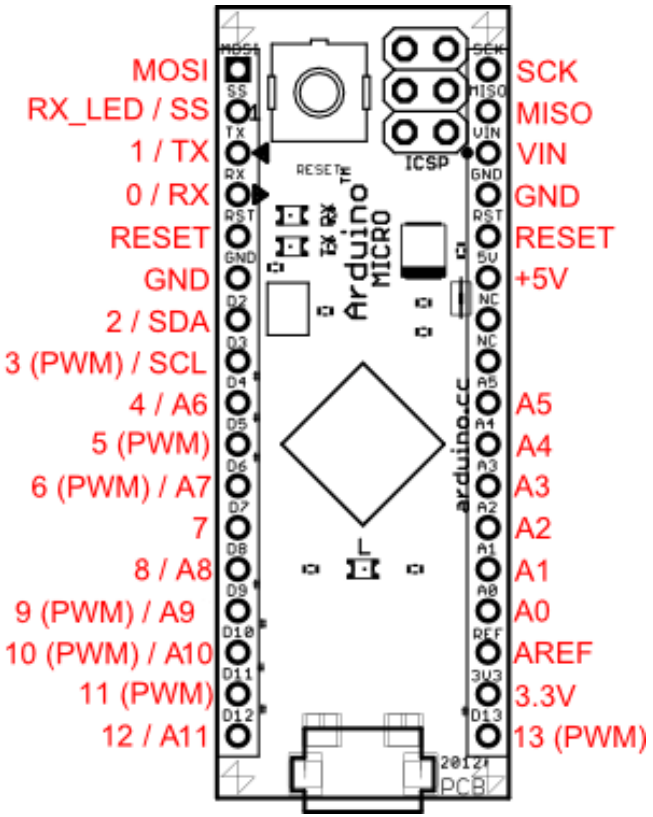
- Open-source Arduino software (IDE) for writing the code and upload to the board
  - https://www.arduino.cc/en/Main/Software
  - Choose the correct OS version (Windows, MacOS, Linux)
- After install, run
- setup() – run once
- loop() – run over and over



I/O Pin Control and Programming

Digital I/O Pins

- Digital I/O Pins – can be used as input or output
- Eg. in Arduino Micro number internally assigned: 0-13, and A0-A5 (A0-A5 are also analog I/O pin). 13 is connected to an internal LED.
- Function: pinMode(pin, mode);
- Two parameters
  - pin: eg. for Arduino Micro, we have pin no. 0-13, A0-A5
  - 3 modes: INPUT, OUTPUT, INPUT_PULLUP



Important Note: Should strictly follow on the capital/non-capital case.

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Pin Configuration

- Configure an I/O pin as output

e.g. pinMode(7, OUTPUT);

  - Logic 0 = 0V; Logic 1 = 5V;
  - source/sink 20mA;

- Configure as input pin

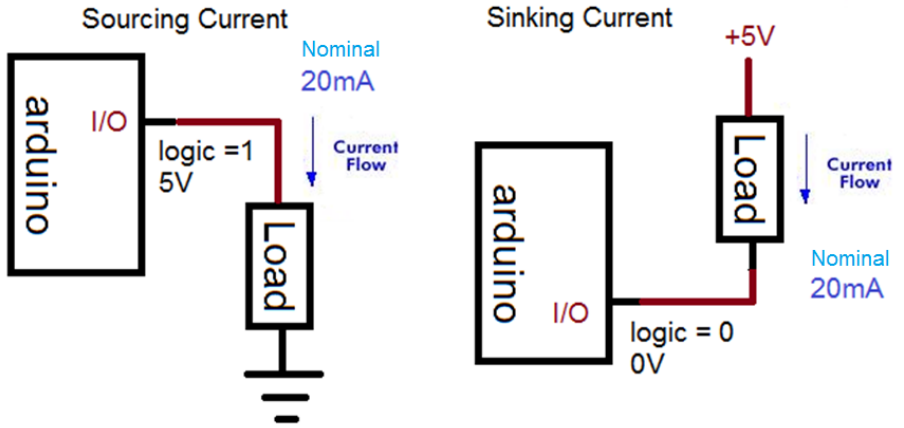
e.g. pinMode(6, INPUT);

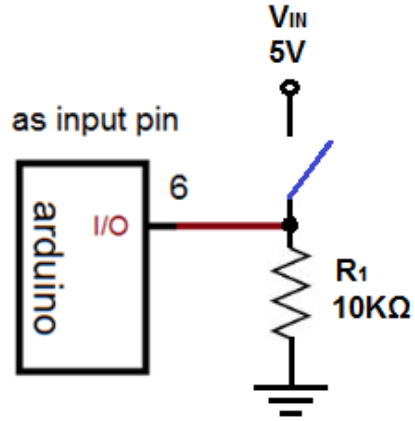
  - High impedance => take very little current (no affect the other part)
  - Non-connected pin will pick up noise, getting undetermined state at the pin.

- Configure as input with pull-up

e.g. pinMode(5, INPUT_PULLUP);

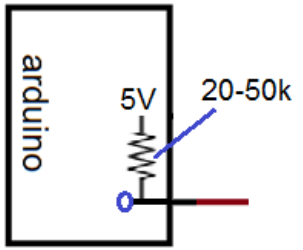
Note: For each line, it should be end with ;





high impedance state

as input with pull-up resistor

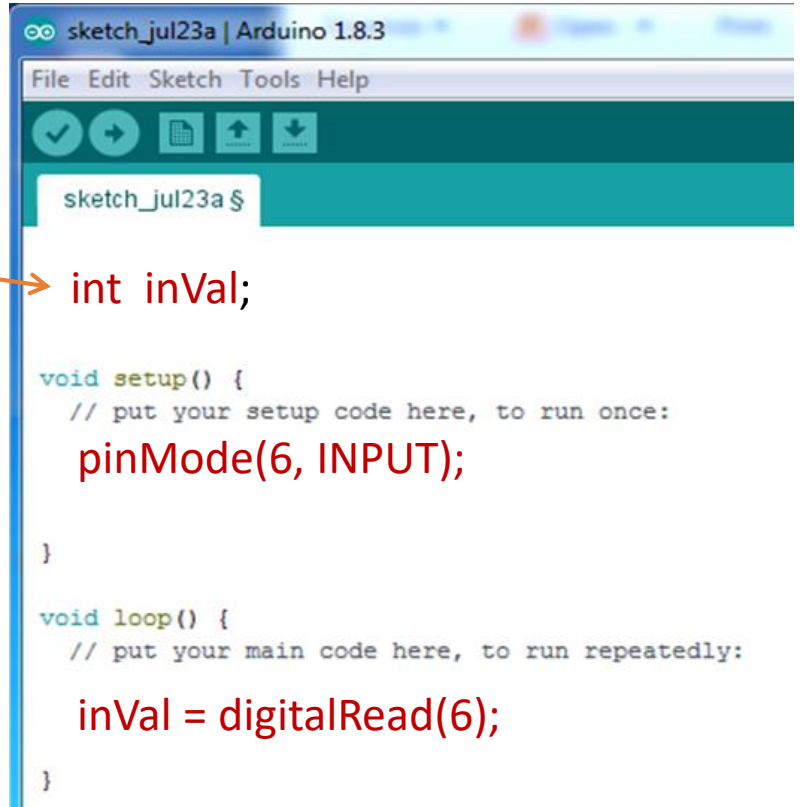


Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Digital Input

- digitalRead(**pin**)
  - parameter: **pin** – pin number (the pin should should be assigned as INPUT or INPUT_PULLUP)
- **digitalRead** is to read the state of the **pin.** But where to store it after read?
  - Ans: Use a variable
    - Define a variable
    - Give a name to a variable
    - Since the digitalRead will give 0 or 1, we can define the type of variable as an integer.
    - eg. int inVal

It means to declare inVal as an integer.



