ARDUINO & BASIC CONTROL

Contents

- ❑ Arduino

- ❑ Basic Control
  - ▪ Serial Input / Output
  - ▪ Hardware Control Basic

What is Arduino

- Arduino is an open source hardware/software programming platform based around Atmel microcontrollers.

  :Open source means that circuit schematics and source code of software used in designs is freely available and can be modified by enthusiasts.

  - Arduino Hardware(Single-board microcontroller)

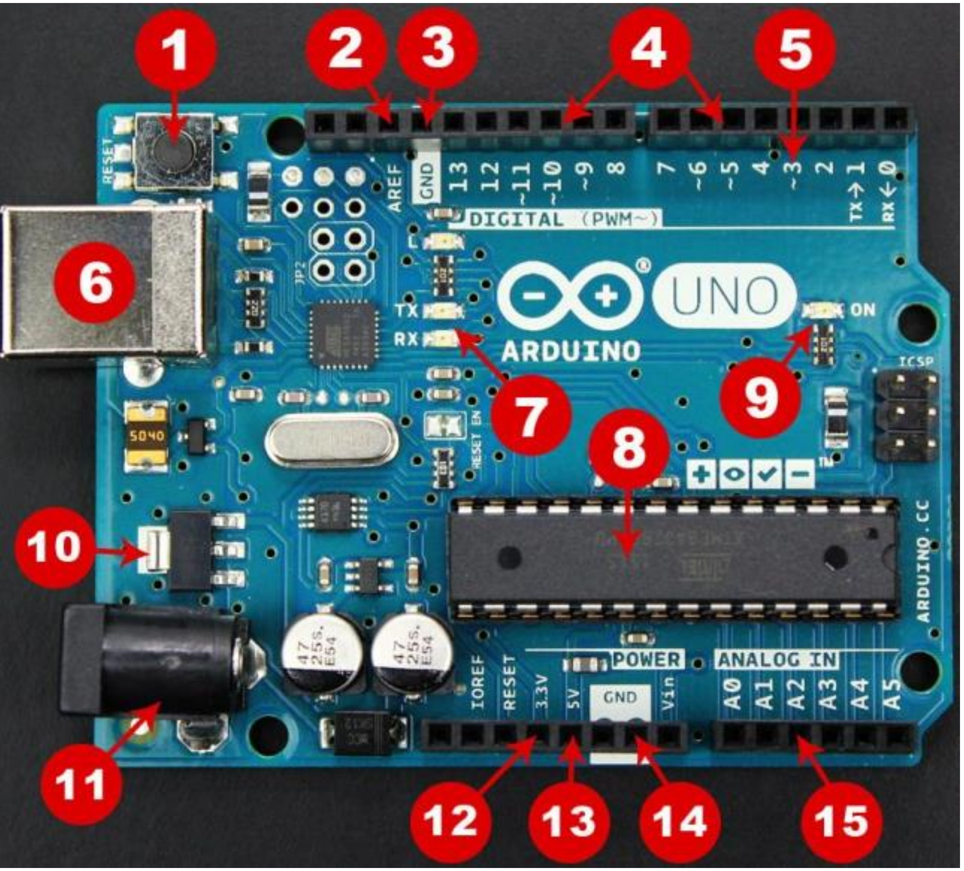
    : Microcontroller built onto a single printed circuit board that provides all of the circuitry necessary for a useful control task.

  - Arduino Software(IDE)
    - The open-source **Arduino Software** (**IDE**) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux.
    - The environment is written in Java and based on processing and other open source **software**.

Why Arduino

- ❑ **Inexpensive**: Arduino boards are relatively inexpensive compared to other microcontroller platforms.
- ❑ **Cross-platform**: The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems.
- ❑ **Simple, clear programming environment**: The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users
- ❑ **Open source and extensible hardware:** The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it.
- ❑ **Open source and extensible software**: The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries.

Arduino Microcontroller Board



1. Reset Button: restart any code loaded to board

2. AREF: Stands for “Analog Reference” and is used to set an external reference voltage

3, 14: Ground Pin

4. Digital Input/Output: Pins 0-13 can be used for digital input or output

5. PWM: The pins marked with the (~) symbol can simul ate PWM output

6. USB Connection: Used for powering up board and uploading software

7. TX/RX – Transmit and receive data indication LEDs

8. ATmega Microcontroller: This is the brains and is where the programs are stored

9. Power LED Indicator

10. Voltage Regulator: controls the amount of voltage going into the Arduino board

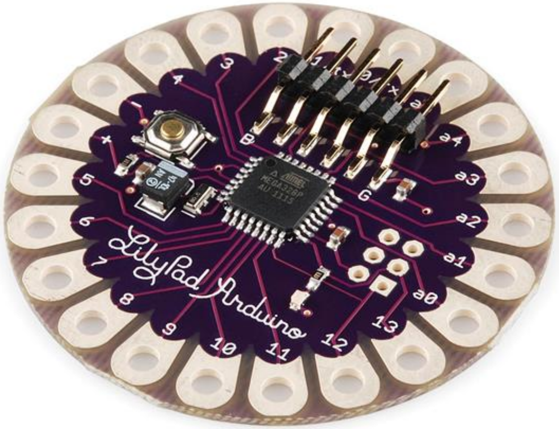
11. DC Power Jack: for powering board with a power supply

12. 3.3V Pin: supplies 3.3 volts of power to projects

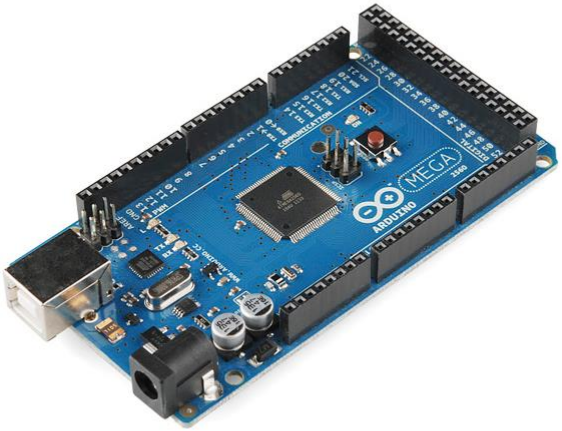
13. 5V Pin: supplies 5 volts of power to projects

15. Analog Pins: can read the signal from an analog sensor and convert it to digital

The Arduino Family

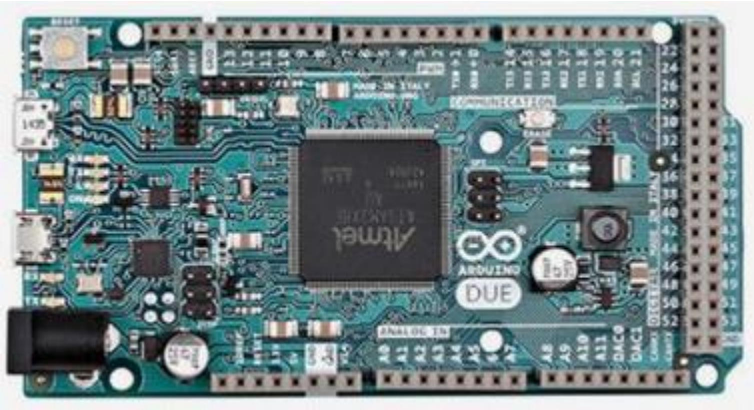


LilyPad Arduino

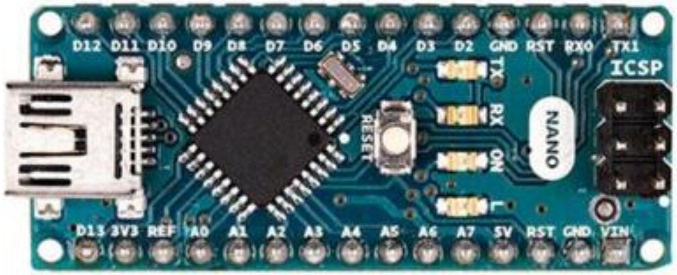


Arduino Mega (R3)

