Workshop Syllabus:

Table of Contents

[0 What is Arduino? 2](#_Toc32927496)

[Introduction 2](#_Toc32927497)

[The IDE 2](#_Toc32927498)

[How Arduino Upload Code 4](#_Toc32927499)

[1 Basic Programming 5](#_Toc32927500)

[1.1 The Structure – A Quick View 5](#_Toc32927501)

[1. 2 Serial.println() – Debugger 101 5](#_Toc32927502)

[1.3 Variables and Constant – The Changing and Unchanged 7](#_Toc32927503)

[1.4 Operators – + - x / 7](#_Toc32927504)

[1.5 Control Statement – Time for making decision 8](#_Toc32927505)

[1.6 Repetition – Let the machine do the boring stuff 8](#_Toc32927506)

[Array – Working with a chunk of values 8](#_Toc32927507)

[Function – Structure Your Code into Block 9](#_Toc32927508)

[Concept of Library 9](#_Toc32927509)

[2 Fundamental of Electronics 10](#_Toc32927510)

[2.1 How’s a bread board looks like? 10](#_Toc32927511)

[2.2 Series and Parallel 10](#_Toc32927512)

[2.3 Voltage Divider 10](#_Toc32927513)

[2.4 Ohms Law, voltage source and current source. 10](#_Toc32927514)

[2.5 GPIO, Pull-up Pull-down and Open Drain Push Pull 10](#_Toc32927515)

[2.6 Passive Component – Resistor (photoresistor, potential meter) 11](#_Toc32927516)

[2.7 Types of Signal – Analogue, Digital 12](#_Toc32927517)

[2.8 Machine Interface – Serial Communication, I2C, SPI 12](#_Toc32927518)

[3 Input 15](#_Toc32927519)

[3.1 Types of Buttons? How to connect a button to MCU? 15](#_Toc32927520)

[3.2 Limit Switch 16](#_Toc32927521)

[3.3 Motion Sensor 17](#_Toc32927522)

[3.4 Distance Sensor 19](#_Toc32927523)

[3.5 IR Sensor 19](#_Toc32927524)

[4 Output (Visual, Physical and Audio) 20](#_Toc32927525)

[4.1 LED 20](#_Toc32927526)

[4.2 Servomotor 21](#_Toc32927527)

[4.3 LCD Display 21](#_Toc32927528)

[5 Communication, I2C SPI Serial Revisit 23](#_Toc32927529)

[5.1 Bluetooth Module 23](#_Toc32927530)

[5.2 I2C 24](#_Toc32927531)

[Optional: NODEMCU For IoT using MQTT 24](#_Toc32927532)

[Putting All Together 24](#_Toc32927533)

# 0 What is Arduino?

## Introduction

The Arduino is a project that forked (or “refer”) from the Wiring project, which aims to provide easy and user-friendly interface for the artist to prototype their hardware. It’s an open-source hardware and software project, their IDE, which used to compile the Arduino code is derived from Processing, an IDE that used to create electronics art using JavaScript.

Followed by the link is the schematics of a typical Arduino UNO board. <https://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf>

You can make one Arduino and even sells it, but you need to propagate the open source license and credit the author.

There’s some drama happened between the author and Arduino and Wiring, check this out when you’re free <https://arduinohistory.github.io/>

## The IDE

IDE stands for Integrated Development Environment. Inside an IDE there’s usually a text editor, a build tools and a debugger. To install the Arduino IDE, goes to here: <https://www.arduino.cc/en/main/software>

Figure below shows an Arduino IDE:

A screenshot of a cell phone

Description automatically generated

Here are the explanations of the components inside the IDE:

1. File
   1. New: Open a new sketch (in Arduino, a file consist of Arduino code is called a “sketch”)
   2. Open: Open a sketch (each sketch has to be contained inside a folder)
   3. Examples: Very important section, consists of some built-in example which you can plug and play for some widely used sensor and motor.
   4. Page Setup and Print: If you want to do your report and include your code, consider print it in a nice manner using these two functions.
   5. Preference: If you don’t like the font size of your code editor, you can customize here. You can also set line number visible so that you can know how long your code is. (hopefully not long!)
2. Edit
   1. If you are a web blogger or online tutorial writer, the copy for forum and coy for HTML is very useful because you can copy your code and display it on website as it would display in the Arduino IDE!
   2. The rest of the options in this tab is just some keyboard shortcut to increase the font size (ctrl + plus), select all (ctrl + A) and etc.
3. Sketch
   1. Before uploading a sketch, one should click the compile first to verify your code has no syntax error.
   2. Ctrl + R to verify the code, Ctrl + U to upload the code to Arduino.
   3. Show sketch directory will guide you to the file explorer where the sketch lives in.
   4. You will rarely use this “Upload using programmer”. Programmer here is refer to a ICSP that can write the code into your Arduino using another Arduino but upon writing your bootloader will be erase as well. So next time when uploading the code you have to use ICSP to upload, because without bootloader your ATMEL chips will not know how to read a Arduino code from the serial port.
   5. If you don’t want your friend to copy your code, you can press this “export to binary file” and send the binary file to your friend. Your friend can learn how to use command line to upload a binary file to his or her Arduino (the command line tool for uploading Arduino code is called AVRDUDE)
   6. Library is a set of precompiled code that you can reuse. Include Library is a very important part of Arduino. It allows you to write less code for doing more things.
4. Tools
   1. Auto format (Ctrl + T), if you think your code looks ugly, try ask the IDE to auto format for you!
   2. Serial Monitor to debug your code
   3. Serial Plotter looks like an oscilloscope, capable for debugging for most of the hobbyist project.
   4. At the board session, you need to select the board version and the com port (USB port) you’re using in order to upload your code.
   5. Manage library will help you to download and install library from the website.
   6. Programmer and Burn Bootloader helps us to burn a bootloader to a ATMEL chip to make it a “Arduino” (without the bootloader, you Arduino is just a ATMEL chip)

