(U4) MICROCONTROLLERS (CLASS ACTIVITIES)

1 NAME AND SURNAMES:					Group:
2 NAME AND SURNAMES:					
Day/Date	Signatures		Day/Date	Signatures	
1/	Name1:	Name2:	2/	Name1:	Name2:

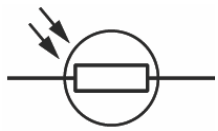
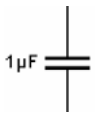
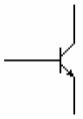
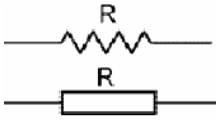
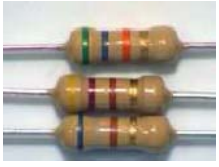
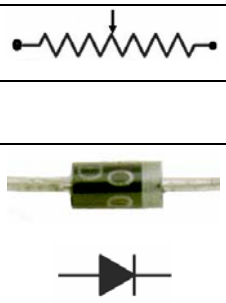
Introduction

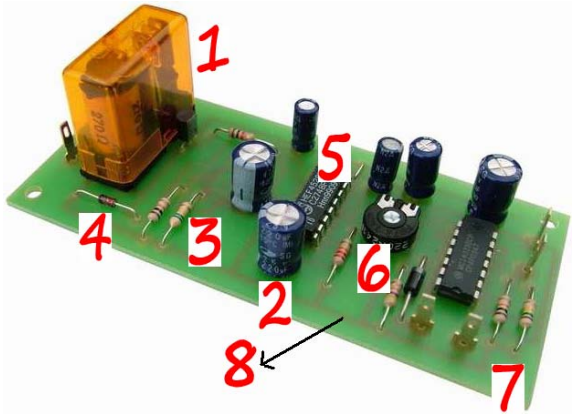
⊙ **Electricity versus Electronics**. Decide if the circuits below are electric or electronic circuits and label the properties with the words **ecity** or **enics**.

- They work with high voltages and currents:
- Electrical components include light bulbs, motors, relays, etc.:
- They use components capable of operating and controlling small currents:

- They use bigger components that need much more power to work:
- Uses: lighting houses, powering motors… (high output):
- Uses: they are used in telecommunications, sound, photography, video, etc.:

⊙ **Electronics**. Identify the components below (potentiometer, rectifier diode, capacitor, LED, resistor, transistor and LDR) :



1 Relay

2

3

4

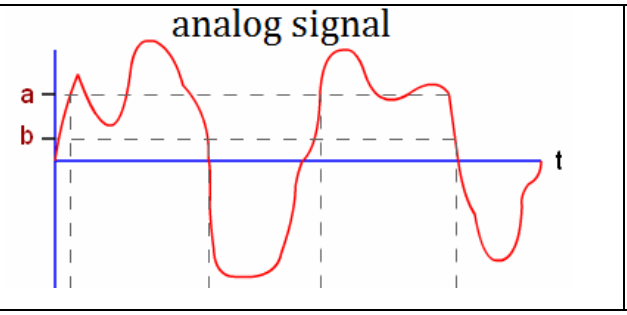
5

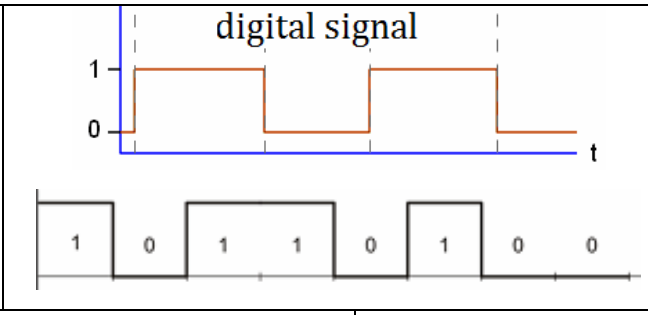
6

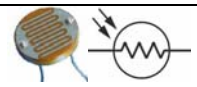
7 Printed Circuit borrad (PCB)

8

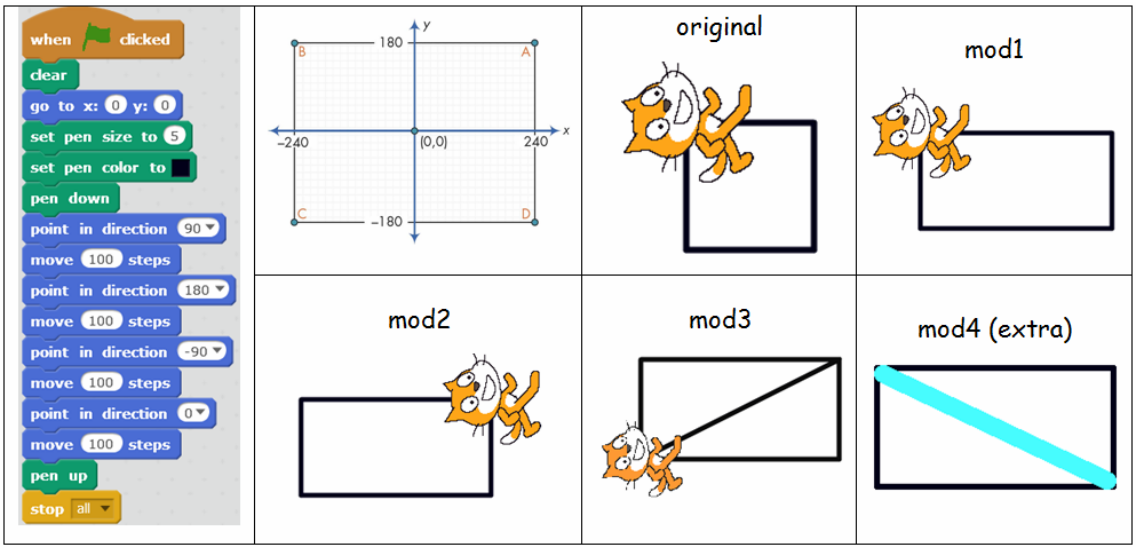
⊙ Analogue signals versus Digital Signals

