

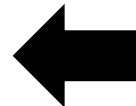
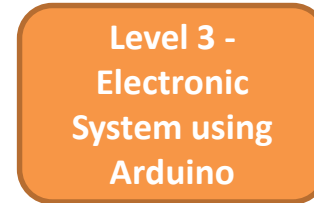
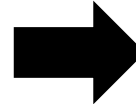
Module 2 - mBlock Programming



Program Outline

Outcomes:-

- participants are able to:-
1. describe how internet works
 2. describe 'digital technology'
 3. describe how computers work



Outcomes:-

- participants are able to:-
1. Able to execute simple programming functions
 2. able to read digital and analog inputs
 3. able to display digital output

Outcomes:-

- participants are able to:-
1. read data sheet of basic electronics components
 2. construct simple electronic circuits
 3. design a simple electronic system on open source platform

Outcomes:-

- participants are able to:-
1. describe IoT concept
 2. develop small scale website
 3. develop a small electronic system that is able to control via apps

PRE-LEARNING PREPARATION

Please ensure that you have the following:

1

PERSONAL COMPUTER

Running Windows, Linux or MacOS with a USB port

2

ARDUINO BOARD with USB Cable

This guide uses UNO, but you can use any version of the ARDUINO board out there

3

ELECTRONIC COMPONENTS

Contains all necessary components and parts for all exercises

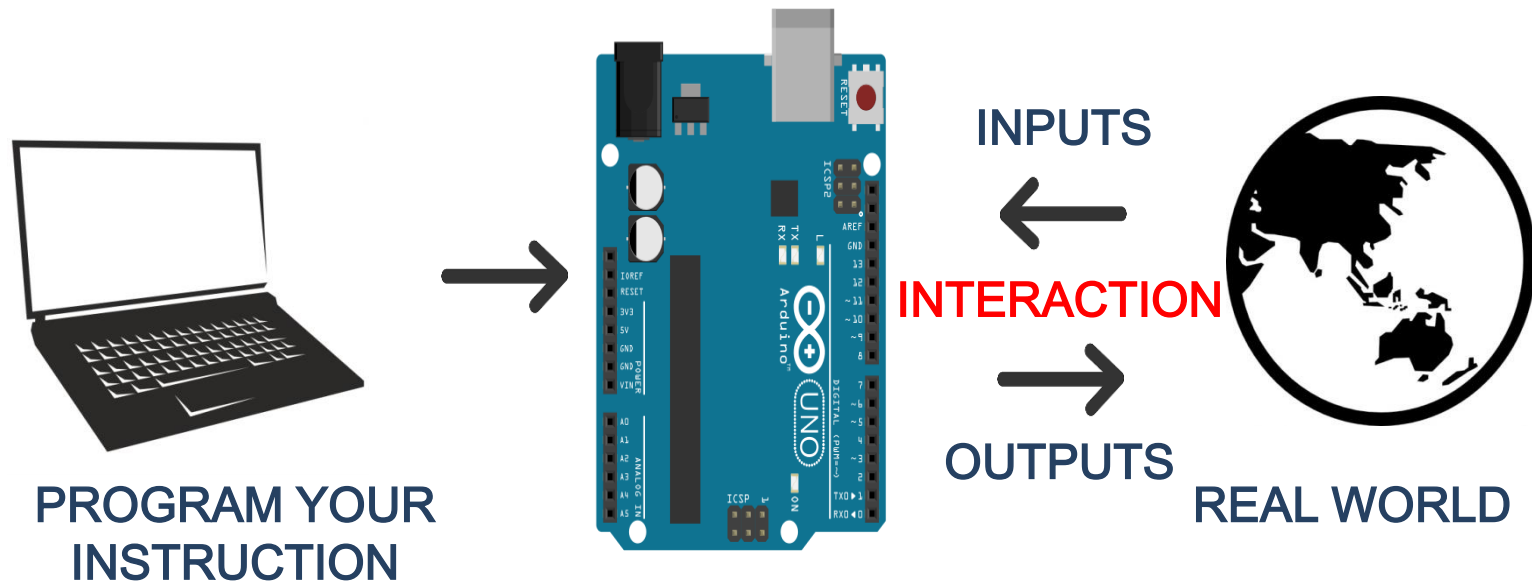
4

mBlock and ARDUINO SOFTWARE

- Referred to as an Integrated Developers Environment (IDE).
- Download the latest version according to your operating system (Windows, MacOS or Linux) at <http://arduino.cc/en/main/software>
- Once downloaded, click the executable file and follow the instructions
- A shortcut will be create on your desktop along with an Arduino folder in Mydocument

Survey – Pre-program

<https://goo.gl/Zpp1Gm>



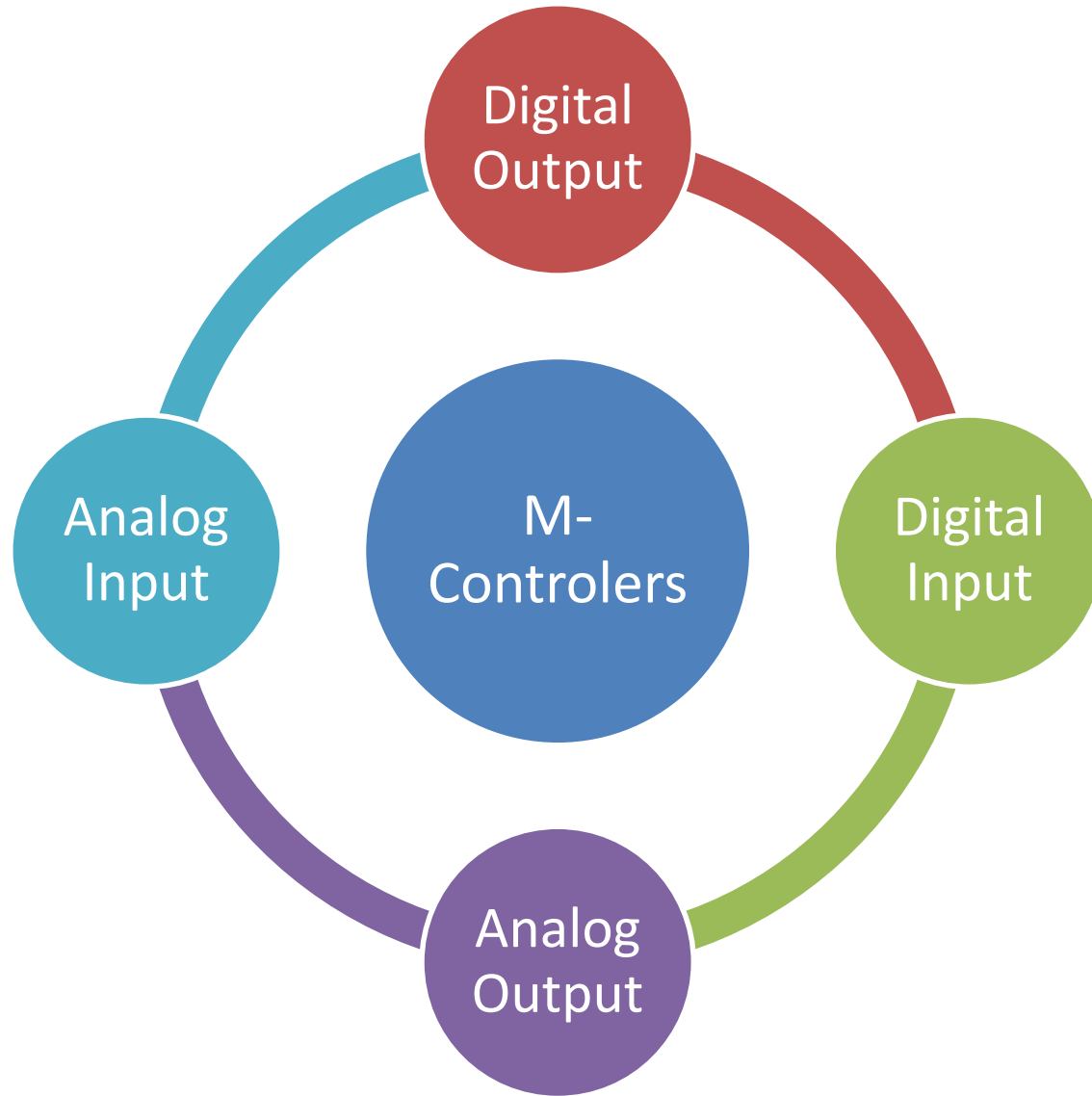
Microcontrollers are **dedicated** to one task and run one specific program