Arduino Leonardo

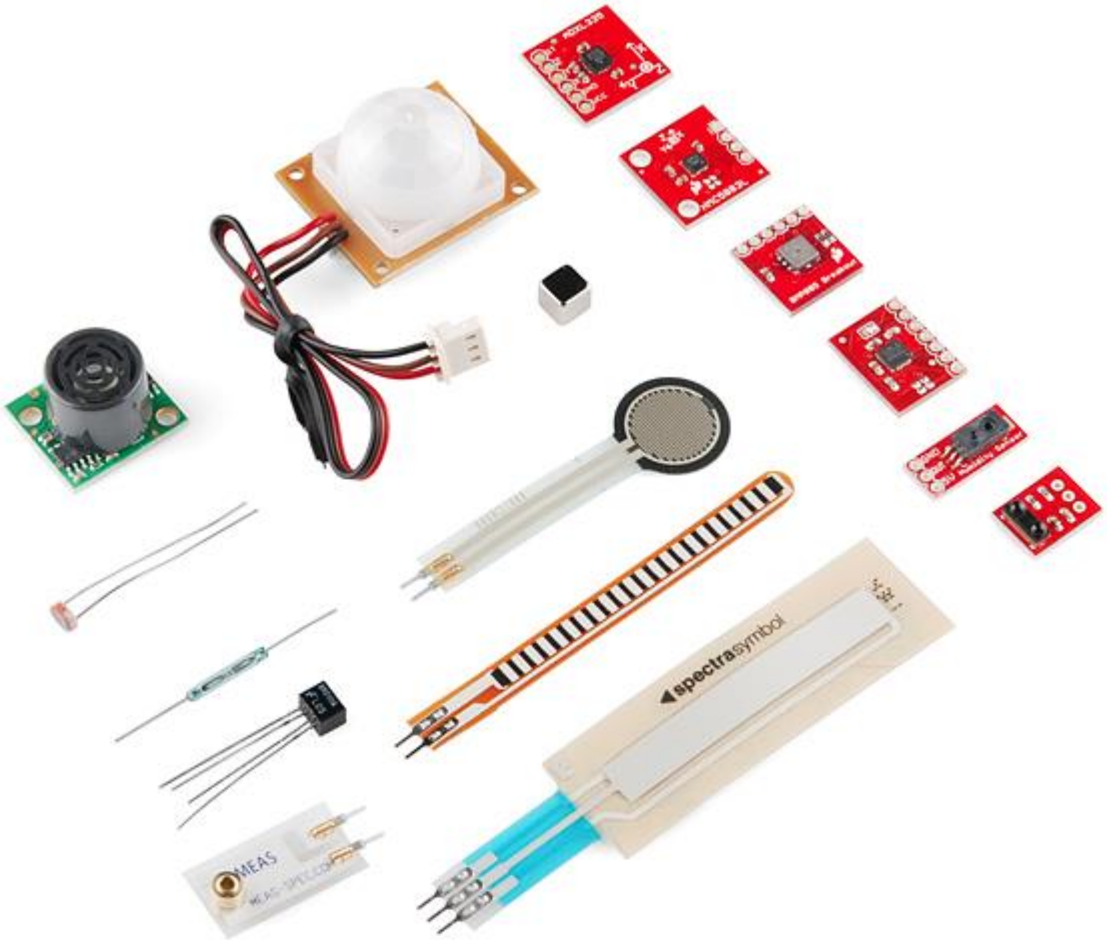


Arduino Due

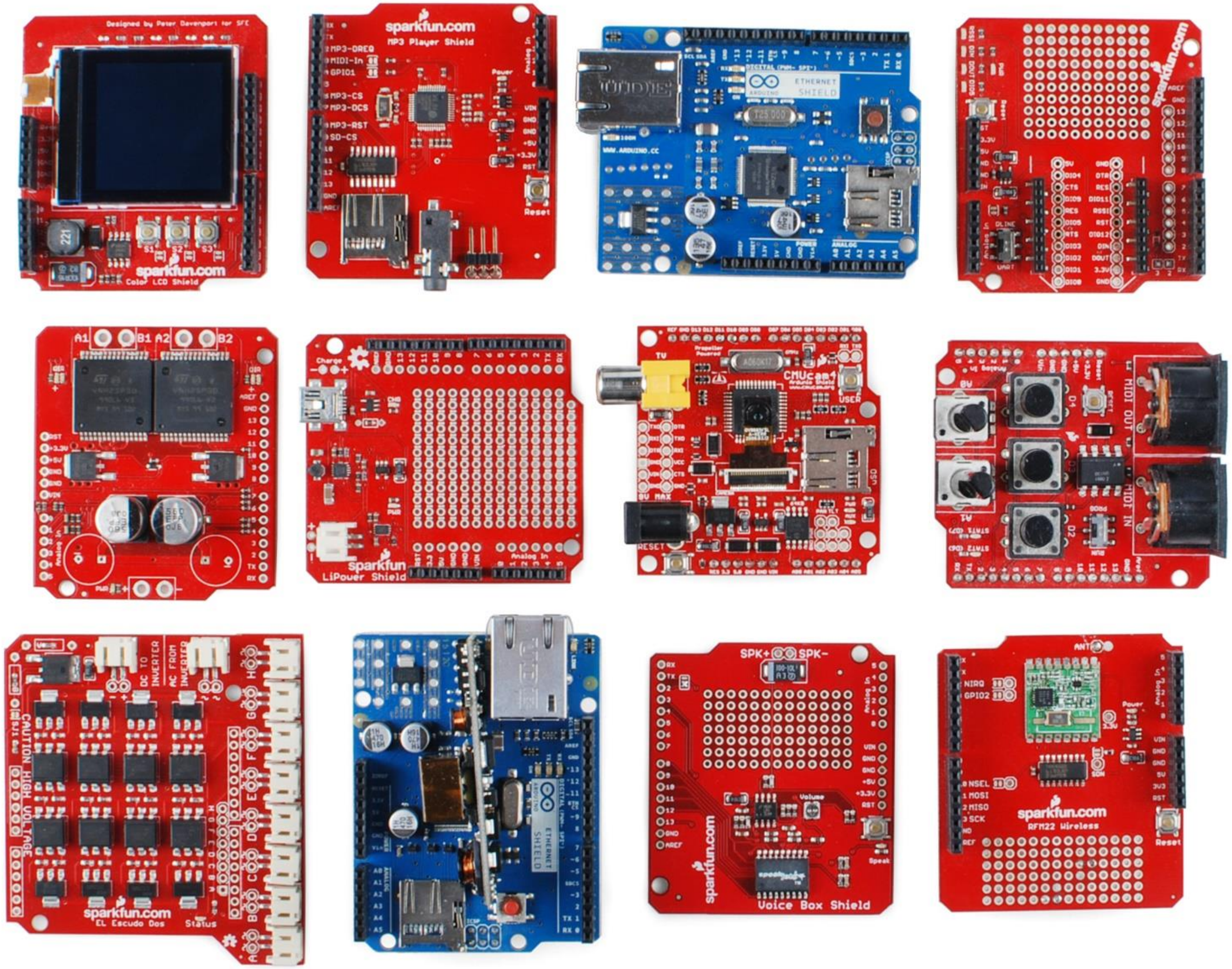

D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
ICSP
RESET
TX
RX
ON
L
NANO
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN

Arduino Nano

The Extended Family - Sensors

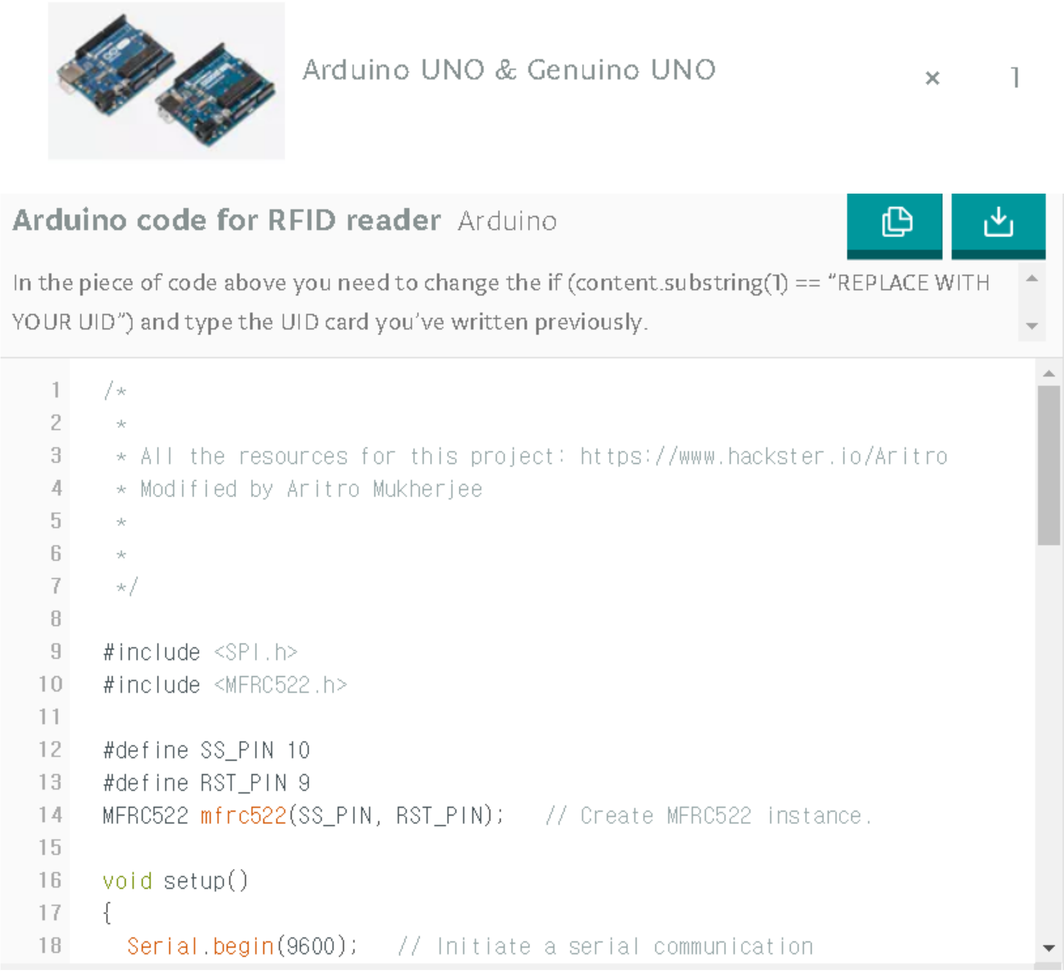


The Extended Family - Shields



Projects using Arduino UNO

Security Access Using RFID Reader



Arduino UNO & Genuino UNO × 1

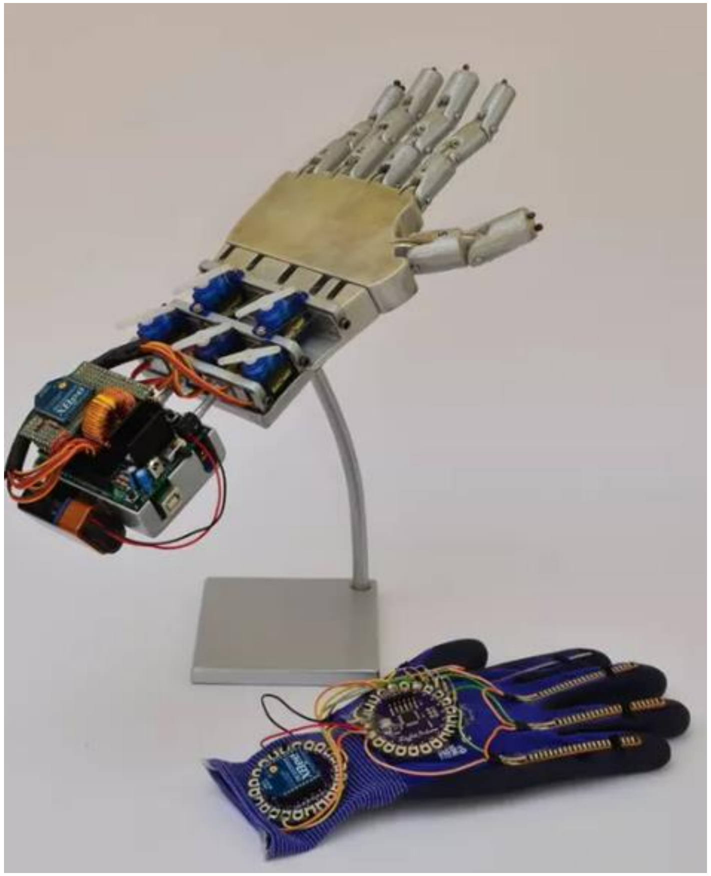
Arduino code for RFID reader Arduino

In the piece of code above you need to change the if (content.substring(1) == "REPLACE WITH YOUR UID") and type the UID card you've written previously.

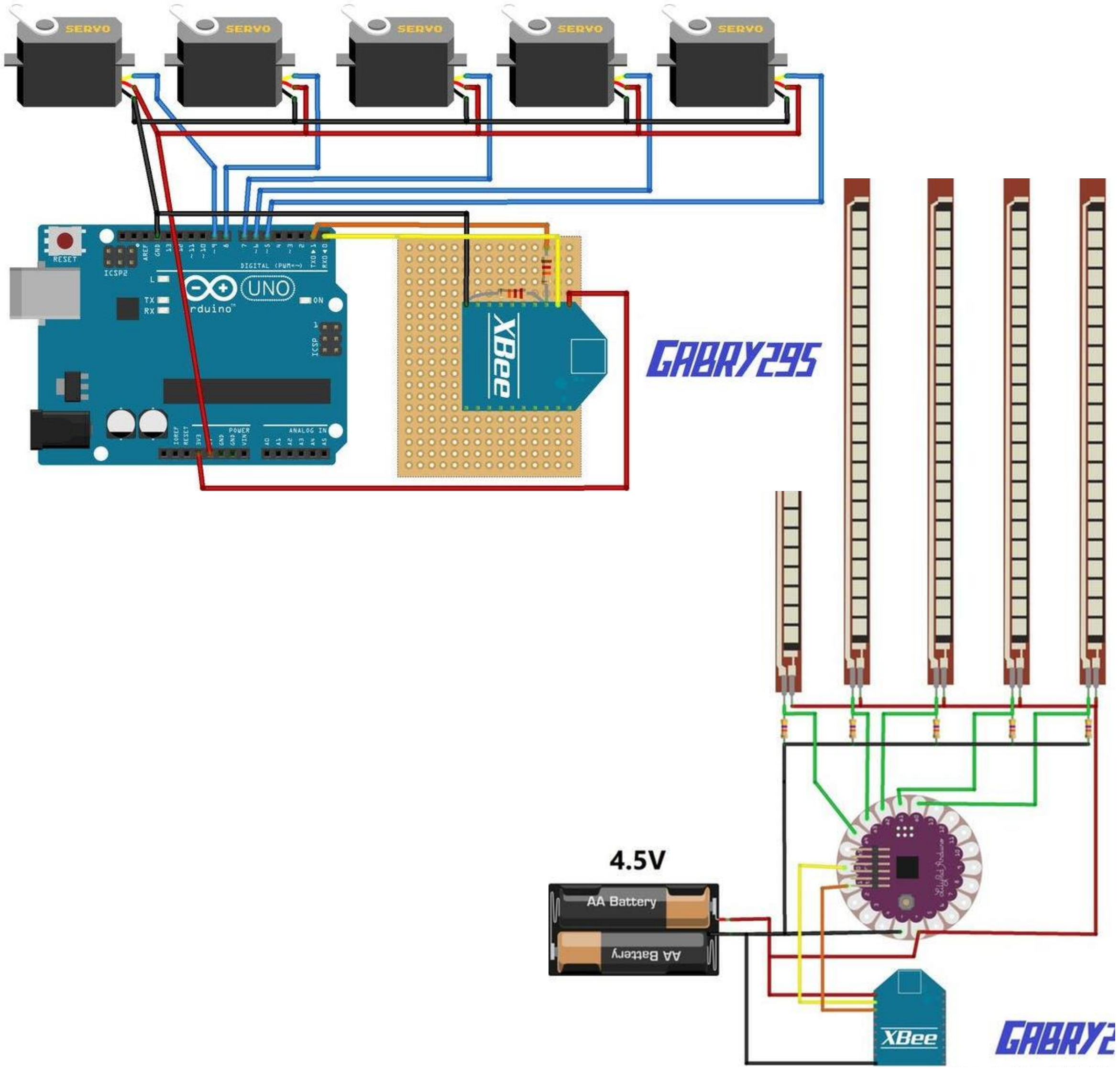
```
/*
 *
 * All the resources for this project: https://www.hackster.io/Aritro
 * Modified by Aritro Mukherjee
 *
 *
 */

#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance.

void setup()
{
  Serial.begin(9600);   // Initiate a serial communication
```

