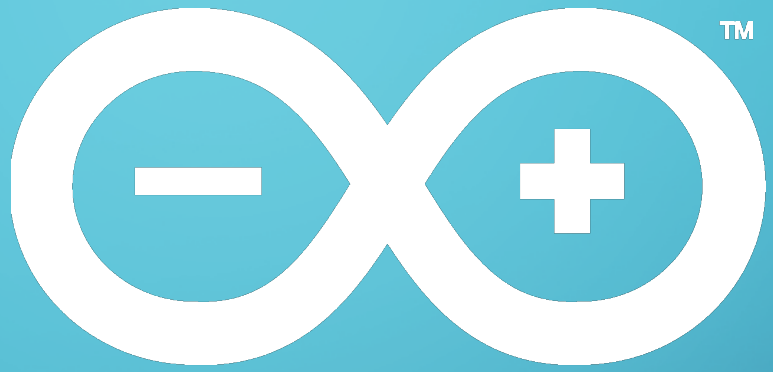


ARDUINO WORKSHOP

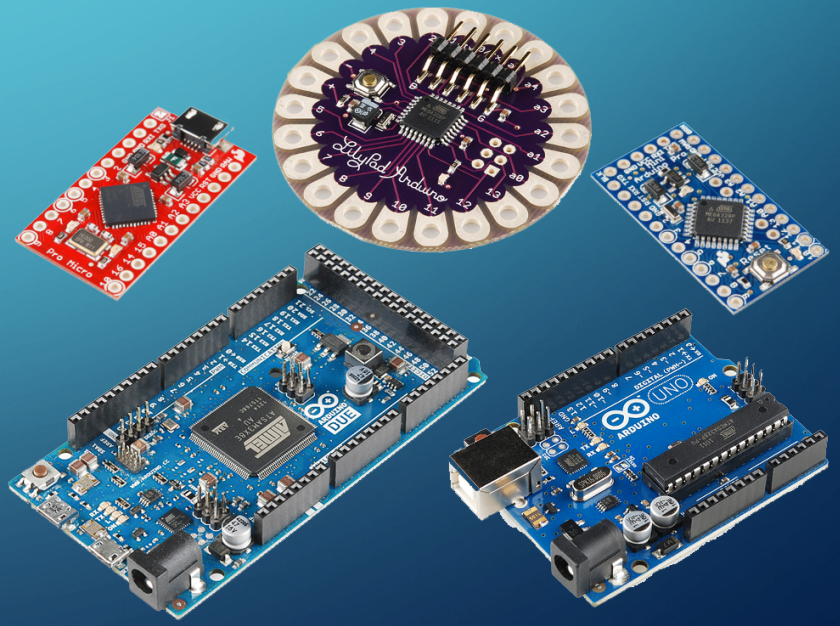
SUTD IEEE STUDENT BRANCH

WORKSHOP OUTLINE

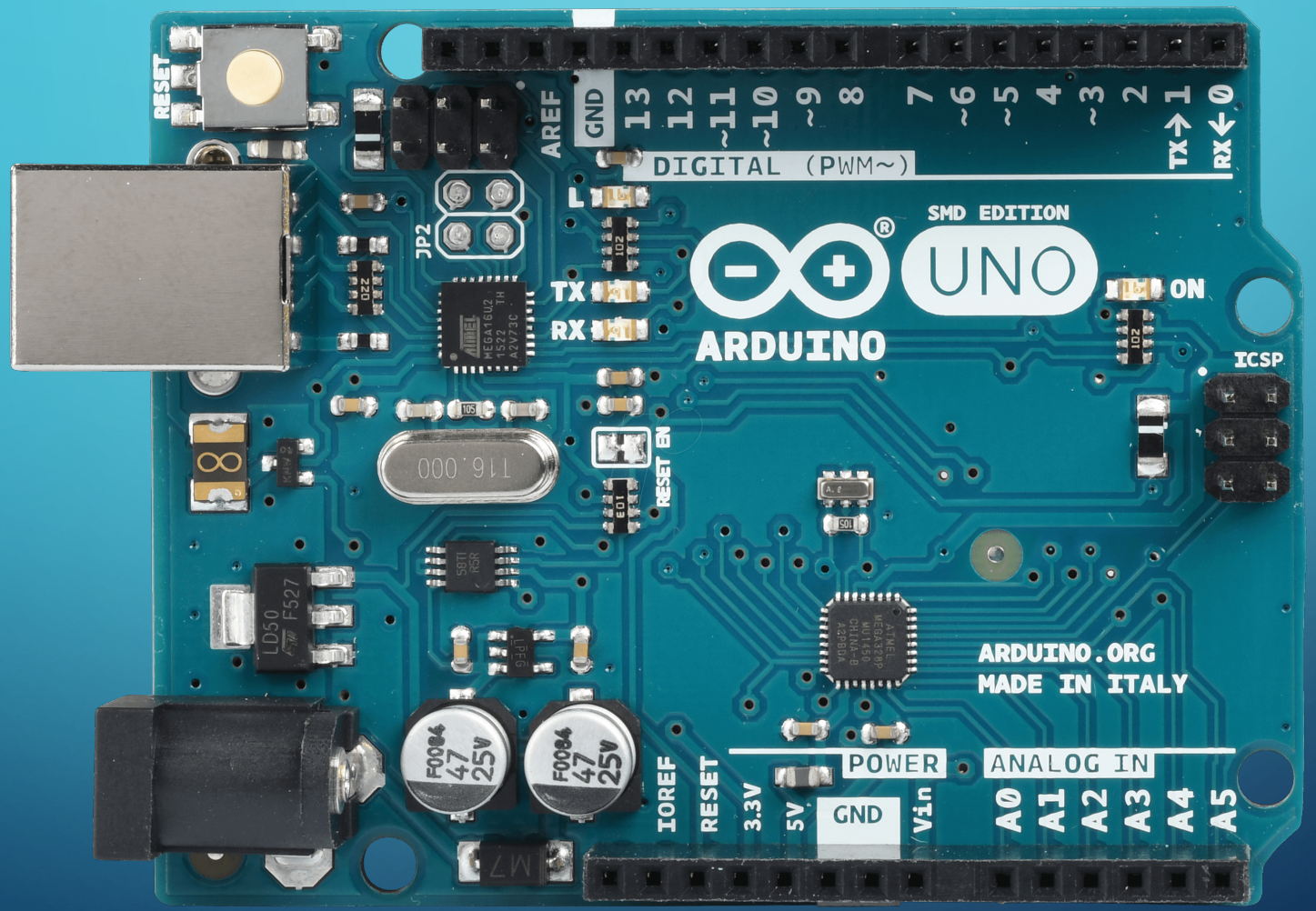
1. Introduction to Arduino
2. Electronics Concept
3. Programming with C++
4. Self Exploration