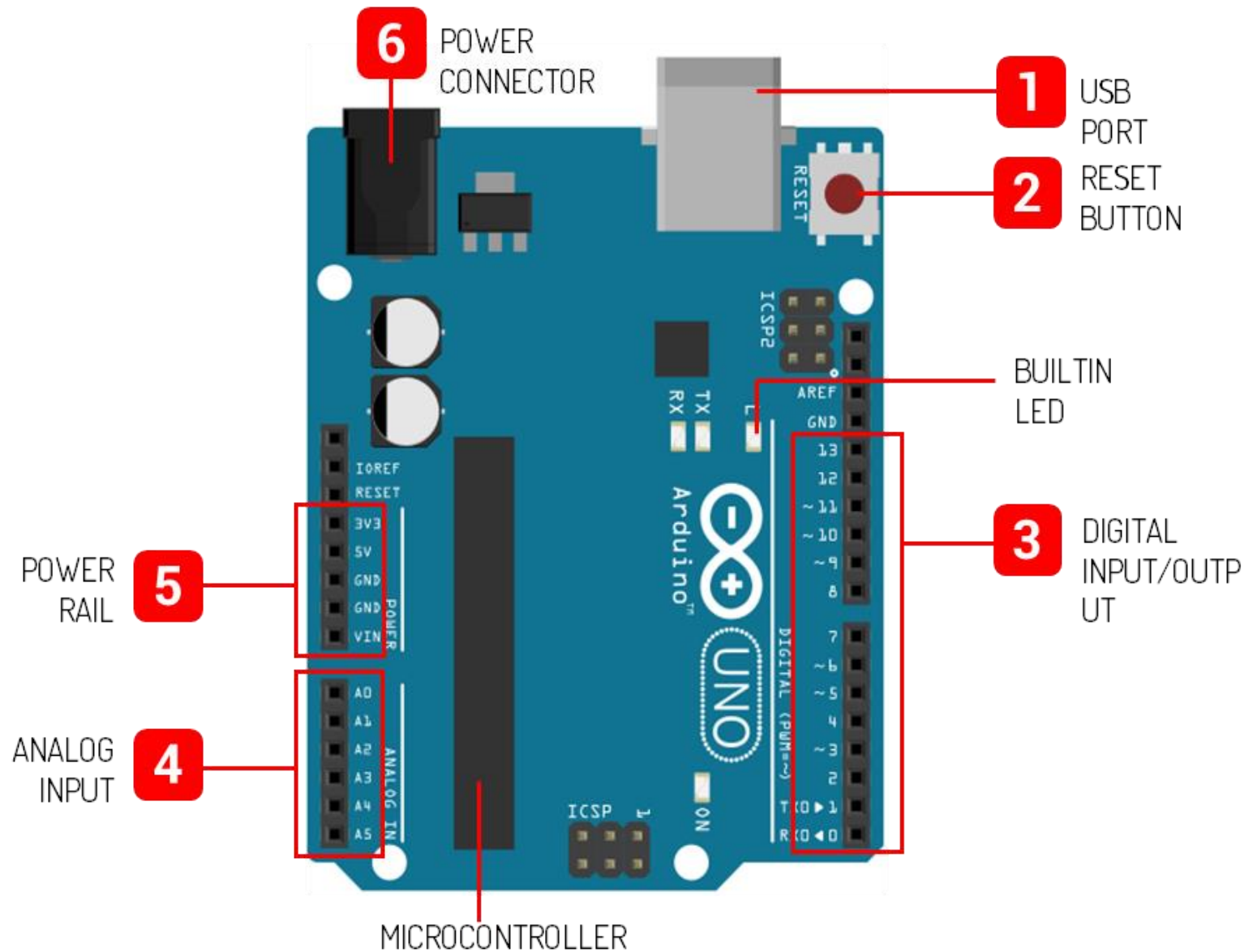
Examples of tasks could be:

- i. **Received from inputs** via ports (read from external hardware)
- ii. **Process the data**, store in file registers, arithmetic operations (added, subtracted, logic gates), etc.
- iii. **Control outputs** (control hardware)