Arduino code for RFID reader

Projects using Arduino UNO



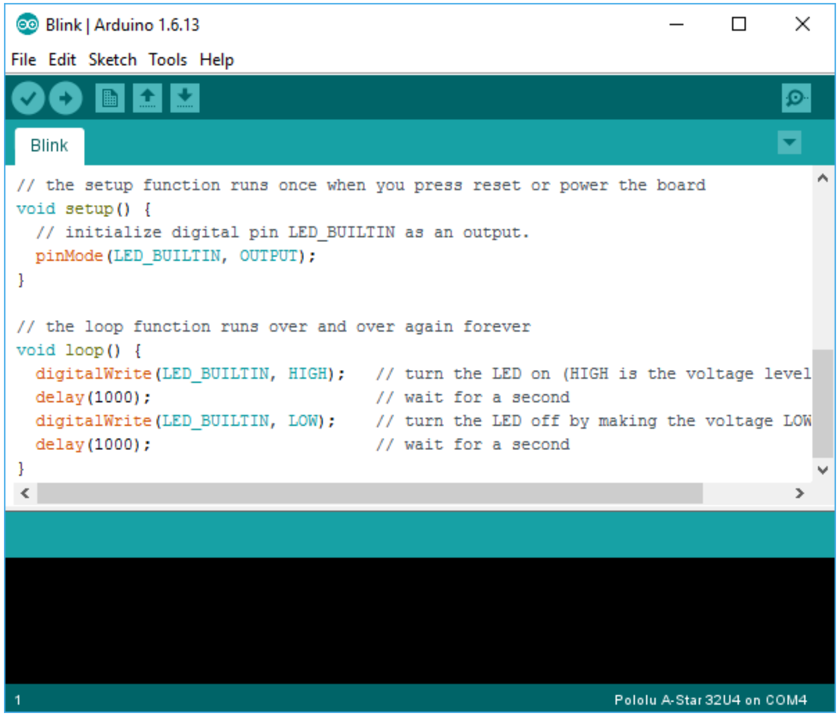
Remote Controlled Robotic Hand



Arduino Software(IDE)

❑ Arduino IDE(Integrated Development Environment)

- contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus.
- connects to the Arduino hardware to upload programs and communicate with them



```
// 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
}
```

Arduino Software(IDE) - Sketches

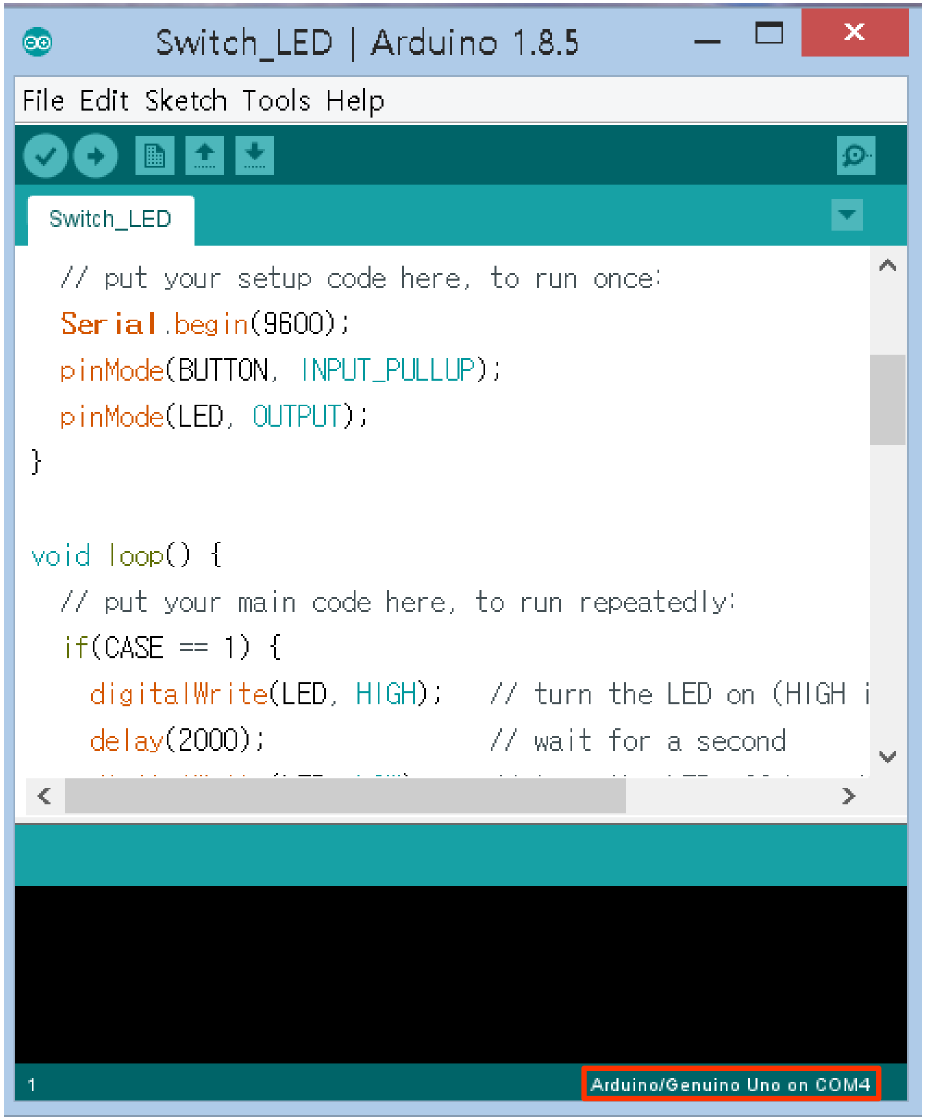
❑ Writing Sketches

- Programs written using Arduino Software (IDE) are called sketches
- Sketches is written in the text editor and are saved with the file extension .ino.
- The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.
  - ✓ Verify: Checks your code for errors compiling it.
  - → Upload: Compiles your code and uploads it to the configured board.
  - 🗋 New: Creates a new sketch
  - ↑ Open: Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.
  - ↓ Save: Saves your sketch
  - 🔍 Serial Monitor: Opens the serial monitor

Arduino Software(IDE) - Uploading

❑ Uploading

- Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus.
- Press the upload button or select the Upload item from the Sketch menu.
- Current Arduino boards will reset automatically and begin the upload
- The RX and TX LEDs blink as the sketch is uploaded.
- The IDE will display a message when the upload is complete, or show an error.



