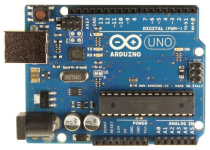


CoderDojo Toronto, Canada, 22 June 2014

Arduino and Robotics: Make what you imagine

Training by Artash Nath, Grade 2. Module written by Vikas Nath. Co-Founders, HotPopRobot.com

For free training on Arduino and Electronics, email us at: vikas.nath@gmail.com

I. What is Arduino?

Arduino is like a tiny but powerful computer which can be programmed to follow instructions. We can connect sensors, displays and motors to Arduino so that it can sense the world around and interact with it. Arduino can be compared to the human brain. Just as the human brain thinks and executes actions of the human body, Arduino too act as a controller and controls actions of different components attached to it.

Arduino are the Lego blocks of the digital age. We can make almost any project we can imagine using Arduino. These projects could be on robotics, space applications, education, games, health and more. Some examples of projects you can make using Arduino:

- Flashing lights
- A rover that can avoid obstacles or follow a line
- A motion sensor which detects movement
- An automated water sprinkler
- A robot which monitors data and sends it to base station over wireless

What do you want to make? Making things with Arduino is simple. By the end of this training you will have a better understanding of Arduino and be able to start working on simple projects of your own.

→ Check out our 90 second video on making a Rover – Curious Bot – using 3 Arduinos *at: https://www.youtube.com/watch?v=icAnX6dGjFY Curious Bot won the NASA SpaceApps 2014 Peoples' Choice Award, Toronto. It was among the top 5 NASA global finalists in the same category. Curious Bot was also reported in the "The Star" and the "Metro" newspapers.*

More Arduino and Electronic projects made by Artash are at www.HotPopRobot.com

---

2. Main components of Arduino

Arduino is an open-source device. This means you can make your own Arduino using simple and easily available electronic components. Arduino has 3 main parts:

- **Voltage Regulator:** Arduino operates at 5 volts. The voltage regulator reduces the input voltage, which can be between 7 volts and 12 volts, to 5 volts. We can power the Arduino using normal batteries or from the USB port of the computer.

- **Microcontroller:** This is an integrated circuit which can store data and carry out instructions. We can delete old data and upload new data on the microcontroller. This means we can make new projects using the same Arduino. The most commonly used microcontrollers are Atmel ATMEGA8 and ATMEGA168.

- **Crystal Oscillator:** This acts as the time keeper. For example, if we want a light to be switched on after 2 minutes, the crystal oscillator helps us do this. The frequency of the Arduino crystal is around 16 Mhz. This means it makes 16 million cycles per second and this is used for time keeping.

3. Different Types of Arduinos

There are different types of Arduinos which we can use. Just like computers, some Arduinos have faster speed and memory than others. Some have more connecting pins which allows us to connect more number of sensors, motors and displays to Arduino. Some of the popular Arduinos are:

- *Arduino UNO:* The most common version which can be used for a large number of projects. It has 14 digital pins and 6 analog pins.

- *Arduino Mega:* More powerful than Arduino UNO in terms of memory and pins. It has 54 digital pins and 16 analog pins. It is also more costly.

- *Arduino Nano / Mini:* Very small sized Arduino which are used when there is no space to use regular Arduino UNO. They are often used for making permanent projects that you do not want to break.

- *Arduino Lilypad:* Used for creating “Wearables”. We can stitch it on our clothes to make very interesting projects, for instance body temperature sensors and health monitors.

4. Arduino Software

The Arduino software allows us to write codes on computers and upload them to Arduino. The code we write to control Arduino is a variation of the C and C++ programming languages. When we write a programme for Arduino we are actually learning C. Isn’t that cool?

Arduino can only follow instructions which are written in the programme. It is important to practice writing simple programmes, so that when we start making robotic applications, we can fully control the robots.

The latest Arduino software can be downloaded from: www.Arduino.cc. See Annex for more details on installing the software and the driver. The Arduino software is available for installation on Windows, Mac and Linux.

Now that we have been introduced to Arduino hardware and software, let us learn about how to use Arduino for Robotics.