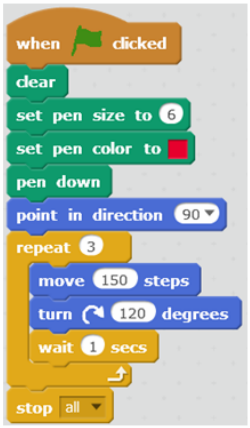




Signals and components	Examples
✓  (open or closed): **digital**	Open 0 Closed 1
✓ Marks of an exam:	
✓ Promotion of a student (to the next year)	
✓  Speed of a motor:	
✓  Resistance of a LDR dependent of the amount of light	
✓  Push Switch (press or release):	
✓  Resistance of a potentiometer:	
✓  Turn on and turn of a LED:	
✓ Infrared Sensor (discerning between Light and dark colours):	
✓ Infrared Sensor (black and white):	
✓  Buzzer (it produces an audible sound:	

⊙ **Programming_1** (with Scratch)





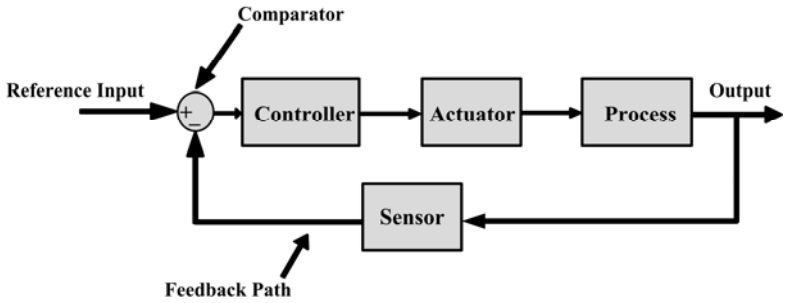
01 (polygon1)	02 (polygon2)	03 (polygon3)
	Change the size (10) and the colour (blue) of the pen and the side (300 steps	Draw a square with 200 step sides
04 (polygon2)	**05 (polygon2)**	**06 circle**
Draw a pentagon/hexagon with 110 step sides	Draw a pentagon/hexagon with 110 step sides	Draw a circle of 1 step radius

⊙ **Programming_2** (with Scratch)

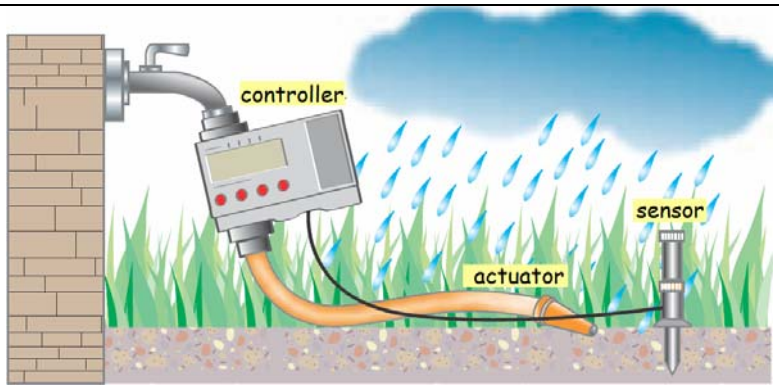
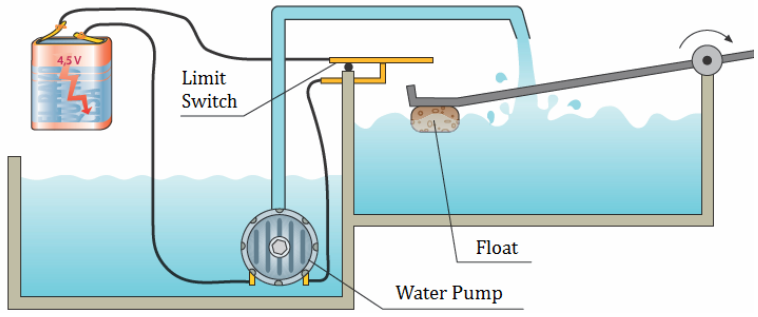
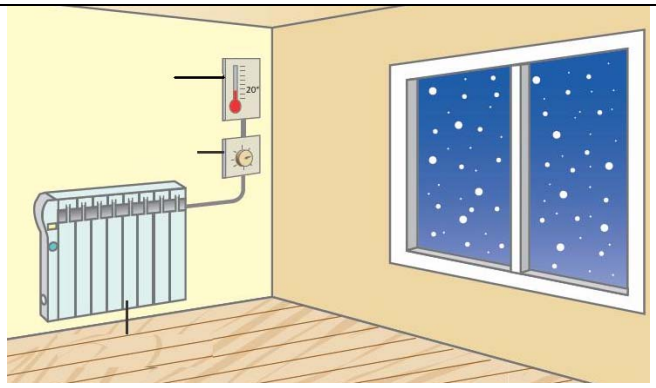
Day/Date	Signatures		Day/Date	Signatures	
3/	Name1:	Name2:	4/	Name1:	Name2:

Automation Control Systems

1. **Automatism**. Automation can be defined as the application of machines to tasks once performed by human beings or, increasingly, to tasks that would otherwise be impossible. Automation generally implies the integration of machines into a self-governing system.

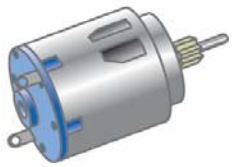


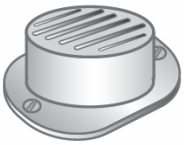
⊙ Complete the table below (label the first picture with the

	Input signal: humidity (in the earth) Sensor: humidity sensor (the "nail") Controller: electronic controller Actuator: the tap (with a hose) Process: irrigation system control Output signal: water
	Input signal: Sensor: limit switch Controller: Actuator: Process: water level control Output signal:
	Input signal: Sensor: Controller: Actuator: Process: Output signal:

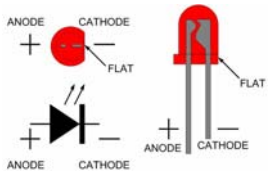
⊙ Identify the electronic components below and classify the into the correct group:

(light dependant resistor)

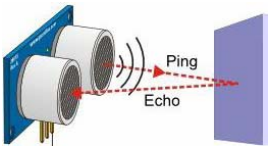


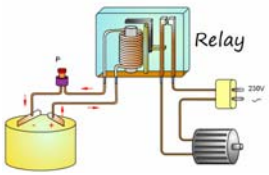


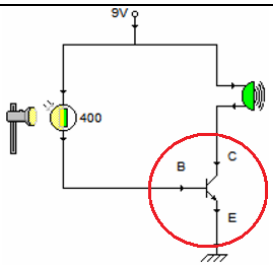
(limit switch)