WHAT IS ARDUINO?

- Open source physical computing platform
- Sense and control the physical world via software



ARDUINO WITH UNO BOARD



THE ARDUINO WORKFLOW

Input → **Board** → **Output**

-Software (Programming)

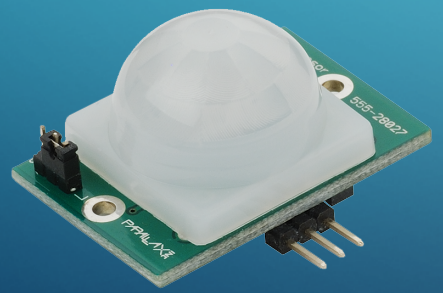
-Various Sensors

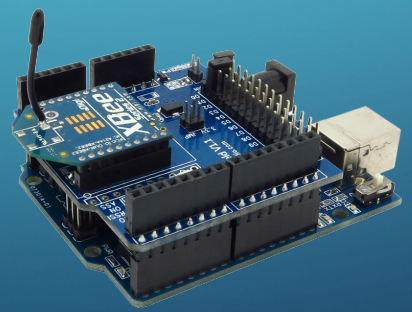
-Connections

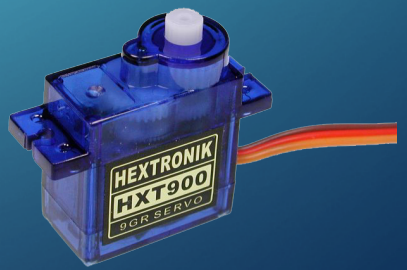
-Modules

-Shields

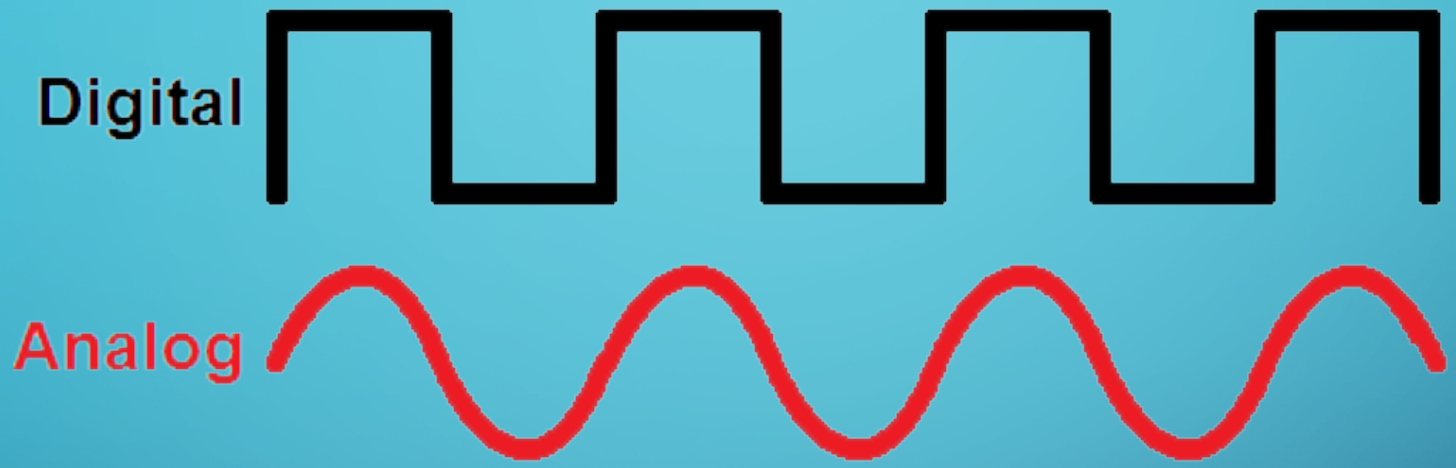
-Actuators







INPUTS: DIGITAL VS ANALOG



DIGITAL

- Discrete Values
- High/Low
- On/Off
- 0/1

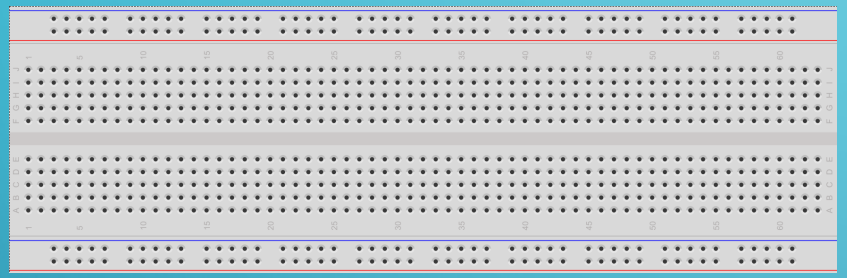
- eg: Switches

ANALOG

- Continuous Values
- Real Numbers
- Arduino maps 0-5V to values 0 -1023

- eg: LDR/ IR Distance Sensor

BOARD : CONNECTIONS



BREADBOARD

- Columns Connected (Terminal Strips)
- Rows Connected (Power Rails)