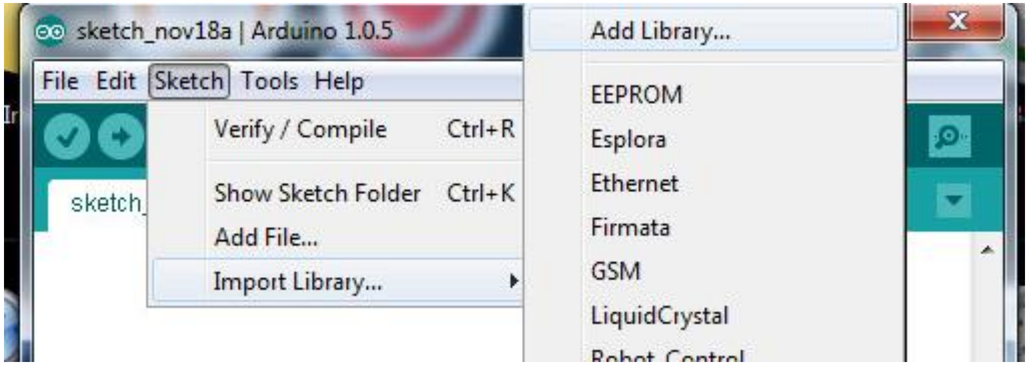
```
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(BUTTON, INPUT_PULLUP);
  pinMode(LED, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  if(CASE == 1) {
    digitalWrite(LED, HIGH);   // turn the LED on (HIGH i
    delay(2000);               // wait for a second
```

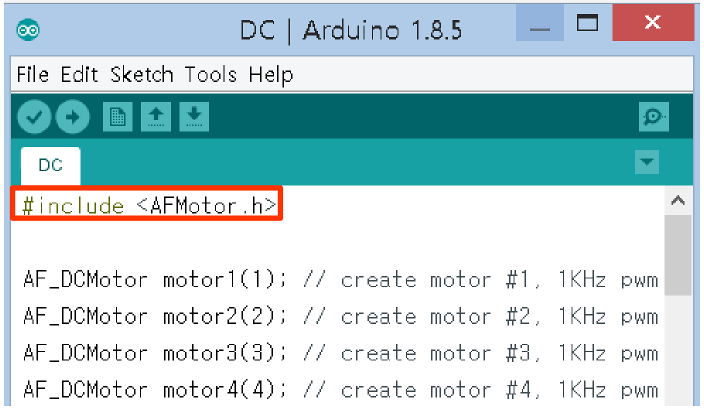
1 Arduino/Genuino Uno on COM4

Arduino Software(IDE) - Libraries

❑ Libraries

- Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data.
- To use a library, select it from the Sketch > Import Library menu
- This will insert one or more #include statements at the top of the sketch and compile the library with your sketch.
- Other libraries can be downloaded from a variety of sources or through the Library Manager



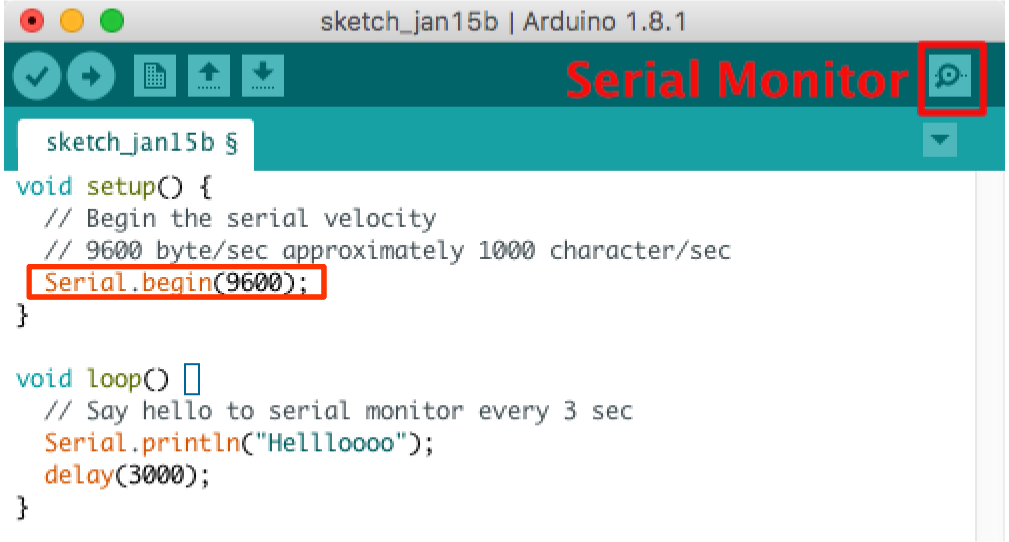


```
#include <AFMotor.h>


AF_DCMotor motor1(1); // create motor #1, 1KHz pwm
AF_DCMotor motor2(2); // create motor #2, 1KHz pwm
AF_DCMotor motor3(3); // create motor #3, 1KHz pwm
AF_DCMotor motor4(4); // create motor #4, 1KHz pwm
```

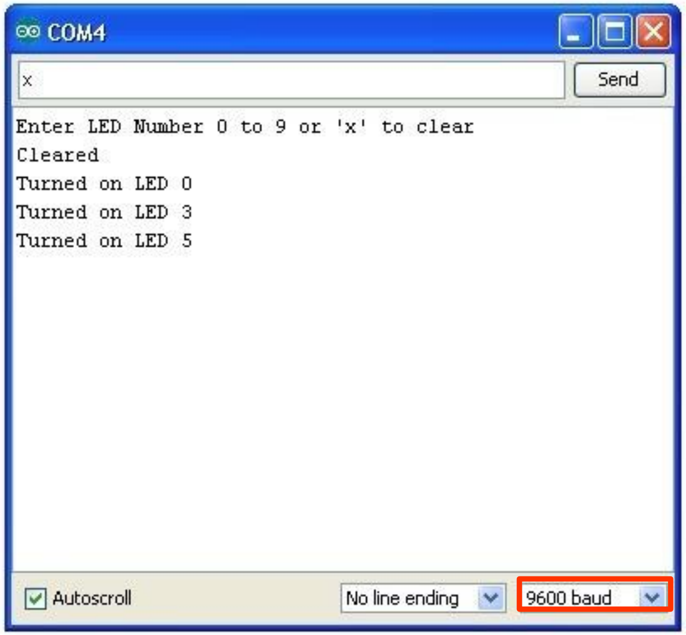
Arduino Software(IDE) - Serial

❑ Serial Monitor

- This displays serial sent from the Arduino board over USB or serial connector.
- To send data to the board, enter text and click on the "send" button or press enter.
- Choose the baud rate from the drop-down menu that matches the rate passed to Serial.begin in your sketch.



```
void setup() {
  // Begin the serial velocity
  // 9600 byte/sec approximately 1000 character/sec
  Serial.begin(9600);
}

void loop() {
  // Say hello to serial monitor every 3 sec
  Serial.println("Hellloooo");
  delay(3000);
}
```



```
Enter LED Number 0 to 9 or 'x' to clear
Cleared
Turned on LED 0
Turned on LED 3
Turned on LED 5
```

