Basic Microcontrollers

Basic Tutorial for Arduino

By,
Manas Shrivastava
MNNIT, Allahabad

Contents

- Control systems - open loop and closed loop
- Structure of Arduino
- What can I do with Arduino?
- Open loop control system with Arduino
- Closed loop control system with Arduino
- Sensors
- Devices
- Coding
- Examples
- Troubleshooting

Control Systems

- Any system that controls a process based on inputs given to it is a control system.
- In our everyday life, we are surrounded by various control systems.
- For example, the maintenance of a definite temperature inside a refrigerator is achieved through a control system.
- A simple cam-follower mechanism is an example of a control system.
- Touch screen of phone functions through a control system.
- As we can see, there are infinite control systems all around us.
- Control systems are not just mechanical or electrical, even the facing of a sunflower towards the sun and the entire operation of the human body are also a few examples of controls systems.

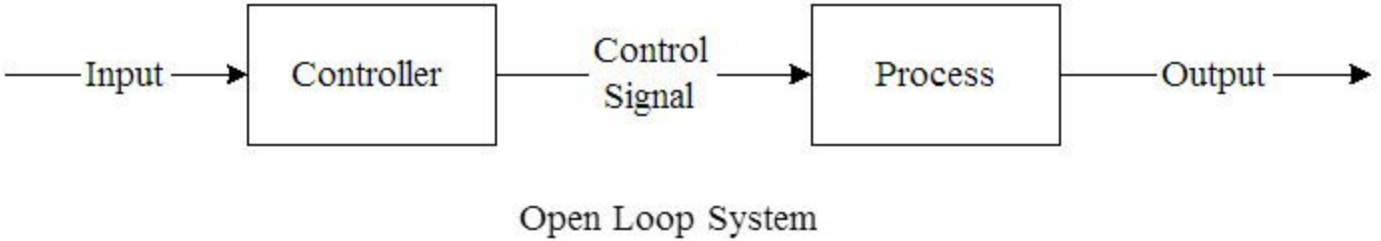
A broad classification of control systems

There are broadly two kinds of control systems - Open loop control and Closed loop control systems.

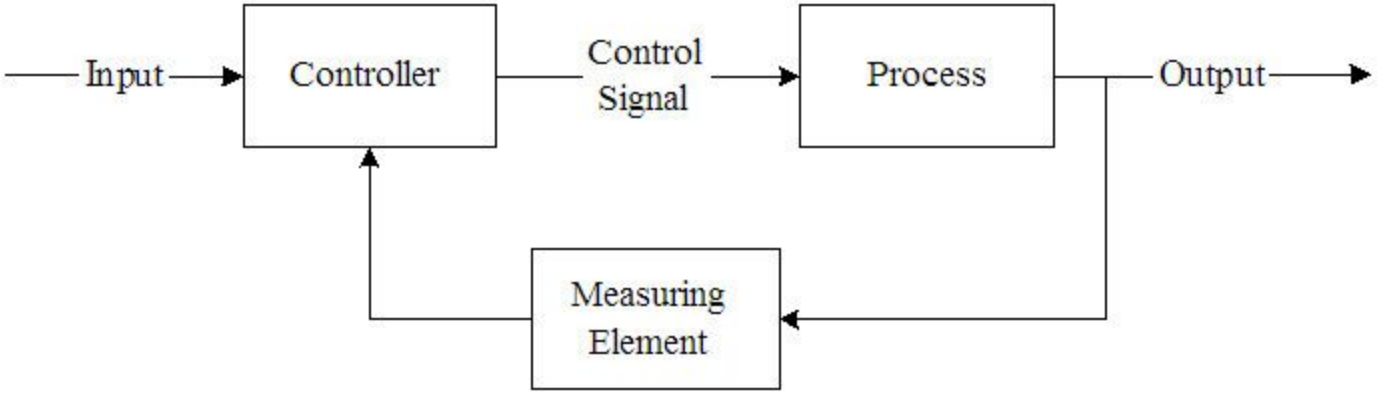
Open loop control systems - These systems do not take any feedback from the output conditions. We just give an input and get an output. Easy and cheap :)

Closed loop control systems - These systems take their job seriously and check whether the system is producing the desired output or not? This requires feedback from the output conditions and thus, increases the complications and cost of the system.