Processor, Storage and RAM all in one tiny package







mBlock Console

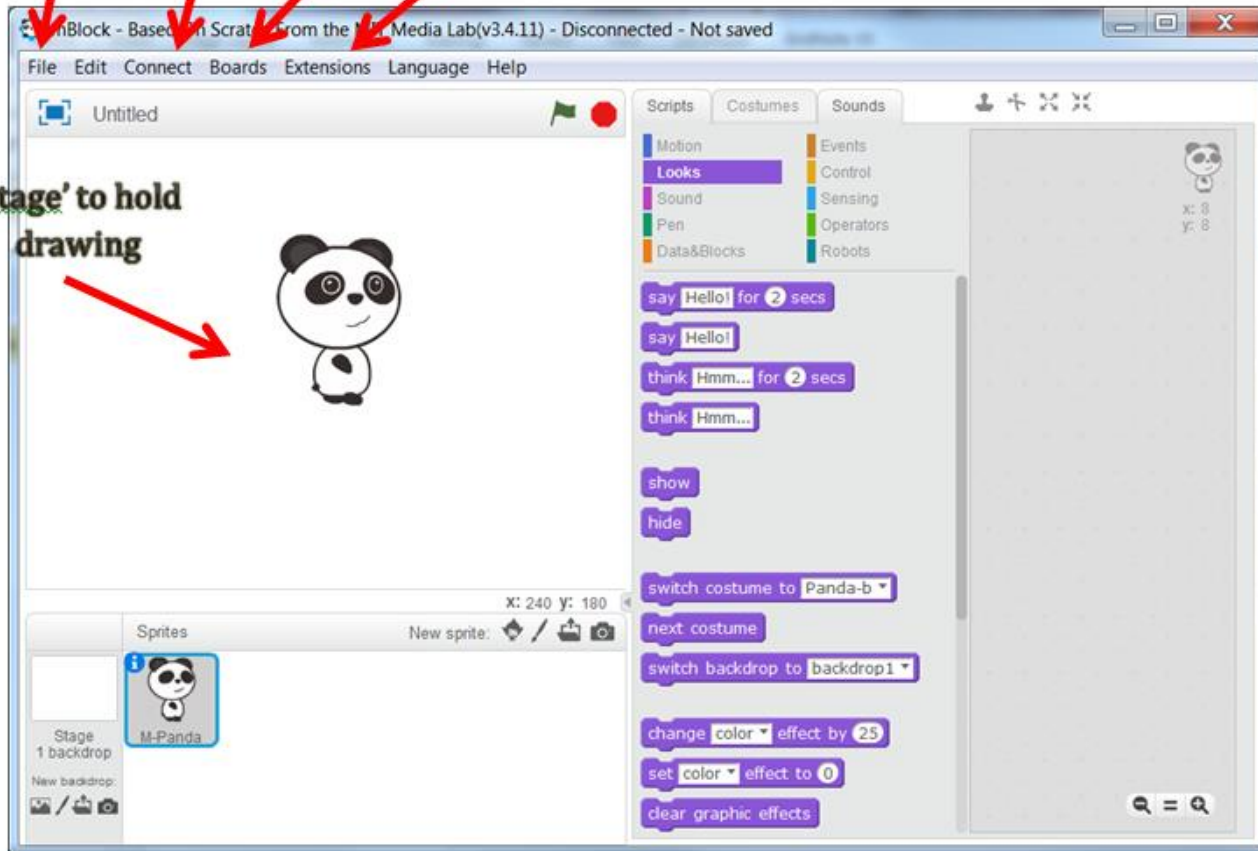
Save / Load files

Select the targeted hardware

Connection with firmware

Introduce new blocks

'stage' to hold drawing



mBlock Console

Connection with firmware

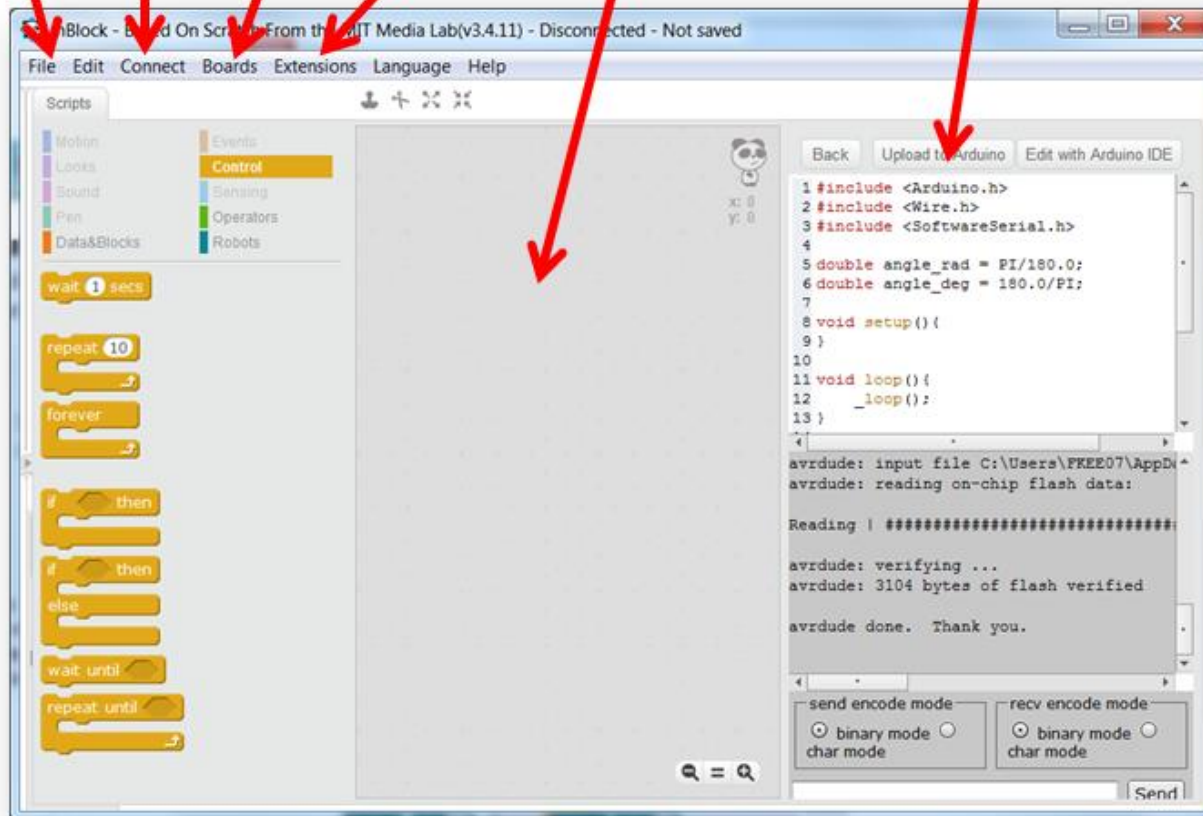
Save or load files

Select the targeted hardware

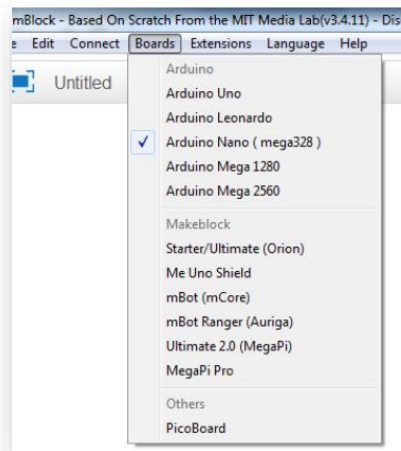
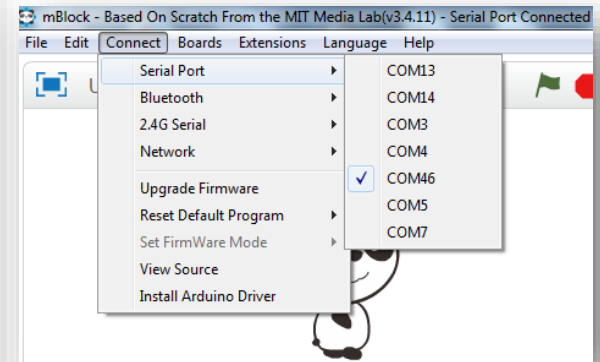
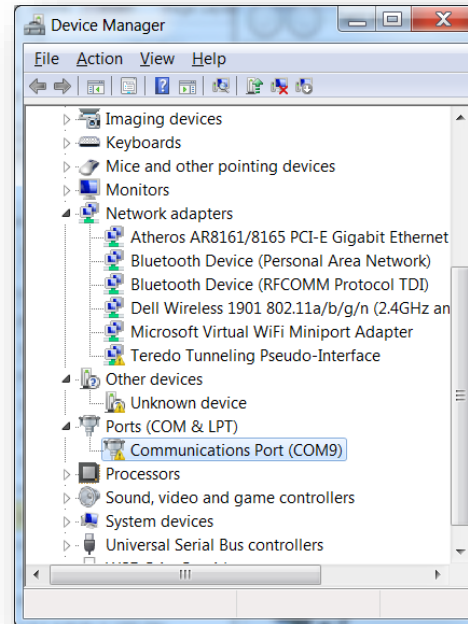
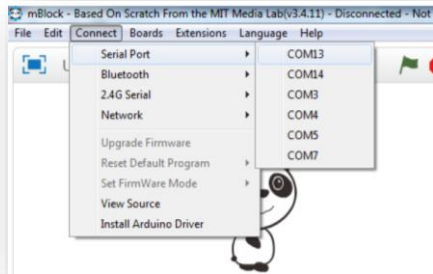
Introduce new blocks

Design Canvas

Equivalent Arduino codes



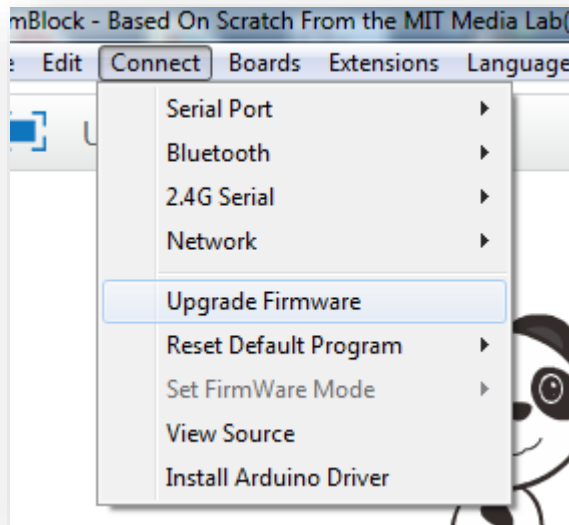
Successful Connection



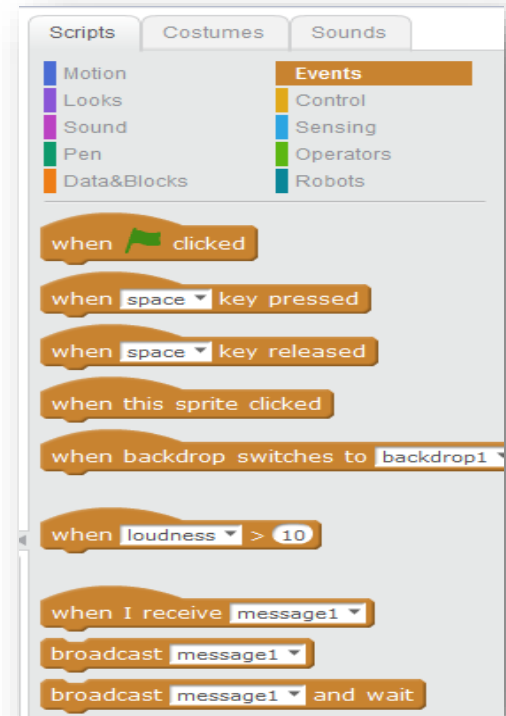
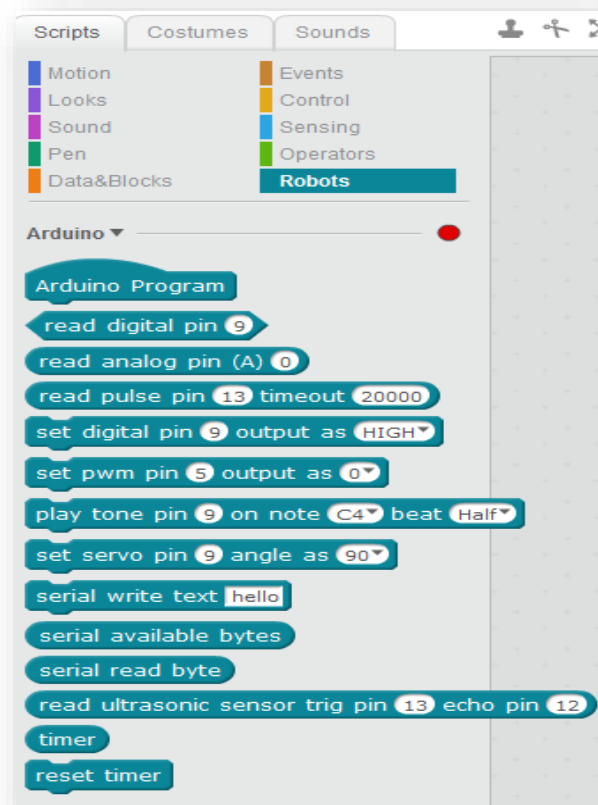
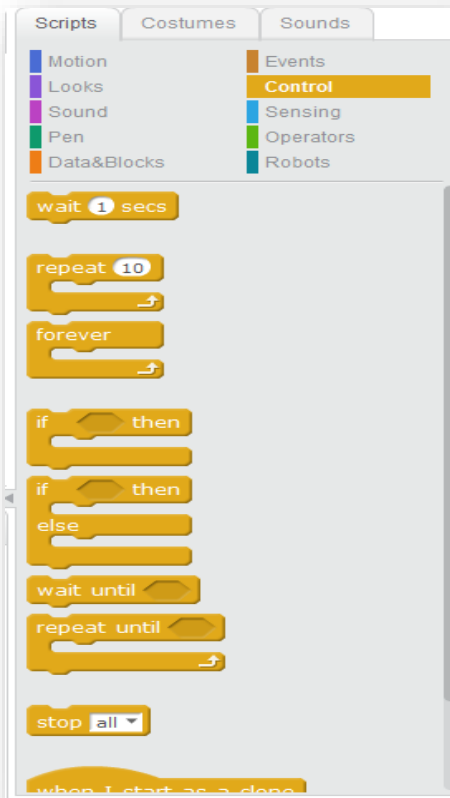
Two Ways Communication

- Connection between mBlock to Arduino
 - **“Connect”** menu, select **“Upgrade Firmware”**. Wait until the upgrade is complete
 - This allow the mBlock to talk to the Arduino. Cable needs to be connected at all time

***Try this**



mBlock Functions

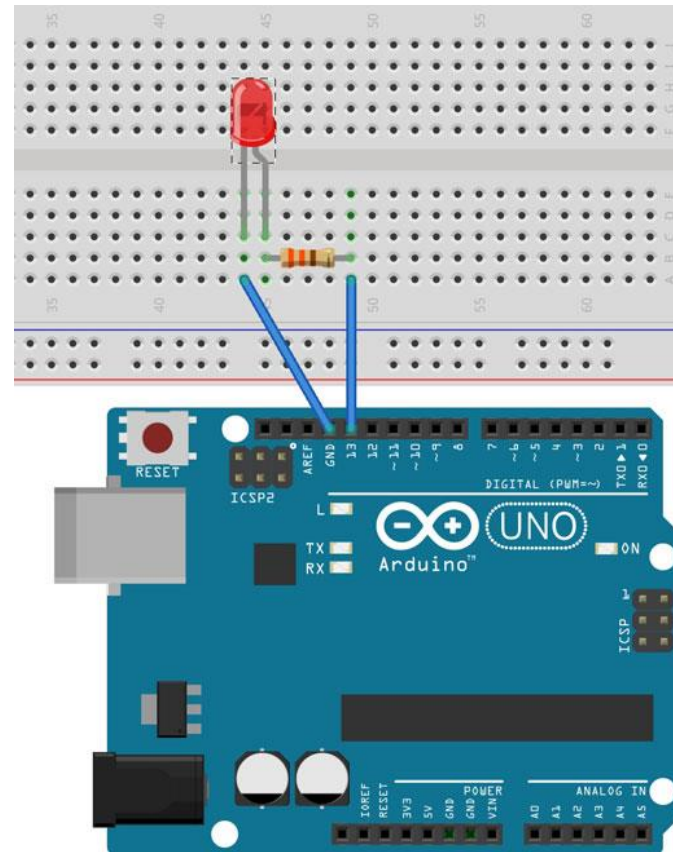


Logic Sequence Programming

Required Components

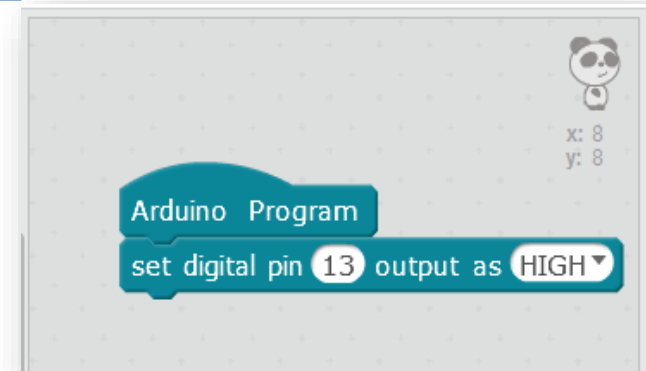
1. Arduino Uno (1 unit)
2. LED (1 unit)
3. Resistor (1 unit)
4. Jumpers (2 units)