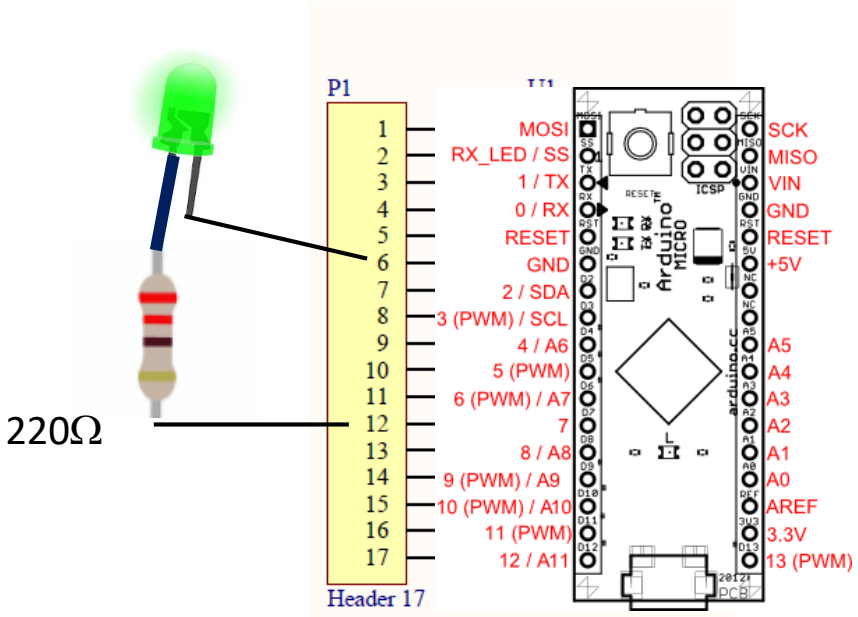
Department of Electrical Engineering
香港城市大學
City University of Hong Kong

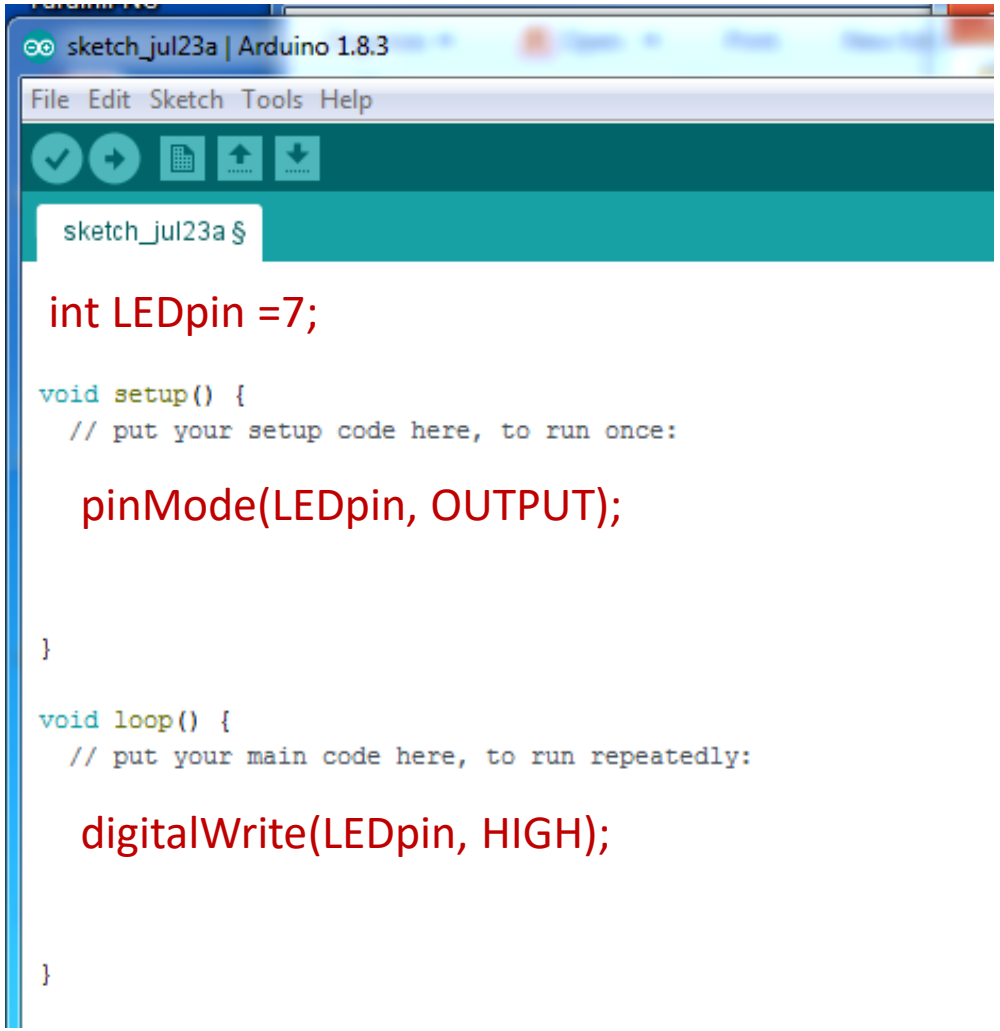
Digital Output

- digitalWrite(**pin, val**);
  - 2 parameters: **pin** – pin number (the pin should have been assigned as OUTPUT)
  - **val: 0 (LOW) or 1 (HIGH)**
  - Note: **LOW** and **HIGH** are defined constants for the I/O pin
- digitalWrite is to output **HIGH** voltage (5V) or **LOW** voltage (0V)

Example: Light up a LED

Department of Electrical Engineering
香港城市大學
City University of Hong Kong





Example: Light up a LED

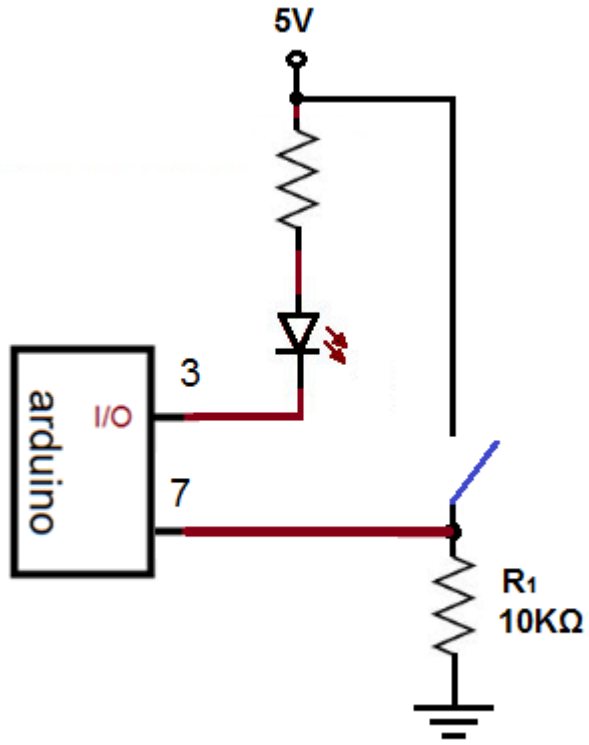
Department of Electrical Engineering
香港城市大學
City University of Hong Kong

What is the function of this program?

single line comment.
With // , the compiler ignores all texts after.

```
int ledPin = 3;  // LED connected to digital pin 3
int inPin = 7;
int val = 0;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(inPin, INPUT);
}

void loop()
{
  val = digitalRead(inPin);
  digitalWrite(ledPin, val);
}
```

Remember:
Only read an input pin, write an output pin!



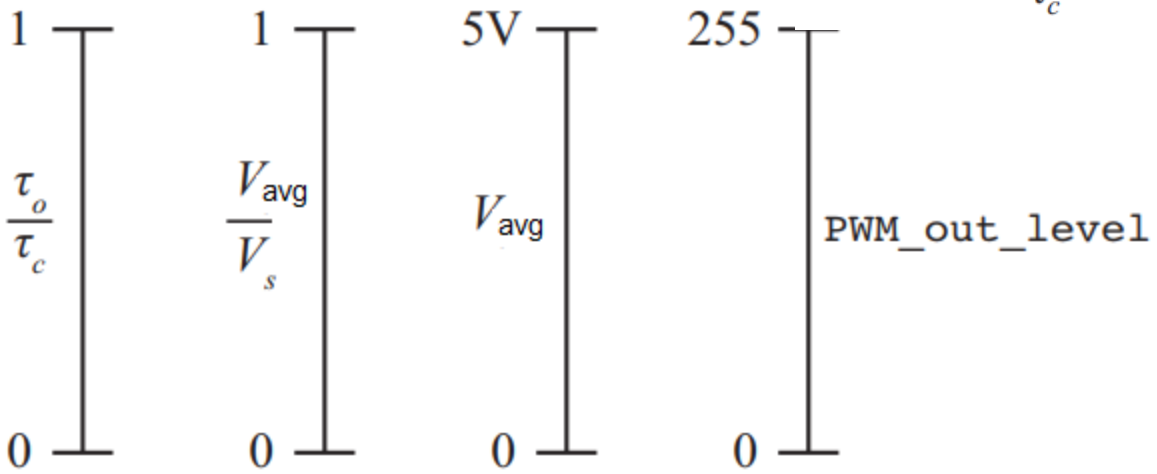
Circuit

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Analog Output

- Using Pulse Width Modulation (PWM)

$$\texttt{PWM\_out\_level} = 255 \times \frac{\tau_o}{\tau_c} = 255 \times \frac{V_{\text{avg}}}{V_s}$$