Depending upon our application, we choose open or closed control system.



Open Loop System



Closed Loop System

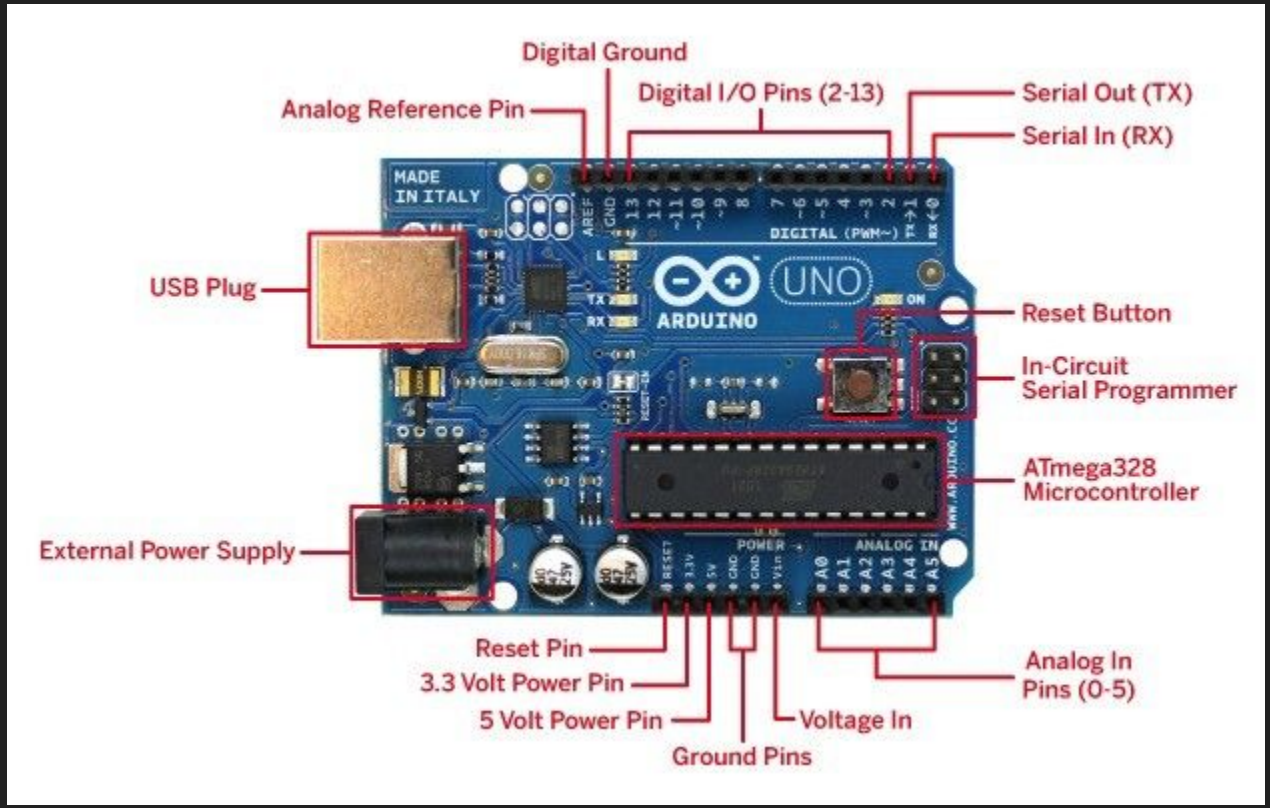
Examples of open loop control systems

- Cam-follower pair
- Induction Cooker
- Air conditioner in a car
- Remote of TV
- Manual Washing machine
- Regulator of fans
- Spring damper system
- Water taps
- Shells from cannons

Examples of closed loop control systems

- Valve opening in IC Engines by camshaft.
- Air conditioners/ Refrigerators
- Guided bombs and missiles
- Autopilot in aircrafts
- Adaptive brightness in smartphones
- Automatic doors in buildings
- Automatic electric iron
- Water level control through balloon in tanks

Structure of Arduino



What can we do with an Arduino?