Arduino Software(IDE) - Structure

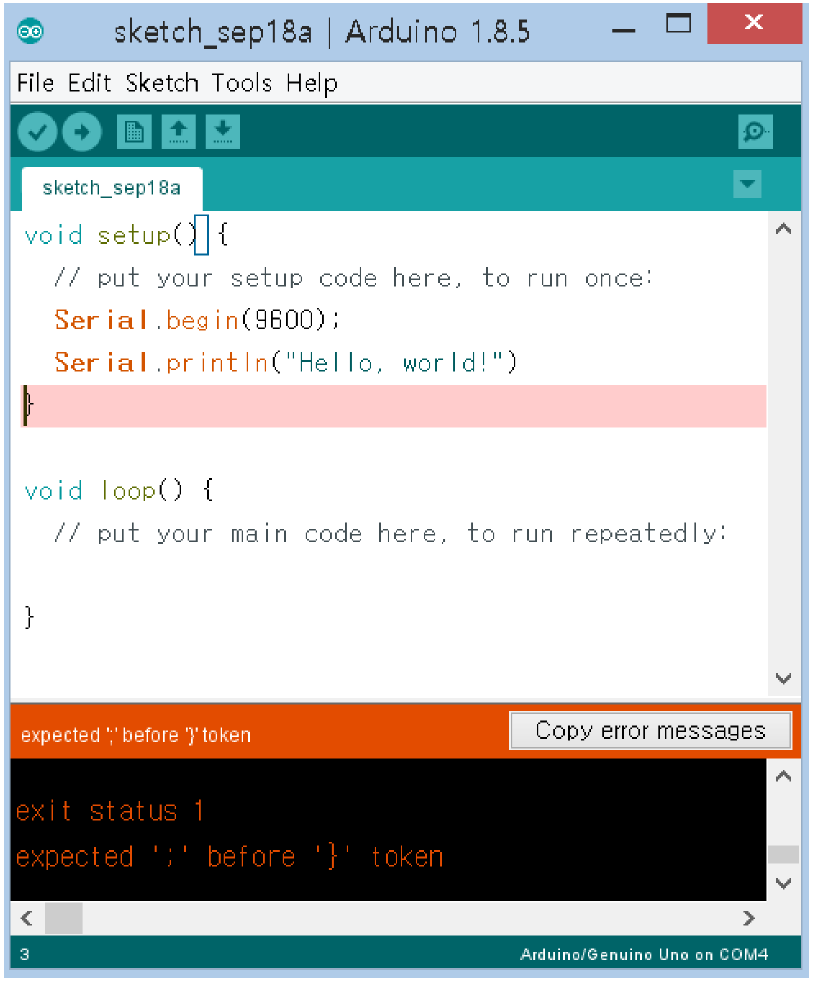
- the setup( ) function

  : Statements in the setup() function are run only once, every time that the sketch is run.

- the loop( ) function

  : Statements in the loop() function will run continuously from top to bottom and then back to the top

- Text Message by the Arduino IDE
  - The message area gives feedback while saving and exporting and also displays errors.
  - The console displays text output including complete error messages and other information


sketch_sep18a | Arduino 1.8.5
File Edit Sketch Tools Help
sketch_sep18a
Copy error messages
Arduino/Genuino Uno on COM4

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("Hello, world!")
}

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

}
```

expected ';' before '}' token

```
exit status 1
expected ';' before '}' token
```

3

Arduino Data Types

Type	Keyword	Width	Description
Boolean	bool	1bit	True(1) or False(0)
Character	char	1byte	a character value in the ASCII table
	unsigned char		datatype for numbers from 0 to 255.
Byte	byte	1byte	8-bit unsigned number, from 0 to 255
Integer	short	2byte	16-bit value, from -32768 to 32767
	int	2byte	16-bit value, from -32768 to 32767
	unsigned int	2byte	16-bit value, from 0 to 65535
Word	word	2byte	16-bit unsigned number, from 0 to 65535
Long	long	4byte	32 bits, from -2,147,483,648 to 2,147,483,647
	unsigned long	4byte	32 bits, from 0 to 4,294,967,295
Floating point	float	4byte	32 bits, -3.4028235E+38 ~ 3.4028235E+38
Double floating point	double	-	Arduino Uno: 4bytes, Arduino Due: 8bytes
String	string		Character array or an object of String class
Valueless	void		no information

Arduino Constants

Type	Description
true	defined as 1, Any integer which is non-zero is true
false	defined as 0
HIGH	- Reading: a voltage greater than 3.0V is present at the pin - Writing: the pin is at 5 volts
LOW	- Reading: a voltage less than 1.5V is present at the pin - Writing: the pin is at 0 volts
INPUT	high-impedance state for reading a sensor. To assure a proper reading when the switch is open, a pull-up or pull-down resistor must be used
OUTPUT	low-impedance state that can provide a substantial amount of current to other circuits
Integer Constants	Integer constants are numbers that are used directly in a sketch: decimal(7), binary(B1111011), octal(0173), hexadecimal(0x7B)
Floating Point Constants	floating point constants are used to make code more readable: 0.005, 10.0, 2.34E5(2.34*10^5 = 234000), 67e-10(67.0*10^-10= 0.0000000067)

Arduino Type Conversion

- Convert a value into another type
  - **char() :** Converts a value to the char data type
  - **byte() :** Converts a value to the byte data type
  - **int() :** Converts a value to the int data type
  - **word() :** Converts a value to the word data type
  - **long() :** Converts a value to the long data type
  - **float() :** Converts a value to the float data type

#1 задача: Arduino Data Handling

❑ Task(Задание)

: Define variables and Assign values and then Converts values to corresponding data types as below table.

Variable Name	Data Type	Value	Conversion Type
f_sw	boolean	false	byte
mark	character	‘A’	word
id	byte	0x1F	character
rank	unsigned integer	123	float point
average	floating point	3.141592	integer

❑ Use Tip(Подсказка)

- Refer to Arduino data types and constants

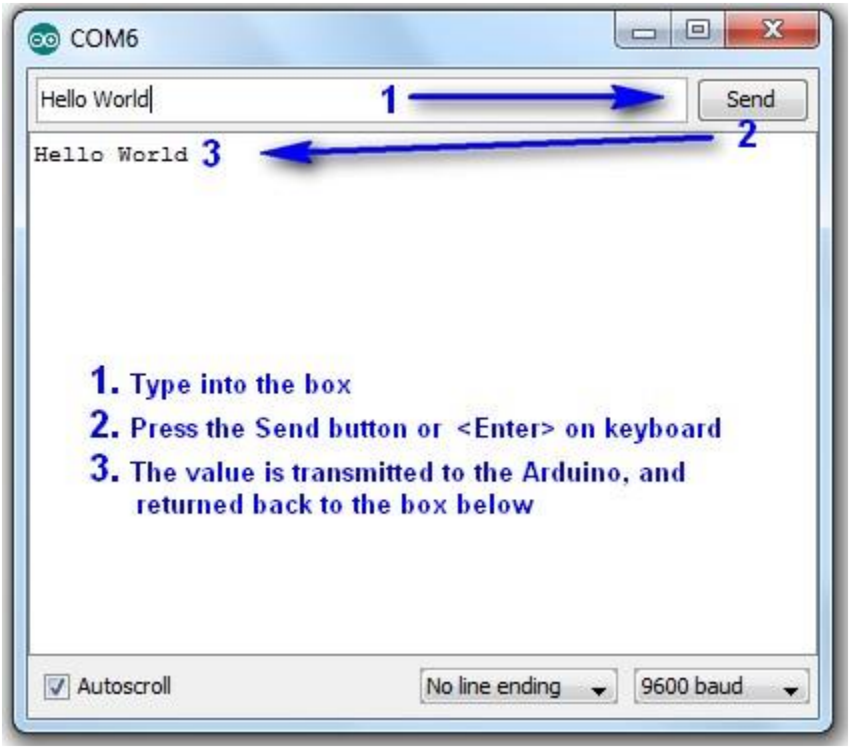
Contents

- ❑ Arduino

- ❑ Basic Control
  - Serial Input / Output
  - Hardware Control Basic

Serial in Arduino

- ❑ Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART)
- ❑ A user can enter data in the input field in the serial monitor window to send values and data to the Arduino



Hello World

```
Hello World
```

☑ Autoscroll

1. Type into the box
2. Press the Send button or <Enter> on keyboard
3. The value is transmitted to the Arduino, and returned back to the box below

Serial Functions

❑ Serial.begin()

: Sets the data rate in bits per second (baud) for serial data transmission

- Syntax
  - ➢ Serial.begin(speed)
    - speed: use one of these rates: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200
- Example

```
void setup() {
    Serial.begin(9600);               // opens serial port, sets data rate to 9600 bps
}

void loop() { .. }
```

❑ Serial.end()

: Disables serial communication, To re-enable, call Serial.begin()

- Syntax
  - ➢ Serial.end()

Serial Functions

❑ Serial.available()

: Get the number of bytes (characters) available for reading from the serial port.

- Syntax
  - ➢ Serial.available()
    - Returns: the number of bytes available to read
- Example