## How Arduino Upload Code

This session explains how the code in Arduino IDE is uploaded to Arduino board. We split it to two parts, first part is for the layman, who only wants to upload code and get thing tested and done. The second part is a detail description of how a code is compiled and flashed into Arduino.

For the engineers with deadline, before uploading your Arduino code, follow the steps below:

1. Plug in the Arduino board
2. Go to tools tab, select the right COM port.
3. Again, in the tools tab, select the board name, for example, Arduino UNO.
4. Compile the code by clicking verify, the tick icon at the top left, or simply Ctrl + R
5. Flash the code into Arduino, press Ctrl + U

For the curious folks, of all these steps are possible and are so simple is because of the following points:

1. The controller chips we program is nothing but a normal AVR chip, however, an Arduino chip is an AVR chip that’s a bootloader pre-installed.
2. A powerful IDE that wraps around AVR microcontroller tools.

When you click the “Upload” icon, the IDE verify and compile the C++ code in the text editor to binary file, which is a set of instructions that the microcontroller will understand, you won’t see this file because the IDE later send to your board by using a tools called AVRDUDE (AVR download/upload/manipulate). In order to inspect the file, you can click Sketch --> Export compiled binary.

Without the Arduino IDE, we can still upload the code using command line, by typing code such as this:

*“C:\Program Files (x86)\Arduino\hardware\tools\avr/bin/avrdude” -C”C:\Program Files (x86)\Arduino\hardware\tools\avr/etc/avrdude.conf” -v -patmega2560 -cwiring -PCOM9 -b115200 -D -Uflash:w:”C:\Users\SEBAST~1\AppData\Local\Temp\arduino\_build\_415832/yeasu\_v0.ino.hex:I”*

To know more about how to compile and upload code manually, visit this channel: <https://www.youtube.com/watch?time_continue=446&v=Lwpxrq9v1Yw&feature=emb_logo>

Prior the code is flashed to the ROM, the bootloader, a program that’s pre-installed inside the AVR chip, will “press” the reset button upon the arrival of the hex code from our computer. A series of instructions will be carried out and eventually flash the code into the ROM. If you don’t have a bootloader installed, we need a special device (In-system programmer or ISP) to replace the bootloader software. More info please visit: <https://www.arduino.cc/en/Hacking/MiniBootloader>

# 1 Basic Programming

## 1.1 The Structure – A Quick View

Structuring an Arduino Program – As a new programmer to Arduino, you want to understand the building blocks of an Arduino Program, or we often call it as a “sketch”

For example, see the hello world program **(1\_1\_1.ino),** this will be your first sketch in our workshop.

Some of the things might be strange for you, for example:

***delay(millisecond) – keep the program at last instruction for N millisecond. For example, if your last line of code is set the LED to high, then LED will be on high state N second after the code proceed to next line.***

***pinMode(pin, mode) – A GPIO can be set as input or output pin, this function allow us to choose which pin as which mode (input, output)***

***digitalWrite(pin, state) – If the pin is a digital pin, we use digitalWrite to manipulate the output state, we use digitalRead() to read the pin current state.***

These are pre-written functions that’re related to Input/Output (I/O). Check the in-built functions out here: <https://www.arduino.cc/reference/en/#functions>

## 1. 2 Serial.println() – Debugger 101

See example code **(1\_2\_1.ino)**

In this example, notice that the code won’t be running until you open the serial monitor. Some of the new things here are:

***Serial.begin(9600) – Start the serial comport at 9600 baud rates (bits per second)***

***Serial.print()***

***Serial.println()***

***int number = 0;***

***number++***

**Why Serial no need to import library? (Maybe you will know later what’s a library)**

Serial is an in-built function in Arduino language, which is inherited from C++ programming language. So, you no need to manually import Serial, it’s inside the language itself. There’s quite a lot of functions that’s pre-written inside Arduino, check them out again:

<https://www.arduino.cc/reference/en/#functions>

**Why 9600? What does it signify?**

Open the device manager and you will see the COM port, double click one of it and go to Port Settings:

A screenshot of a cell phone

Description automatically generated

Yes, you guess why, because the default bit rate is set to 9600 in your laptop by the driver. 9600 means that for every second the maximum bit you can send is 9600 bits. If you have some unreadable text for example like this `3??f<ÌxÌ􁂠􁂠􁂠ü`³??f<`3??f<ÌxÌ􁂠􁂠􁂠ü`³??f< then consider checking the baud rate.

**Why we use Serial.print and Serial.println instead of just printing “hello world”?**

Serial.print and Serial.println use the COM port of your Arduino UNO to send message back to our laptop and Arduino IDE display that message on the Serial Monitor (Ctrl + M).

Very often, especially in embedded system design, we always reserve a COM Port, to assist us in debugging our program. We can put Serial.println() inside a huge chain of if else statement and see which if else fail to work by printing out the state of the program to serial monitor. Also, we can use serial to send numerical data and port to MATLAB to perform some mathematical operation (FFT) or even plotting!

