MAKE THINGS INTERACTIVE :WEEK 6

GRAP2757 / Elective

Dr Karen ann Donnachie

Marina Fujiwara
MUDA-ZUKURI /Wasted Creation


Marina Fujiwara
MUDA-ZUKURI /Wasted Creation

Marina Fujiwara
MUDA-ZUKURI /Wasted Creation

Chindogu was created by Japanese artist Kenji Kawakami in the 1990s, who describes these inventions as "un-useless." He coined the term chindogu using a combination of the Japanese words chin, meaning "strange" or "odd," and dougu, which means "device" or "tool." But chindogu is more than a mashup of words (a portmanteau, if you will); it's a philosophy. There are 10 tenets of chindogu, according to the chindogu society:

1. A chindogu cannot be for real use. If you end up using your invention on the regular, you have failed.

2. A chindogu must exist. No thought experiments allowed.

3. There must be the spirit of anarchy. Build your invention free from the constraints of utility or cultural expectations.

4. Chindogu are tools for everyday life. Everyone everywhere must be able to understand how it works without any special technical or professional background info.

5. Chindogu are not tradeable commodities. Finally, something in your life that you just can't turn into a side hustle.

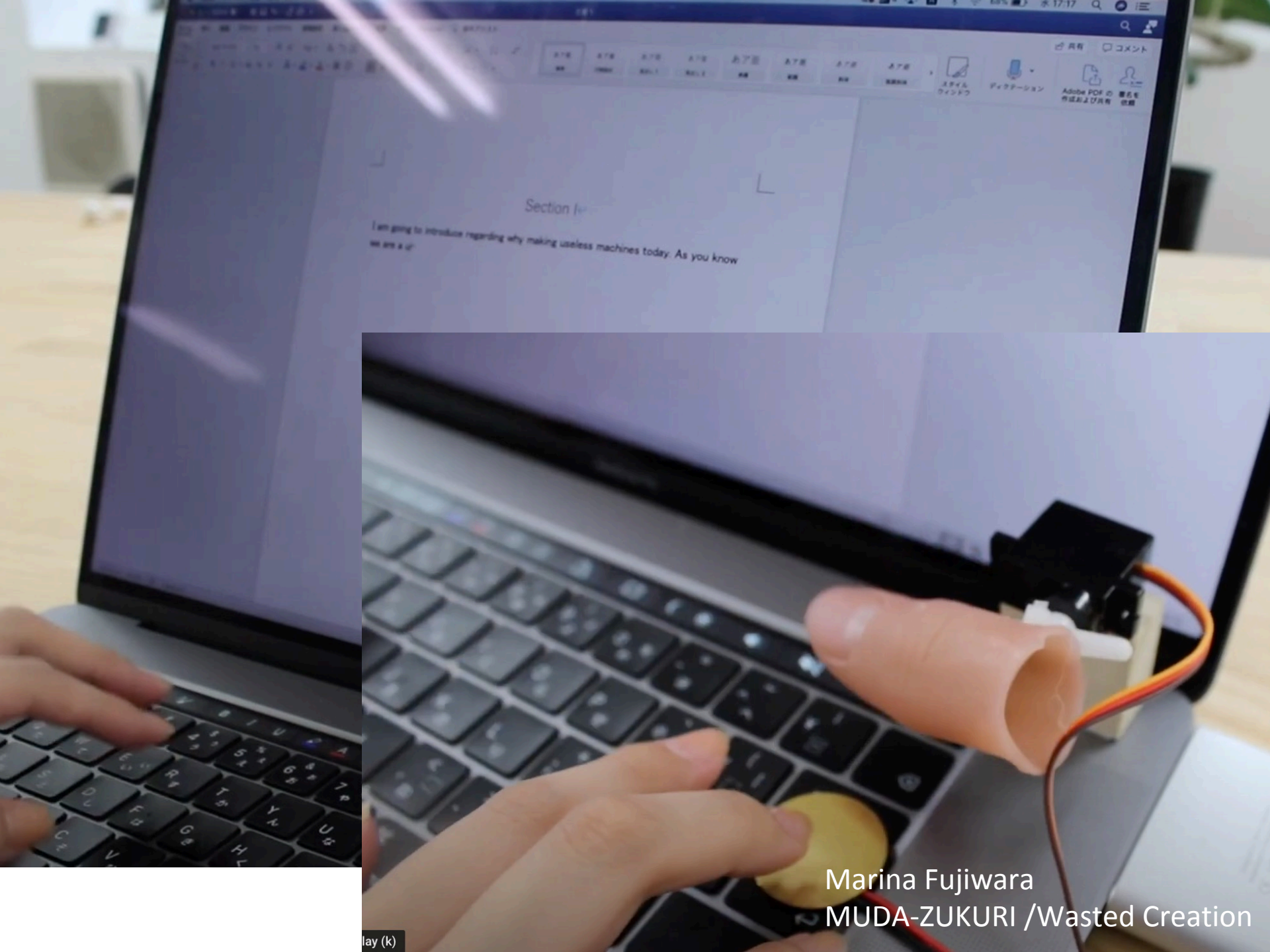
6. Humor must be the sole reason for creating chindogu. Creating an elaborate way to solve a tiny problem is just funny. Roll with it.