Circuit Assembly

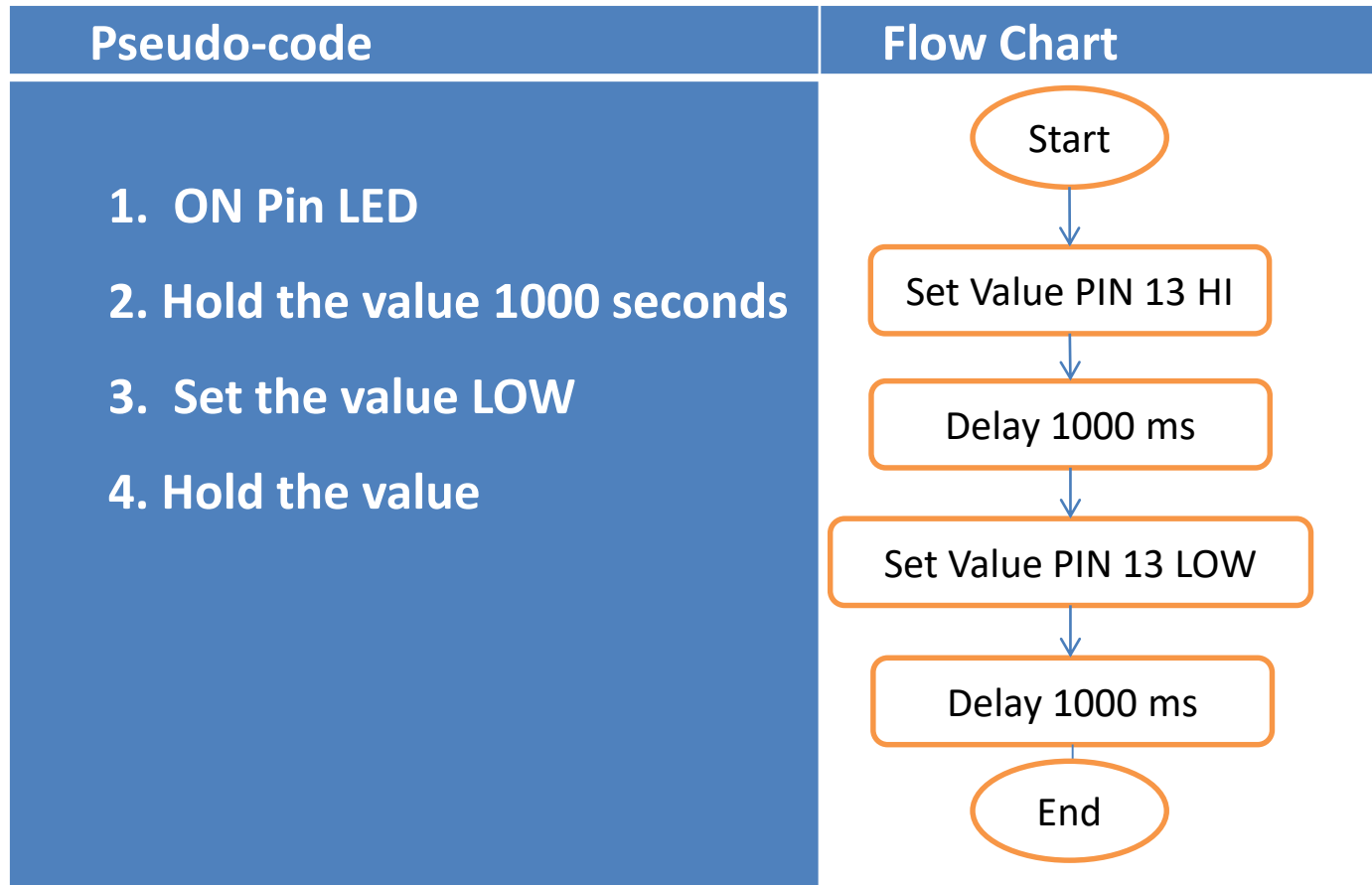


Digital Output – LED ON

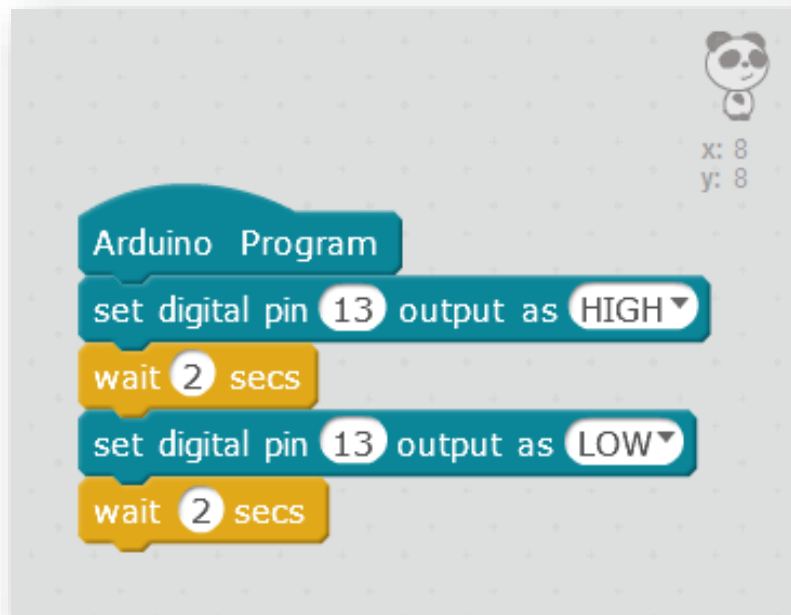
Pseudo-code	Flow Chart
<p>Pin in use 13</p> <p>1. ON Pin LED</p>	<pre>graph TD; Start([Start]) --> SetValue[Set Value PIN 13 HI];</pre> <p>The flowchart starts with an oval labeled 'Start'. An arrow points down to a rounded rectangular process box labeled 'Set Value PIN 13 HI'. From the bottom of this box, an arrow points down to a grey rectangular area containing a block programming interface.</p>



Digital Output – LED Blinking



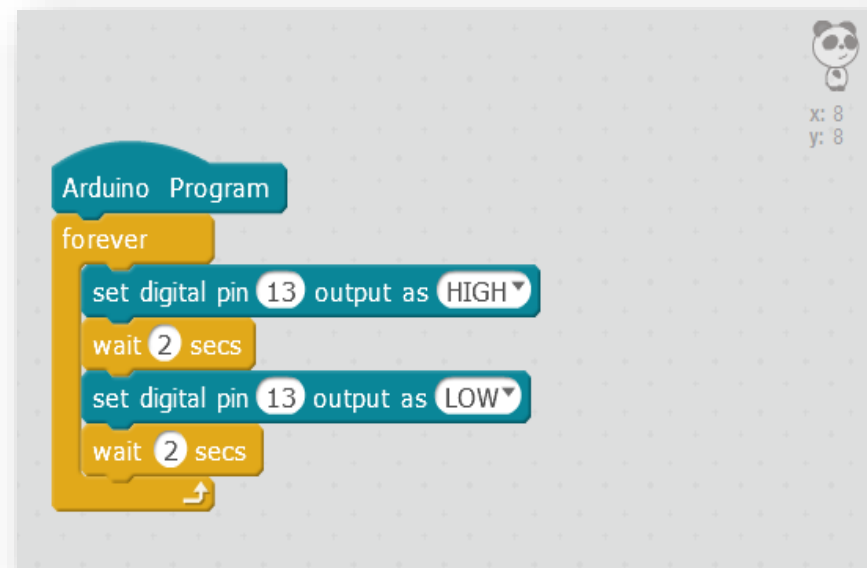
Digital Output – LED Blinking



Digital Output – LED Blinking Continuously (*at a 1s rate*)

Pseudo-code	Flow Chart
<ol style="list-style-type: none">1. ON Pin LED2. Hold the value 1000 seconds3. Set the value LOW4. Hold the value5. Repeat	<pre>graph TD; Start([Start]) --> SetHI[Set Value PIN 13 HI]; SetHI --> Delay1[Delay 1000 ms]; Delay1 --> SetLow[Set Value PIN 13 LOW]; SetLow --> Delay2[Delay 1000 ms]; Delay2 --> SetHI;</pre> <p>The flowchart illustrates a continuous loop for LED blinking. It begins with an oval labeled 'Start'. The process then follows a sequence of four rectangular process boxes: 'Set Value PIN 13 HI', 'Delay 1000 ms', 'Set Value PIN 13 LOW', and 'Delay 1000 ms'. A feedback arrow connects the bottom of the final delay box back to the top of the 'Set Value PIN 13 HI' box, forming a continuous loop.</p>

Digital Output – LED Blinking Continuously (*at a 1s rate*)



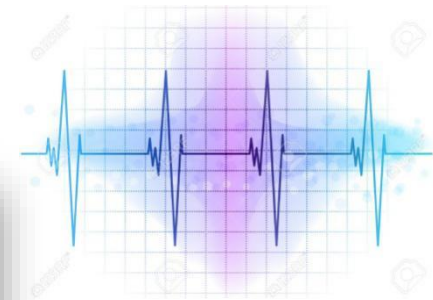
Digital Output – LED Blinking

Continuously (*Blink to mimic a heartbeat*)