5. Introduction to Input and Output in Arduino

It is important to become familiar with "Input" and "Output" (I/O) pins in Arduino. They are fundamental to Arduino and all the projects we will make.

- **What is "Input" and "Output"?**

Example: When I hear a sound, I turn my head around.
Here "hearing" is input while "turning of head" is output.

Example: When I see a friend, I give him a high five.
Here "seeing" is input while "giving high five" is output.

Example: When I touch a hot thing, I quickly remove my hand.
Here "touch" is input while "removing my hand" is output.

"Input" means reading or sensing the data. "Output" means taking action based on the data which is read.

- **Configuring Arduino pins as Input pins or Output pins**

You will see that Arduinos have several pins or slots on them from where a wire can be connected. Each of these pins can be programmed to act as an Input pin or an Output pin. For instance, in Arduino UNO there are 14 digital pins and all of them can be programmed as either Input pins or Output pins.

When a pin is programmed as an "Input" pin it is ready to receive data, for instance from a sensor. When a pin is configured as an "Output" pin, that pin is ready to send out data, say to a motor or a speaker.

Example: In case of an obstacle-avoiding robot, the robot has to first detect an obstacle. It can do so by receiving data from an ultrasonic or infrared sensor (Input). It can then use this data to control the motors (Output) to avoid the obstacles.

The pin on which the sensor is attached has to be configured as an Input pin so that it can read the incoming data. The pin which controls the motor has to be configured as an output pin so that Arduino can switch on or switch off the motor.

Example: In case of a fire-fighting robot, the robot has to gather the temperature data (Input) using a sensor and then sound a buzzer (Output) depending on the temperature.

The pin on which we attach the temperature sensor has to be configured as an Input pin. The pin on which we attach the buzzer has to be configured as an Output pin.

INPUT AND OUTPUT COMMANDS

Each pin on Arduino is numbered. In Arduino UNO you will note that the digital pins are numbered 0 to 13.

To make a pin, say pin number 7, as an Input pin, we use the command:
pinMode (7, INPUT);

To make a pin, say pin number 12, as an Ouptut pin, we use the command:
pinMode (12, OUTPUT);

If we are connecting a temperature sensor to pin 7, we have to make pin 7 as an input pin. And if the buzzer is connected on pin 12, then we have to make pin 12 as an output pin.

We can connect multiple sensors and multiple output devices to the same Arduino.

CAUTION: Arduino provides a maximum of 40mA current per pin. Do not load the Arduino with any device which draws more than 40mA, for instance motors. To power motors (as we will do in this training) we use transistors. Never connect them directly to Arduino.

6. Introduction to Analog and Digital and HIGH and LOW

All Arduinos have digital pins and analog pins. For instance Arduino UNO has 13 digital pins and 6 analog pins. There are always more digital pins than analog pins as digital pins are used more often.

As most projects use digital pins, for the purpose of this training, we will limit ourselves to digital pins only.

Digital pins can have two states: ON or OFF. ON state is known as HIGH and OFF state is known as LOW. When a digital pin is configured as an output and set to HIGH, it means that +5V is available on that pin. And when the digital pin output is set to LOW, it means that 0V is available on that pin.

If we connect a bulb between digital pin output set to HIGH (+5V) and digital pin output set to LOW (0V), then the bulb will light up because of the potential difference.

When a digital pin is configured as an Input, it will report HIGH if a voltage of 3 volts or more is present at the pin and LOW if a voltage of 2 volts or less is present at the pin.

Analog pins are mostly used to read analog sensors. In Analog pins, the voltage is not limited to either 0V or 5V. It can be set to be anywhere in between. In Arduino UNO, analog pins are numbered as A0, A1, A2, A3, A4 and A5.

HIGH and LOW COMMANDS

Any digital pin on Arduino can be set as HIGH (+5V) or LOW (0V).

To set digital pin, say pin number 7 to HIGH, we use the command:
digitalWrite (7, HIGH); // +5V is available from pin 7

To set digital pin, say pin number 12 to LOW, we use the command:
digitalWrite (12, LOW); // 0V is available from pin 12 and acts as a current sink.