- A wide range of operations is possible by making use of the Arduino IDE.
- Project examples-

1. Remote controlled Quadcopter drone
2. Theft detector
3. Home automation circuit
4. Automatic sweet dispenser
5. Sorting machines
6. Bomb defuser(and the opposite as well)
7. Spyware
8. Robots carrying out various functions
9. Robotic Arm
10. Health tracker

Open loop and closed loop control system with Arduino

Open loop - Arduino is the microcontroller, we give input through the code, and anything like a motor, LED, buzzer, can be used as an actuator to give motion, rotation, sound, light, etc as outputs.

Closed loop - We add sensors to the above mentioned system for feedback, and change the input accordingly. Sensors include resistive, inductive or capacitive transducers, ultrasonic distance sensors, IR sensors, gas sensors, etc.

Sensors

- For measuring distance:
  - Ultrasonic Distance sensors
    - Ultrasonic waves are triggered, and the time that they take to reflect back is measured to calculate distance of object.
    - Used for a range of 3cms to 4m.
    - Cheap and easily available.
  - Laser Distance sensors
    - Very accurate with high range and fast response time
    - Costly
- For differentiating between light and dark colours:
  - IR sensors: Emit Infrared rays, dark surfaces absorb them, light surfaces reflect them. A reverse biased diode receives the reflected rays and detects bright surfaces. These are cheap and can detect only dark or white surfaces.
  - Colour sensors: They can detect any colour depending upon the wavelength of light reflected back, they are costly and can detect any distinct colour.

- For measuring Rotation:
  - Potentiometers: They are cheap and easy to use.
  - Rotary Encoders: They are more precise, and more complex than rotary encoders
- For measuring presence of gas: Various gas sensors use chemicals whose conductivity varies with concentration of particular gas around them.
- For measuring magnetic field intensity: Hall sensors use Hall effect to detect magnetic field variation around them.
- Miscellaneous sensors and combinations:
  - Many other sensors are used according to applications like vibration sensors, flow sensors, temperature and humidity sensors, etc.
  - Combination of transducers can be used to make more applications like joystick, which uses a combination of two potentiometers to detect rotation in X and Y axis.

Devices

- Once the Microcontroller processes the input, we need actuators and devices to get an output.
- Just like sensors, we have many devices according to our applications. Some of them are:
  - Motors - DC, Servo, Stepper, etc
  - Linear actuators
  - LED’s, LCD Displays, OLED displays, LED Matrix, etc
  - Buzzers
  - Relays
  - Various Modules

Coding

- Arduino IDE’s coding is a mixture of C and Java.
- The basic structure of a code comprises of two parts: Setup and Loop
- Setup is the part of the code which just runs once when the Arduino is started, it defines which pins are we using and what do we want from the board to do in this code.
- Loop is where you put your main code to run it throughout the time period of the Arduino being switched on.

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

Input Output commands

- As we already know, there are two kinds of pins - digital and analog
- The input commands are:
  - digitalRead(pin_number);
  - analogRead(pin_number);
- The output commands are:
  - digitalWrite(pin_number, state); //State can be ‘HIGH’/1 or “LOW”/0
  - analogWrite(pin_number, value); //Value can be anything between 0 to 1023 (1024 values)
- // is used for writing single line comments, /* is used to write multi line comments */
- To define the mode of pins (Input or Output) in setup, pinMode(mode); is used. //mode = INPUT or OUTPUT
- To hold the looping code for some time at any instance, we use delay(milliseconds); //milliseconds of the delay that we want