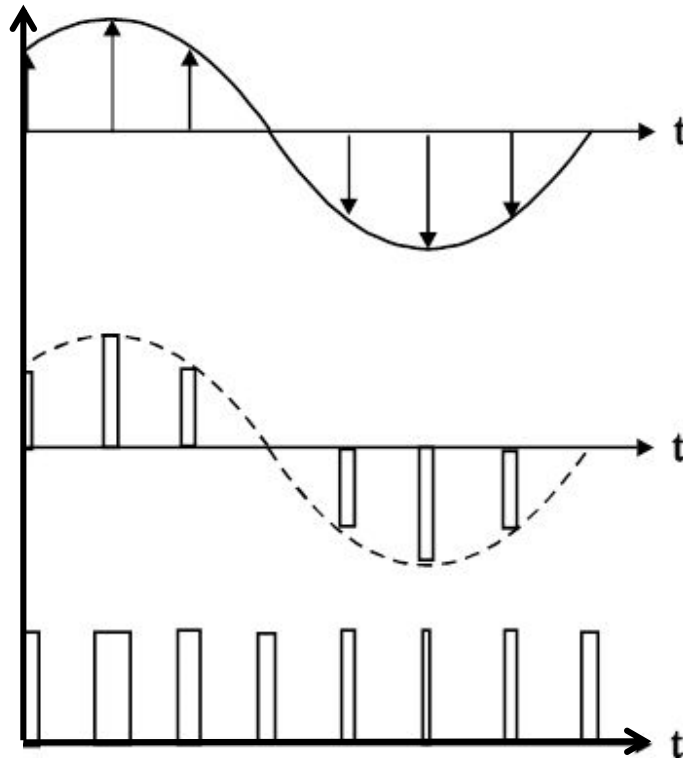
The image shows the Arduino IDE interface. On the left, the 'Arduino Program' is displayed in a block-based format. It consists of a 'forever' loop containing the following blocks: 'set digital pin 13 output as HIGH', 'wait 0.1 secs', 'set digital pin 13 output as LOW', 'wait 0.1 secs', 'set digital pin 13 output as HIGH', 'wait 0.1 secs', 'set digital pin 13 output as LOW', and 'wait 3 secs'. On the right, the C++ code is shown, which implements the same logic. The code includes headers for Wire and SoftwareSerial, defines constants for angles, and sets up pin 13 as an output. The loop function writes HIGH and LOW to the pin with 0.1-second delays, followed by a 3-second delay. A custom _delay function is used for the longer delay.

```
2 #include <Wire.h>
3 #include <SoftwareSerial.h>
4
5 double angle_rad = PI/180.0;
6 double angle_deg = 180.0/PI;
7
8 void setup() {
9   pinMode(13,OUTPUT);
10 }
11
12 void loop() {
13   digitalWrite(13,1);
14   _delay(0.1);
15   digitalWrite(13,0);
16   _delay(0.1);
17   digitalWrite(13,1);
18   _delay(0.1);
19   digitalWrite(13,0);
20   _delay(3);
21   _loop();
22 }
23
24 void _delay(float seconds){
25   long endTime = millis() + seconds * 1000;
26   while(millis() < endTime)_loop();
27 }
28
29 void _loop(){
30 }
```



Analog Output - PWM

Analog Signal Representation



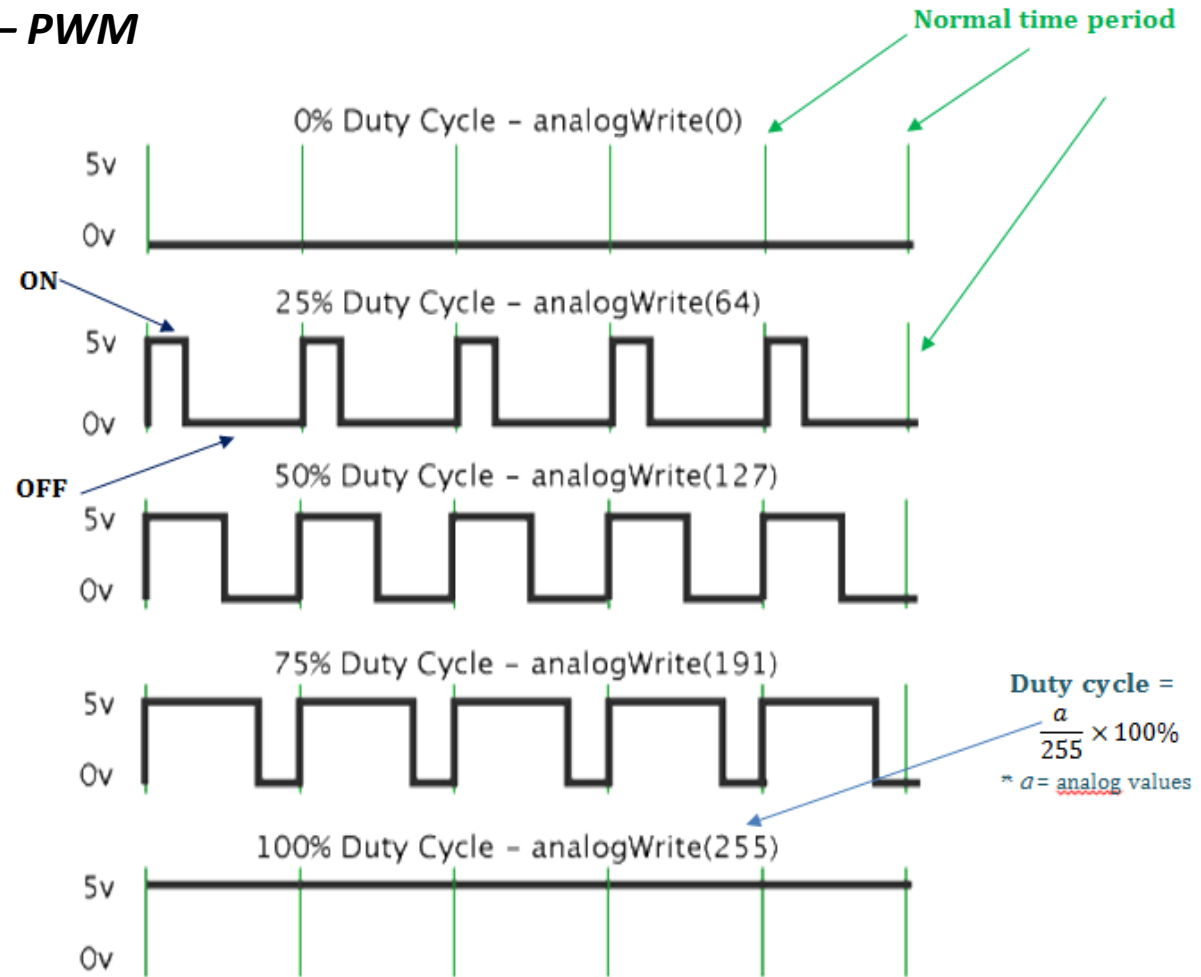
Analog Signal

Pulse Amplitude Modulation

Pulse Width Modulation

Analog Output - PWM

Pulse Width Modulation – PWM



Analog Output

Arduino Uno Pin Assignment – Analog Output

`analogWrite (pin, value);`

Pin = A0, A1, A2, A3, A4, A5

Value = From 0 to 255



Analog Output Fading Light

A Scratch 'set' block with the variable 'red' and the value '0'.

Set the initial value of Variable **red** as 0, and then the onboard LED is initially off.

A Scratch 'change' block with the variable 'red' and the value '1'.

Set the varied value of Variable **red** and the brightness increases by 1 each time

A Scratch 'wait' block with the value '0.05' and the unit 'secs'.

Avoid excessively fast change of the red light. Wait to control its changing speed.

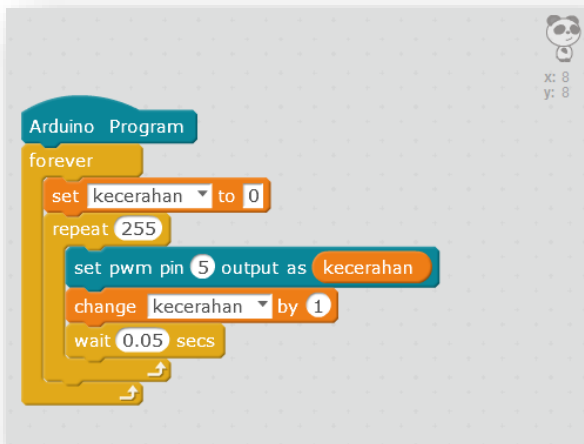
A Scratch 'repeat' block with the value '60' and a loop icon.

You can adjust the number “60” here and set the maximum brightness the onboard Led can achieve.

