International Journal of Research In Science & Engineering e-ISSN: 2394-8299
Volume: 2 Issue: 6- December 2016 p-ISSN: 2394-8280

A Study on Arduino Microcontroller & Its Applications

Neha Urkude[1], Mukta Gaur[2], B.S. Gujar[3]

urkude.neha@gmail.com[1],gaurmukta@gmail.com[2],bsgurjar@yahoo.co.in[3]

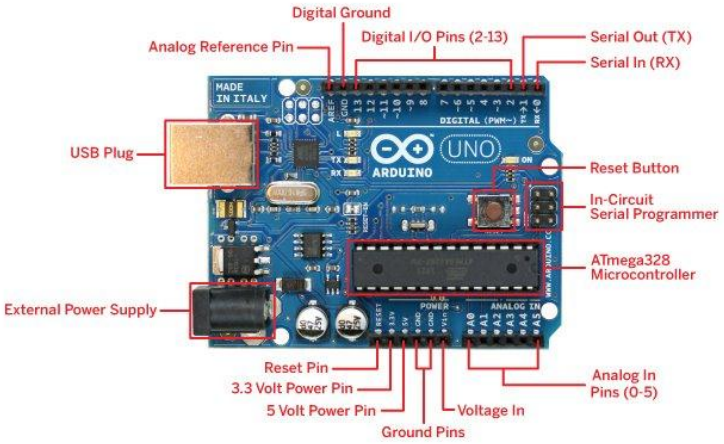
Dr. DY Patil Institute of Engineering Management and Research, Pune, Maharashtra,India

Abstract: - Arduino is an open-source electronics 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, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing. Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike. This paper will reflect the introductory part and different features of Arduino.

1. Introduction

The Arduino microcontroller is an easy to use yet powerful single board computer that has gained considerable traction in the hobby and professional market. The Arduino is open-source, which means hardware is reasonably priced and development software is free. This guide is for students in ME 2011, or students anywhere who are confronting the Arduino for the first time. For advanced Arduino users, prowl the web; there are lots of resources. The Arduino project was started in Italy to develop low cost hardware for interaction design. An overview is on the Wikipedia entry for Arduino. The Arduino hardware comes in several flavors. In the United States, Sparkfun is a good source for Arduino hardware.

This guide covers the Arduino Uno board (Sparkfun DEV-09950, $29.95), a good choice for students and educators. With the Arduino board, you can write programs and create interface circuits to read switches and other sensors, and to control motors and lights with very little effort. Many of the pictures and drawings in this guide were taken from the documentation on the Arduino site, the place to turn if you need more information. This is what the Arduino board looks like.



The Duemilanove board features an Atmel ATmega328 microcontroller operating at 5 V with 2 Kb of RAM, 32 Kb of flash memory for storing programs and 1 Kb of EEPROM for storing parameters. The clock speed is 16 MHz, which translates to about executing about 300,000 lines of C source code per second. The board has 14 digital I/O pins and 6 analog input pins. There is a USB connector for talking to the host computer and a DC power jack for connecting an external 6-20 V power source, for example a 9 V battery, when running a program while not connected to the host computer. Headers are provided for interfacing to the I/O pins using 22 g solid wire or header connectors.

The Arduino programming language is a simplified version of C/C++. If you know C, programming the Arduino will be familiar. If you do not know C, no need to worry as only a few commands are needed to perform useful functions. An important feature of the Arduino is that you can create a control program on the host PC, download it to the Arduino and it will run automatically. Remove the USB cable connection to the PC, and the program will still run from the top each time you push the reset button. Remove the battery and put the Arduino board in a closet for six months. When you reconnect the battery, the last program you stored will run. This means that you connect the board to the host PC to develop and debug your program, but once that is done, you no longer need the PC to run the program.

2. What You Need for a Working System

1. Arduino Duemilanove board
2. USB programming cable (A to B)
3. 9V battery or external power supply (for stand-alone operation)
4. Solderless breadboard for external circuits, and 22 g solid wire for connections
5. Host PC running the Arduino development environment. Versions exist for Windows, Mac and Linux

3. Installing the Software

Follow the instructions on the Getting Started section of the Arduino web site, http://arduino.cc/en/Guide/HomePage. Go all the way through the steps to where you see the pin 13 LED

blinking. This is the indication that you have all software and drivers successfully installed and can start exploring with your own programs.