7. Main Arduino Commands

int (name) = pin number;
This allows us to name the pin instead of remembering the pin number.
Example, if we are attaching an LED to pin 12, then we can name pin 12 as led.
int led = 12;

pinMode(pin, mode)
Configures the specified pin to behave as an input or an output.
pinMode (7, INPUT); // pin 7 has been configured as an INPUT pin.
pinMode (led, OUTPUT); // pin named led has been configured as an OUTPUT pin.

digitalWrite(pin, value)
Writes the value to a specified pin. It can be either HIGH or LOW.

digitalRead (7, LOW); // pin 7 has been set to LOW.
digitalRead (led, HIGH); // pin named led has been set to HIGH.

digitalRead(pin, value)

Reads the value from a specified pin. It can be either HIGH or LOW.

if (digitalRead(sensor) == LOW) { // here the "if" condition is true when input on switch pin is LOW
digitalWrite (led, LOW); } // tells Arduino to switch off the LED

void setup()

Initial set of instructions given to Arduino. These are used only when the programme starts or is reset. Instructions given in setup() including initializing of variables, pin modes, using libraries and others.

void loop()

Instructions given under loop() continue to be executed endlessly as long as the conditions are being met. It allows the programme to change and respond to different conditions.

delay(milliseconds)

This executes a delay (in milliseconds) in carrying out the next instruction. 1000 milliseconds make 1 second.

delay(2000); // It will delay the execution of the next command by 2 seconds.

Comments

Comments are not programming instructions. They are written by the coder and help us in writing and understanding the code.

- Anything we write between /* and */ is ignored by the Arduino programme.
- Anything we write after // on the same line is also ignored by the programme.

For example:

/*
This is a Block comment.
It can be written in several lines and paragraphs.
It is ignored by the programme.
*/

// This is a line comment. It is ignored by the programme.
//Comments make it easier to write and understand codes.

EXAMPLES: Let us make some projects using Arduino

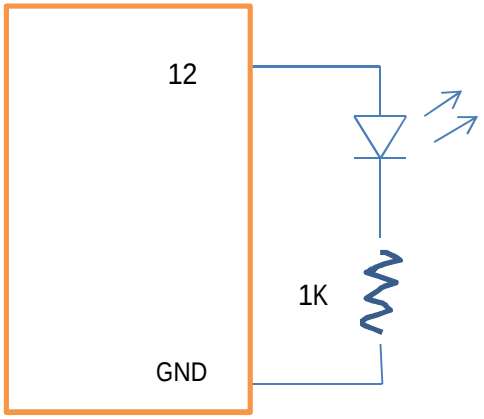
Example 1: Blinking Light

In this example, we will learn about the digital **output** and HIGH and LOW commands.

Components needed: Arduino, Light Emitting Diode (LED), Breadboard and Resistance (1K).

Using connecting wires connect the positive end of the LED (the longer terminal) to pin 12 of the Arduino. Connect the negative end of the LED (the shorter terminal) to the 1K resistance and connect the other end of the resistance to the Ground (GND) terminal of the Arduino.

Example 1: Circuit Diagram for the Blinking Light



Open the Arduino software and write the following code. *You do not need to write the comments as they are for our understanding and will be ignored by the programme.*

```
/*
  Blinking light example
  Turns on a Light Emitting Diode (LED) for one second and then off for one second, repeatedly.
 */
 int led = 12;                    // We have connected the LED to pin 12 and give it the name led

  void setup() {                  // the setup routine runs once at the start of programme
  pinMode (led, OUTPUT);      // initialize the led pin as an output
}

  void loop() {                   // the void loop routine runs over and over again
  digitalWrite(led, HIGH);       // turn the led on (HIGH means 5 volts)
  delay(1000);                    // wait for 1000 ms or 1second
  digitalWrite(led, LOW);        // turn the led off  (LOW means 0 volts)
  delay(1000);                    // wait for 1000 ms or 1second
}
```

Compile the code: Once you have completed writing your code, it is important to compile it and check for errors. Compilation is done by clicking the verify button. You may also go to **Sketch > Verify / Compile option.** If there is an issue with the code or the syntax then an error message will show. Else it will simply say, "Done Compiling" at the bottom of the window. If there are errors, they need to be corrected before we can proceed ahead.