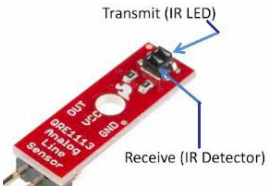


(Light Emitting Diode)









(Infrared sensor)

				Infrared sensor

✓ Sensors:

✓ Actuators:

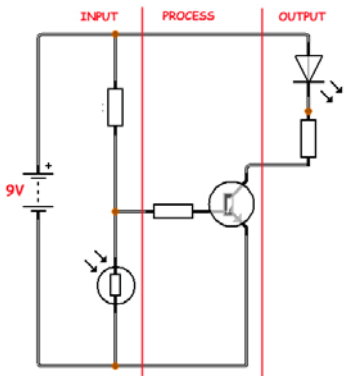
✓ Controllers: Relay

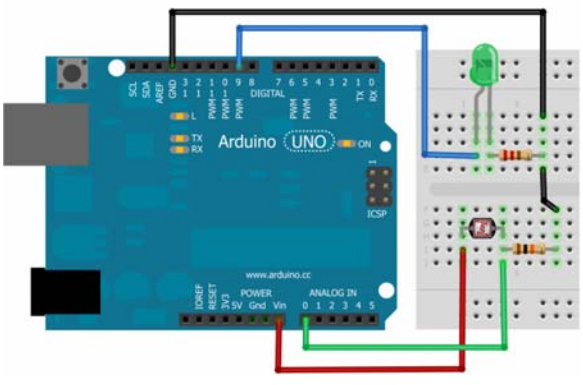
2. **Electronic control** versus **programmed control**. In the circuits below:

⊙ Label the pictures below with electronic or programmed control.

⊙ Identify the following components: LED, LDR, transistor, resistor, wires, power supply, controller and controller board.

⊙ Complete the table.





Input signal:

Sensor:

Controller:

Actuator:

Process:

Output signal:

Day/Date	Signatures		Day/Date	Signatures	
5/	Name1:	Name2:	6/	Name1:	Name2:

ARDUINO

CTC UNO

CTC101

CREATIVE TECHNOLOGIES IN THE CLASSROOM

BLOCK 1 PROGRAMMING

Programming

- Processing
- Line
- Screen and Pixels
- Variables
- Setup and draw

Projects

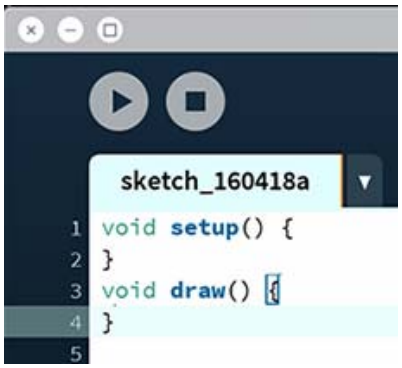
- Red snake (shortest project)
- Post it clock
- Catch the apple (longest project)

Processing

Processing is an open source **programming language** and environment for people who want to create images, animations, and interactions.

- ✓ Program➔Instructions➔Code
- ✓ IDE (Integrated Development Environment)➔Text Editor
- ✓ **Compiling**: IDE translate **programming code** (instructions) into **machine language**
- ✓ **draw()**: programs that continually run (loop)
- ✓ **setup ()**: code that runs just once

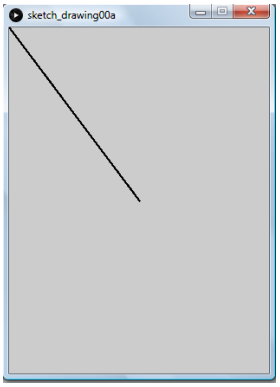
Processing IDE interface

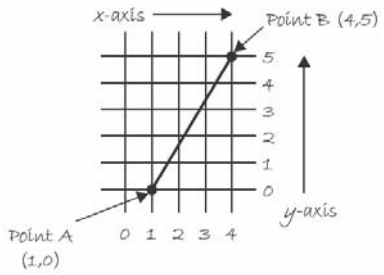


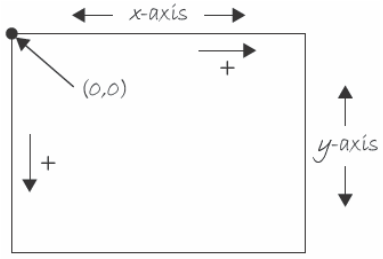
Program

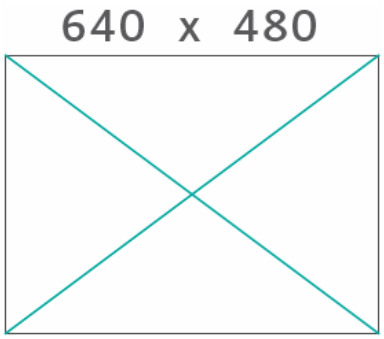
```
sketch_01_line
1 void setup(){
2   size(300,400);
3
4 }
5
6 void draw(){
7   line (0,0,150,200);
8
9 }
```

Result



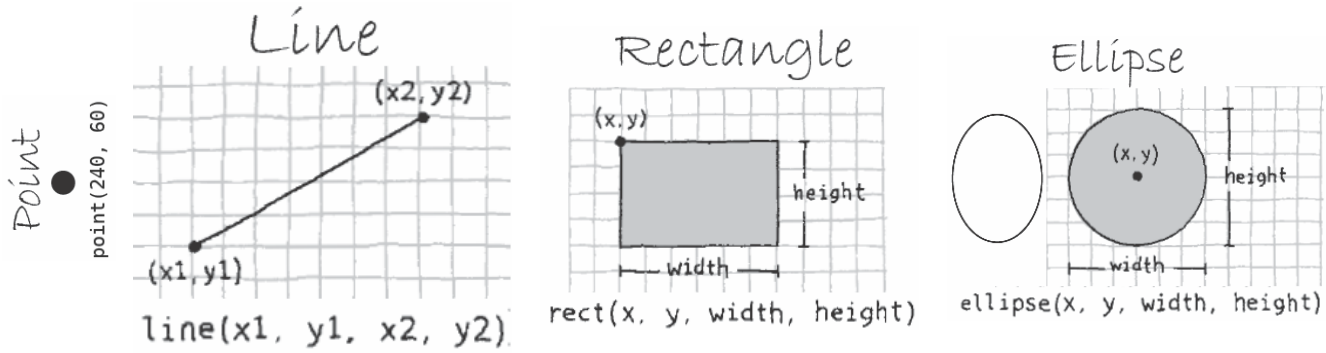






lines

1) Create the **following free shapes** (call the teacher for the ellipse):