See example **(1\_2\_2.ino),** it shows what kind of data we can send through serial monitor. If you don’t understand how we format the data, don’t worry you will be learning later. Here you will also learn how to use Serial.plotter, yet another tool for debugging your program.

**What are the other functions of Serial?**

We will use other serial functions heavily when we are talking about serial communication with another machine (laptop or even another Arduino). For the full documentation, check this out: <https://www.arduino.cc/reference/en/language/functions/communication/serial/>

**What is the other Arduino in-built function? Why we use them?**

Again, there’re many in-built functions in Arduino, and there’re categorise by their functionality, such as Math, Random Numbers, Bits and Bytes, Characters, time and so on, again refer to the documentation: <https://www.arduino.cc/reference/en/#functions>

## 1.3 Variables and Constant – The Changing and Unchanged

**What are the data types available in Arduino?**

For primitive data types, there’re byte, char, double, float, int, long, short, unsigned char, unsigned int, unsigned long, void, bool.

For other derived data types, array, String(), size\_t.

To see each of their size occupied in Arduino, check out example **(1\_3\_1.ino)**

**How to define a constant in Arduino? What are their advantages and drawbacks? Common use case?**

Use case would be defining pin number using constant, defining update interval time, defining the Wi-Fi password, ssid and so on.

Let’s see an example on how constant might be helpful at **(1\_3\_2.ino)**, you will also learn the technique of delta time to blink our LED in example 1\_1\_1.ino without using delay. Also, you will use a special operator called ternary operator to do the decision. It’s important to get rid of delay, since we don’t want to put so many delays, during the delay time our CPU is totally idle and wasted.

**What are the variables common use case?**

When storing real time clock, like millis() we saw at last session. We use unsigned long, since you know that seconds won’t go negative, and millis() use the unit of millisecond, which will end up to a very big integer.

Boolean are widely used, the HIGH and LOW is actually Boolean true or false.

Byte or often known by embedded programmer uint8\_t (unsigned integer 8) can store 8-bit binary from 00000000 to 11111111 which correspond to 0 to 255 in decimal.

Char is same as byte, but it’s signed, means it store number from -128 to 127. The compiler will interpret it as character. If you expect the output is a character, then use char.

Float will be useful when we want to map a range of analogue value, say 0 to 1023 to 2V to 4.5V, the display of 2V to 4.5V required certain level of precision.

## 1.4 Operators – + - x /

**What operators in Arduino have?**

Operators can be categorized into 5 types: arithmetic operators, comparison operators, Boolean operators, pointer access operators, bitwise operators, compound operators. Each of the operators, we picked some important one and show them in example, so that you know how to use them in your project.

Let’s see example **(1\_4\_1.ino)** to add, subtract, multiply and divide some value.

Let’s see example **(1\_4\_2.ino)** to know how to increment or decrement a variable.

Let’s see example **(1\_4\_3.ino)** to round up and down value.

Let’s see example **(1\_4\_4.ino)** to constraint some value.

Let’s see example **(1\_4\_5.ino)** to generating random number. The circuit is shown below:

A circuit board

Description automatically generated

## 1.5 Control Statement – Time for making decision

**What are the control statements in Arduino?**

If, else, else if, switch case break, ternary operator (see example 1\_3\_2.ino)

**What’s their common use case? How to use them practically?**

See example **(1\_5\_1.ino)** which demonstrate how to use a switch case for our LED blinking project.

See example **(1\_5\_2.ino)** which demonstrate how to compare numerical value and character, which is also stored numerically in Arduino, and we like to call it ASCII. Comparing string might not be straight forward, we have to use a built-in function strncmp() to do it.

For the example above, we also demonstrate how to read the input from serial monitor using Serial.read(). Now, could you write your own version using if else and elseif?

## 1.6 Repetition – Let the machine do the boring stuff

**I already know for loop while loop and do while loop, but what and how can I use them in my project?**

As you already have seen so many examples on looping, in fact the void loop() is an infinity loop which let you run your code over and over in real time, now let’s put more examples on loop to strengthen your skills:

you can make your own void loop() by writing these codes:

for(;;), while(true)

Basically, they’ll run whatever is inside their body forever.

Serial.setTimeout(), Serial.parseInt() or use while(Serial.available() > 0), you should learn how to distinguish if(Serial.available()) and while(Serial.available()>0). See example **(1\_6\_1.ino) (1\_6\_2.ino)**

## 1.7 Array – Working with a chunk of values

**What are the types of array?**

Array is just a chunk of same type of values. You can have an array of integer, an array of float, an array of character etc.

**How to initialize and how’s the assignment of array? How to use an array?**

One can declare an array without assign them value like this:

*int arr[5];*

Here int is the variable type in the array, 5 in the bracket signify how many member or value inside an array. arr is just the name of the array.

To assign value in an array, you can try the followed:

*int arr[5] = { 1, 2, 3, 4, 5};*

Here we give the array some value from 1, 2, 3, 4, 5. Each value is having a “position” information. The value at the left most, is said to be at the “0-th” position, we like to call the position as index of the element. Followed is the values of the array and their corresponding position information:

value : 1 2 3 4 5

index : 0 1 2 3 4

We can use the index or position number to get a value from an array. For example, arr[4] = 5, arr[0] = 1 etc.