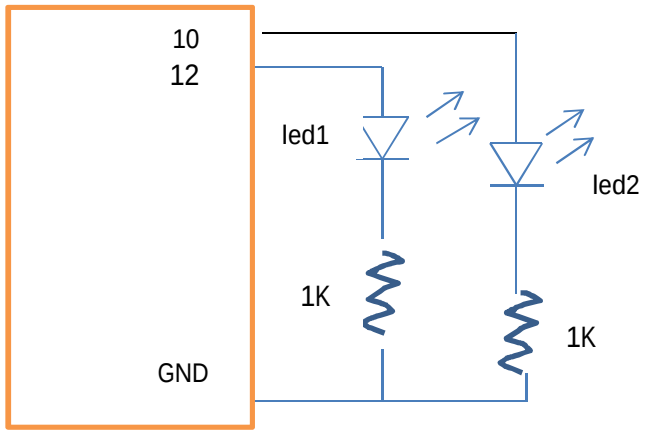
Save and Upload the code: When the sketch has been verified, save it. Under **Tools> Board** make sure the correct Board (example Arduino UNO is selected). Then go to **File >Upload** and transfer the code to the Arduino or simply click the upload button. *You will see the TX/RX LEDs on Arduino blinking during this time. This indicates that data is being transferred between the Arduino and the computer.*

Once the data has been uploaded, you will find that the LED has started to blink on and off. You can change the duration for which LED is on or off by changing the "delay".

Exercise 1: Can you connect 2 LEDs and programme them so that when one is on, the other is off?

Components needed: Arduino, Two Light Emitting Diode (LED), Breadboard and Two Resistances (1K).

Exercise1: Circuit Diagram for the Blinking 2 LEDs



The code is very simple and easy. The steps are as follows:

1. Give the second led a name, example led2, and initialize it, say on pin 10.
2. Make pin 10 an output pin using pinMode command.
3. In the void loop, we have to put the status of led2 as LOW (ie. Off) when led1 is HIGH. And status of led2 is HIGH (ie. On) when led1 is LOW.

```
/*
Complete code for blinking 2 lights
 */
 int led1 = 12;                  // We have connected led named led1 to pin 12
 int led2 = 10;                  // We have connected led named led2 to pin 10

 void setup() {                 // the setup routine runs once at the start of programme
  pinMode (led1, OUTPUT);     // initialize the led1 pin as output
  pinMode (led2, OUTPUT);     // initialize the led2 pin as output
}
void loop() {                   // the void loop routine runs over and over again
  digitalWrite(led1, HIGH);     // turn the led1 on
  digitalWrite(led2, LOW);      // turn the led2 off
  delay(1000);                  // wait for 1 second
  digitalWrite(led1, LOW);      // turn the led1 off
  digitalWrite(led2, HIGH);     // turn the led2 on
  delay(1000);                  // wait for 1 second
}
```

Compile the above code and upload it on Arduino and you will see one LED is on while the other is off, and they continue to blink. You can change the delays between on and off, or you can make different on and off patterns.

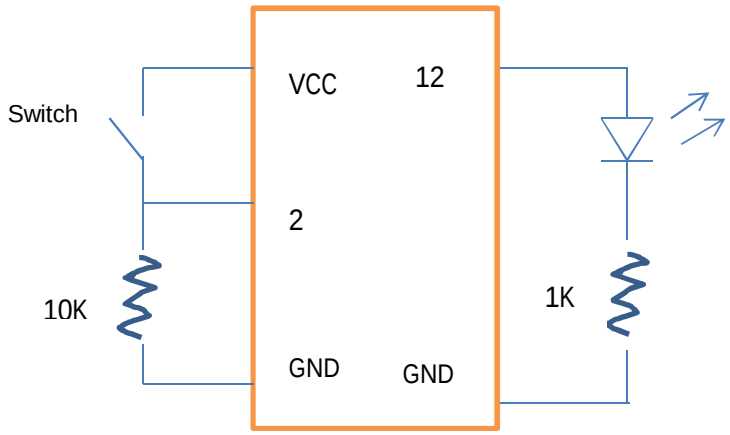
Example 2: Controlling LED using a Switch

In this example we will learn about digital **input** function, and "if" and "else" commands. We will make the LED on when the switch is on and turn the LED off when the switch is off.