Now, Vs = 5V

Duty Cycle = $\frac{\tau_o}{\tau_c}$

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Analog Output

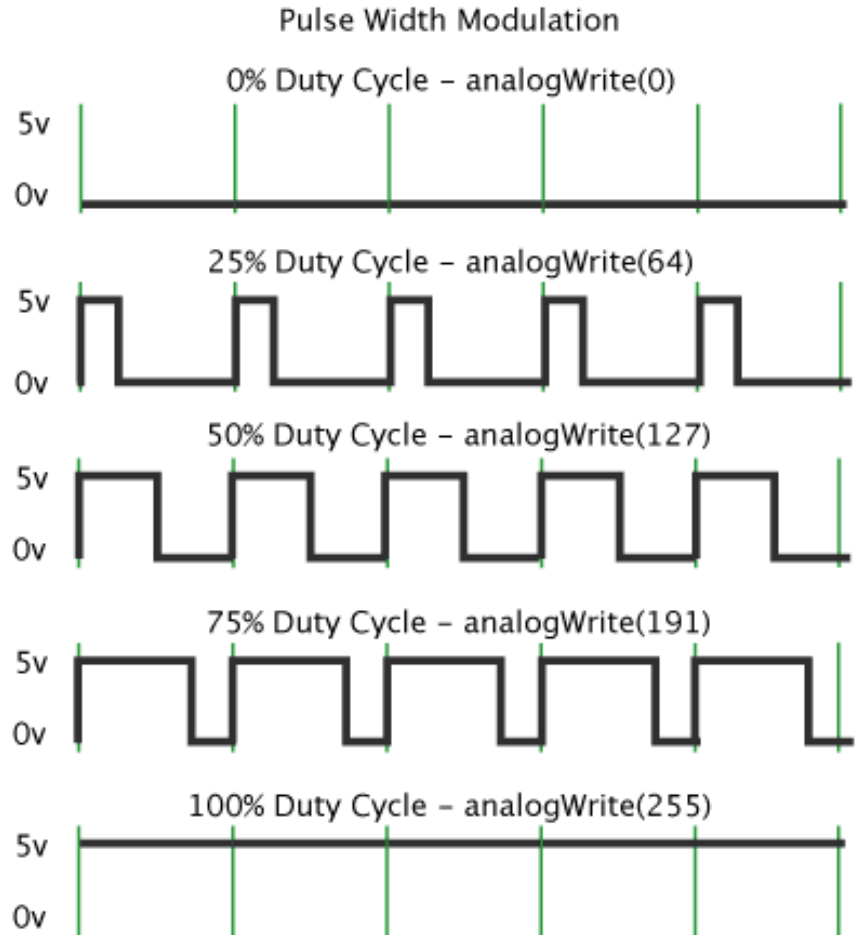
- analogWrite(pin, value)

  Two parameters:

  Pin: PWM pins

  value: 0 - 255

- The pin should be configured as OUTPUT
- analogWrite is to change the duty cycle
  - As a result, the Vavg can be changed



Department of Electrical Engineering
香港城市大學
City University of Hong Kong

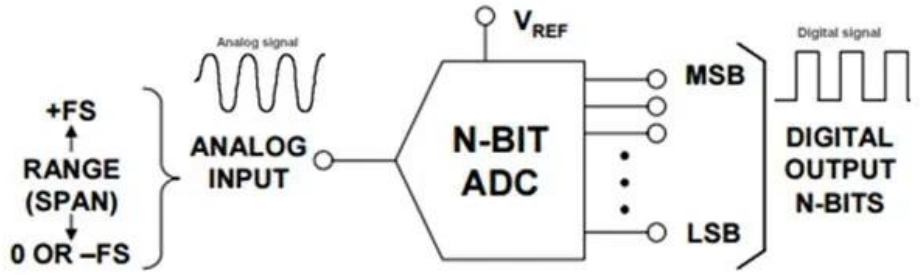
Usages of Analog Output

- Dim a LED
- Provide variable speed control for DC motors.
- Generate an analog output between 0% and 100% of 5V if the output is filtered.

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Analog Input

- Digital: HIGH (logic 1; 5V) and LOW (logic 0; 0V)
- Analog: vary within a range, cannot be discretely classified; Theoretically infinite number of possible values within that range
- In microcontroller, everything is in digital!
- Analog-to-Digital Converter: Convert an analog signal (voltage) into a digital value

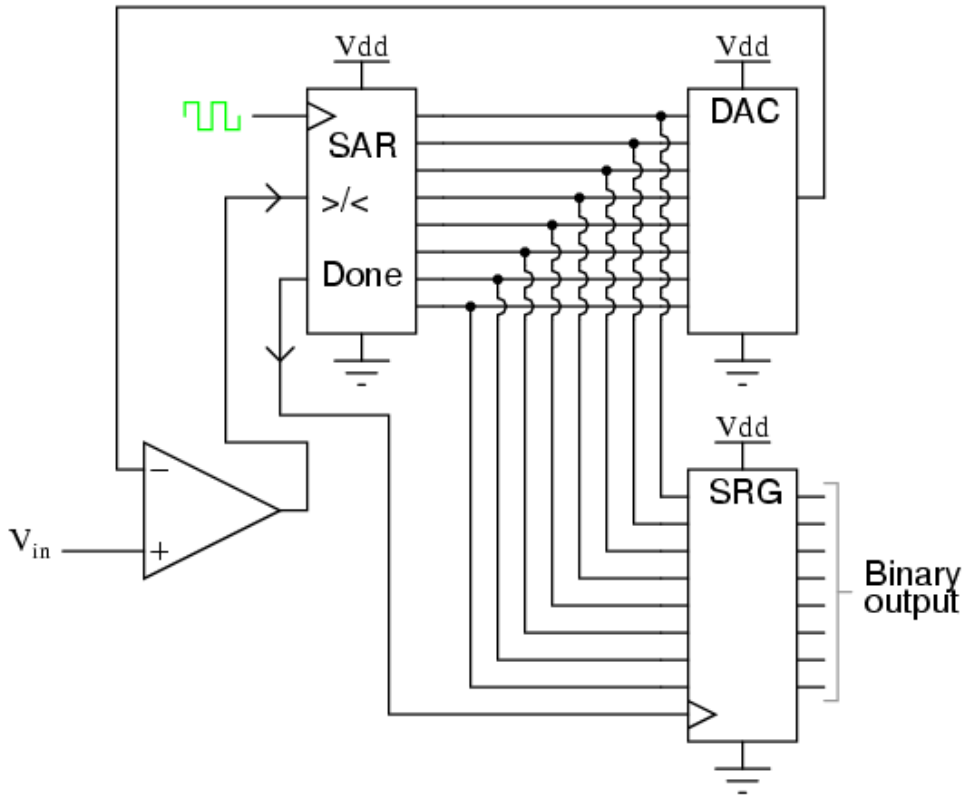


Department of Electrical Engineering
香港城市大學
City University of Hong Kong

ADC in Arduino Micro

- Successive Approximation ADC
  - Compare the result by setting the bits one by one (MSB first). eg: 3 bit case: Vin > V_{100}? Yes, then set MSB = 1. Continue upto LSB

- Features in Arduino
  - 10-bit
  - 0.5 LSB integral nonlinearity
  - ±2LSB absolute accuracy
  - 12 multiplexed channel
  - 0-Vcc input voltage range