**What can we do to an array? Append? Insert?**

Once an array is initialized, we cannot append or insert extra variable in the array with fixed size. However, one can change the value of the member inside array, for example, now we want the value at first position to be 9, we do:

*arr[0] = 9;*

**Can the array size be dynamically changed?**

Unfortunately, no, unlike array in Python or Matlab, once you declare an array, its size is fixed to that particular one.

**Is string an array of character? If so, can we make our own character string?**

Yes, a string is nothing but an array of character, terminated with a “\0” or null symbol at the end of the array to signify the end of the string. For example: char arr[6] = { ‘H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘\0’ }

**See example (1\_4\_5.ino) for how to use two-dimensional array.**

## 1.8 Function – Structure Your Code into Block

**How to write function in C and in Arduino?**

Writing function in C is the same as writing one in Arduino basically a function must have the following form:

A close up of a logo

Description automatically generated

Some concept you got to be clear:

1. Return type must be specified: void, int, float, char…
2. Parameters may be empty.
3. Function Body must enclose with curly bracket.
4. A function with return type void can exist without return statement.
5. Once functions encounter return state, it will immediately stop the process and return the corresponding value.
6. Any variable(s) that is/are defined in the function body, only have lifetime when the function is executing. When function is done executed, the variable will be erased from the memory.

Below are the instructions on how you can create function and put them into separated files:

1. First thing of the blinking LED light example, you want to have a function, say blink() to execute the blinking routine, so now create a new tab called it as blink.ino: A screenshot of a cell phone

   Description automatically generated
2. Click new tab to open a tab, the IDE will prompt you to enter a name for the new sketch, just use any name and hit enter:A screenshot of a cell phone

   Description automatically generated
3. After that, copy the helloWorld() function inside the new tab:A screenshot of a cell phone

   Description automatically generated

You should use this to break down your code into separated files in order to ease your eye from seeing too many codes in one page! Also, you can team work with your friend to write different function for different component, for example, one file is for the ultrasonic sensor, one file is for the LCD display screen.

## 1.9 Library

**What the heck is library?**

Library are files written in C or C++ which provide us sketches with extra functionality, for example, control a LED matrix, read from a sensor.

**Where we can see our library?**

To use an existing library in a sketch simply go to the Sketch menu, choose "Import Library", and pick from the libraries available. This will insert an #include statement at the top of the sketch for each header (.h) file in the library's folder. These statements make the public functions and constants defined by the library available to your sketch. They also signal the Arduino environment to link that library's code with your sketch when it is compiled or uploaded.

**How to download library?**

To install your own library, create a folder inside ARDUINO/hardware/libraries with the name of your library. The folder should contain a C or C++ file with your code and a header file with your function and variable declarations. It will then appear in the Sketch | Import Library menu in the Arduino IDE.

Task 1: We’re going to use LCD display to display “Hello world”, however, you know that control the LCD is very complicated, so can you find a library to install in Arduino IDE?

Task 2: Let’s say your lecturer ask you to build a colour sorting machine next time, and the sensor you have got is this:

## 1.9 Project Sharing

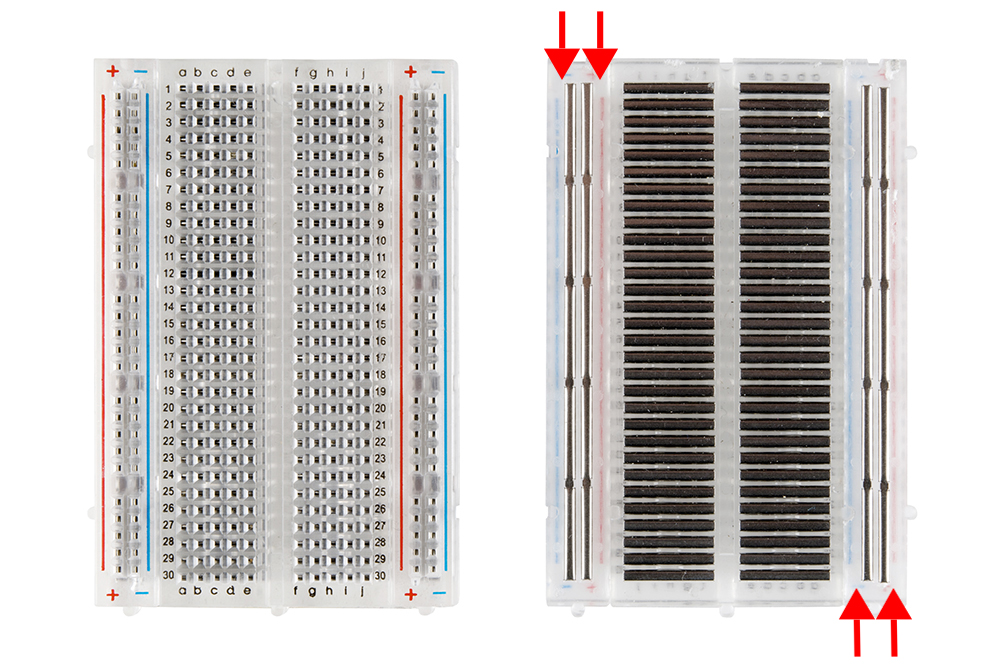
Here’s the project that I’m current working on. It’s about to control the rotator in order to track the satellite on Sky. Let’s go through this code and see how much we can comprehend.