Components needed: Arduino, Tactile Switch, Light Emitting Diode (LED), Breadboard, 1K Resistance and 10K resistance.

Using connecting wires connect the positive end of the LED (the longer terminal) to pin 12 of the Arduino. Connect the negative end of the LED (the shorter terminal) to the 1K resistance and connect the other end of the resistance to the Ground (GND) terminal of the Arduino. Take the tactile switch and insert it on the breadboard. Connect one end of switch to pin 2 and the other end to VCC (5V). Take a 10K resistance and connect it between pin 2 and ground (GND).

Example 2: Circuit Diagram for Controlling LED using a Switch



Open the Arduino window and write the code given below. Here we will use the "if" and "else" conditionality commands. **"if" tells Arduino what to do when a condition is met. "else" tells Arduino what to do when the "if" condition is <u>not</u> met.**

```
if (condition is met )
{ do this }
else { do that }
```

```
/*
Code for controlling LED using a Switch. When Switch is on, LED is on, and Switch is off, LED is off.
 */
int led = 12;      //initialize LED on pin 12
int sw = 2;        //initialize Switch on pin 2

void setup() {
pinMode (led, OUTPUT);     //set the led pin to Output mode
pinMode (sw, INPUT);         //set the switch pin to Input mode
}

void loop() {
if (digitalRead(sw) == LOW) {        // here the "if" condition is true when input on switch pin is LOW
digitalWrite (led, LOW);              //tells Arduino to switch off the LED
}
else {                                // here the "else" condition is true when input on switch pin is not LOW
digitalWrite(led, HIGH);              //tells Arduino to switch on the LED
 }
}
```

Compile the above code and upload it on Arduino. Now if you press the switch, the LED will glow and when you leave the switch, then the LED will turn off.

Exercise 2: Can you change the programme so that the LED is on when the switch is off, and the LED is off when the switch is on?

It is very simple and we only need to change the "If" and "Else" command in void loop(). The revised code will read as below:

```
/*
Code for controlling LED using a Switch. When Switch is off, LED is on and  when Switch is on, LED is off.
 */

int led = 12;      //initialize LED on pin 12
int sw = 2;        //initialize Switch on pin 2

void setup() {
pinMode (led, OUTPUT);     //set the led pin to Output mode
pinMode (sw, INPUT);          //set the switch pin to Input mode
}


void loop() {
if (digitalRead(sw) == LOW) {         // here the "if" condition is true when input on switch pin is LOW
digitalWrite (led, HIGH);               // tells Arduino to switch on the LED
}
else {                                  // here the "else" condition is true when input on switch pin is not LOW
digitalWrite(led, LOW);                 // tells Arduino to switch off the LED
 }
}
```

Example 3: How to run and control a motor

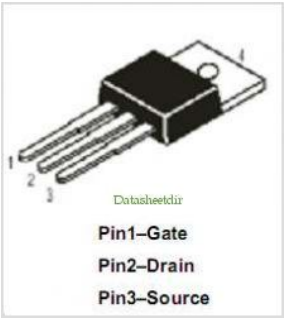
Controlling motors is fun and motors are an important part of robotic applications, whether they are rovers or androids.

Since Arduino do not produce enough current, we cannot connect motors directly to the Arduino pins. Instead we use semiconductor devices – transistors and diodes – to power them.

Arduino controls the transistor and the transistor in turn controls the motor. A transistor is an electronic switch. A small current to the base of the transistor can switch it on or off, and control bigger currents. A diode is a one-way gate. It allows current to flow in one direction. It prevents the back flow of current from motor to the Arduino.

In this example we will learn how to use motors with Arduino and control them. We will use a MOSFET transistor. A MOSFET is a special kind of transistor with a built in diode, and it will simplify our circuit.