Department of Electrical Engineering
香港城市大學
City University of Hong Kong

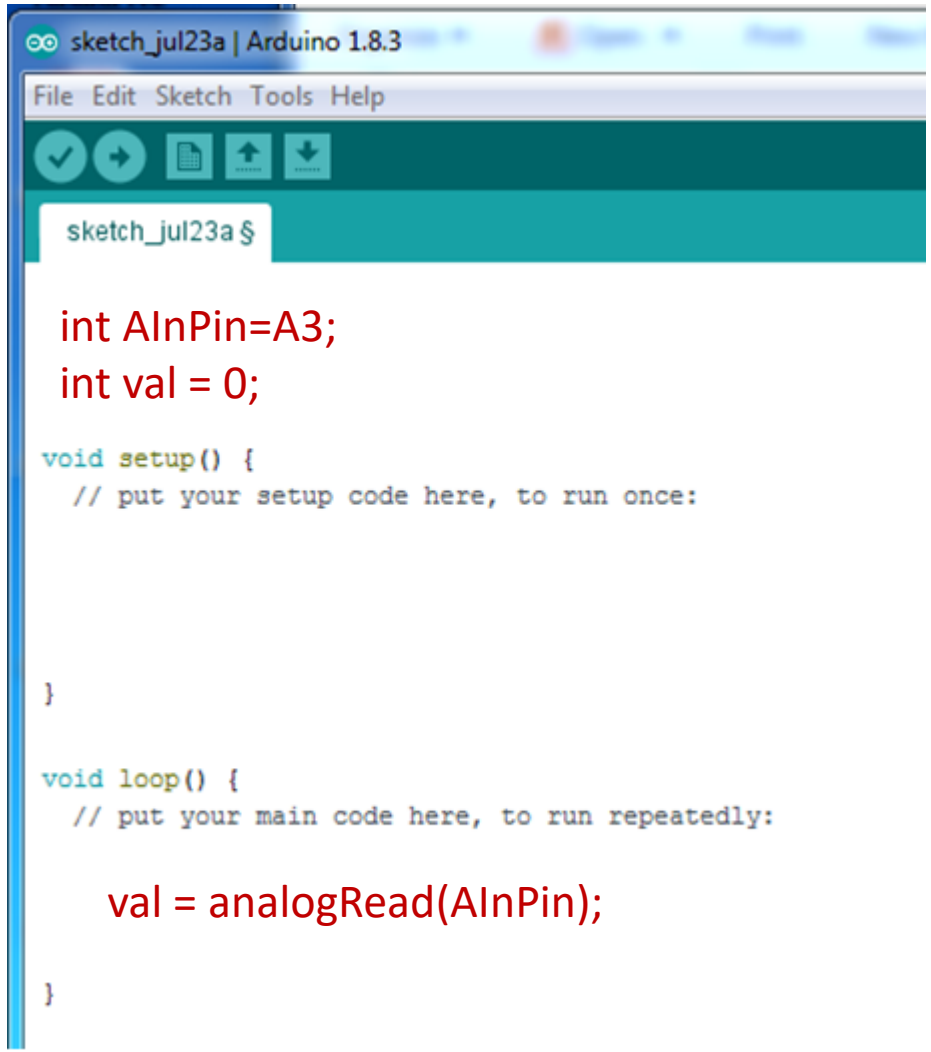
Analog Input

- analogRead(pin)

  Parameter

  pin no: e.g. A0-A11 in Micro

- Reference: 5V (can be changed using the AREF pin and the analogReference())
- Output: 0 – 1023 (10 bits)



Ext Reference Voltage for Analog Input

- The default reference voltage for analog input is 5V.
- analogReference(type)
  - One parameter: type = EXTERNAL
  - The voltage applied to AREF pin (0—5V only)
  - Set it before using analogRead()

Control structure and comparison operators

Program Control

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Output is depended on Input states
  - Conditions need to be met
  - Eg. If A>0 then X=0; If A>=0 and B<0 then Y=1; ...
  - How to do so?

Comparison Operators

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

equal to:	`==`
not equal to:	`!=`
less than:	`<`
greater than:	`>`
less than or equal:	`<=`
greater than or equal:	`>=`

- Binary operators and yield either true (1) or false (0).
- Same precedence levels and associate left to right.
- Comparison operators have lower precedence than the arithmetic operators (eg. `+`, `–`, `*`, `/`).

What is wrong?

```
p < = q;
r => s
```

```
if 2 < index < 6
  digitalWrite(3,HIGH);
else
  digitalWrite(3,LOW);
```

What is the output
If index = 0?

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Notes on Equality

- The `==` works perfectly correctly with integer operands.
- Don't confuse (`==`) with (`=`).
- `a == 2;` determine whether the integer variable `a` is equal to *2*, and yield either true or false.
- `a=2;` assign the value of *2* to variable `a`. It is always true.

Boolean Operators

logical and: `&&`

logical or: `||`

negation: `!`

- `&&` and `||` are binary, both acting on two expressions and yielding the logical value true or false. Both operators treat their operands as logical values.
- `!` is unary operator. The single operand is taken to represent either logical true or false.

```
a = !5

x = 5;
b = 3+!x;
```

What are a and b?

a =
b =

Control Structures

- if
- if … else
- for
- while
- do … while
- switch case
- break
- continue
- return
- goto

No Capital !

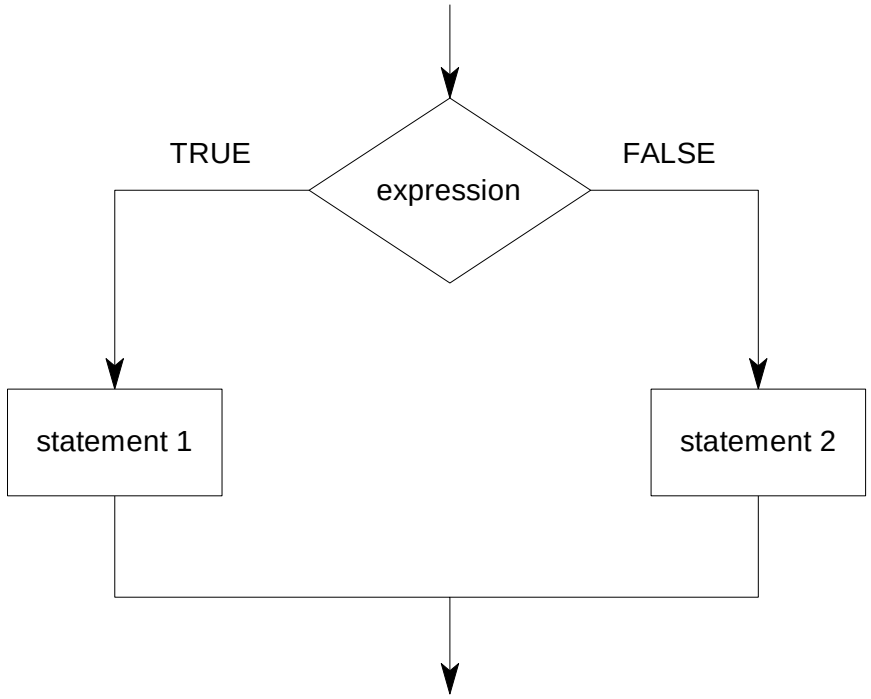
You can check these functions from
https://www.arduino.cc/en/Reference

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

if statement

Good habit: Align and indent (2 spaces or a tab)

```
if (expression)
  statement1;
else
  statement2;
```



- Both `statement1` and `statement2` may be single statement or compound statements.

- **else** can proceed another **if** test, so that multiple, mutually exclusive tests can be run at the same time

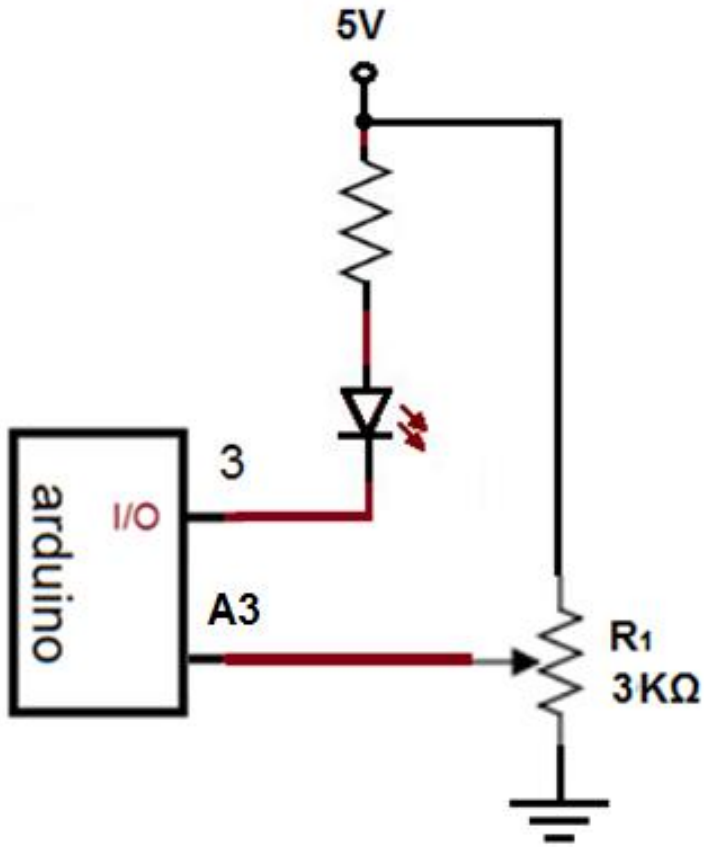
```
if (expression1)
   statement1;
else if (expression2)
   statement2;
else
   statement3;
```