7. Chindogu is not propaganda. This is not the place for your clever commentary on the dumpster fire that is the current state of the world. As the tenet makes clear: "Make them instead with the best intentions."

8. Chindogu are never taboo. If you demand sexual innuendo, cruel jokes and sick humor, the International Chindogu Society would ask that you find it literally anywhere else on the internet. That's not chindogu's jam.

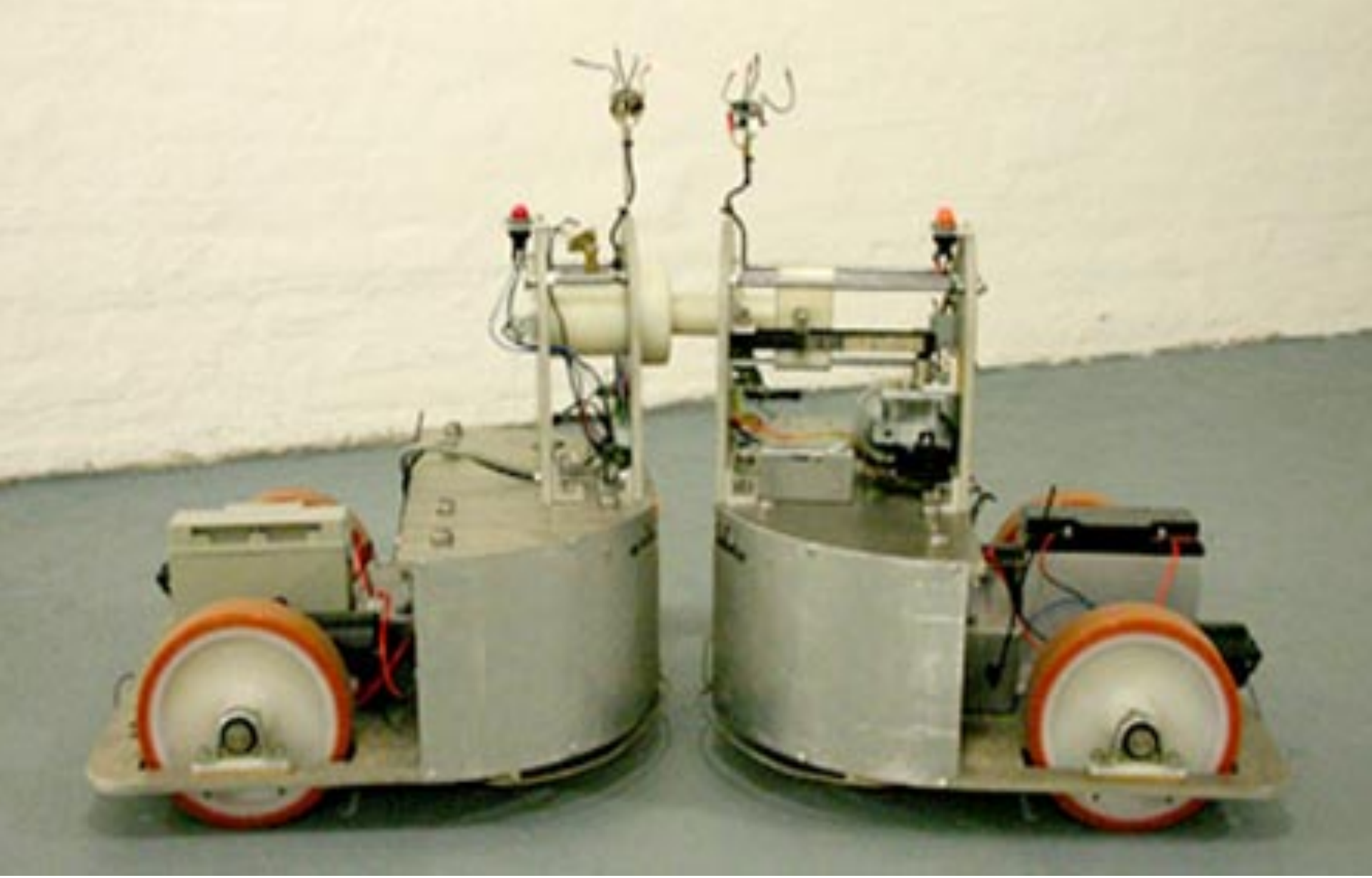
9. Chindogu cannot be patented. Consider chindogu the openest of open source. They're meant to be shared and delighted in, not owned and collected.