The link is here: <https://github.com/rebeccaljl/YeasuRotatorController>

# 2 Fundamental of Electronics

## 2.1 How’s a bread board looks like?

Behind every bread board is nothing but some soldered lead attach to the board underneath the white cover. Let’s see how it’s looked like inside:



Electrical current will pass through the wire, so make sure two legs of the component are not place at the **same row.**

## 2.2 Series and Parallel

## 2.3 Voltage Divider

## 2.4 Ohms Law, voltage source and current source.

Typical ATMEL chip can source 800mA current. So be careful of your motor, and other sensors, make sure when they’re operating, the current the sunk total not more than 800mA. Normally, we can calculate the current by looking at the datasheet. Say, a motor with maximum power of 5W with supply voltage of 5V, the current it will take is I = P/V = 1A, so definitely our Arduino cannot supply the motor. You might need to use an external power supply.

## 2.5 GPIO, Pull-up Pull-down and Open Drain Push Pull

When GPIO is at output mode, it can be treated as the model like below:

A close up of a logo

Description automatically generated

Output pins can be driven in three different modes:

1. Open-drain – A transistor connects to low.  
   A screenshot of a cell phone

   Description automatically generated

When the signal form MCU is high, Q1 short through, Pin will be shorted to ground.  
When the signal from MCU is low, Q1 is open, then what’s the state of the Pin???? (ans: nobody knows! So, what should we do…)  
So, who bother to know this? As we go to the I2C session, you will see what’s the problem.

1. Push-pull – A transistor connects to high, and a transistor connects to low. Basically, takes the form of a normal GPIO output mode.

Input pins can be a gate input with a:

1. Pull-up – A resistor connected to high
2. Pull-down – A resistor connected to low

Pull up and pull down are widely used in button circuit. Let’s draw an example!

## 2.6 Passive Component – Resistor (photoresistor, potential meter)

How a potential meter works?

A close up of a logo

Description automatically generated

We can form a voltage divider circuit from the diagram above. However, can you see the disadvantages of using a voltage divider?

Potential circuit is commonly used when we want to create analogue value.

How a photoresistor works?

More light intensity 🡪 Resistance drop. (valance band to conduction band)

## 2.7 Types of Signal – Analogue, Digital

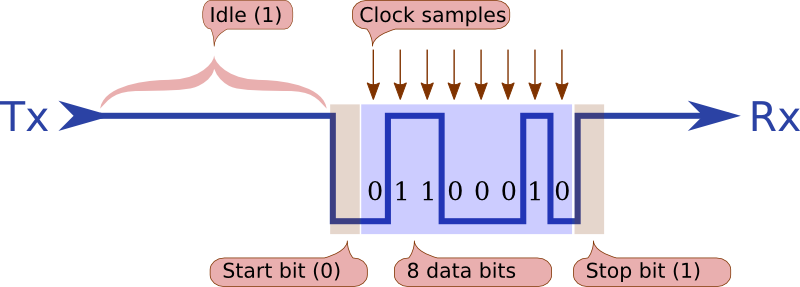
Signal can be categorized into two types, one is digital, which only holds a finite set of possible value. For example, the digital signal only takes two values, whether high or low. A button, which can only be pressed or release, a switch, can be open or close. In contrast, analogue signal takes infinite set of value, but these values are limited to a range of maximum and minimum values. For example, the light intensity from dim to bright, the angle of servo motor, the volume of a sound, the input from a potentiometer…

In order to read analogue signal to a digital system, such as Arduino, we need to have analogue to digital converter or ADC. An ADC is graded by its resolutions and sampling rate. For example, Arduino has a 10-bit ADC, which can divide the signal from 0 to 5V into 2^10 division (so how small is per division, Ans 4.88mV). If your signal level goes below this resolution, the ADC will not do the conversion for you!

## 2.8 Machine Interface – Serial Communication, I2C, SPI

Serial, I2C and SPI are serial communication protocol, they’re like the agreement between machine, which defined how they’re going to talk to each other. Serial means bit by bit, in contrast to parallel, which bit are send parallelly at the same time from one end to another.

Arduino Serial is the simplest to use protocol for beginner, it’s the Recommended Standard 232 or simply RS232, which originally introduced in 1960 for serial communication transmission of data.



This is how the RS232 works:

1. When the streamline has nothing to do, i.e. “idle” time, there will be a continuous “1” bits.
2. When data arrive, in order to signal the start of a byte, a start bit is always indicated(0V).
3. After the receiver sees the start bit, it waits for 1.5 times the sample time (what’s the sample time?), so that it skips the start bit and sample half-way through the next bit.
4. The purpose of stop bit is to ensure that there definitely is a 1-bit between each byte. Without the stop bit, if a byte ended in a zero, then the hardware won’t be able to tell between that and the start bit of the next cycle.

Few things to be careful when using Arduino serial:

1. The baud rate of the sender and the receiver has to be synchronised, i.e. same baud rate
2. Don’t plug RS232 cable wire straight into TX RX of Arduino, since the old RS232 use 12V and -12V and Arduino use TTL (0V to 5V) to represent 0 and 1.
3. In Arduino there’s only one serial port, when you upload your code, the serial port is occupied. To extend the serial port, we can use software to implement the protocol. There’s a library called SoftwareSerial that lets you do serial communications in a software rather than hardware. This has advantage that you can use different pin configurations for serial communications.