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Example

sketch_jul23a §

```
int LED1=3;
int AinPin=A3;
int val;


void setup() {

  pinMode(LED1, OUTPUT);
}

void loop() {

  val = analogRead(AinPin)
  if ((val > 400) && (val < 600))
    digitalWrite(LED1, HIGH);
  else
    digitalWrite(LED1, LOW);
}
```



Department of Electrical Engineering
香港城市大學
City University of Hong Kong

for Statement

- The syntax of the for statement

```
for (expression 1; expression 2; expression 3)
     statement;
```

- **`expression 1:`** **initialize the loop.**
- **`expression 2:`** **it is evaluated and if it is logical true, then the statement is executed.**
- **`expression 3`** **is then evaluated, and the control is passed back to the beginning of the loop and to re-evaluate `expression 2`.**

```
sum = 0;
for (i=1; i<=m; i++)
  sum += i;
```

Rewrite with while

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Example 1

Example

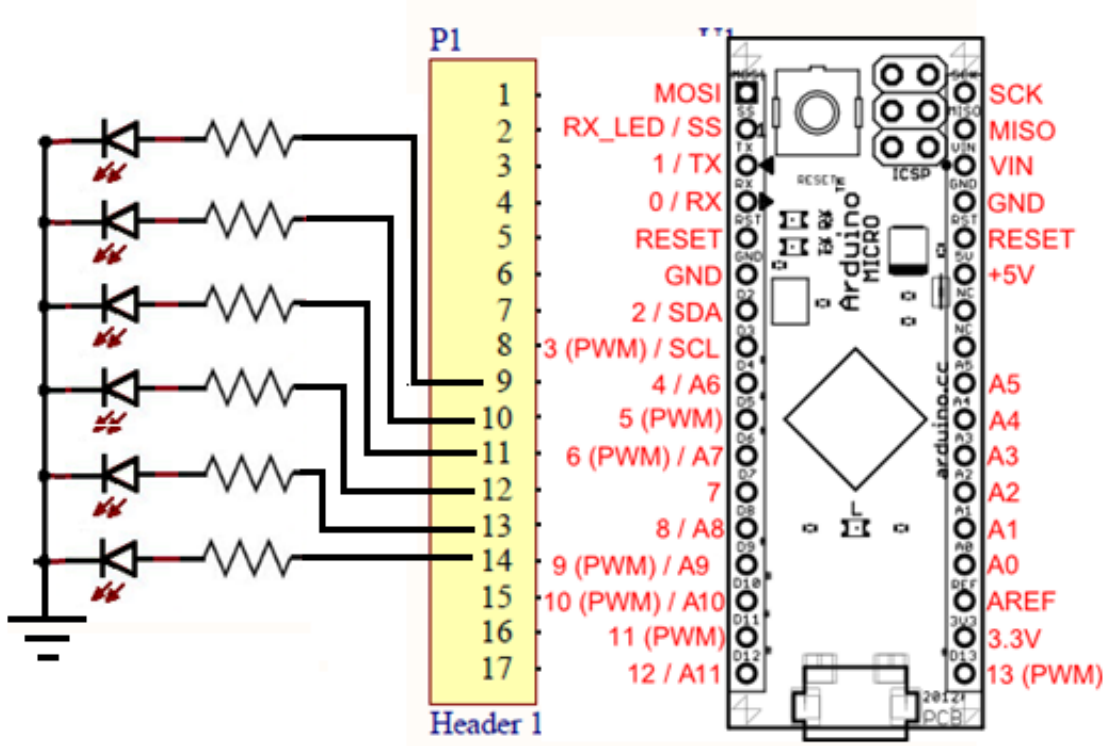
```
// Dim an LED using a PWM pin
int PWMpin = 10; // LED in series with 470 ohm resistor on pin 10

void setup()
{
  // no setup needed
}

void loop()
{
   for (int i=0; i <= 255; i++){
      analogWrite(PWMpin, i);
      delay(10);
   }
}
```

Pause for 10 millisecond

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Example 2

Given six LEDs, turn on them one by one for 300 milliseconds.




```
void setup() {

  int i;
  for (i=4; i<10; i=i+1)
    pinMode(i, OUTPUT);
}


void loop() {

  int i;
  for (i=4;  i<10;  i++)
  {
    digitalWrite(i, HIGH);
    delay(300);
    digitalWrite(i, LOW);
    delay(300);
  }
}
```

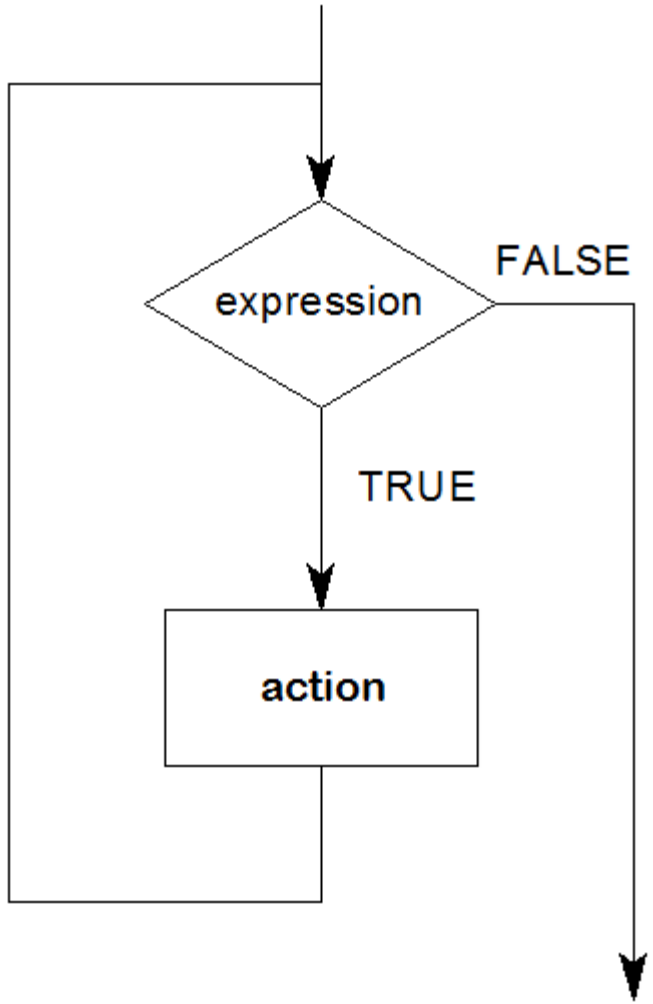
Department of Electrical Engineering
香港城市大學
City University of Hong Kong

while loop

- The syntax of the while statement is

```
while (expression)
{
    action;
}
```

- **while loops will loop continuously until the (expression) become false.**



do – while statement

- The **do** statement is in the form of:

```
do
{
   statement;
} while (expression);
```

- The distinction between **do**-*statement* with **while** or **for** statement is that the conditional test is performed after executing the loop.

- The loop body is then guaranteed to be obeyed at least once.

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

switch / case Statement

- **control expression** must be of an **integral** type, including **char** (character data type).
- **constant expressions** associated with each **case** keyword must also resolve to an **integer** *(or* **char***)*
- Resulting value is compared with each **case** label in turn.
- If a case label value equals the value of the expression, control is passed to the first statement associated with that *case* label.
- All statements through to the end of the **switch** are then executed / reach the **break**

```
switch (expression){
  case constant-exp-1:
       statement 1;
      break;
  case constant-exp-2:
       statement 2;
       break;
  :
  :
  default:
       statement D;
   break;
}
```

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Example

```
n=2;
switch(n){
  case 1:
    digitalWrite(1,HIGH);
  break;
  case 2:
    digitalWrite(2,HIGH);
  break;
  default:
    digitalWrite(2,HIGH);
    digitalWrite(3,HIGH);
  break;
}
```

What is the output?

- default keyword is an optional. If the expression does not equal to any of the constant expressions, and no default is present, no statement is executed.