2) Create the programs below (copy the firs two and create the other one)

sketch_05_mix1

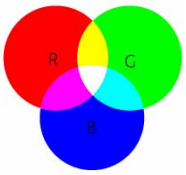
```
size(480, 120);
ellipse(140, 0, 190, 190);
// The rectangle draws on top of the ellipse
// because it comes after in the code
rect(160, 30, 260, 20);
```

sketch_05_mix2

```
size(480, 120);
smooth();
ellipse(75, 60, 90, 90);
strokeWeight(8); // Stroke weight to 8 pixels
ellipse(175, 60, 90, 90);
ellipse(279, 60, 90, 90);
strokeWeight(20); // Stroke weight to 20 pixels
ellipse(389, 60, 90, 90);
```

The single parameter to **strokeWeight()** sets the **width** of drawn lines:

Screens and Pixels



3) Functions background(), fill(), and stroke() allow us to use colours for the display window (**background**), to **fill** shapes and set the width of outlines (**stroke**). The values of the parameters

are in the range of 0 (00000000) to 255 (11111111), where **255 is white**, 128 is medium gray, and **0 is black**.

✓ Create the program below to see how to use colours

sketch_11_colours1

```
size(480, 120);
noStroke();
smooth();
background(0, 26, 51); // Dark blue color
fill(255, 0, 0); // Red color
ellipse(132, 82, 200, 200); // Red circle
fill(0, 255, 0); // Green color
ellipse(228, -16, 200, 200); // Green circle
fill(0, 0, 255); // Blue color
ellipse(268, 118, 200, 200); // Blue circle
```

sketch_11_colours2

```
size(480, 120);
noStroke();
smooth();
background(204, 226, 225); // Light blue color
fill(255, 0, 0, 160); // Red color
ellipse(132, 82, 200, 200); // Red circle
fill(0, 255, 0, 160); // Green color
ellipse(228, -16, 200, 200); // Green circle
fill(0, 0, 255, 160); // Blue color
ellipse(268, 118, 200, 200); // Blue circle
```

Variables

A variable stores a value in memory so that it can be used later in a program. The variable can be used many times within a single program, and the value is easily changed while the program is running. When you make your own variables, you determine the **name**, the **data type**, and the **value**. Variables must first **be declared**, which sets aside space in the computer's memory to store the information.

4) Create the following programs:

sketch_21_variables1

```
size(480, 120);
smooth();
int y = 100;
int d = 130;
ellipse(75, y, d, d); // Left
ellipse(175, y, d, d); // Middle
ellipse(275, y, d, d); // Right
```

sketch_21_variables2

```
size(480, 120);
smooth();
line(0, 0, width, height); // Line from (0,0) to (480, 120)
line(width, 0, 0, height); // Line from (480, 0) to (0, 120)
ellipse(width/2, height/2, 60, 60);
```

5) Do the same thing over and over: use a **for Loop** (Repetition)

sketch_22_repetition

```
size(480, 120);
smooth();
strokeWeight(8);
line(20, 40, 80, 80);
line(80, 40, 140, 80);
line(140, 40, 200, 80);
line(200, 40, 260, 80);
line(260, 40, 320, 80);
line(320, 40, 380, 80);
line(380, 40, 440, 80);
```

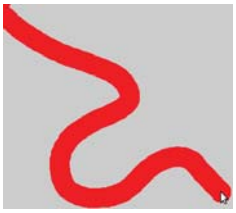
sketch_23_repetition

```
size(480, 120);
smooth();
strokeWeight(8);
for (int i = 20; i < 400; i += 60) {
line(i, 40, i + 60, 80);
}
```

Day/Date	Signatures		Day/Date	Signatures	
7/	Name1:	Name2:	8/	Name1:	Name2:

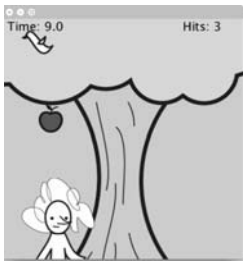
Project: Red Snake

6) Create the following project (web): Red Snake

Draw a circle	Make the circle move	Gradually shift colour (1&2)	Use sin() function	Changing the shape
				

Project: Post-it Clock

Project: Catch the apple



Day/Date	Signatures		Day/Date	Signatures	
9/	Name1:	Name2:	10/	Name1:	Name2:

BLOCK 2 Sports

Control boards (Arduino)

- What is the CTC Board
- Digital signals
- Count in Binary
- Blink and Beep
- Digital inputs

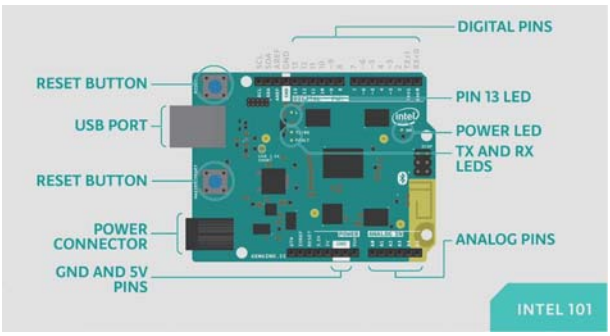
Projects

- Basketball
- Digital Die (longest project)
- Fencing (shortest project)
- Pong®
- Racing
- React
- Simon Says

What is the CTC Board

Arduino (Arduino 101) is an **open-source prototyping platform** based on easy-to-use hardware and software. Arduino boards are able to read **inputs** (light on a sensor, a finger on a button….)

and turn it into an **output** (activating a motor, turning on an LED…). You can **program** the board by sending a **set of instructions** (to the microcontroller on the board) using the **IDE** (Integrated Development Environment) for Arduino.



☉ Arduino boards connect to your computer using a **USB cable**. It must be connected in order to **upload a program** as well as supplying power to the board. We could also power the board by using an external power source like a battery or a power supply.