Some simple calculation for the curious cat:

If the baud rate is 9600, how fast in the sample rate (time spread between sample and sample) in microsecond? (ans: 104.16us)

I2C is a serial communication protocol invented and developed by Philips. It uses only two wire to connect up to 128 devices with 7-bit addressing. The two wire are SDA, used for transmitting data and SCL for setting the clock frequency. The clock signal is always generated by the master.

A close up of a logo

Description automatically generated

Communication via I2C takes the following step:

1. Start condition: Before the address can be sent, the master device **pulls SDA low while leaving SCL high.** If two master were to send data at the same time, whichever pulls SDA low first wins the race and gain control of the data bus.
2. Master put the address frame.
3. Slave acknowledge the address.
4. (Optional) Master put the address which the data is stored inside the slave.
5. Slave acknowledge the address.
6. Finally, master read/write the data to the address.
7. Stop condition: pulling the SDA to high while keeping SCL high.

Few things to be aware of when using I2C:

1. Normally the I2C port in microcontroller is open drain, meaning that they can put signal low but cannot pull signal to high. So, you use a 4.7k Ohms resistor connect two bus to 5V.
2. If above cannot, try short the ground of two system to each other.

A close up of a piece of paper

Description automatically generated

# 3 Input

## 3.1 Types of Buttons? How to connect a button to MCU?

A circuit board

Description automatically generated

Task 1: Combine the schematic with a LED. Be careful of the circuit, too much current will kill the LED! Suggest use a resistor to series with it.

A circuit board

Description automatically generated

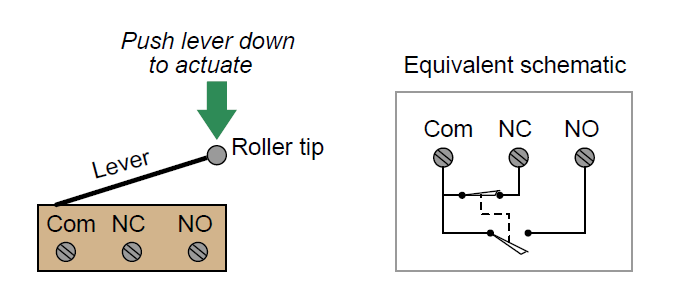
Task 2: Write an Arduino code, that will allow user to press a button. Upon pressing, the LED will light up, and you need to hold it in order to let the LED light up constantly. (Hint: If you don’t have an LED, remember there’s a built-in LED in Arduino)

**(3\_1\_1.ino)**

Task 3: Make the button a toggle button, say, when press the button and release, the LED will stay on until the next time the user presses the button and release. Remember the action here is press and release, just like you switch on your TV and radio in your house.

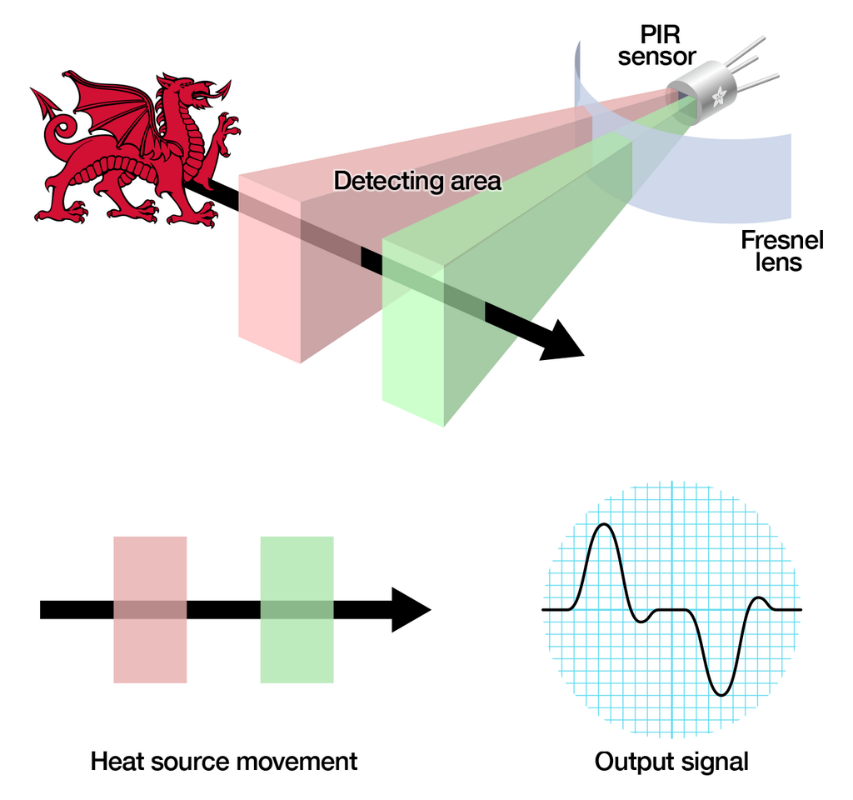
**(3\_1\_2.ino)**

## 3.2 Limit Switch



(Noting much to explain…... if the tip is pressed, COM and NO will be shorted. If not pressed, it’s short to NC. NC stands or normal close, NO stands for normal open.