Analog Output Fading Light

Solution 2

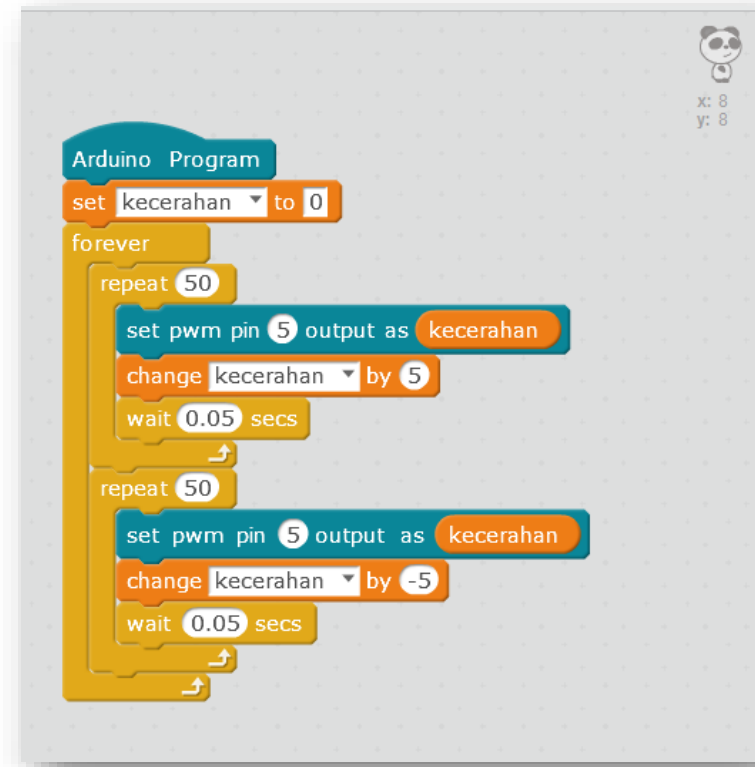
Solution 1



Arduino Program

```
forever
  set kecerahan to 0
  repeat 255
    set pwm pin 5 output as kecerahan
    change kecerahan by 1
    wait 0.05 secs
```

The code for Solution 1 is a Scratch script for an Arduino program. It starts with a 'forever' loop. Inside the loop, it sets a variable named 'kecerahan' to 0. Then, it enters a 'repeat' loop that runs 255 times. In each iteration, it sets the PWM output of pin 5 to the current value of 'kecerahan', increments 'kecerahan' by 1, and waits for 0.05 seconds.



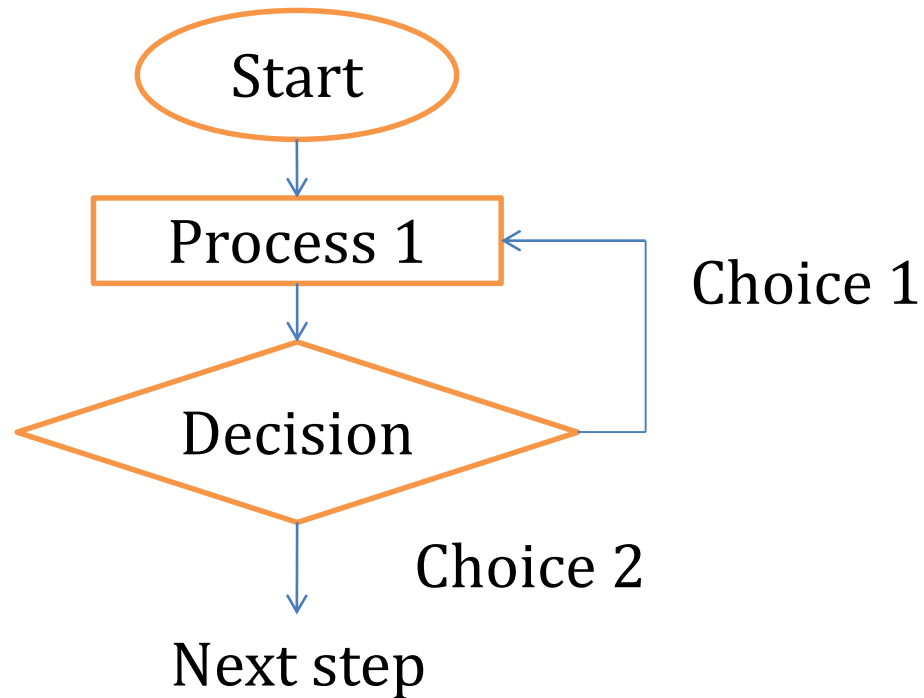
Arduino Program

```
set kecerahan to 0
forever
  repeat 50
    set pwm pin 5 output as kecerahan
    change kecerahan by 5
    wait 0.05 secs
  repeat 50
    set pwm pin 5 output as kecerahan
    change kecerahan by -5
    wait 0.05 secs
```

The code for Solution 2 is a Scratch script for an Arduino program. It starts by setting a variable named 'kecerahan' to 0. Then, it enters a 'forever' loop. Inside the loop, there are two 'repeat' blocks, each running 50 times. The first 'repeat' block sets the PWM output of pin 5 to the current value of 'kecerahan', increments 'kecerahan' by 5, and waits for 0.05 seconds. The second 'repeat' block sets the PWM output of pin 5 to the current value of 'kecerahan', decrements 'kecerahan' by 5, and waits for 0.05 seconds.

Digital Input

Decision making process

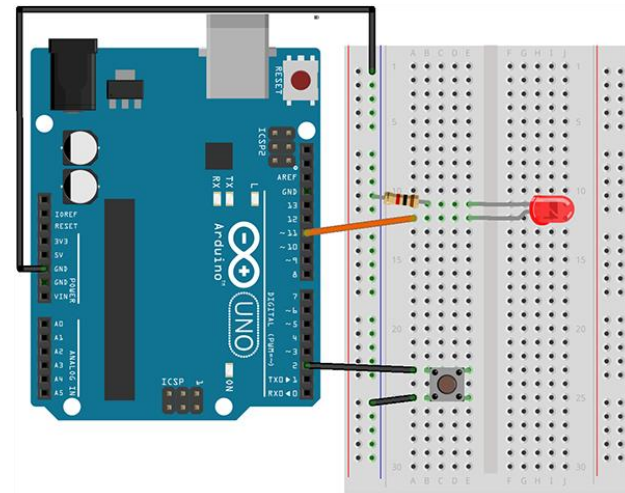


Selection Programming

Required Components

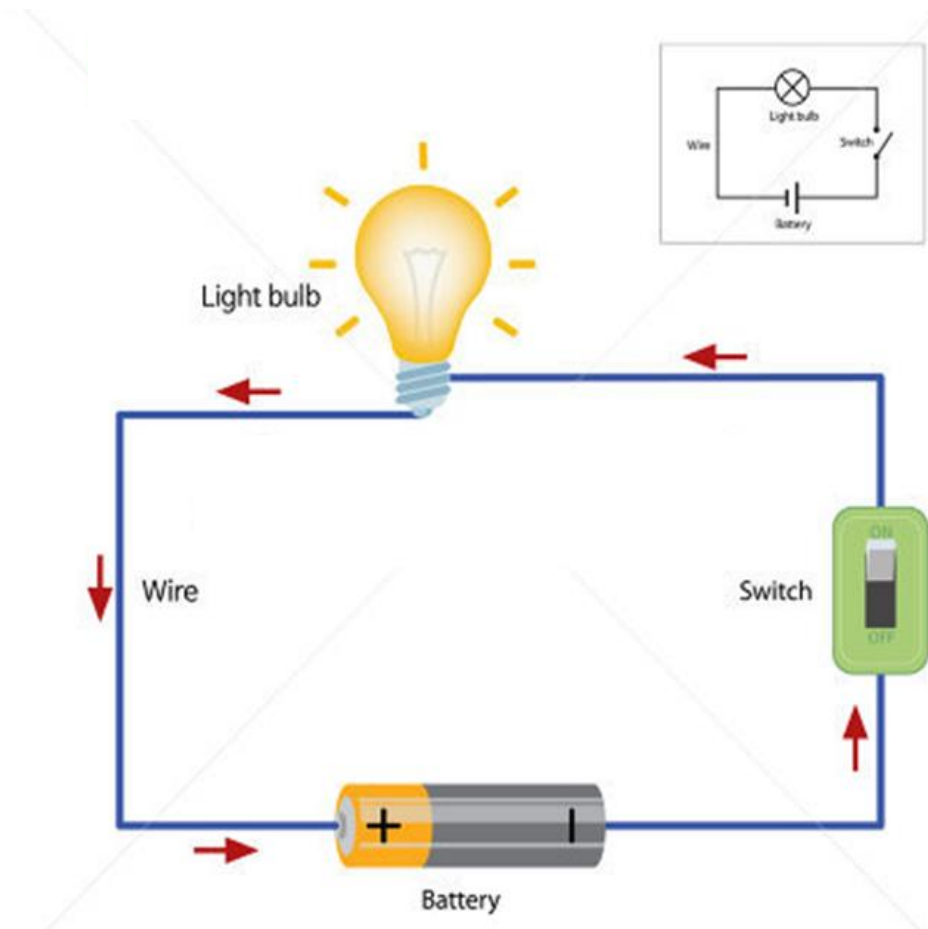
1. Arduino Uno (1 unit)
2. LED (1 unit)
3. Resistor (1 unit)
4. Jumpers (2 units)
5. Push button (1 unit)

Circuit Assembly

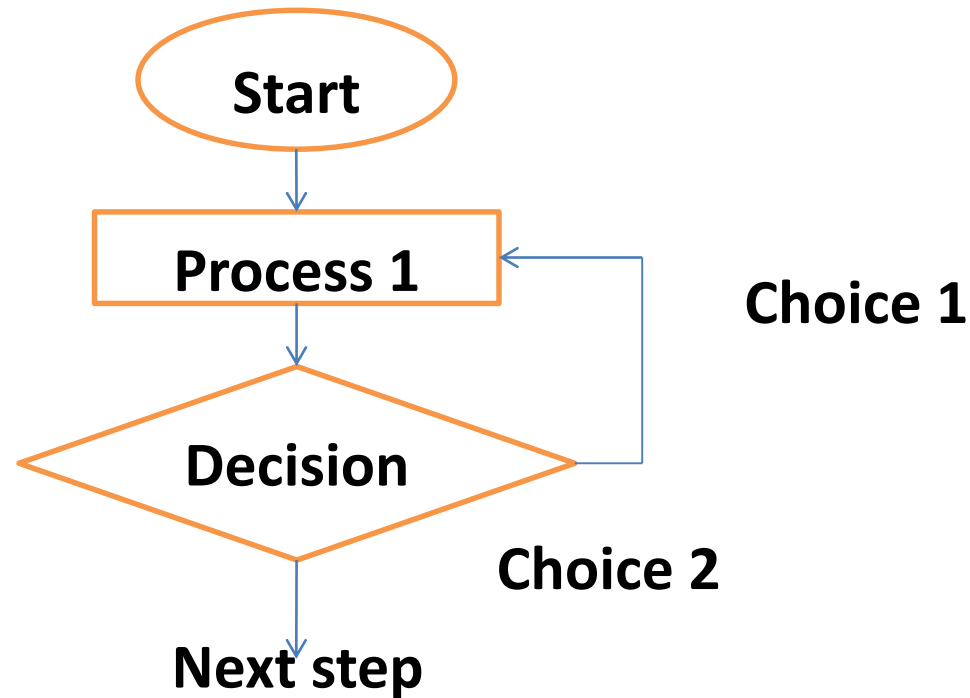


Switch

Solution 1



Push Button LED



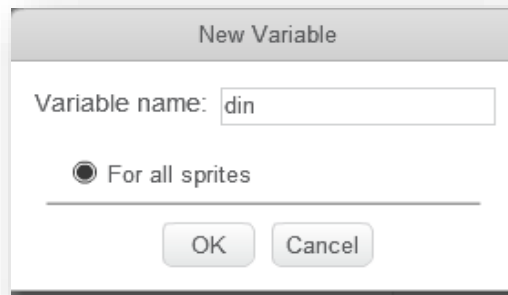
A 'decisions' is normally translated either as :-