JUMPER WIRES

- M to M
- M to F
- F to F

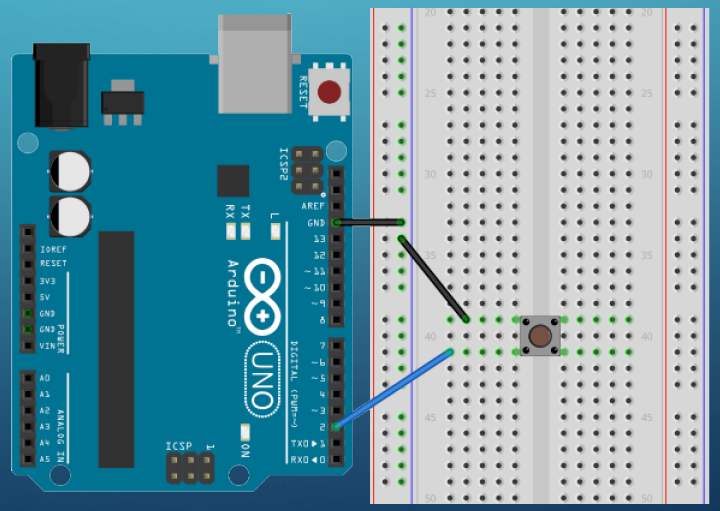
BOARD : WIRING

1. Shorting The Circuit

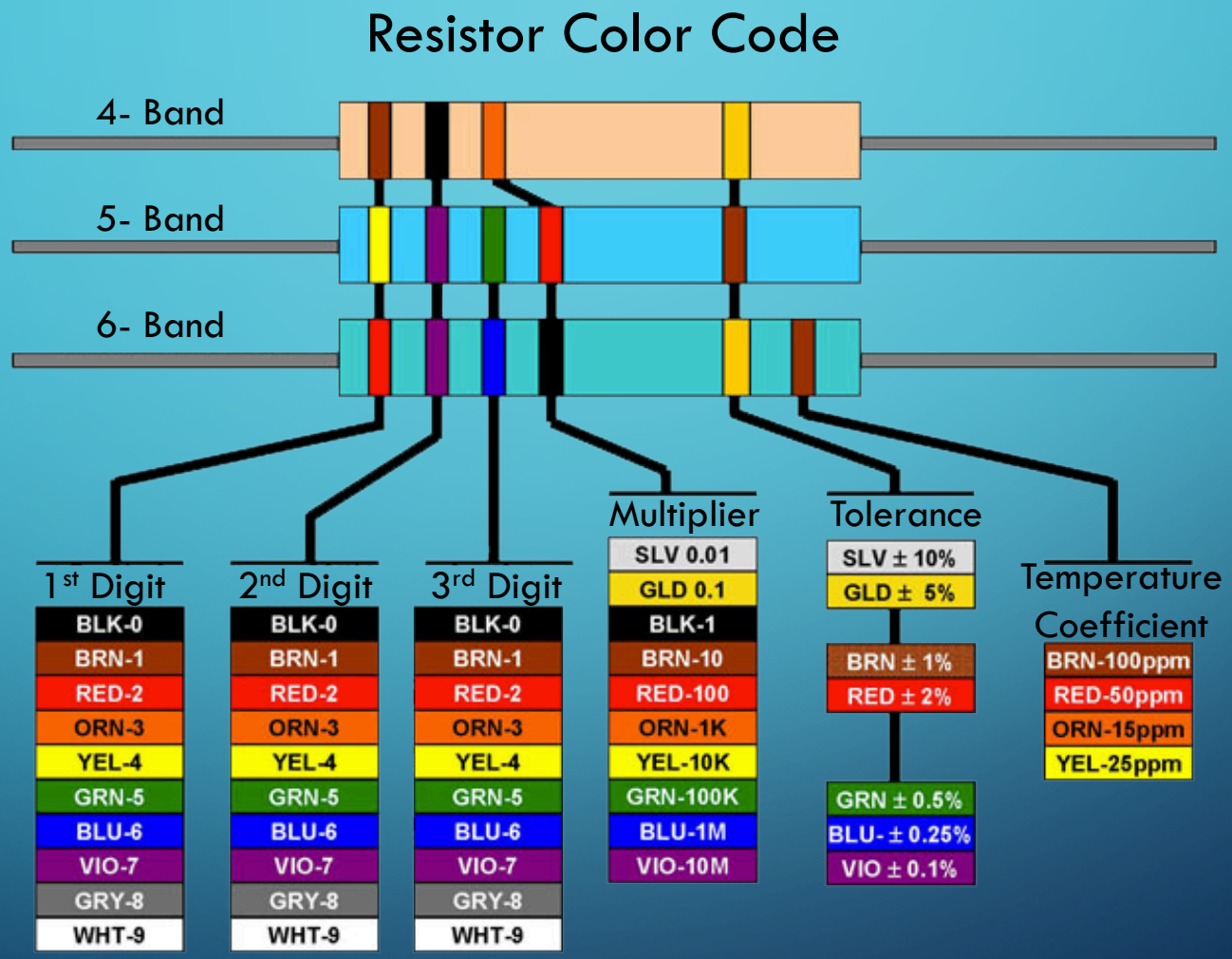
- Do NOT connect 5V directly to Ground

2. Floating Pins

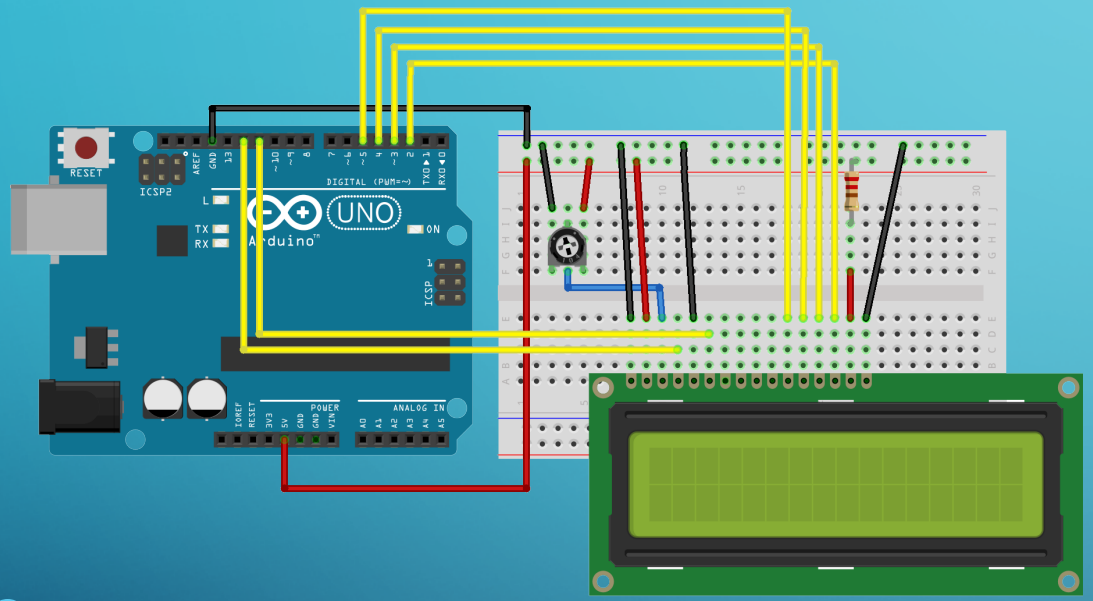
- Inputs must have definite connections…otherwise the input value will float (Give False Positive/Negative)
- Floating happens because electrical components act as “antennas” and are susceptible to electrical noise.
- Can be solved by adding pull up (to 5V) and pull down resistors (to 0V).