10. Chindogu are without prejudice. Race, religion, gender, age, ability — none of these matter to chindogu. These inventions should be equally (almost) useless to everyone who sees them.


Section I
I am going to introduce regarding why making useless machines today. As you know
Marina Fujiwara
MUDA-ZUKURI /Wasted Creation

Simone Giertz, *Shi**ty Robots*


Massage me, 2014



Sexed Robots, Paul Granjon 2005

SUN Yuan & PENG Yu
Can’t Help Myself (2016)

Mari Velonaki’s installation. Robot arms hammer their way out of a room

A tool to decive and slaughter,
Caleb Larsen, 2009

Ken Kawamoto, *Tempescope* (weather-forecast-box)

The Sigh Collector
http://www.instructables.com/id/Sigh-Collector/

Richard Clarkson, *Cloud*

MAKE THINGS INTERACTIVE :DEBUGGING ERRORS

ARDUINO BOARD / PORT ERRORS
SYNTAX MISTAKES & TYPOS
COMPONENT ISSUES

1 ARDUINO BOARD / PORT ERRORS

“BOARD NOT IN SYNC”

There are a couple of reasons this can happen (the worst reason is your Arduino board is fried, or…):

- You have something attached to pins 0/1 and trying to communicate via serial (! shields sometimes use these pins)
- You have not chosen the correct com port or board under the tools menu

HOW TO FIX:

- Check board/port/tx/rx pins
- Press the reset button on the Arduino couple of times and re-upload the code
- Disconnect and reconnect the Arduino to your computer
- Restart the Arduino IDE

1 ARDUINO BOARD / PORT ERRORS

“Serial Port Already in Use”

This is a relatively simple one. Surprisingly it means the serial port is in use, ie. perhaps you have the serial monitor open somewhere in your application.

The error happens because you are asking the Arduino to use the same serial port to do two different things at the same time.

! If you ever want to use Bluetooth, managing serial ports can get complicated

1 ARDUINO BOARD / PORT ERRORS

“Sketch too large” or Sketch uploads, but does nothing

Your sketch could be too big or the upload may have been interrupted.

Check the size of your sketch, it must be <30kb.