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

break

- **`break`** is used to exit from a **`while`**, **`for`** and **`do`** loop, bypassing the normal loop condition. It is also used to exit from a **`switch`** statement.
- Execution of a **`break`** statement causes immediate termination of the innermost enclosing loop.

```
for (x = 0; x < 255; x++)
{
    analogWrite(PWMpin, x);
    sens = analogRead(sensorPin);
    if (sens > threshold){      // bail out on sensor detect
       x = 0;
       break;
    }
    delay(50);
}
```

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

continue

- When a **`continue`** statement is executed, control is immediately passed to the test condition of the nearest enclosing loop.
- All subsequent statements in the body of the loop are ignored for that particular loop iteration.
- Its use is restricted to **`while`**, **`for`** and **`do`** loops.

```
for (x = 0; x < 255; x++)
{
    if (x > 40 && x < 120){      // create jump in values
        continue;
    }

    analogWrite(PWMpin, x);
    delay(50);
}
```

Serial Communications

Arduino and PC via USB

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Built-In Communications to PC

- Serial is used for communication between the Arduino board and a computer/other devices.
- Communication between PC and Arduino board (via USB) is based on a built-in serial monitor. (Click the serial monitor button in the toolbar)
  - Select the same baud rate used in the program
  - The RX and TX LEDs on the board will flash when data is being transmitted via the USB connection to the computer
- In Arduino Micro, Serial class refers to USB communication and it is used in programming.

Program with Serial Communication

- Serial.begin(val)
  - One parameter
    - val: baud rate, eg. 9600, 57600 …
  - It is used to set the baud rate for the serial communications

```
//----- SETUP SERIAL PORT -----
Serial.begin(9600);
while(!Serial)
  ;            //Leonardo/Micro should wait for serial init
```

- Serial is true if the port is ready.
- Serial.end (): Disable serial communication

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Sending to Serial Monitor (Arduino→PC)

- Sending text or string of texts

```
Serial.print("Hello");         //Write string no new line
Serial.println(my_variable);  //Write a value with line break at end
```

- Sending Variable as ASCII

Serial.print(78, BIN) gives "1001110"

Serial.print(78, OCT) gives "116"

Serial.print(78, DEC) gives "78"

Serial.print(78, HEX) gives "4E"

Serial.println(1.23456, 0) gives "1"

Serial.println(1.23456, 2) gives "1.23"

Serial.println(1.23456, 4) gives "1.2346"

Default is two decimals.

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Receiving from Serial Monitor (PC →Arduino)

- Serial.available()
  - Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes).

- Serial.parseInt()
  - Get the first valid integer number from serial buffer. Characters that are not integers are skipped.

- Serial.readString()
  - Read characters from serial buffer and output the string (a string of characters)

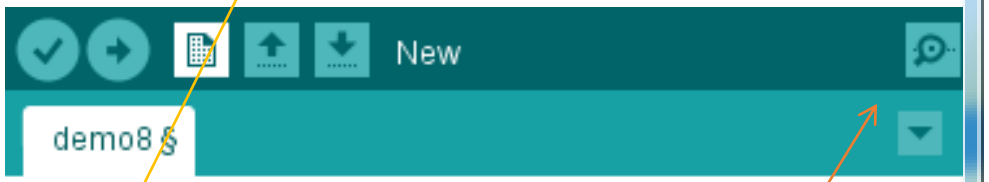
Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Example 1

Read an integer

Serial Monitor

Setup serial comm




```
int val  ;

void setup() {
  int i;

  Serial.begin(9600);
  while (!Serial)
  ;
  for (i=4; i<10; i++)
    pinMode(i,OUTPUT);
}
```

```
void loop() {
  int i;

  if (Serial.available() > 0)
    val = Serial.parseInt();
  Serial.println(val, DEC);
  if ((val > 400) && (val < 600)) {
  for (i=4; i<10; i++)
  {
    digitalWrite(i, HIGH);
    delay(200);
    digitalWrite(i, LOW);
    delay(200);
  }
  }
  else
  for (i=9; i>=4; i--)
  {
    digitalWrite(i, HIGH);
    delay(200);
    digitalWrite(i, LOW);
    delay(200);
  }
}
```

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Example 2

- String is a variable that contains characters
- str1.length returns the size of the str1 (str1 is a String)
- Say, str1 = “123”. Then
  - str1[0] = ‘1’;
  - str1[1] = ‘2’;
  - str1[2] = ‘3’.

  Again, ‘1’ is a character (in ASCII), not number 1. Instead, ‘1’ = 49 in ASCII

demo9 §

```
void loop() {
  int i;
  int size;
  String str1;

  if (Serial.available() > 0)
  {
      str1 = Serial.readString();
      size = str1.length();
      val = 0;
      for (i=0; i<size; i++)
        val = val*10 + str1[i]-'0';
  }

  Serial.println(val, DEC);
```