Power
The Uno board can be powered via the **USB connection** or with an external power supply.
5V. This pin outputs a regulated 5V from the regulator on the board.
GND. Ground pins.

Input and Output
The Uno has **6 analog inputs**, labeled A0 through A5.
Each of the **14 digital pins,** labeled 0 through 13 on the Uno can be used as an input or output, using ***pinMode(), digitalWrite(), and digialRead()*** functions.

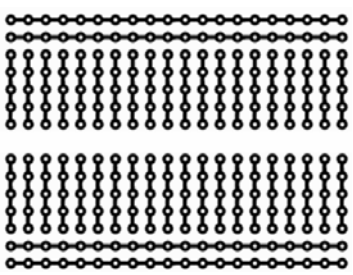
In addition, some pins have specialized functions:
PWM: **3, 5, 6, 9, 10,** and **11**. Provide 8-bit PWM output with the ***analogWrite()*** function.

LED: 13. There is a **built-in LED** driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

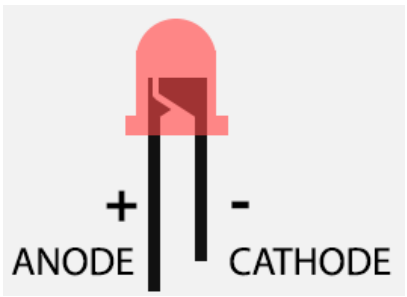
Reset. Bring this line LOW to reset the microcontroller.

☉ Make sure the IDE is configured correctly: the board (Tools➔Board➔Arduino Genuino 101) and the port (Tools➔Serial Port)

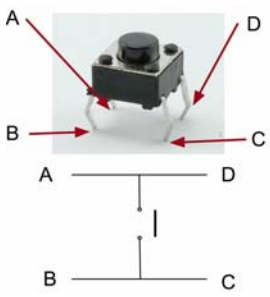
Breadbord



Diode

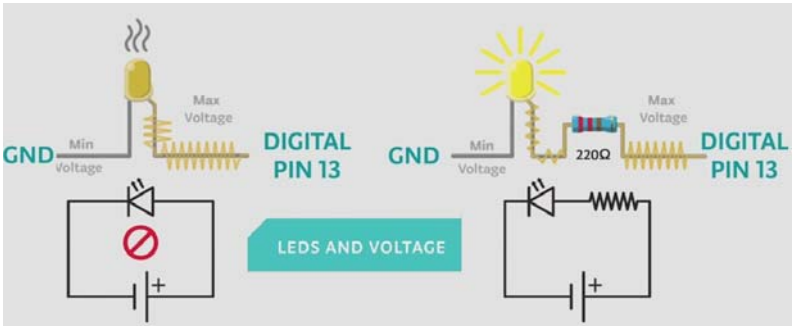


Push button

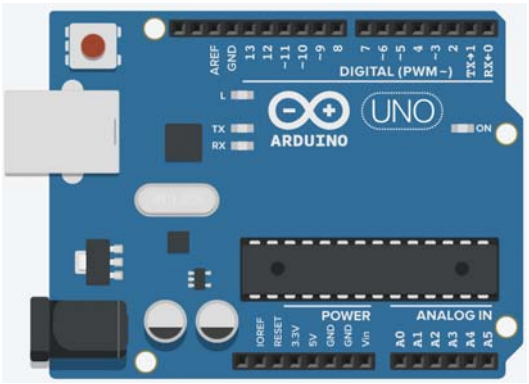


☉ Activity in : create the first 2 programs and call the teacher.

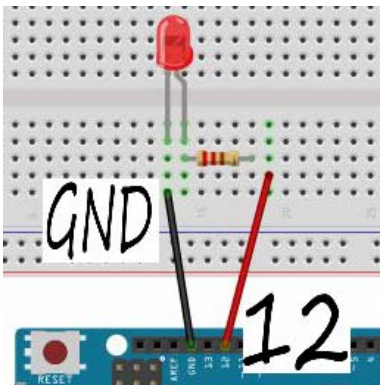
- Built-in LED blink: only the Arduino board + program
- LED Blink: circuit + the same program (notice the picture below)



Built-in LED blink **LED Blink** **LED Control**



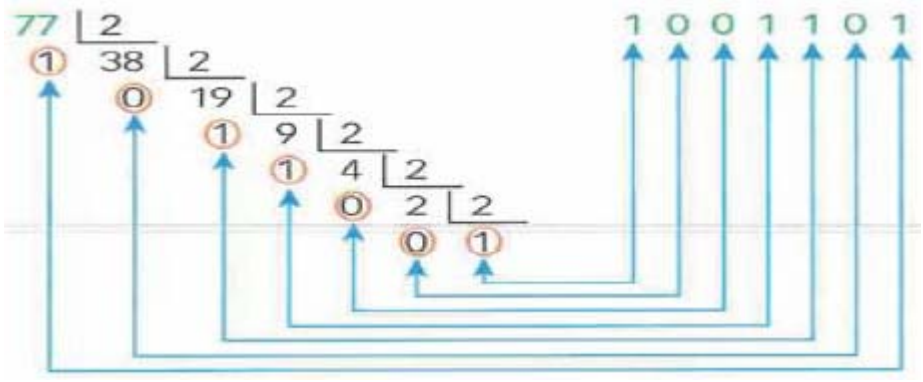

```
void setup() {

  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```



Digital Signals (outputs)

- Decimal System: $2165 = 2 \cdot 10^3 + 1 \cdot 10^2 + 6 \cdot 10^1 + 5 \cdot 10^0 = 2000 + 100 + 60 + 5$
- Binary System: $11001 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 24 + 23 + 1 = 25_{10}$
- Convert the decimal numbers below into binary numbers:

Decimal Numbers	Binary Numbers	Hexa-decimal Numbers
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

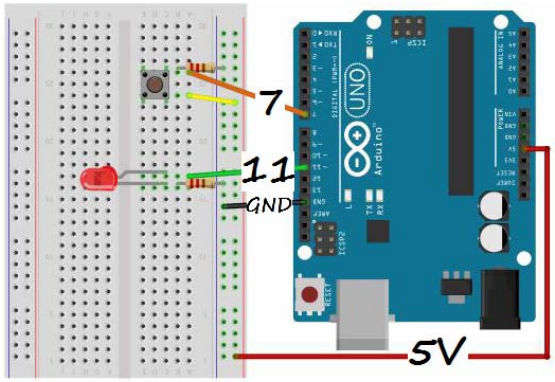


18: 37: 64: 112: 128:

⊙ 2 Activities in our General Website: the first 16 binary numbers and check your binary numbers

⊙ Activity in TINKERCAD: LED Control (circuit + program) and Piezo_S peaker1 (**extra**)

LED Control



Piezo_Speaker1 (beep)

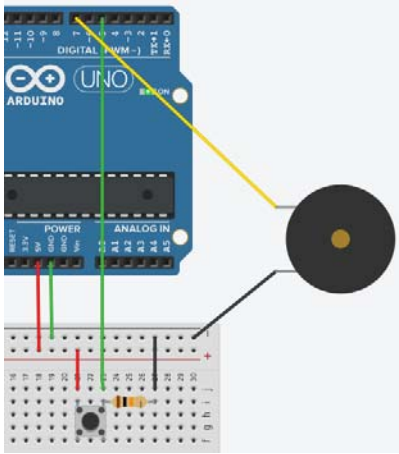
```
1  int speakerPin = 7;
2
3  void setup() {
4  pinMode(speakerPin, OUTPUT);
5  }
6
7  void loop() {
8  digitalWrite(speakerPin, HIGH);
9  delay(1000);
10 digitalWrite(speakerPin, LOW);
11 delay(1000);
12 }
```

Digital Inputs

on	off
1	0
5V	0V
HIGH	LOW

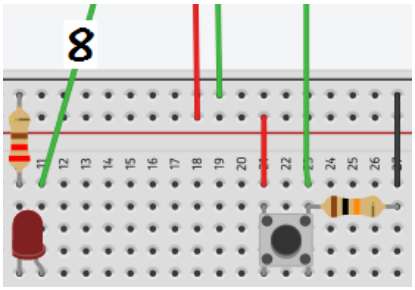
⊙ Activity in TINKERCAD: Piezo_S peaker2 and Piezo_S peaker_mod (**extra**)

Piezo_Speaker2 (beep)