4. Connecting a Battery

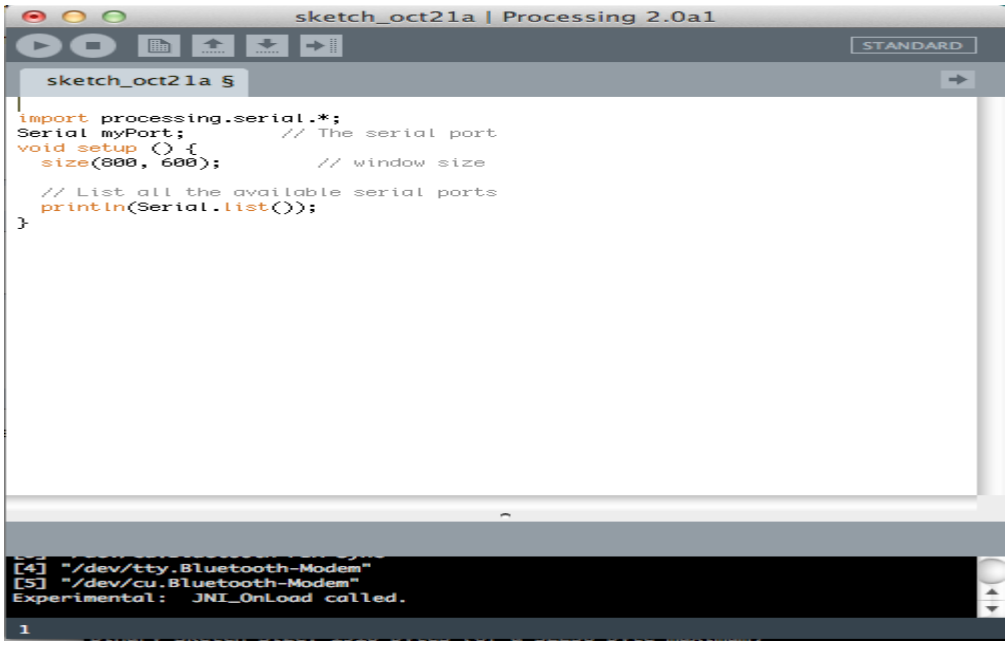
For stand-alone operation, the board is powered by a battery rather than through the USB connection to the computer. While the external power can be anywhere in the range of 6 to 24 V (for example, you could use a car battery), a standard 9 V battery is convenient. While you could jam the leads of a battery snap into the Vin and Gnd connections on the board, it is better to solder the battery snap leads to a DC power plug and connect to the power jack on the board. A suitable plug is part number 28760 from www.jameco.com. Here is what this looks like. Disconnect your Arduino from the computer. Connect a 9 V battery to the Arduino power jack using the battery snap adapter. Confirm that the blinking program runs. This shows that you can power the Arduino from a battery and that the program you download runs without needing a connection to the host PC.

5. Moving On

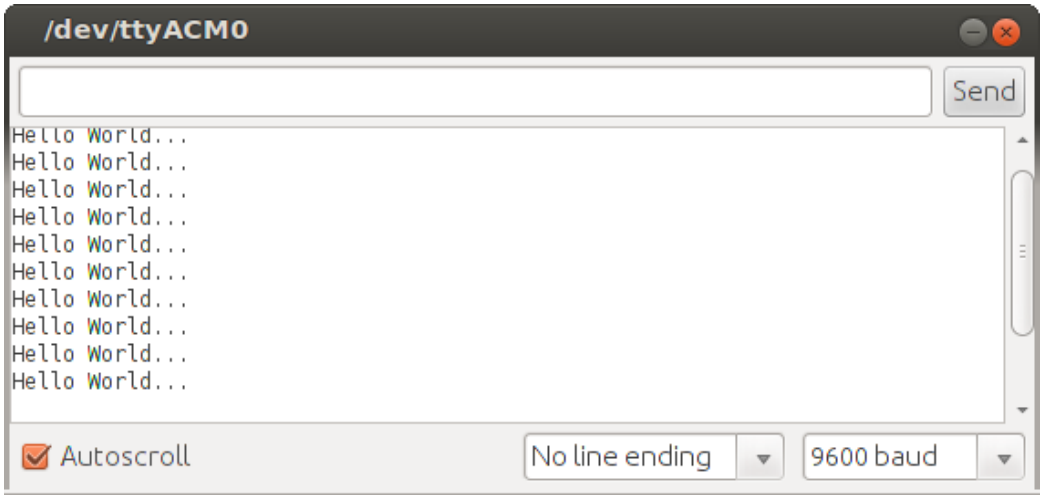
Connect your Arduino to the computer with the USB cable. You do not need the battery for now. The green PWR LED will light. If there was already a program burned into the Arduino, it will run. Start the Arduino development environment. In Arduino-speak, programs are called “sketches”, but here we will just call them programs. In the editing window that comes up, enter the following program, paying attention to where semi-colons appear at the end of command lines.

```
void setup()
 { Serial.begin(9600);
Serial.println("Hello World"); }
 void loop()
 {}
```