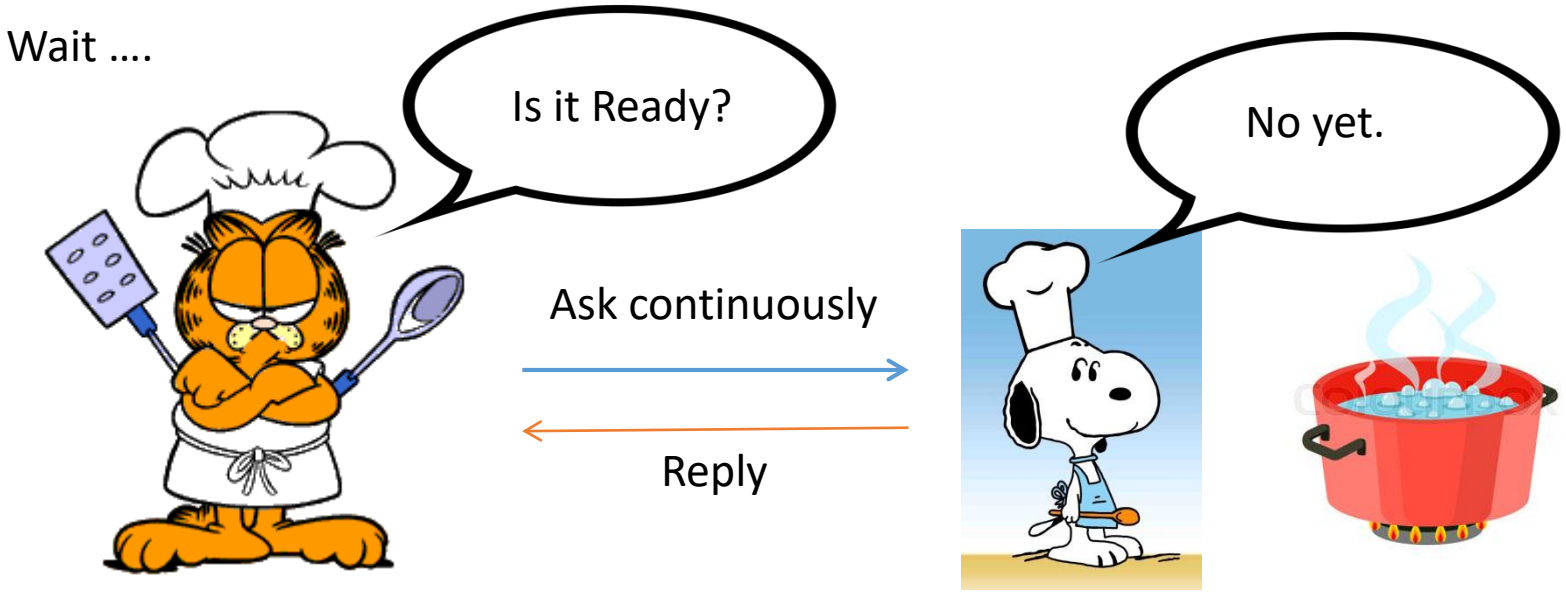
Read a String

Interrupt and timer

Event Handling

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

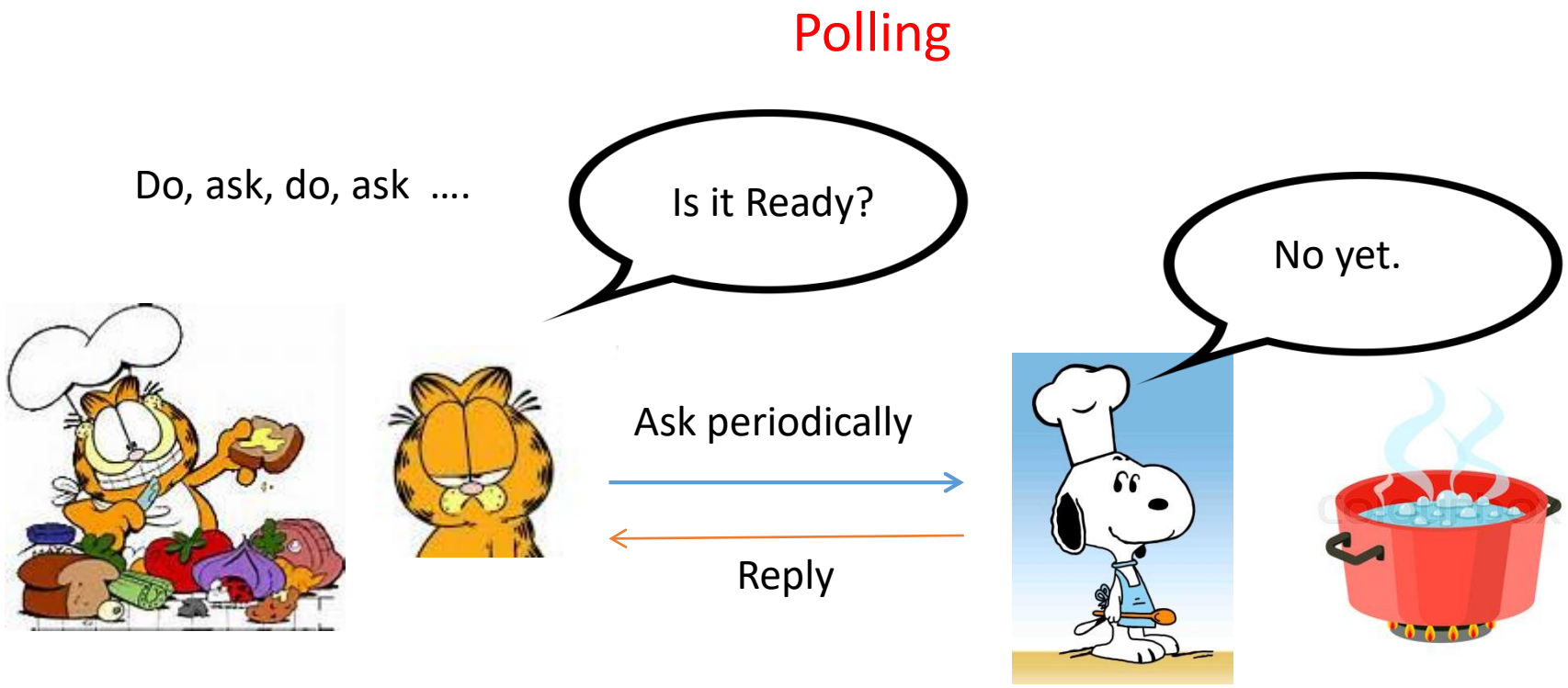
- Say, if we want to check whether an event occurs or not, what can we do?



Event Handling

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Polling
  - May miss, and hence respond with a delay.



Event Handling

- Better way …

Doing other things

Interrupt

Hi, it is done!

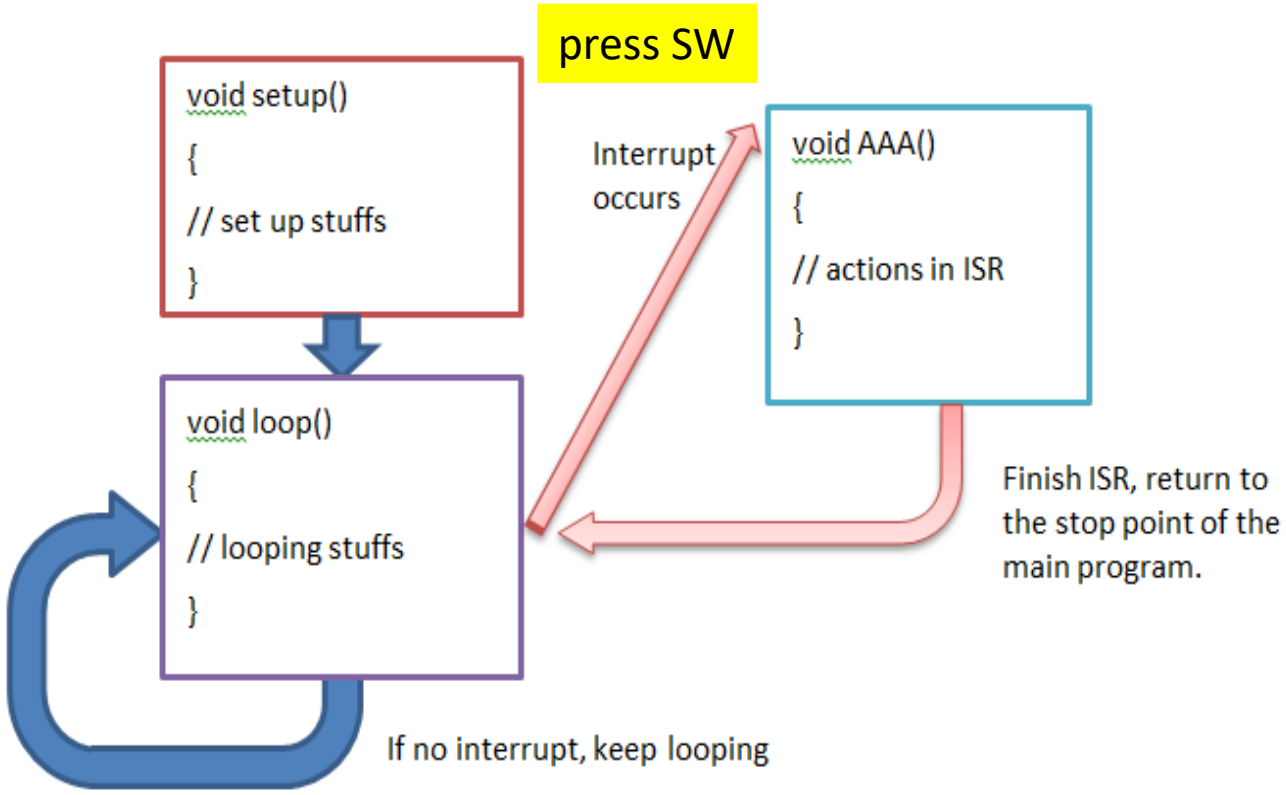
OK, I come !

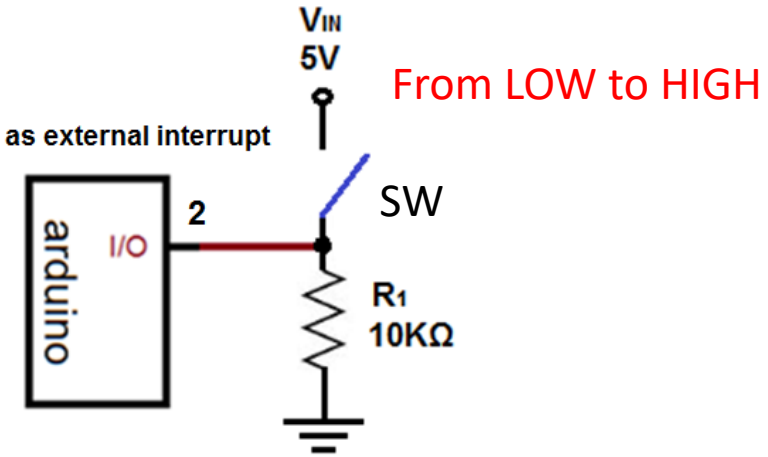
- An interrupt is a signal that tells the processor to immediately stop what it is doing and handle some high priority processing.
- Timer Interrupt: based on the time
- Event Interrupt: based on an event

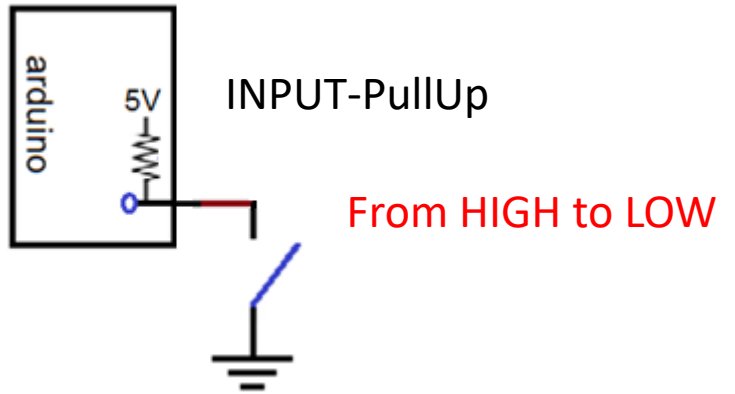
Department of Electrical Engineering
香港城市大學
City University of Hong Kong

External Interrupt

- Triggered by an external event, using a digital pin.
- When interrupt, run the Interrupt Service Routine (ISR is a procedure containing the actions required after an interrupt.)