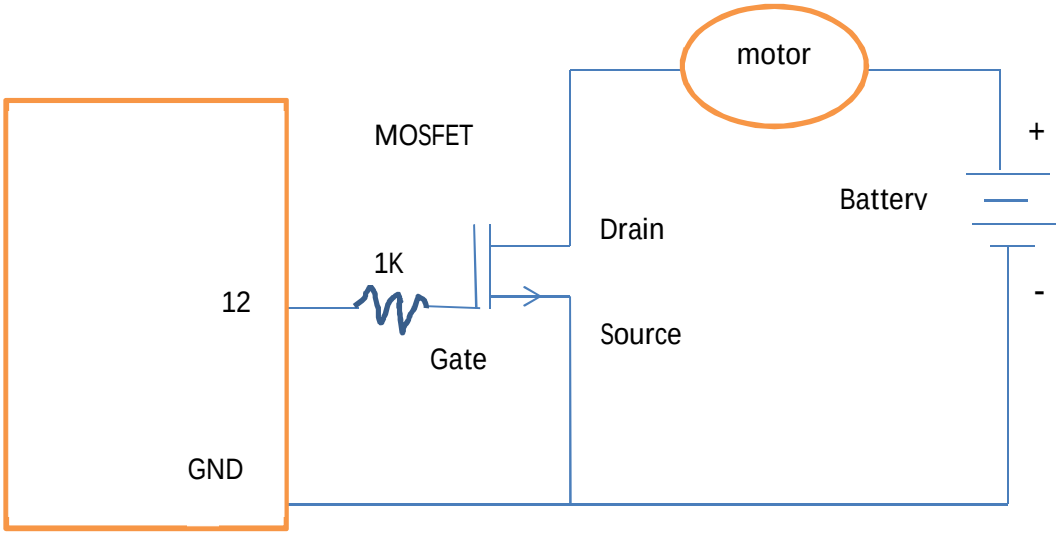
Components needed: Arduino, Breadboard, Motor, Battery Holder, Batteries, MOSFET and 1K Resistance.



Using connecting wires connect the gate pin of the MOSFET to pin 12 via a 1K resistance. Connect the Drain pin of the MOSFET to one end of the motor. The other end of the motor will connect to the positive end of the external battery pack. The negative end of the battery will connect to the Source pin of the MOSFET and also to the ground (GND) pin of the Arduino. And the circuit is complete.

You will note from the circuit that Motors have their own power supply and do not draw power from the Arduino.

Example 3: Using motors with Arduino and controlling them.



```
/*
In this example we will learn how to connect motors to the Arduino and then turn them on or off.
*/

int motor=12;              // Initialise the gate pin of the MOSTFET which controls the motor on pin 12.

void setup(){
  pinMode(motor,OUTPUT);     // Make the gate pin of the MOSFET an Output pin
}

void loop(){
  digitalWrite(motor,HIGH);      // Switch on the gate switch of MOSFET. This switches ON the Motor.
  delay(2000);
  digitalWrite(motor,LOW);      // Switch off the gate switch of MOSFET. This switches OFF the Motor.
  delay(2000);
}
```

Compile the above code and upload it on Arduino. And voila, the motor will turn on for 2 seconds and then off for 2 seconds. You can change the delay settings to have the motor go on and off for different time periods.

Exercise 3:

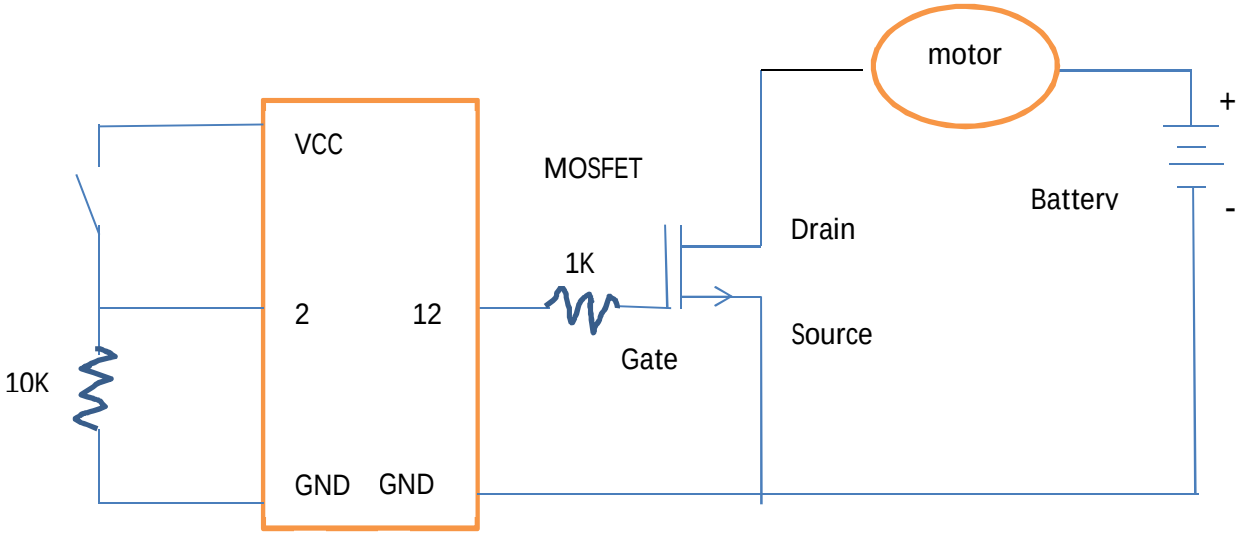
Can you now control the motor using a switch? Modify the programme and make the motor turn on when the switch is on, and off when the switch is turned off? *(Hint: Use what you learnt in example 2)*

Components needed: Arduino, Breadboard, Motor, Battery Holder, Batteries, MOSFET, Tactile Switch, 10K Resistance and 1K Resistance.

Take the tactile switch and insert it on the breadboard. Connect one end of switch to pin 2 and the other end to VCC (5V). Take a 10K resistance and connect it between pin 2 and ground (GND).

Now initialise switch (sw) on pin 2 and make it as an INPUT pin as we did in example 2. Modify the programme using “if” and “else” command. When the digital input on pin 2 is low (ie. when the switch is off), the current to the gate of the MOSFET (pin 12) is Off (LOW) causing the motor to stop. And when digital input on pin 2 is not low (ie when the switch is on), the current to the gate of the MOSFET is On (HIGH) and the motor starts to turn.

Exercise 3: Controlling motors using Input function / Switch




```
/*
In this example we will learn how to control motors using Input functions / switch.
*/
int motor=12;              // Initialise the gate pin of the MOSTFET which controls the motor on pin 12.
int sw = 2;                // Initialise the switch  on pin 2.

void setup(){
  pinMode(motor,OUTPUT);     // Make the gate pin of the MOSFET an Output pin
  pinMode(sw, INPUT);          // Make the pin connected to Switch an Input pin
}

void loop() {

if (digitalRead(sw) == LOW) {        // here the "if" condition is true when input on switch pin is LOW

digitalWrite (motor, LOW);              // tells Arduino to switch off the motor

}

else {                                  // here the "else" condition is true when input on switch pin is not LOW

digitalWrite(led, HIGH);               // tells Arduino to switch on the motor

 }

}
```

You have now mastered the basics of Arduino. You are now ready for the advanced session on Arduino and Sensors. Happy learning and making!

ANNEX

Installing Arduino Software and Drivers (for Windows)

Do not connect the Arduino to the computer before installing

1. Download the latest version of the software from the Arudino website www.arduino.cc (Select "Download").
2. Click "Windows Installer" and select "Run".
3. Allow programme to make changes to the computer.
4. Accept terms of license for Arduino.
5. Install Software, USB Driver, Short Cuts and Associate .INO files.
6. Install in folder c:\Program Files\
7. Arduino starts to extract files and install them. Say "Yes" to software installation.
8. Installation is complete.
9. Open Arduino from Start menu and the Arduino programming screen should open up.
10. **Connect your Arduino UNO to the computer.**
11. Go to Tools > Board and select Arduino UNO.
12. You are ready!

Thank You!

The HotPop Robot Factory is co-founded by Artash, Arushi, Rati and Vikas to bring discussions on science, space and technology in our everyday conversation. It encourages kids to make things, including low-cost robots and electronic projects - so that they become producers and not just consumers of technology.

For more information, write to Vikas Nath at vikas.nath@gmail.com

Website: www.HotPopRobot.com

Follow us on Twitter @wonrobot