```
int inputPin = 5;
int piezoPin = 7;
int buttonState = 0;
void setup() {
pinMode(inputPin, INPUT);
pinMode(piezoPin, OUTPUT);
  }
void loop()
{
buttonState = digitalRead(inputPin);
  if (buttonState == HIGH) {

    digitalWrite(piezoPin, HIGH);
  }
  else {
    digitalWrite(piezoPin, LOW);
   }
}
```

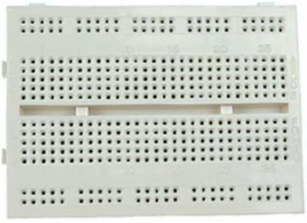
Piezo_Speaker2_mod

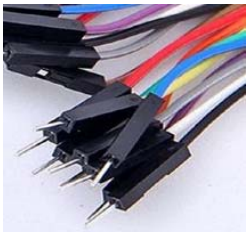


Day/Date	Signatures		Day/Date	Signatures	
11/	Name1:	Name2:	12/	Name1:	Name2:

Project: Fencing

- Identify the components you need for the project:




red-green-blue


- Follow the steps in the web.

Day/Date	Signatures		Day/Date	Signatures	
13/	Name1:	Name2:	14/	Name1:	Name2:

BLOCK 3 Magic

Control boards (Arduino)

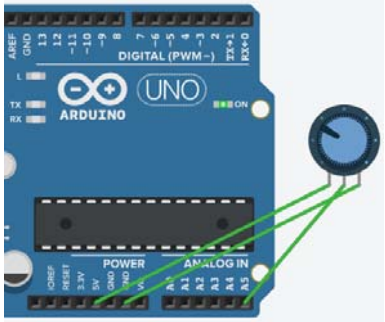
- Reading analogue signals
- Writing analogue signals
- Light sensor
- Serial port
- Sending to the computer
- Receiving from the computer

Projects

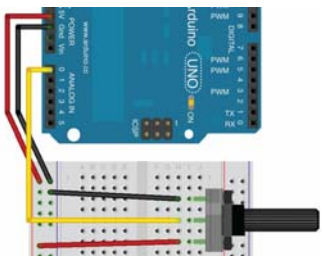
- Binary LP
- Cookie Monster (longest project)
- P.O.V (shortest project)
- Knock Knock box
- Drawdio
- Boombox
- Sequencer

Analogue Signals

- Analogue **inputs**: A0, A1, A2, A3, A4 and A5. `analogRead()`
- Analogue values: **0 V**: 0 (0000000000); **2,5 V**: 512 (1000000000); **5 V**: 1023 (1111111111)
- Analogue components: potentiometer. Practice with the following example(**analog_01**):



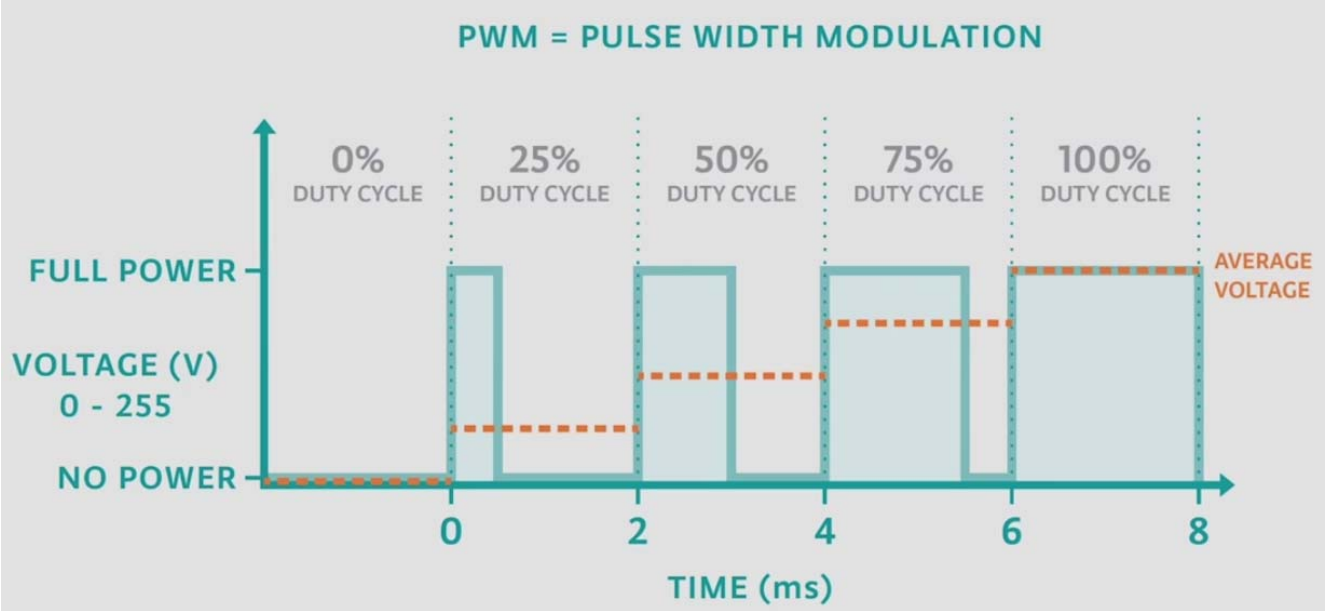
Potentiometer: 0-5 volts



Indicate the following values:

- 2^0: 1 (wiper to the left)
- 2^5:
- 2^8:
- 2^{10}:

- How many different values can `analogRead` read? (and for that we need to use ___ bits)
- PWM (Pulse Width Modulation) `analogWrite()`

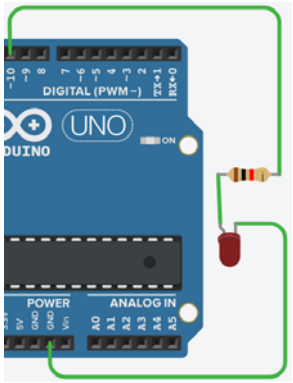


- Digital pins with ~ (tilde): analogue outputs
- `analogWrite` uses 8 bits to write the values: from 00000000 ( ) to 11111111 ( ).