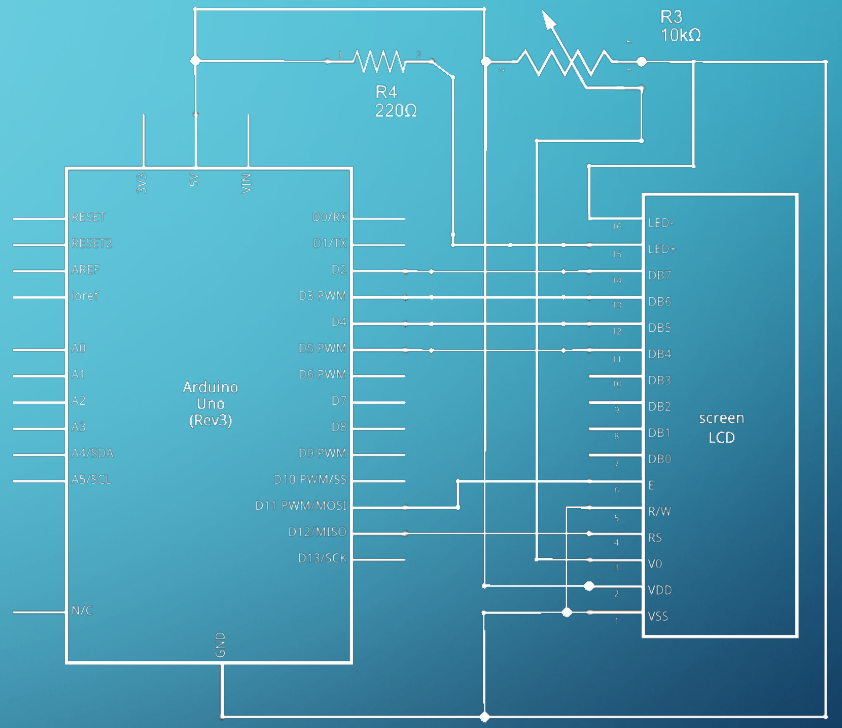
BOARD : WIRING WITH RESISTORS



BOARD : WIRING/SCHEMATIC DIAGRAMS



Wiring Diagram



Schematic Diagram

ACTUATORS : TYPES

1. Light : LED/LCD/Lasers

2. Sound : Buzzers/Beepers/Speakers

3. Motion : Servo/Stepper/DC Motors

4. Power Control : Current Transmission

5. Software : Database/App Updates

BASICS OF CODING

<*Hello World*/>

1. Types of Languages

- High Level → Easy to comprehend, complex in nature
- Low Level → Hard to comprehend, simple in nature

Different Purposes

```
1  print "Hello World"
```

Python

```
1  class HelloWorld{
2      public static void main(String[] args){
3          System.out.println("Hello World")
4      }
5  }
```

Java

```
1  00100010010010000110
2  01010110110001101100
3  01101111001000000101
4  01110110111101110010
5  01101100011001000010
6  0010
```

Binary

BASICS OF CODING

<*Hello World*/>

2. Code logic is generally applicable to all languages.

3. Use good coding practices (structure/comments/effeciency)

4. Practice, practice and practice

PROGRAMMING WITH C++

1. Technical Reasons

   - Maps written code efficiently to machine language for the CPU

     → Arduino runs machine code compiled from C++

2. Open Source Reasons

   - Highly portable language

     → Can be easily run on other devices.

   - Very famous language

     → Well documented resources present

PROGRAMMING (CONCEPTS)

1. Variables
   - Used to store data values
   - Have naming conventions
   - Local VS Global
2. Boolean
   -True / False
3. Integer
   -Whole Numbers
4. Float
   - Real Numbers (with d.p)
5. String
   -Sequence of characters in quotation
6. Function
   -Blocks of reusable code

Other: Lists/Tuple/Dictionary/Class

PROGRAMMING (SYNTAX)

1. ; semicolon ends a statement
2. {} curly braces contain a function
3. () brackets contain conditions/arguments
4. // single line comment
5. /**/ multi line comment
6. = Assignment Operation
7. == Equal Operation

PROGRAMMING BASICS

1. If Else Statements

2. For/While Loops

3. Functions

PROGRAMMING

→ Arduino IDE

→ Simplified C++

1. Program Structure

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Initializes everything upon start/reset (e.g. pins/functions/etc)

Main code block goes here

* void loop() and void setup() are a must for every Arduino Code
* void means that nothing is returned when the function is executed