Interrupt Service Routing

- ISR should be short.
- Do not call time functions (eg. millis, delay) as they may be wrong
- Better disable interrupt when ISR is called, to avoid a nested case.

How to Program an Interrupt

- attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);
  - For Micro, Digital Pins: 0,1,2,3,7 can be used
  - Which Pins can be used in Mega?
  - ISR (Interrupt Service Routine) to call when the interrupt occurs. ISR has no parameters and return nothing
  - mode: define when the interrupt should be triggered.
    - **LOW** to trigger the interrupt whenever the pin is low,
    - **CHANGE** to trigger the interrupt whenever the pin changes value
    - **RISING** to trigger when the pin goes from low to high,
    - **FALLING** for when the pin goes from high to low.

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Other Related Commands

- detachInterrupt(digitalPinToInterrupt(pin));
  - Turn off the given interrupt
- noInterrupts()
  - Disable interrupts
  - No parameter
- interrupts()
  - Re-enable interrupts
  - No parameter

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Example

Making the variable as a constant, and cannot be changed in other places.

```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
  digitalWrite(ledPin, state);
}

void blink() {
  state = !state;
}
```

A **volatile** variable is one to be changed in the ISR

Interrupt service routine

Timer

- Here, it is referred to a library called timer.h
- https://playground.arduino.cc/Code/Timer
- Imagine your have a stop watch
  - You can do something according to the specified time
- Timer t

A class

- Attach up to 10 events to a timer

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Programming the Timer

int **update**()

- Must be called from 'loop()'.
- This will service all the events associated with the timer.

int **every**(long period, callback)

- Run the 'callback‘ (the procedure you design, eg. situp) every 'period' milliseconds.
- Return the ID of the timer event.

int **every**(long period, callback, int repeatCount)

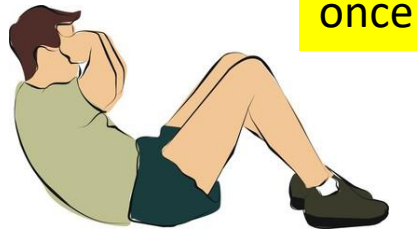
- Run the 'callback' every 'period' ms for a total of 'repeatCount' times.
- Return the ID of the timer event.

Timer t

t.update

t.every(1000, situp)

once per second



t.every(1000, situp,10)

once per second
but only do ten times.

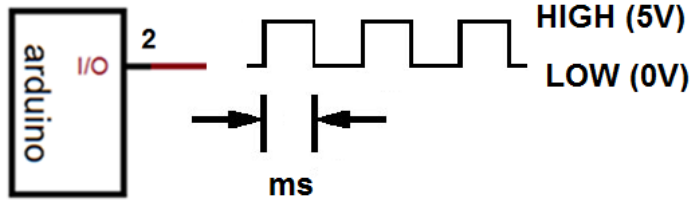
Programming the Timer

int **oscillate**(int pin, long period, int startValue)

- Toggle the state of the digital output 'pin' every 'period' milliseconds.
- The pin's starting value is specified in 'startValue', which should be HIGH or LOW.
- Return the ID of the timer event.

int **oscillate**(int pin, long period, int startValue, int repeatCount)

- Toggle the state of the digital output 'pin' every 'period' milliseconds 'repeatCount' times.
- The pin's starting value is specified in 'startValue', which should be HIGH or LOW.
- Return the ID of the timer event.



Programming the Timer

int **pulse**(int pin, long period, int startValue)

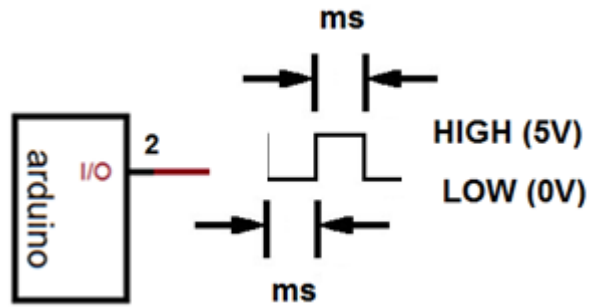
- Toggle the state of the digital output 'pin' just once after 'period' milliseconds.
- The pin's starting value is specified in 'startValue', which should be HIGH or LOW.
- Return the ID of the timer event.

int **stop**(int id)

- Stop the timer event running.
- Return the ID of the timer event.

int **after**(long duration, callback)

- Run the 'callback' once after 'period' milliseconds.
- Return the ID of the timer event.



Example 1

- Note: t.oscillate returns an integer (the timer event ID) but we didn't store it up. This is still ok.
- What are the two events?
  - Event 1:
  - Event 2:

```
Timer t;
int pin = 13;

void setup()
{
 Serial.begin(9600);
 pinMode(pin, OUTPUT);
 t.oscillate(pin, 100, LOW);
 t.every(1000, takeReading);
}

void loop()
{
 t.update();
}

void takeReading()
{
 Serial.println(analogRead(0));
}
```

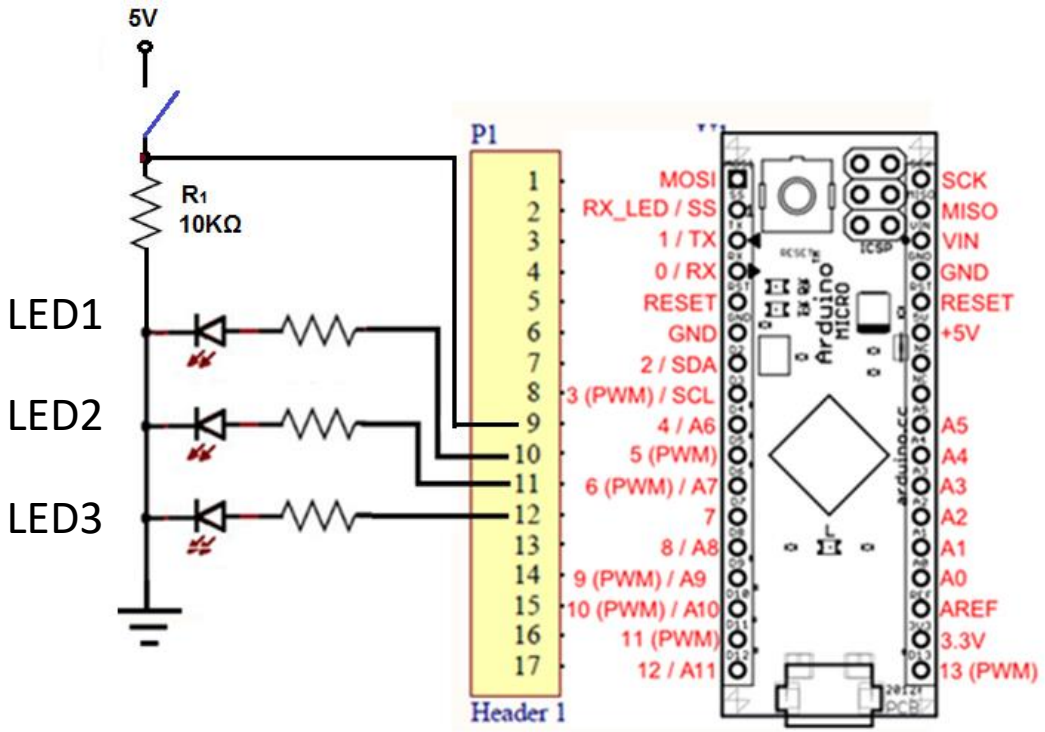
Serial.begin(9600) :
Setup of communication between Arduino and PC.
Serial.printlin : write to PC via USB and display in a Serial monitor.

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Example 2

Given three LEDs

1) Blink LED1 on/off every 300 milliseconds.
2) Light up LED2 once after 500ms and then after 500ms off
3) Check input every 500ms, if Yes, turn on LED3 and always on.




```
#include “Timer.h”
Timer t;
Int e1,e2,e3;

void setup()  {
  pinMode(4, INPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  e1 = t.oscillate(5, 300, LOW);
  e2 = t.pulse(6, 500, LOW);
  e3 = t.every(500, checkP4);
}

void loop()  {
   t.update();
}

void checkP4() {
  if (digitalRead(4) == HIGH) {
     digitalWrite(7, HIGH);
      t.stop(e3);
  }
}
```

END