```
analogRead(analogPin)
analogWrite(pwmPin, pwmValue)
map(value, fromMin, fromMax, toMin, toMax)
```

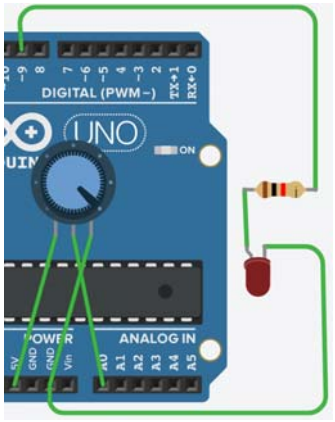
- Map(sensorValue, 0, 1023, 0, 255): the data from the potentiometer (0-1023) needs to be converted to fit into the smaller range (0-255) using it to dim the LED.
- Practice with the following examples

analog_02




```
1  int ledPin = 10;
2  int fade = 0;
3
4  void setup() {
5  // nothing here
6  }
7
8  void loop() {
9  analogWrite(ledPin, fade);
10 delay(10);
11 fade = fade + 10;
12 if (fade > 255) fade = 0;
13 }
```

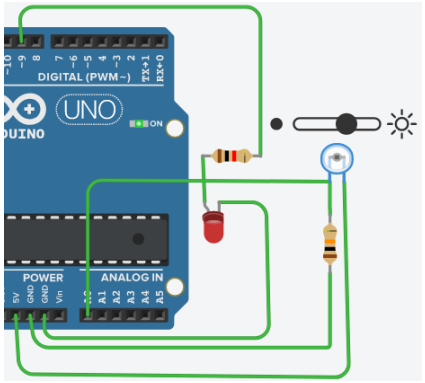
analog_03

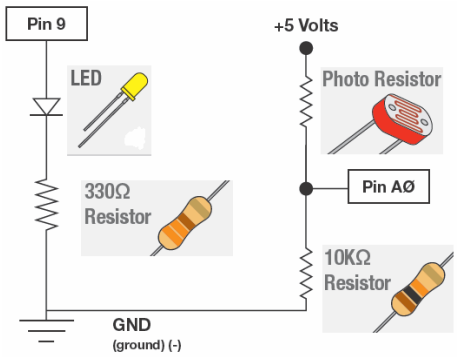


Light Sensor

- LDR (Light Dependent Resistor) is an analogue light sensor that detects the amount of light present, and depending on this amount it will return a specific analogue value.

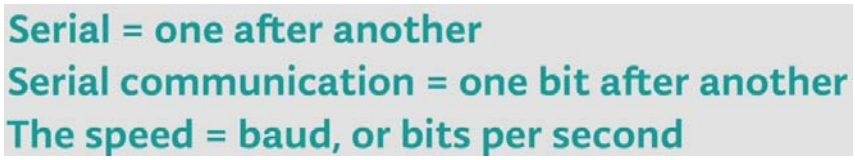
analog_04: LDR analog_04 (serial port)





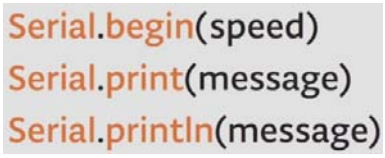

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  int sensorValue = analogRead(A1);
  Serial.println(sensorValue);
  delay(1000);
}
```

Serial Port



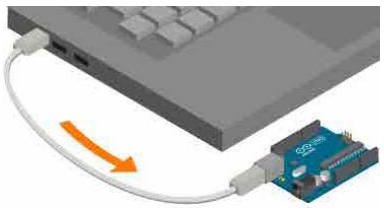
☉ The way the Arduino boards "talk" to the computer is through something called a serial port (USB port). Through the serial port you can send or receive text (string) and communicate with other software as well. For example you can read from a button and send the button state data to a processing sketch (a program) that changes the colour of the screen when the button is pushed.

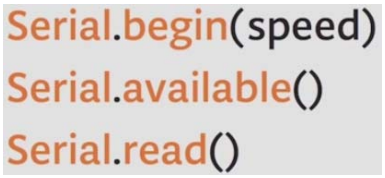
☉ Pin 1, **TX or transmission**, sends the data from the Arduino



☉ Printing light sensor values (analog_04: LDR)

☉ The Pin 0, **RX or reception**, receives the data to the Arduino






```
1   int ledPin=13;
2   int incomingByte;
3
4   void setup() {
5   Serial.begin(9600);
6   pinMode(ledPin,OUTPUT);
7   }
```

```
8   void loop() {
9   if(Serial.available()>0){
10  incomingByte=Serial.read();
11  if(incomingByte=='H'){
12  digitalWrite(ledPin, HIGH);
13  }
14  if(incomingByte=='L'){
15  digitalWrite(ledPin,LOW);
16  }
17  }
18  }
```

⊙ Serial.available(): checks if there are any signals coming from the serial port. Returns either true or false.