PROGRAMMING

1. Setup & Read/Write Functions

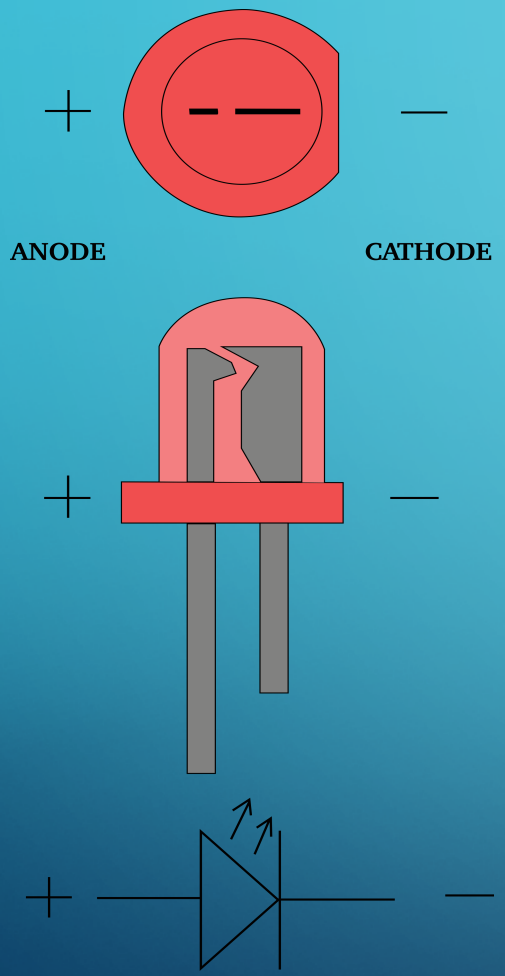
```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

pinMode (pinNumber, INPUT/OUTPUT)

Only for digital pins

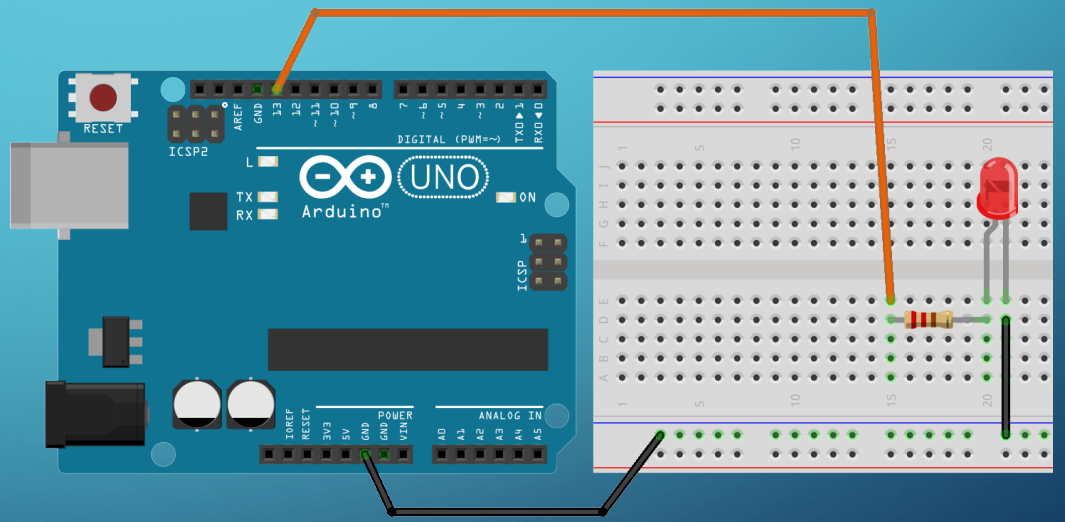
delay(ms) : Pauses the program for a specified time period, to see fast electronic actions/to prevent mechanical “bouncing”.

COMPONENT #1 : LED

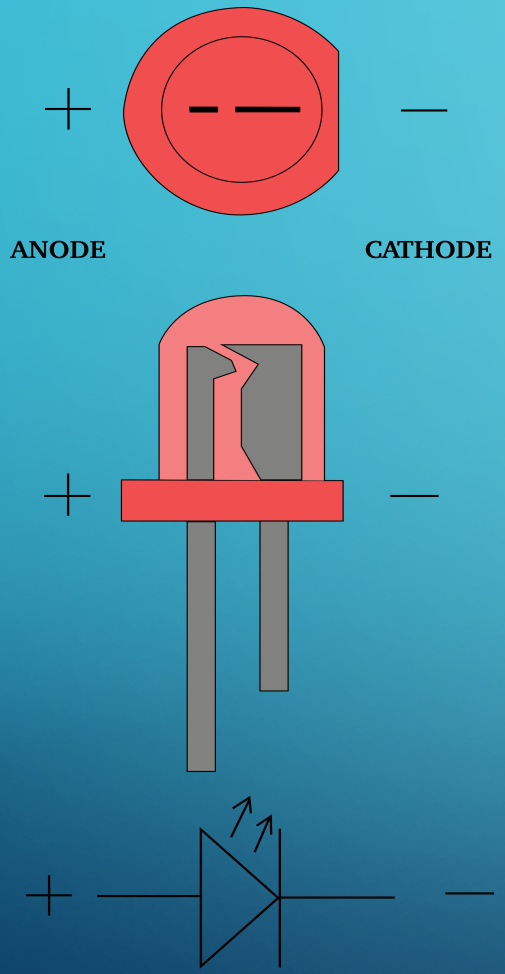


- LED's are forward bias

- Only work when current flows from anode to cathode



COMPONENT #1 : LED



LED EXERCISES

1. Make the LEDs blink slower/faster