- If-else, or
- while

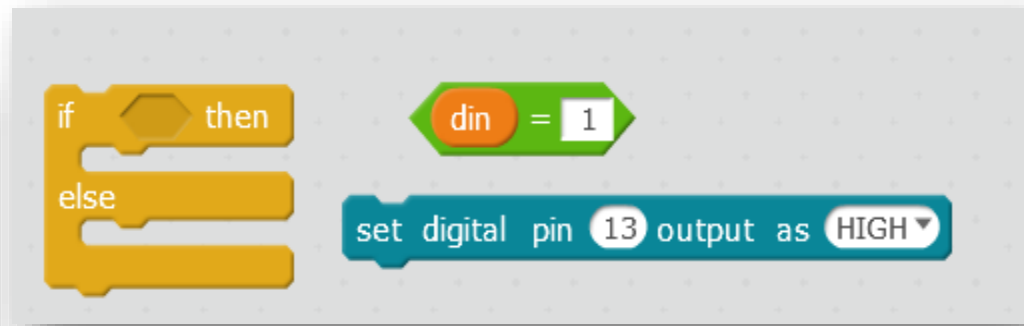
statements in a programming language

mBlock Variable

Create variable in mBlock



Conditional statement



mBlock Variable

Conditional statement

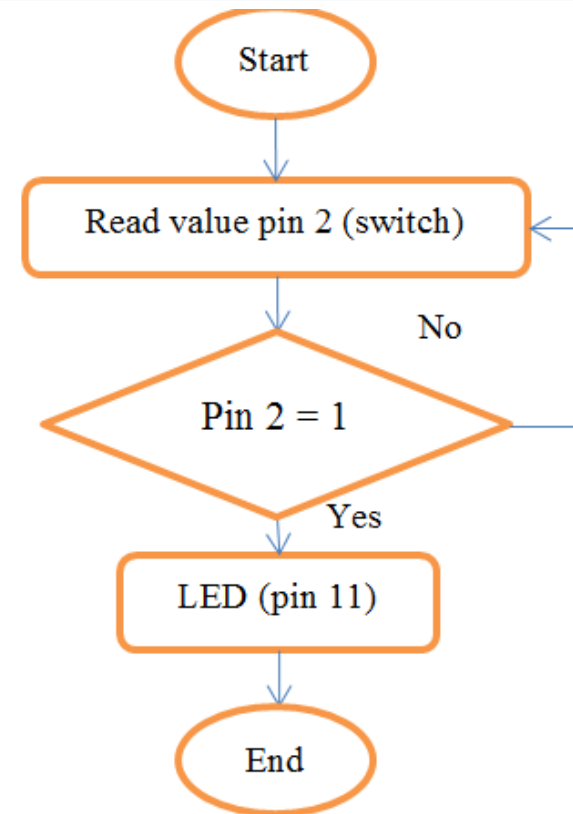
RELATIONSHIP	OPERATOR
Equal to	==
Not equal to	!=
Less than	<
Greater than	>
Less than or Equal to	<=
More than or Equal to	>=

Push Button LED

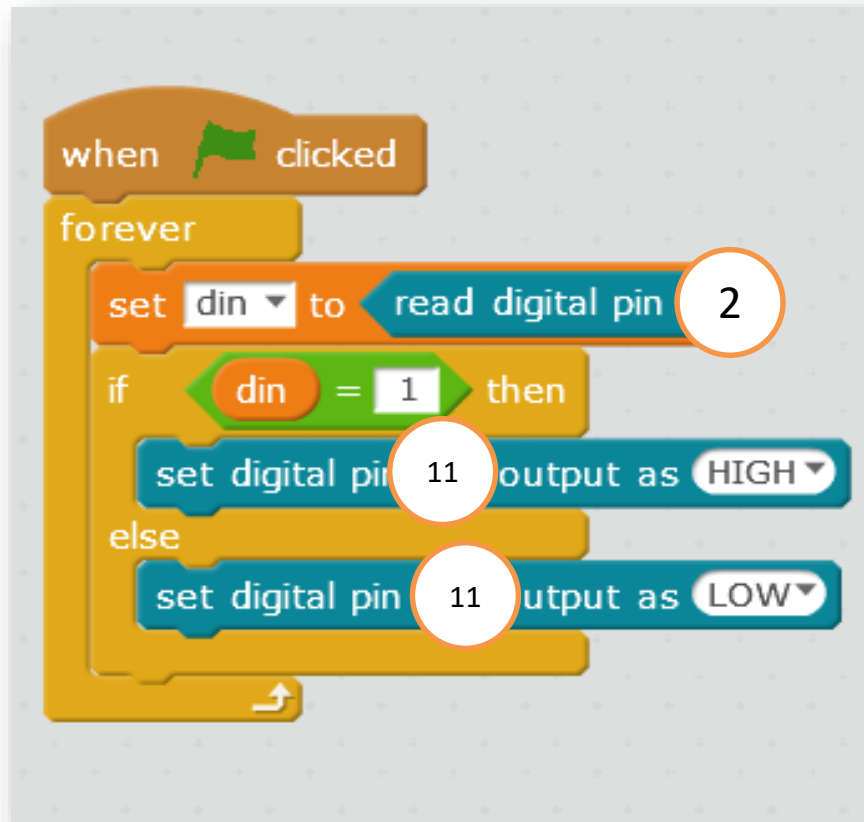
Pseudo-code

1. Declare
 - a. Switch as Input (pin 2)
 - b. LED as output (pin 11)
2. If the switch is pressed, LED ON, otherwise LED off