HOW TO REDUCE THE SIZE OF YOUR SKETCH

- Use integers rather than decimals/floats where possible
- Only include necessary libraries
- Use ‘const’ for variables where possible
- Identify code repetition and make a function to handle the calls

1 ARDUINO BOARD / PORT ERRORS

“java.lang.StackOverflowError”
Your sketch could be choking on some formatting of string variables.

HOW TO FIX:
Ensure your quotes are complete, backslashes are rightly used etc.

2 SYNTAX MISTAKES & TYPOS

Accidentally Using the Assignment Operator = instead of Comparison ==

Arduino programs may generate unexpected results if a programmer accidentally uses an assignment operator (=), instead of a comparison operator (==) in an if statement. To check if something is equal to a value use ==

CORRECT:
This if statement returns false (as expected) because x is not equal to 10:
int x = 0;
if (x == 10)

INCORRECT:
This if statement returns true (maybe not as expected), because 10 is true:
int x = 0;
if (x = 10)

& This if statement returns false (maybe not as expected), because 0 is false:
let x = 0;
if (x = 0)
An assignment always returns the value of the assignment.

2 SYNTAX MISTAKES & TYPOS

USING PROTECTED WORDS FOR VARIABLE OR FUNCTION NAMES

This can really cause unexpected results. Across Arduino, processing, javascript and Python it is best to avoid using these words as your variables (as tempting as it may be ☺

PLEASE ALSO SEE THIS LIST:
https://www.arduino.cc/reference/en/

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
In	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

2 SYNTAX MISTAKES & TYPOS

ACCIDENTALLY DUPLICATING VARIABLES OR LIBRARIES

Arduino code will attempt to do everything you ask, if you add the include code for the same library, you will bloat out your sketch.

Also if you have multiple copies or versions of the library in your folders, it can cause conflicts. You can temporarily move one to another location if you are getting errors around your libraries.

2 SYNTAX MISTAKES & TYPOS

MIS-MATCHED BRACKETS

There are some Arduino errors that all point back to the same issue:

- “does not name a type” or
- “expected declaration before” or
- “expected unqualified-id before” or
- “expected initializer before” or
- “a function-definition is not allowed here before ‘{’ token” or
- “‘functionName’ was not declared in this scope” or
- “expected ‘}’ at end of input” or

These errors usually mean you forgot a ‘{’ or put in an extra ‘}’ in the previous function.

Instead “expected primary-expression before ‘}’ token” usually means a missing semi-colon in the function.

HOW TO FIX:

- Check your brackets match.
- Indent your code to help understand at a glance
- Complete your statements (check semi-colons)

2 SYNTAX MISTAKES & TYPOS

MISPLACED SEMICOLON SURPRISE

Because of a misplaced semicolon,
this code block will execute *regardless of the value of x:*

```
if (x == 19);
{
  // code block
}
```

USING ‘long’ INSTEAD OF ‘unsigned long’ FOR VARIABLES THAT REPRESENT A TIME IN MILLISECOND

This will cause your code to reach the end of the storage limit for this variable and will break your timings and mess with your code.

2 SYNTAX MISTAKES & TYPOS

USING INTEGERS INSTEAD OF FLOATS (& VICE VERSA)

Floating-point numbers are those which contain a decimal, while integers, on the other hand, are whole numbers.

If your program is performing a mathematical calculation of dividing 20 by 6, the answer would be a decimal number (3.33). If you use an integer to store the answer to 20/6, then you'd get a '3' instead of 3.33. The 0.33 difference seems like nothing, but if you need precision in your project, then this difference is going to be significant.

The compiler won't count this mistake as an error, but it can cost you a lot by hindering the way your project works.

2 SYNTAX MISTAKES & TYPOS

MIS-NUMBERING ARRAYS

An array is a collection of values stored one after another.

The size of an array tells you how many elements are present in it.

For instance, if you have a variety of size 6, it means there are six elements in it. They can be numbers, strings, or alphabets. What's important to note is the element numbering inside the array.