2. Attach two more LEDs

PROGRAMMING : IF ELSE STATEMENTS

General Format

```
if (buttonPressed is true) {
    Turn on LED }
else {
    Turn off LED}
```

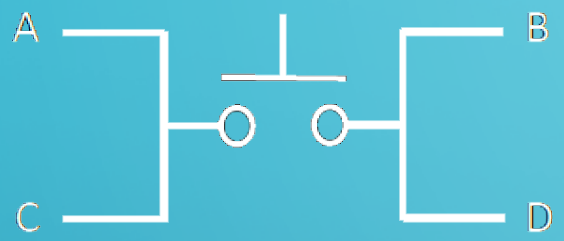
Condition Comparators

- ==
- !=
- <
- >
- <=
- >=

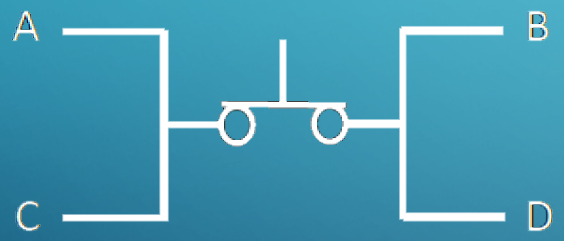
Additional Comparators

- And : &&
- Or : ||
- Not : !

COMPONENT #1 : LED + BUTTON



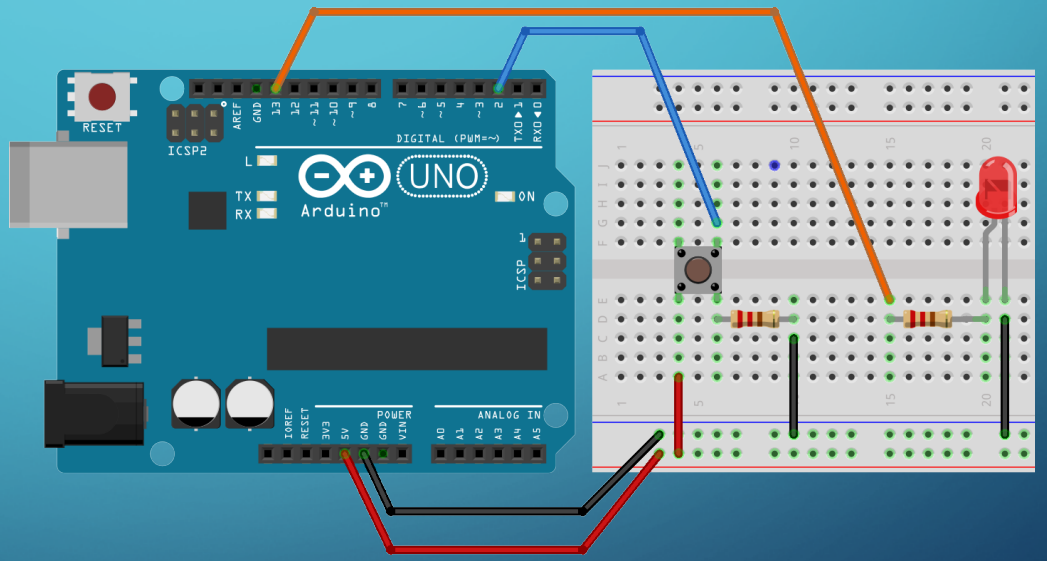
Not Pressed



Pressed

- Button Press

- Only work when both contacts are connected.



PROGRAMMING : LOOPS

1. While Loop

- While(condition is met){
  insertCodeHere;}