## 3.3 Motion Sensor

A picture containing object, antenna

Description automatically generated

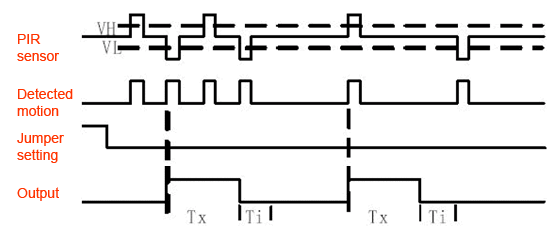
**How to configure your PIR sensor?**

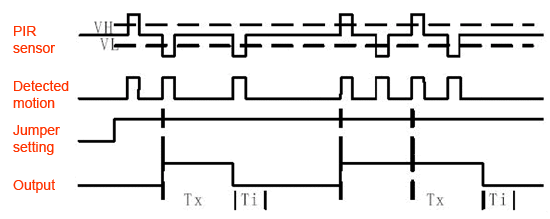
There’re few things you can do with your PIR sensors, there’re:

1. The delay-time Tx and Ti
2. The mode of triggering
3. The sensitivity

"Tx" timeout: how long the LED is lit after it detects movement - this is easy to adjust on Adafruit PIR's because there's a potentiometer. "Ti" timeout which is how long the LED is guaranteed to be off when there is no movement. This one is not easily changed but if you're handy with a soldering iron it is within reason.

The mode of triggering is configured by putting the jumper at the H or L position. H position is retriggering mode, and It works like this: when motion is detected, it set the state pin high, and if motion is still there, it will keep the state pin high until the motion is removed. L position is non-repeatable triggering mode, and it works like this: when motion is detected, it set the pin high for Tx second, and after Tx second, it reset the pin to low, regardless the present of motion. If you can read the timing diagram below is a plus point for your understanding:





Top one: Non-repeatable trigger mode, Bottom: Repeatable trigger mode

Sensitivity can be tuned using the second resistor, the sensitivity can be up to 7m in theory.

Task 1: Wiring the PIR module. Write a code and use serial plotter to show the pin state and compare the two mode.

**(3\_3\_1.ino)**

Task 2: Light up a LED when a motion is detected.

**(3\_3\_2.ino)**

Task 3: Suggest a method, in Vetro, the light is on when motion is detected. Do you think using retriggering mode is better or non-repeatable trigger mode is better?

Task 4: What do you think, for your pet feeder, which mode is better? (You don’t want your kitten keep wandering around the machine and get so many foods until it eats so many and get so fat?)

## 3.4 Distance Sensor

Distance sensor emits an ultrasound typically at 40kHz and travel through the air. If there’s any object block the travelling path, the signal bounce back, and capture by the receiver. Knowing that the travelling speed of sound wave in the air, and the time elapsed when travelling, we can know the distance between the sensor and the object. Note that the time of which the wave travel is same as the pulse width of the Echo signal.

A screenshot of a social media post

Description automatically generated

The distance S obtained from above is the distance which the wave travel front and back, so at the end it has to be divided by 2, we obtain

Task 1: Wiring up the module with LED and distance sensor with Arduino.

Task 2: Write a code to display the distance of you and the celling on the Serial.Monitor()/

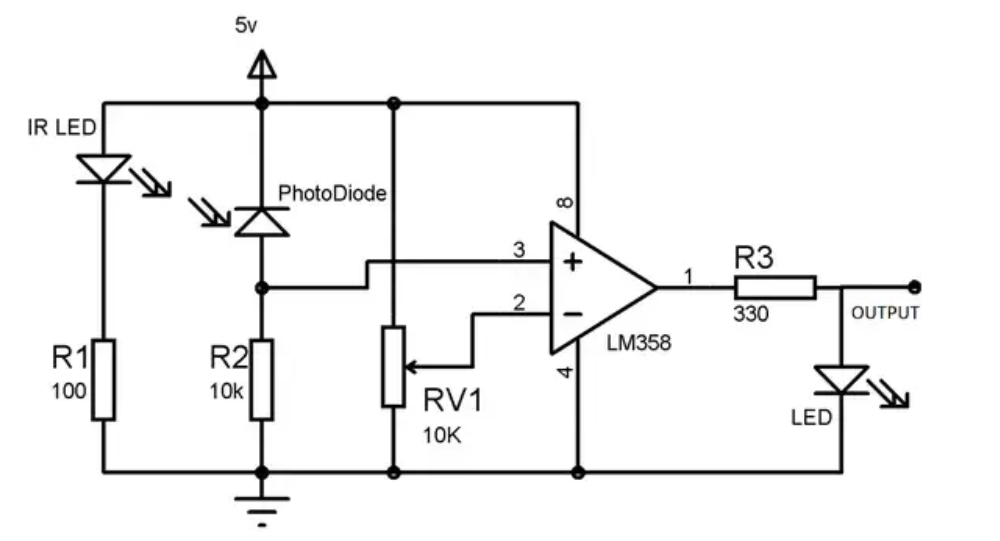
**(3\_4\_1.ino)**

Task 3: Slowing moving an object close to the sensor, if the object is 20cm or less in front of the sensor, light up the LED.

**(3\_4\_2.ino)**

## 3.5 IR Sensor

So far, the simplest sensor, consist of an IR emitter, a photodiode (detector) and a signal amplifier (Op-Amp)



The working principle goes like this:

1. Common sense: white surface reflect radiation while black surface absorbs it.
2. When IR LED emitting light and encounter white surface, all the IR reflected and photon hit on photodiode which is connected reverse bias, the resistance of the photodiode starts decreasing rapidly and the voltage drop across the photodiode also decrease.
3. Using the potentiometer, we can change the IR sensor sensitivity.