Your window will look something like this



Click the Upload button or Ctrl-U to compile the program and load on the Arduino board. Click the Serial Monitor button. If all has gone well, the monitor window will show your message and look something like this :-



6. Troubleshooting

If there is a syntax error in the program caused by a mistake in typing, an error message will appear in the bottom of the program window. Generally, staring at the error will reveal the problem. If you continue to have problems, try these ideas

- Run the Arduino program again
- Check that the USB cable is secure at both ends.

- Reboot your PC because sometimes the serial port can lock up
- If a "Serial port…already in use" error appears when uploading
- Ask a friend for help

7. Solderless Breadboards

A solderless breadboard is an essential tool for rapidly prototyping electronic circuits. Components and wire push into breadboard holes. Rows and columns of holes are internally connected to make connections easy. Wires run from the breadboard to the I/O pins on the Arduino board. Make connections using short lengths of 22 g solid wire stripped of insulation about 0.25" at each end. Here is a photo of a breadboard showing which runs are connected internally. The pairs of horizontal runs at the top and bottom are useful for running power and ground. Convention is to make the red colored run +5 V and the blue colored run Gnd. The power runs are sometimes called "power busses".

8. Reading a switch

The LED exercise shows how the Arduino can control the outside world. Many applications require reading the state of sensors, including switches. The figure to the right shows a picture of a pushbutton switch and its schematic symbol. Note that the symbol represents a switch whose contacts are normally open, but then are shorted when the button is pushed. If you have a switch, use the continuity (beeper) function of a digital multimeter (DMM) to understand when the leads are open and when they are connected as the button is pushed. For this exercise, the Arduino will read the state of a normally-open push button switch and display the results on the PC using the serial.println() command. You will need a switch, a 10 kohm resistor and some pieces of 22 g hookup wire. If you don't have a switch, substitute two wires and manually connect their free ends to simulate a switch closure. The figure below shows the schematic for the circuit on the left and a realization on the right.

For this exercise, the Arduino will read the state of a normally-open push button switch and display the results on the PC using the serial.println() command. You will need a switch, a 10 kohm resistor and some pieces of 22 g hookup wire. If you don't have a switch, substitute two wires and manually connect their free ends to simulate a switch closure. The figure below shows the schematic for the circuit on the left and a realization on the right.

Create and run this Arduino program

```
void setup()
{ Serial.begin(9600);
}
void loop()
{ Serial.println(digitalRead(3));
delay(250);
}
```

Open the Serial Monitor window. When the switch is open, you should see a train of 1's on the screen. When closed, the 1's change to 0's. On the hardware side, when the switch is open, no current flows through the resistor. When

no current flows through a resistor, there is no voltage drop across the resistor, which means the voltage on each side is the same. In your circuit, when the switch is open, pin 3 is at 5 volts which the computer reads as a 1 state. When the switch is closed, pin 3 is directly connected to ground, which is at 0 volts. The computer reads this as a 0 state.

```
void setup()
 { Serial.begin(9600);
}
void loop()
{ while (digitalRead(3) == HIGH) ;
Serial.println("Somebody closed the switch!");
while (digitalRead(3) == LOW) ;
Serial.println("The switch is now open!");
}
```

9. Arduino Hardware

The power of the Arduino is not its ability to crunch code, but rather its ability to interact with the outside world through its input-output (I/O) pins. The Arduino has 14 digital I/O pins labeled 0 to 13 that can be used to turn motors and lights on and off and read the state of switches. Each digital pin can sink or source about 40 mA of current. This is more than adequate for interfacing to most devices, but does mean that interface circuits are needed to control devices other than simple LED's. In other words, you cannot run a motor directly using the current available from an Arduino pin, but rather must have the pin drive an interface circuit that in turn drives the motor. A later section of this document shows how to interface to a small motor. To interact with the outside world, the program sets digital pins to a high or low value using C code instructions, which corresponds to +5 V or 0 V at the pin. The pin is connected to external interface electronics and then to the device being switched on and off.

10. Conclusion

Above content get reflected the ardunio introductory part and its features with proper explanation.