```
byte data = 0;                          // for incoming serial data

void setup() {
            Serial.begin(9600);          // opens serial port, sets data rate to 9600 bps
}

void loop() {
  if (Serial.available() > 0) {          // reply only when you receive data:
            …
   }
```

Serial Functions

❑ Serial.read(), Serial.readBytes()

: Reads serial data

- Syntax
  - ➢ Serial.read(): reads incoming serial data
  - ➢ Serial.readBytes(buffer, length): reads characters from the serial port into a buffer
    - buffer: the buffer to store the bytes in (char[] or byte[])
    - length: the number of bytes to read (int)
- Example

```
byte data = 0;                          // for incoming serial data

void loop() {
   if (Serial.available() > 0) {        // reply only when you receive data:
      data = Serial.read();             // read the incoming byte:
```

Serial Functions

❑ Serial.print(), Serial.println()

: Prints data to the serial port as human-readable ASCII text

- Syntax
  - ➢ Serial.print(val, format)
  - ➢ Serial.println(val, format): followed by a carriage return character ('/r') and a newline character ('\n')
    - val: the value to print - any data type
    - format: specifies the number base
- Example

```
Serial.print(78)                    // gives "78"
Serial.print(1.23456)               // gives "1.23"
Serial.print('N')                   // gives "N"
Serial.print("Hello world.")        // gives "Hello world.“

Serial.println(78, BIN)             // gives "1001110"
Serial.println(78, OCT)             // gives "116"
Serial.println(78, DEC)             // gives "78"
Serial.println(78, HEX)             // gives "4E"
```

Example Code

```
char data = 0;                          // for incoming serial data

void setup() {
          Serial.begin(9600);           // opens serial port, sets data rate to 9600 bps
}

void loop() {

          if (Serial.available() > 0) { // reply only when you receive data:

                    data = Serial.read();                 // read the incoming byte:

                    Serial.print("I received: ");         // say what you got:
                    Serial.println(data);
          }
}
```

#2 задача: Print Text

❑ Task(Задание)

1. Input a character in the serial monitor
2. Print corresponding test as below

Character	Text statement
‘F’	“Go Forward”
‘B’	“Go backward”
‘R’	“Turn Right”
‘L’	“Turn Left”

❑ Use Tip(Использование Совет)

- Set the data rate with Serial.begin()
- Get the number of bytes available for reading from the serial port with Serial.available()
- Read serial data by Serial.read() and prints data to the serial port with Serial.println()

#2-а задача: Print Max value

❑ Task(Задание)

1. Input new value in the serial monitor
2. Compare stored maximum value with the input value
3. Print maximum value

❑ Use Tip(Использование Совет)

- Set the data rate with Serial.begin()
- Get the number of bytes available for reading from the serial port with Serial.available()
- Read serial data by Serial.read() and prints data to the serial port with Serial.print() and Serial.println()

Contents

- ❑ Arduino

- ❑ Basic Control
  - Serial Input / Output
  - Hardware Control Basic

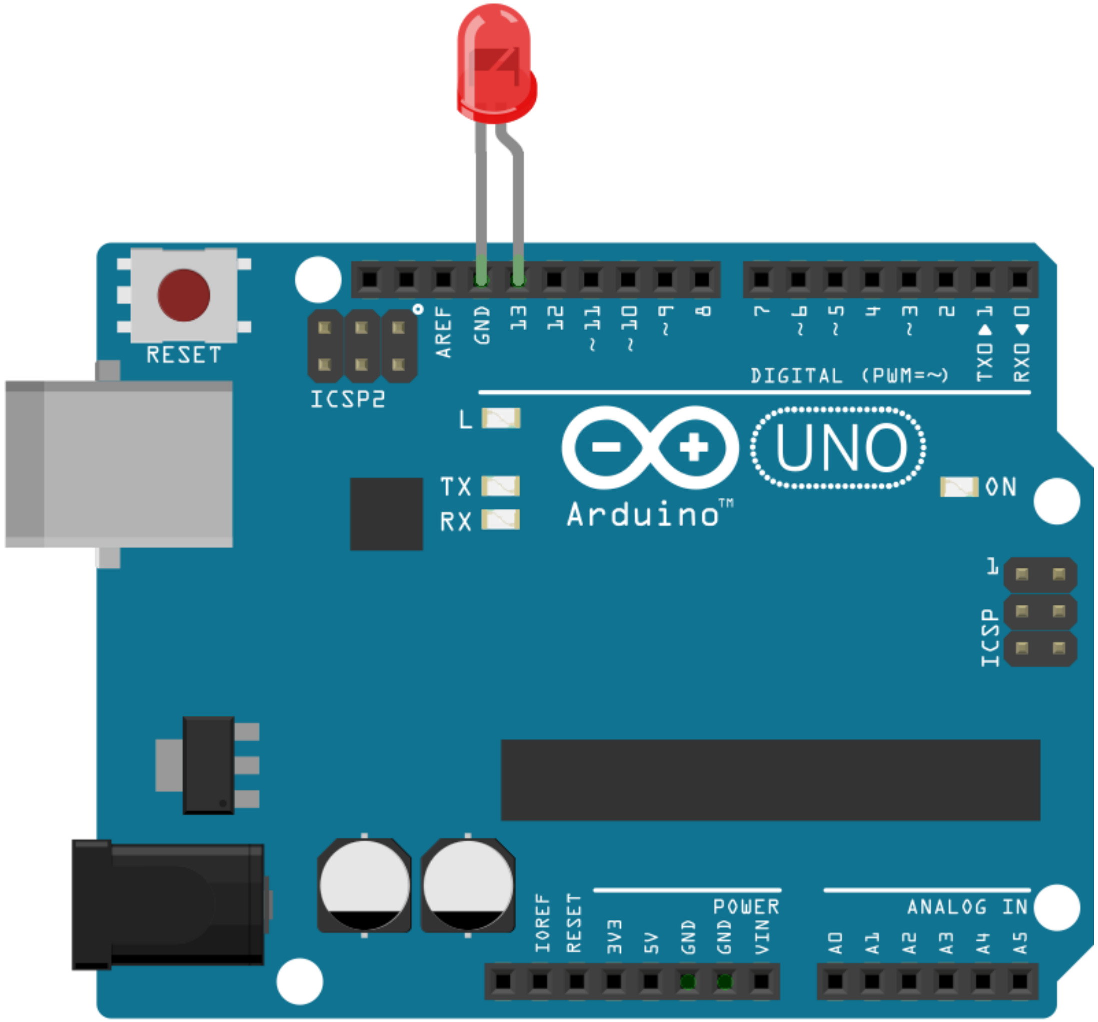
Control Digital I/O

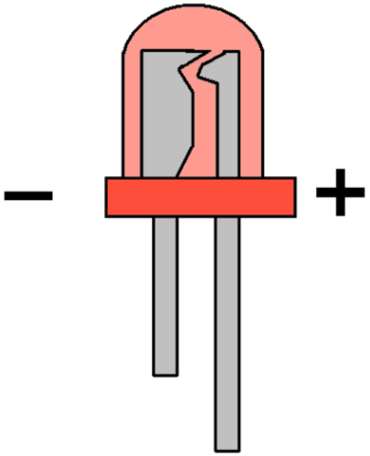
❑ pinMode()

- Configures the specified pin to behave either as an input or an output
- Syntax
  - pinMode(pin, mode)
    - pin: the number of the pin whose mode you wish to set
    - mode: INPUT, OUTPUT, or INPUT_PULLUP
- Example

```
void setup()
{
  pinMode(2, INPUT_PULLUP);                // sets the digital pin 2 as input pull-up

  pinMode(13, OUTPUT);                     // sets the digital pin 13 as output
}
```

Arduino Wiring: LED





Control Digital I/O

❑ digitalWrite()

- Write a HIGH or a LOW value to a digital pin
- Syntax
  - ➢ digitalWrite(pin, value)
    - pin: the pin number
    - mode: HIGH or LOW
- Example