⊙ Serial.read(): Reads a byte from the serial port.

Day/Date	Signatures		Day/Date	Signatures	
15/	Name1:	Name2:	16/	Name1:	Name2:

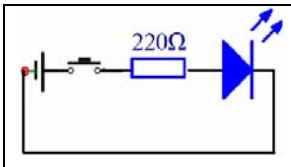
Review: blocks 1, 2 and 3

⊙ Go to our general website➔Unit3➔ICT Activities and complete the following activities:

Introduction to Microcontrollers							
Programming	Scratch	Processing	Processing	Processing			

BLOCK 2 DIGITAL SIGNALS

⊙ **(web) Electronic control** versus **programmed control.** In the circuits below:



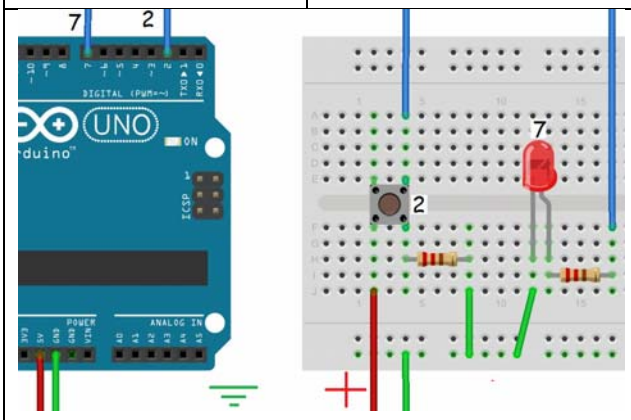
Type of control:

Input component:

Output component:

Function of the resistor:

Possible real application for this circuit:



Type of control:

Output component:

Input component:

Digital components:

Complete the figure below with 7 words:

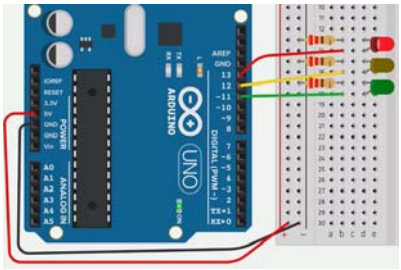
<table>
<tr>
<td>

```
const int buttonPin = ■;
const int ledPin =  ■;
int buttonState = 0;

void setup() {
    pinMode(ledPin, ■■■■); // I
    pinMode(buttonPin, ■■■■); //
}

void loop(){
  // read the state of the pushbu
  buttonState = digitalRead(butto
  // check if the pushbutton is p
  // if it is, the buttonState is
  if (buttonState == ■■■■) {
    // turn LED on:
    digitalWrite(ledPin, ■■■■);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, ■■■■);
  }
}
```

</td>
<td>Complete the text ■■
buttonPin= ;
ledPin= ;
pinMode (ledPin,)
pinMode (buttonPin,)
if (buttonState==)
digitalWrite (ledPin,)
digitalWrite (ledPin,)</td>
<td>Convert the numbers below into binary numbers (6 bits):
0: 12:
2: 15:
6: 18:
9: 23
1 Operation:</td>
</tr>
</table>

☉ Programming (**web**). Create a **basic program** to make 3 LEDs blink (alternatively with a frequency of 0,333 Hz) one after another:

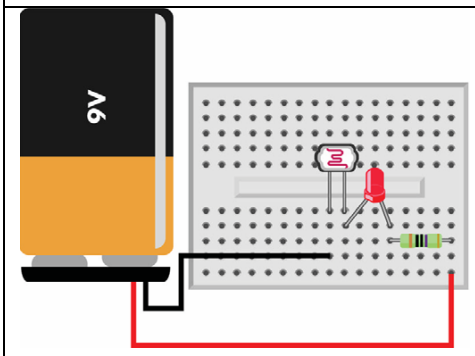
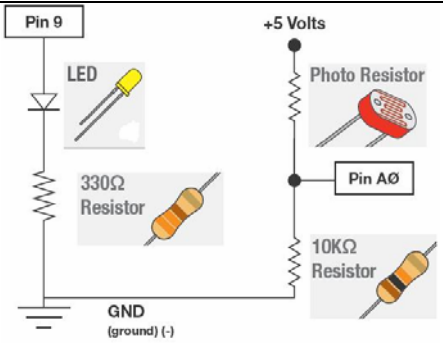
circuit program



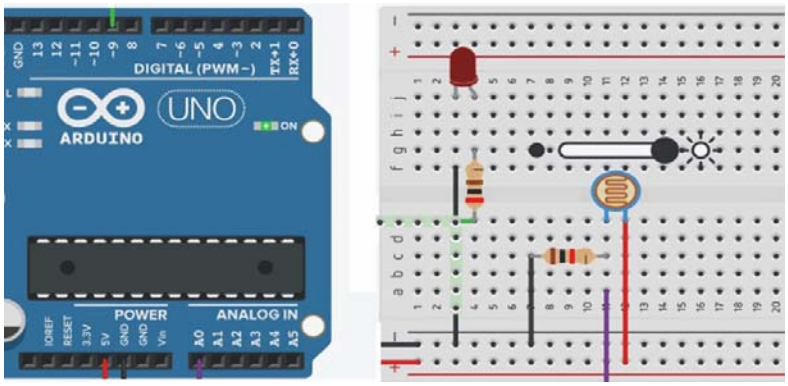
BLOCK 3 ANALOG SIGNALS and SERIAL PORT

☉ In the circuit below:

- Draw its circuit diagram (with symbols)
- (**web**) In an **electronic control**, when the LDR is well lit, the resistance is low/high and current passes through. The brighter it is, the less/more current goes through. So, the LED will be lighting up in night time/day time. LDR are digital /analogue sensor.

electronic control	circuit diagram	**programmed control**
		

- (web) In a **programmed control** as we slide the wiper from left to right the external light increases/decreases and so the resistance of the LDR increases/decreases.
- Wire the circuit below taking into account the circuit above (on the left)



- How can we see the different values from the LDR?
- What is the main difference between both controls?