The counting always starts from 0 for arrays, which means if you have six elements, they go like 0, 1, 2, 3, 4, 5. If you want to access the first element of an array, it would be element 0, not 1. The last element of this array would be 5 and not 6, as many people mistakenly believe.

Many people make the mistake of not numbering the array elements correctly, which renders incorrect values when accessing the array.

3 COMPONENT ISSUES

SERVO NOT WORKING

Servomotors require a high starting current, and this is where the problem begins. The motor needs more current, which lowers the voltage on other pins of the Arduino board, and it resets itself. This cycle continues to go on, and it appears that the servomotor is not starting.

HOW TO FIX:

To fix the Arduino servo issue, you need to provide a proper source of current to the servomotor. A couple of ways to do that:

- One way is to add a capacitor between the 5V and ground connections on the breadboard. The capacitor acts as a charge reservoir, supplying extra current to the servo to allow it to start without any hassle. Keep in mind that the capacitor must be of high value. Anything greater than 470 uF can solve the problem.

- Another solution is to add an external power source for the servomotor. A common practice is to use batteries to power servos. Doing so will fulfill their high starting current requirements and save you from dealing with servo issues.

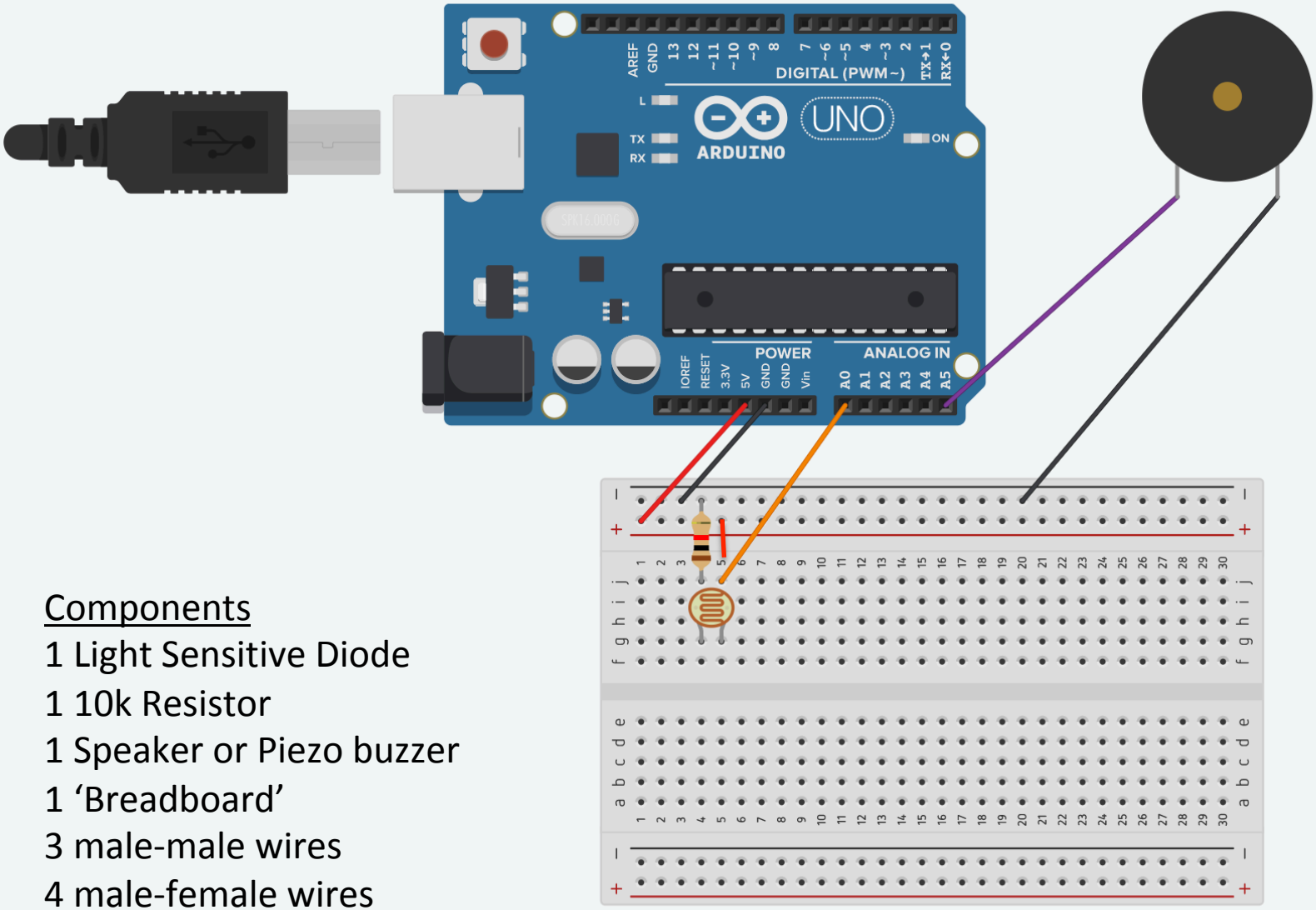
3 COMPONENT ISSUES

GROUND NOT CONSISTENT/COMPLETE

Grounding circuits is essential because it saves your project from unexpected power surges and damage. Grounding in Arduino can sometimes be an issue because the ground pin might refuse to work, or you're not connecting it correctly.

Whatever the case is, you need to fix any Arduino grounding issue as soon as possible to save your project circuitry from damage, like your board getting fried.

HOW TO FIX:
You can fix Arduino grounding issues by first checking the GND pin voltage using a voltmeter.
Check the GND pin and see if there's a loose connection, so you can solder it and fix it back into its place. You won't experience grounding issues after you do this..