Task 1: A simple task, plot the state of the IR sensor to the

# 4 Output (Visual, Physical and Audio)

## 4.1 LED

LED has to be connected with the output pin in series with a resistor, because an LED cannot take too much current.

Let’s check the current limit of the LED from one of the datasheets:

A screenshot of a cell phone

Description automatically generated

Using voltage divider rule, would you be able to find out the value of the resistor?

## 4.2 Servomotor

There’re three types of servo motor:

1. Positional rotation servo: Output shaft rotates in half circuit (180 degrees). It has gear mechanism to prevent turning beyond the limit.
2. Continuous rotation servo: Can turn in any direction independently and continuously.
3. Linear servo: Uses rack and pinion to rotate the motor back and forth, rarely used and rarely see in daily life.

Task 1: Set a rotational servo to rotate to 90 degree for 3 second, then return to 0 degree.

**(4\_2\_1.ino)**

Task 2: Design a candy sorter, that takes the candy from funnel, and sort the candy to different container (how many servomotors we might need?)

Task 3: For mechanical student, could you calculate the maximum weight a servomotor can lift for a given r?

## 4.3 LCD Display

We do not need an LCD shield to interface with Arduino. In fact, we only need 5 digital pins to set up our LCD and display something.

Follow the schematics below to see how to connect to LCD on Arduino UNO:

A circuit board

Description automatically generated

Here’s the description of the pin:

Vcc and ground I don’t need to explain anymore…

Vo pin on which we can attach a potentiometer for controlling the contrast of the display.

The RS pin or register select pin is used for selecting whether we will send commands or data to the LCD. For example, if the RS pin is set on low state or zero volts, then we are sending commands to the LCD like: set the cursor to a specific location, clear the display, turn off the display and so on. And when RS pin is set on High state or 5 volts, we are sending data or characters to the LCD.

Next comes the R / W pin which selects the mode whether we will read or write to the LCD. Here the write mode is obvious, and it is used for writing or sending commands and data to the LCD. The read mode is used by the LCD itself when executing the program which we don’t have a need to discuss about it in this tutorial.

Next is the E pin which enables the writing to the registers, or the next 8 data pins from D0 to D7. So through this pins we are sending the 8 bits data when we are writing to the registers or for example if we want to see the latter uppercase A on the display we will send 0100 0001 to the registers according to the ASCII table.

And the last two pins A and K, or anode and cathode are for the LED back light.

A circuit board

Description automatically generated

Task 1: Display “Hello World” on the screen

**(4\_3\_1.ino)**

Task 2: Connect a potentiometer, as if it’s the angle of the servomotor (0 – 5V correspond to 0 to 180 degree), read the voltage from the potentiometer, convert it to angle, hence display on the LCD screen.

**(4\_3\_2.ino)**

# 5 Communication, I2C SPI Serial Revisit

## 5.1 Bluetooth Module

The common Bluetooth module we use daily is the HC05 or HC06. When operating can draw about 30mA. There’s a microcontroller on the module which handle the data and transmit it out using the Bluetooth antenna and its serial port is enabled to communicate with another microcontroller.

Let’s have a look of the pinout of the typical HC05 module:

A circuit board

Description automatically generated

I will explain what’s the function of each pin:

|  |  |  |
| --- | --- | --- |
| Pin Number | Pin Name | Description |
| 1 | Enable/ Key | This pin is used to toggle between Data Mode (set low) and AT command mode (set high). By default, it is in Data mode |
| 2 | VCC | Powers the module. Connect to +5V Supply Voltage |
| 3 | Ground | Ground pin of module, connect to system ground |
| 4 | Transmitter | Transmits serial data. Everything received via Bluetooth will be given out by this pin as serial data |
| 5 | Receiver | Receive Serial Data. Every serial data given to this pin will be send out via Bluetooth |
| 6 | State | The state pin is connected to the on-board LED, it can be used as a feedback to check if Bluetooth is working properly. |

Some of the default setting for the first-time user:

1. When paring with Bluetooth, the default password is 1234 or 0000
2. Normal data transfer use 9600 8N1
3. AT command transfer use 38400 8N1

In this workshop we will not be using AT command to heck the setting inside the module, we will focus on the data transfer.

Now the following task will let you master the HC05 module:

Task 1: Research how the “state” pin works. Would you be able to show if the Bluetooth is connected on Arduino or LCD using state pin?

**(5\_1\_1.ino)**

Task 2: Research how to use softwareSerial to read data from Bluetooth. Try googling “SoftwareSerial Arduino”.

**(5\_1\_2.ino)**

Task 3: Now download phone app like “Bluetooth Series” or “BLYNK” to try send something over Bluetooth. Here, we use “Arduino Bluetooth Controller” that you can download from the Android market.

**(5\_1\_3.ino)**

## 5.2 I2C

For I2C, we ask two groups to pair up and we will be using I2C to let two Arduinos communicate to each other.

The sender code is in (**5\_2\_1.ino)**

The receiver code is in **(5\_2\_2.ino)**

# Putting All Together

Step of prototyping a product: The Block Diagram, The schematic, Try calculate the power consumption, Testing individual module, Construct the algorithm flow chart, Finally Putting All Together

# Optional: NODEMCU For IoT using MQTT

The MQTT Model

The MQTT Client

The MQTT Server and the host