```
void loop()
{
  digitalWrite(13, HIGH);          // sets the digital pin 13 on
  delay(1000);                     // waits for a second
  digitalWrite(13, LOW);           // sets the digital pin 13 off
  delay(1000);                     // waits for a second
}
```

#3 задача: LED Blinking

❑ Task(Задание)

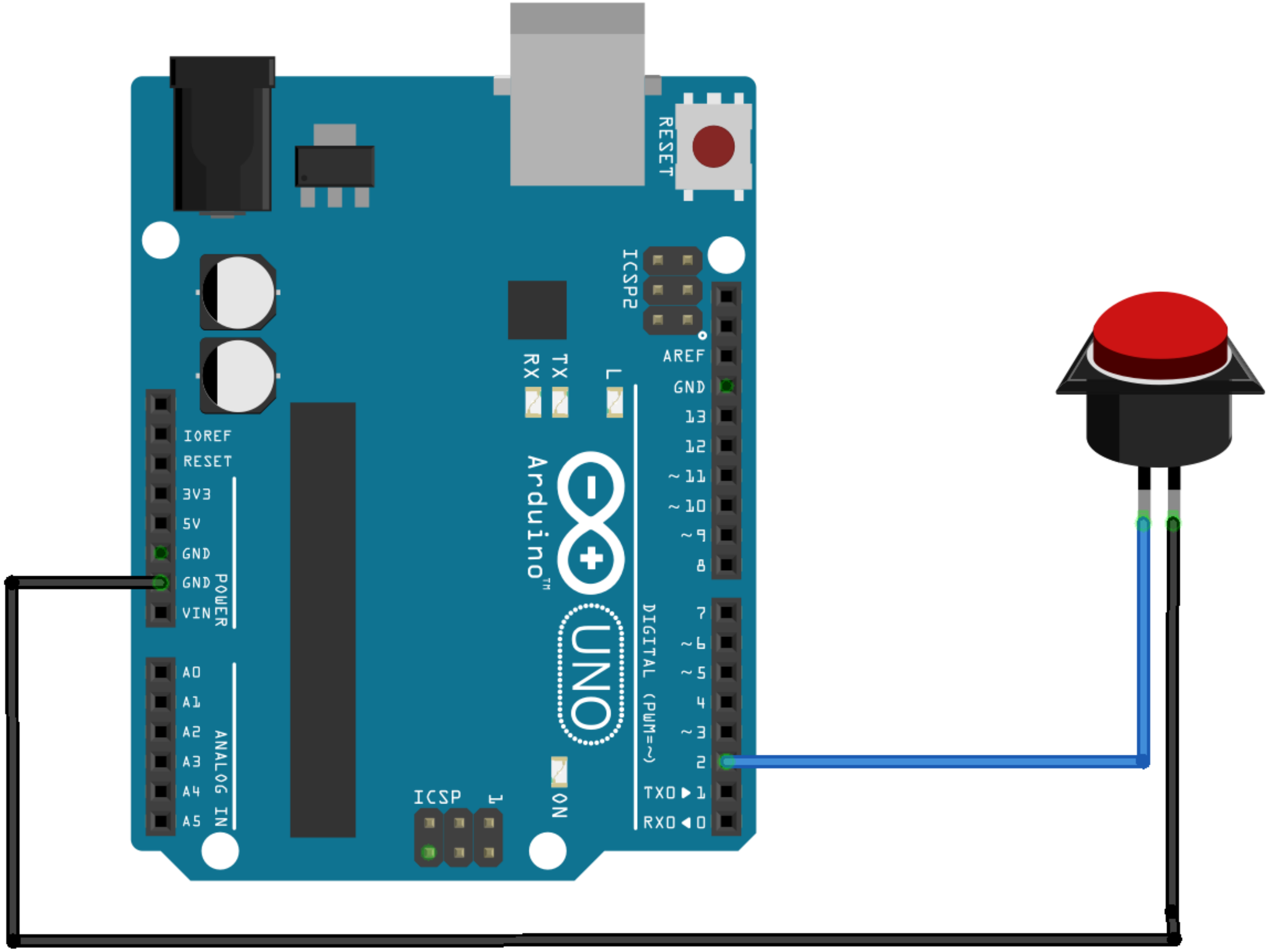
1. Input a value that indicates second (1: 1 second, 2: 2 seconds)
2. Make LED Blink with the input time interval

Second	LED Blinking
1	LED Blinks every 1 second
5	LED Blinks every 5 second

❑ Use Tip(Использование Совет)

- Set the data rate with Serial.begin()
- Get the number of bytes available for reading from the serial port with Serial.available() and Read serial data by Serial.read()
- Configure output pin using pinMode()
- Write HIGH or LOW value to 13 pin using digitalWrite()

Arduino Wiring: Push button



Control Digital I/O

❑ digitalRead()

- Reads the value from a specified digital pin, either HIGH or LOW
- Syntax
  - ➢ digitalRead(pin)
    - pin the number of the digital pin you want to read
    - Returns: HIGH or LOW
- Example

```
void loop()
{
  bool val;
  val = digitalRead(2);     // read the input pin(2)
}
```

#4 задача: Button & LED

❑ Task(Задание)

1. Read digital value from 2 pin connected with push button
2. Turn LED on when push button is pressed
3. Turn LED off when push button is released

❑ Use Tip(Использование Совет)

- Configure input/output pin using pinMode()
- Read digital value from 2 pin using digitalRead()
- Write HIGH or LOW value to 13 pin using digitalWrite()

#4-а задача: Button & LED(2)

❑ Task(Задание)

1. Read digital value from 2 pin connected with push button
2. Turn LED on when push button is pressed
3. Turn LED off when push button is pressed again

❑ Use Tip(Использование Совет)

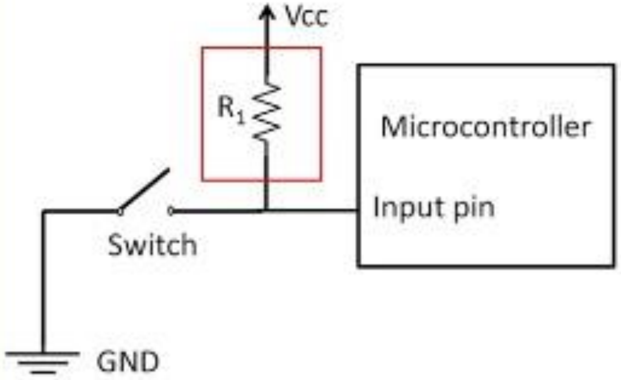
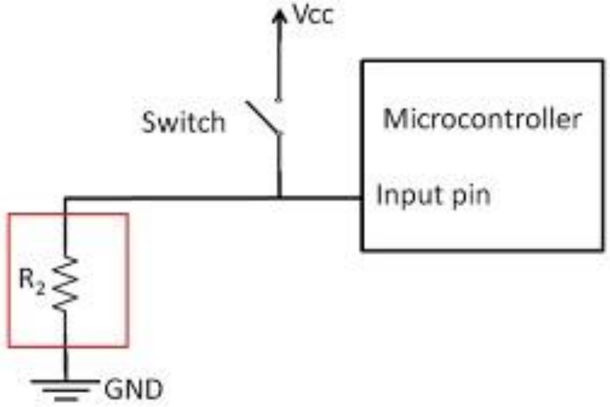
- Configure input/output pin using pinMode()
- Read digital value from 2 pin using digitalRead()
- Write HIGH or LOW value to 13 pin using digitalWrite()

#Appendix

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

#Appendix

	Pull-up Resistor Circuit	Pull-down Resistor Circuit
Circuit Arrangement	 Vcc R_1 Microcontroller Input pin Switch GND	 Vcc Switch Microcontroller Input pin R_2 GND
When **Switch is open**	**R_1** = Pull up resistor Current Path = **Vcc** → input pin ∴ Voltage at input pin = **Vcc** (High)	**R_2** = Pull down resistor Current Path = Input pin → **GND** ∴ Voltage at input pin = **GND** (Low)
When **Switch is closed**	Current Path = **Vcc** → input pin → **GND** ∴ Voltage at input pin = **GND** (Low)	Current Path = Vcc→ input pin → GND ∴ Voltage at input pin = **Vcc** (High)