Components

1 Light Sensitive Diode
1 10k Resistor
1 Speaker or Piezo buzzer
1 ‘Breadboard’
3 male-male wires
4 male-female wires

```
 /* Quasi-theremin instrument

Connect the photoresistor one leg to pin 0, and pin to +5V
Connect a resistor (around 10k is a good value, higher
values gives higher readings) from pin 0 to GND.

-------------------------------------------------

           PhotoR     10K
 +5    o---/\/\/--.--/\/\/---o GND
                  |
 Pin 0 o-----------

-------------------------------------------------
*/

int lightPin = A0;   //define a pin for Photo resistor
int ledPin = 16;     //define a pin for LED

int speakerPin = 5;
int numTones = 10;
int tones[] = {261, 277, 294, 311, 330, 349, 370, 392, 415, 440, 457};
String notes[]={"mid C","C#","D","D#","E","E#","F","F#","G","G#","A"};

void setup()
{
    Serial.begin(9600);  //Begin serial communcation
    pinMode(speakerPin, OUTPUT);
}
```

```
void loop()
{
   long brightness;

   Serial.println(analogRead(lightPin));
// Write the value of the photoresistor to the serial monitor.

    delay(100); //short delay for faster response to light.

   brightness = (analogRead(lightPin));
   if (brightness > 50) {
       tone(speakerPin, tones[(int(brightness/100))]);
        Serial.println(notes[(int(brightness/100))]);
       delay(10);
    }
    else noTone(speakerPin);
    Serial.println(brightness);            // Print to serial port
    delay(200);                           //pause for 200 millis
}
```

HOW CAN WE CUSTOMISE THESE SOUNDS?

```
int numTones = 10;
int tones[] = {261, 277, 294, 311, 330, 349, 370, 392, 415, 440, 457};
//               mid C  C#   D     D#    E     E#     F   F#    G   G#    A
```

You can change these tones (add, take-away).
You can even put them in the order of a tune…

Just update this line to reflect if you have changed the overall number of tones.

```
if (brightness > 200) {
  tone(speakerPin, tones[(int(brightness/100)-1)]);
```

HOW COULD WE CALIBRATE FOR DIFFERENT LIGHTING SITUATIONS?

BEFORE WE START... GITHUB, ANYONE?