Flow Chart



Push Button LED



Analog Input

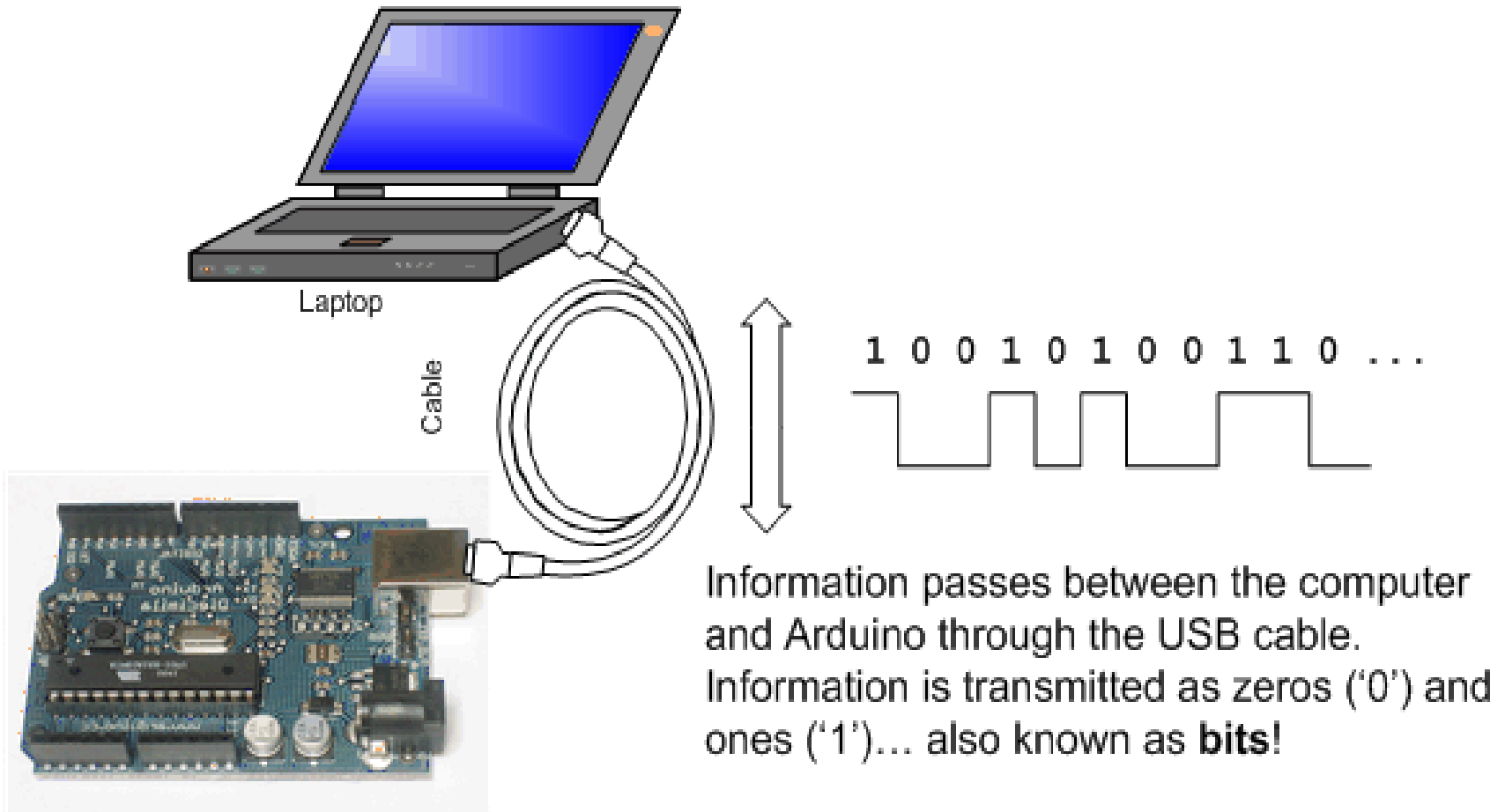
Data Types

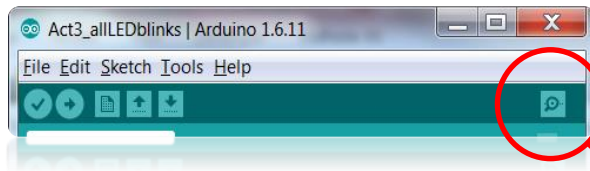
Types		
boolean (8 bit)	simple logical	true/false
byte (8 bit)	- unsigned number	from 0-255
char (8 bit) -	signed number. The compiler will attempt to interpret this data type as a character in some circumstances, which may yield unexpected results	from -128 to 127
unsigned char (8 bit) -	same as 'byte'; if this is what you're after, you should use 'byte' instead, for reasons of clarity	
word (16 bit) -	unsigned number	from 0-65535
unsigned int (16 bit)-	the same as 'word'. Use 'word' instead for clarity and brevity	
int (16 bit) -	signed number This is most commonly what you see used for general purpose variables in Arduino example code provided with the IDE	from -32768 to 32767.
unsigned long (32 bit) -	unsigned number The most common usage of this is to store the result of the millis() function, which returns the number of milliseconds the current code has been running	from 0-4,294,967,295
long (32 bit) -	signed number from -2,147,483,648 to 2,147,483,647	
float (32 bit) -	signed number Floating point on the Arduino is not native; the compiler has to jump through hoops to make it work. If you can avoid it, you should. We'll touch on this later	from -3.4028235E38 to 3.4028235E38.

Serial Communication

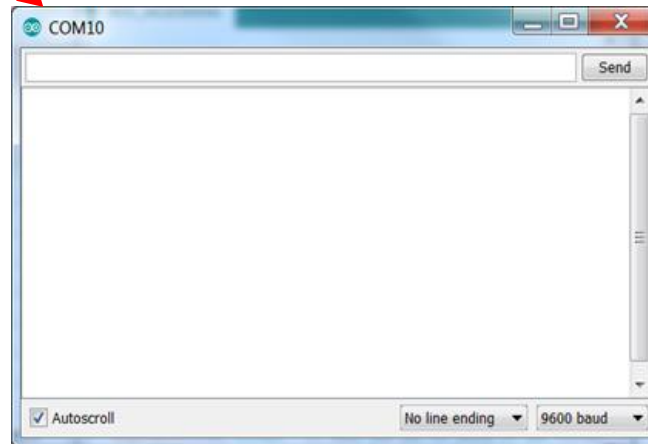
- Serial data transfer is a set of data that is transferred one bit at a time, one right after the other.
- Information of HIGH (1) and LOW (0) is passed back & forth between the computer and Arduino. Just like we used that technique to turn an LED on and off, we can also send data. One side sets the pin and the other reads it. It's a little like Morse code, where you can use dits and dahs to send messages by telegram.
- These values can be displayed on our computer's monitor and send information from the computer or any other serial devices to the Arduino board

Serial Communication





Serial monitor button



Analog to Digital Converter (ADC)

- Pin A0 – A7 on Arduino Nano are spared for analog voltages. Through these pins, the analog signals are converted to digital signals.
- This is the difference between an on/off sensor (which tells us whether something is there) and an analogue sensor, whose value continuously changes.
- The ADC on the Arduino is a 10-bit ADC meaning it has the ability to detect 1,024 (2^{10}) discrete analog levels. Some microcontrollers have 8-bit ADCs ($2^8 = 256$ discrete levels) and some have 16-bit ADCs ($2^{16} = 65,536$ discrete levels). Arduino Nano has 10 bit ADC.
- By using the `analogRead()` function, we can read the voltage applied to one of the pins. This function returns a number between 0 and 1023, which represents voltages between 0 and 5 volts.

Analog Digital Conversion



$$\frac{ADC \text{ Resolution}}{Voltage} = \frac{ADC \text{ Reading}}{Analog \text{ Voltage Measured}}$$

LED ON when LDR detects no light

Circuit and Programming:

LED

PIN3 to resistor 150 Ohm

Resistor to +ve LED

-ve LED to GND

Light Detected Resistor (LDR)

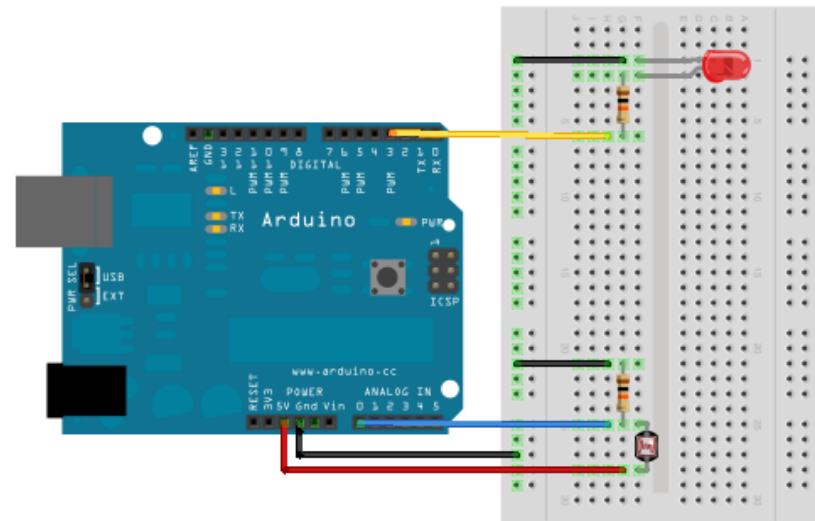
5V to LDRPIN1

PINA0 to LDRPIN2

LDRPIN2 to Resistor 1k Ω

Resistor 1k Ω to GND

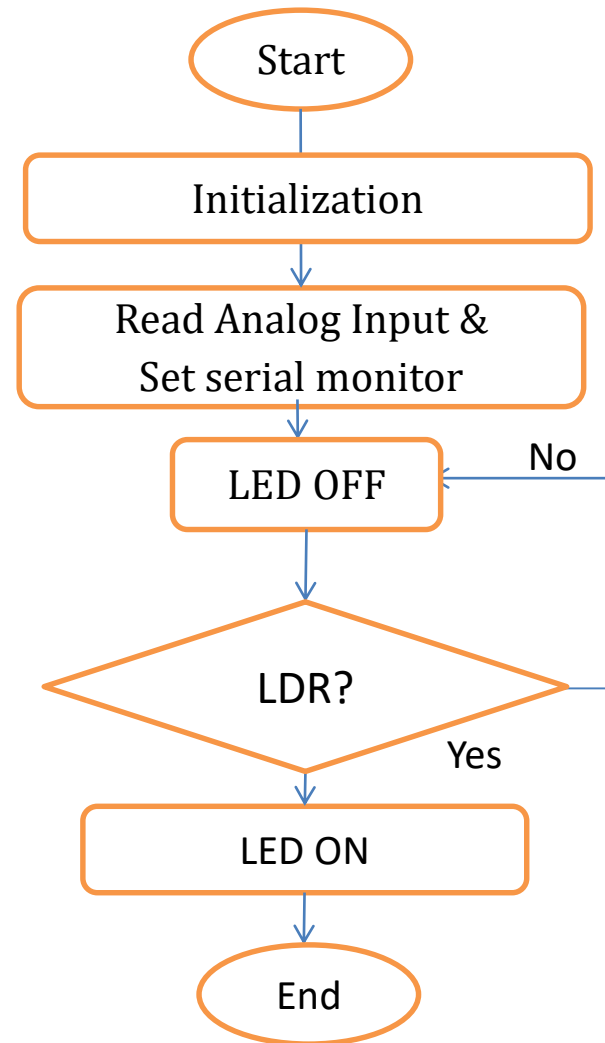
Resistor 1k Ω (brown,black,orange)



Pseudo-code

1. Initialize
 - a. Variable
 - b. Ports
2. Default condition LED OFF
3. SET Serial Monitor ON (to see the analog value)
4. Read Analog Input (LDR)
5. Check the analog value < 120 (Night?)
6. OFF LED if day light.
7. ON LED in night.
8. This will repeat in loop forever.

Flow Chart



LED ON when LDR detects no light

Create variable in mBlock

The image shows the mBlock code editor interface. On the left, a variable declaration panel lists 'cahaya' and 'din' as checked variables. Below this, a list of actions for the 'cahaya' variable is shown: 'set cahaya to 0', 'change cahaya by 1', 'show variable cahaya', and 'hide variable cahaya'. The main workspace contains a 'when clicked' event block followed by a 'forever' loop. Inside the loop, the first block is 'set cahaya to read analog pin (A) 0'. This is followed by an 'if cahaya < 255 then' block containing 'set digital pin 13 output as HIGH'. Below that is another 'if cahaya > 300 and cahaya < 800 then' block containing 'set digital pin 13 output as LOW'. A small panda icon and coordinates 'x: -1 y: -2' are visible in the bottom right corner of the workspace.

```
when clicked clicked
  forever
    set cahaya to read analog pin (A) 0
    if cahaya < 255 then
      set digital pin 13 output as HIGH
    if cahaya > 300 and cahaya < 800 then
      set digital pin 13 output as LOW
```

Code