```
void loop() {
  int i = 0;
  while(i < 100){
    ..............
    ..............
    ..............
    i += 1 //Increment i by 1 after every loop
  }
}
```

2. For Loop

- for (initialization; test; increment/decrement){
  insertCodeHere;}

```
void loop() {
  for(int i = 0; i<100; i++){
    ..............
    ..............
    ..............
  }
}
```

PROGRAMMING ASSISTANCE

- Compiler

- Serial Communication

1. Serial Monitor
2. Serial Commands

```
void setup() {
  Serial.begin(BAUDRATE); //send and receive at this rate
  //CODE
}

void loop() {
  Serial.print("Here is what you want: ")
  Serial.println(//NAME OF VARIABLE YOU WANT);
}
```

Baud Rate: No of bits per second (Standard : 9600)

Difference is println automatically jumps to a new line.

Other Uses: Get Data/Understand Behaviour

SENSOR # 1 : LDR

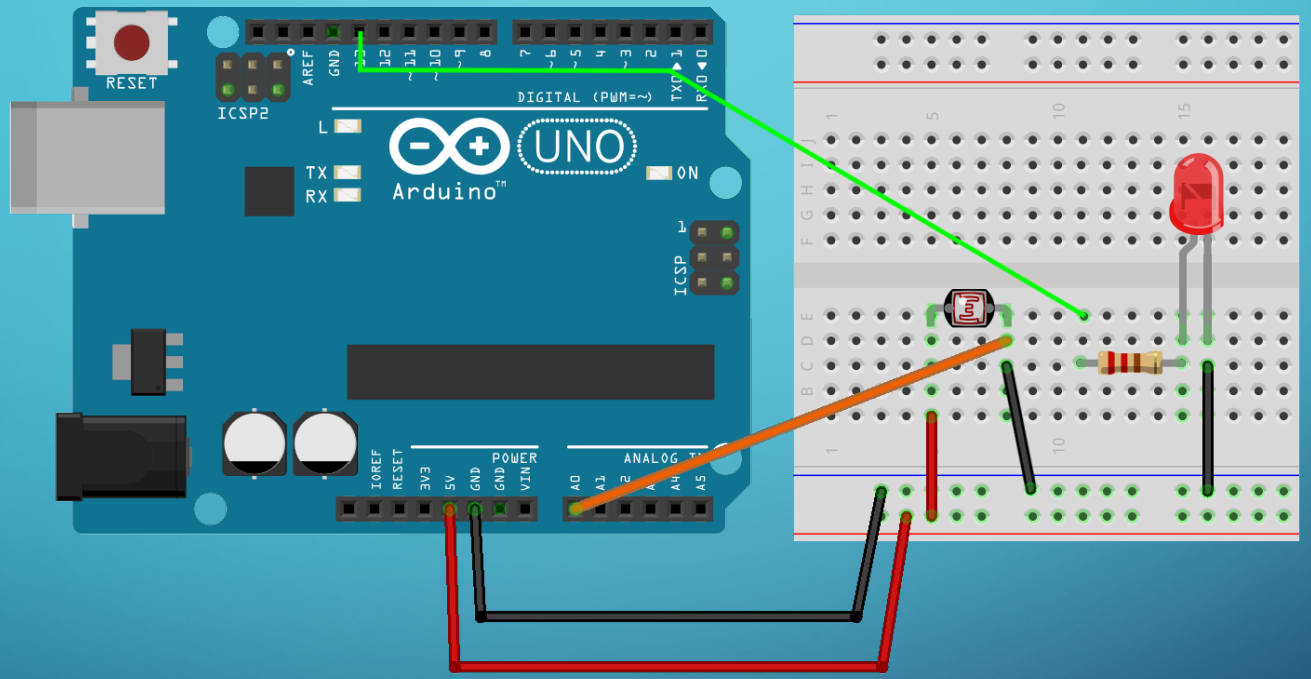
LDR EXERCISES

1. Is the LDR an analog or digital sensor?

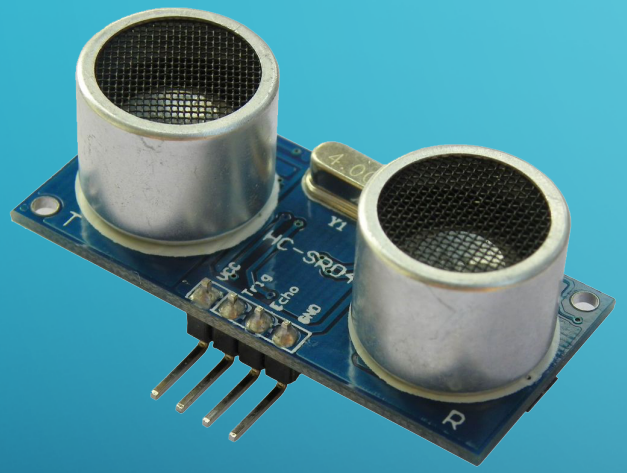
2. Substitute the Push Button with the LDR and use its reading to control on/off state of the LED.

   (Hint: Use a while loop, and then try the same with a for loop)

SENSOR # 1 : LDR


RESET
ICSP2
AREF
GND
DIGITAL (PWM=~)
TX
RX
L
ON
UNO
Arduino™
ICSP
POWER
IOREF
RESET
3V3
5V
GND
GND
VIN
ANALOG
A0
A1
A4
A5

SENSOR # 2 : ULTRASONIC SENSOR

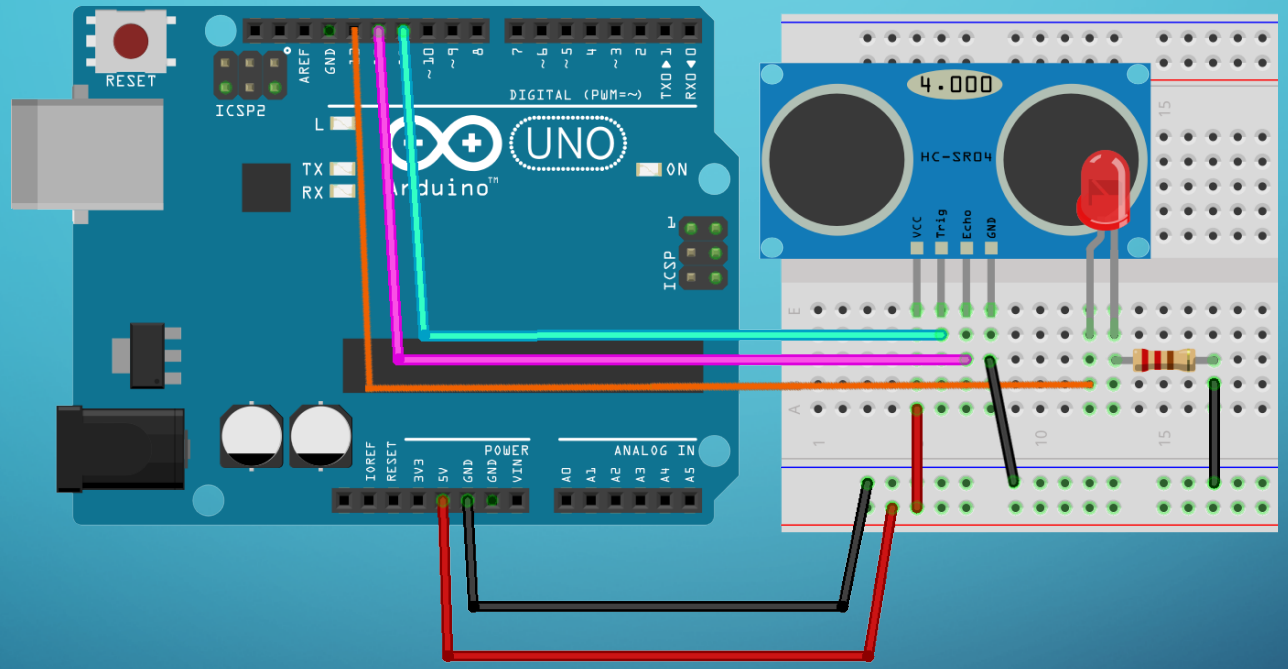


ULTRASONIC EXERCISES

1. Is the ultrasonic sensor an analog or digital sensor?

2. Substitute the LDR with the ultrasonic sensor and use its distance reading to control on/off state of the LED.

SENSOR # 2 : ULTRASONIC SENSOR



PROGRAMMING : FUNCTIONS

Functions : condense code into modular and repeatable form

(+) Modular Code → Easier Troubleshooting

(+) Less Wordy Code → Easier to Read

BUILT IN FUNCTIONS:

eg : digitalWrite() / digtalRead() /delay()

DIY FUNCTIONS:

- (data type to be returned) (function name) (arguments with data type){
    insertCodeHere;
    return whatYouWant;
}

```
int myMultiplyFunction(int x, int y){
  int result;
  result = x*y;
  return result;
}
```

All functions must be created outside of void setup () and void loop().

MOVING FORWARD

1. Do Some Projects
2. Explore Libraries (.h format)
   - #include <libName.h>
3. Explore Components
4. Practice

USEFUL LINKS

1. http://playground.arduino.cc/Projects/Ideas
2. http://www.instructables.com/id/Arduino-Projects/
3. And you can search online for many more:)

THANK YOU ☺

SUTD IEEE STUDENT BRANCH