Variables, control statements, loops, arrays, functions

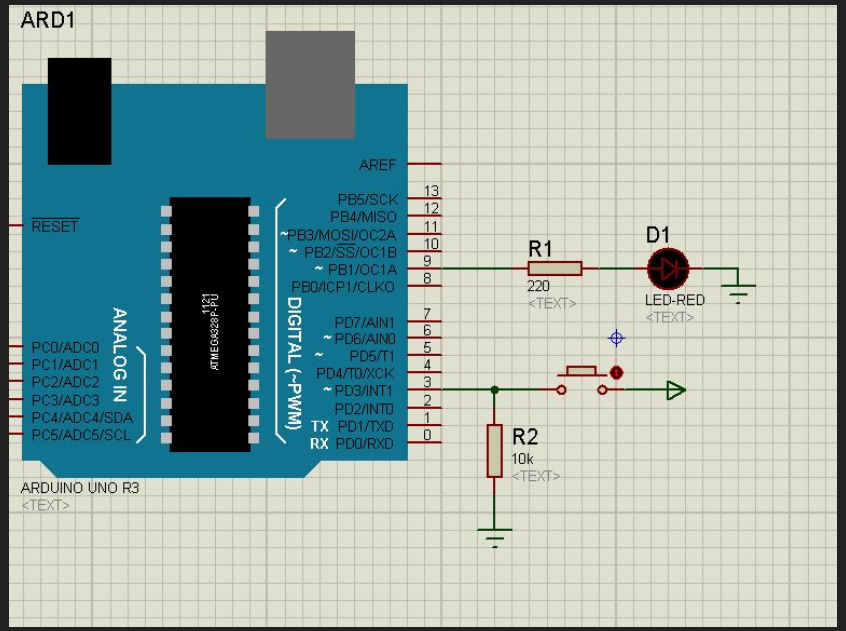
- The above mentioned commands are same as in C, the only difference is that instead of using “||”, we use “or”, and instead of “&&”, we use “and”.

We will be seeing some examples for more clarity now..

Example #1

A program to flash a LED connected to digital pin 9 when a button on digital pin 3 is pressed.

Connections:

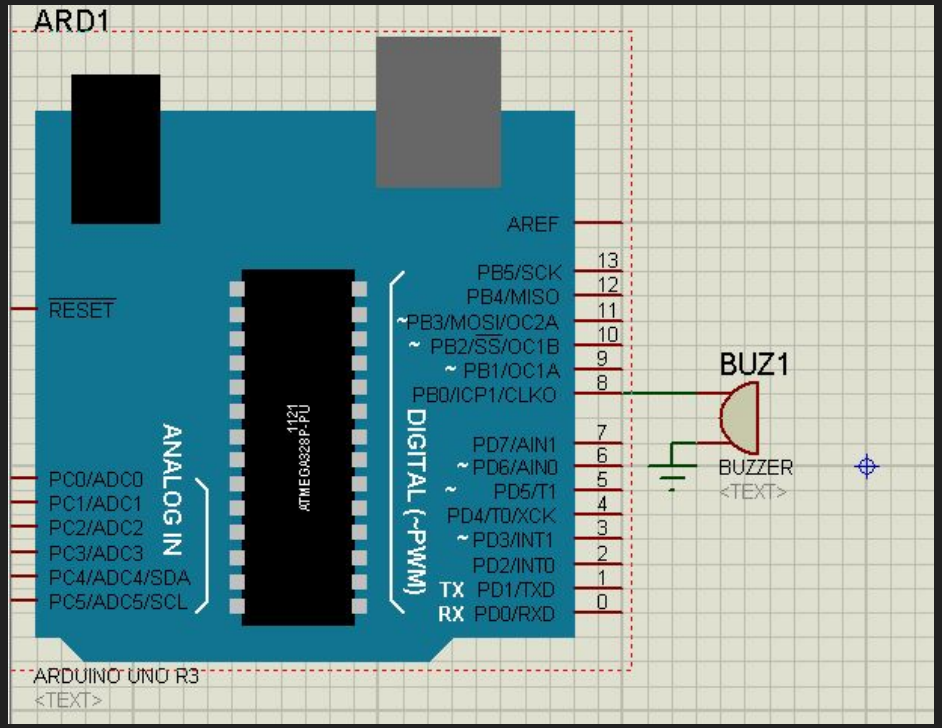


```
void setup()
{
  pinMode(9, OUTPUT); //LED
  pinMode(3, INPUT); //Push button
  digitalWrite(9, LOW); // Initially keep the LED off
}
void loop()
{
  if(digitalRead(3)==HIGH) //button pressed
  {
    digitalWrite(9, HIGH); //led on
  }
  else //button released
  {
    digitalWrite(9, LOW); //led off
  }
}
```

Example #2

A program to make a tune Beepbeep….beepbeep… through a buzzer at pin 8.

Connection:



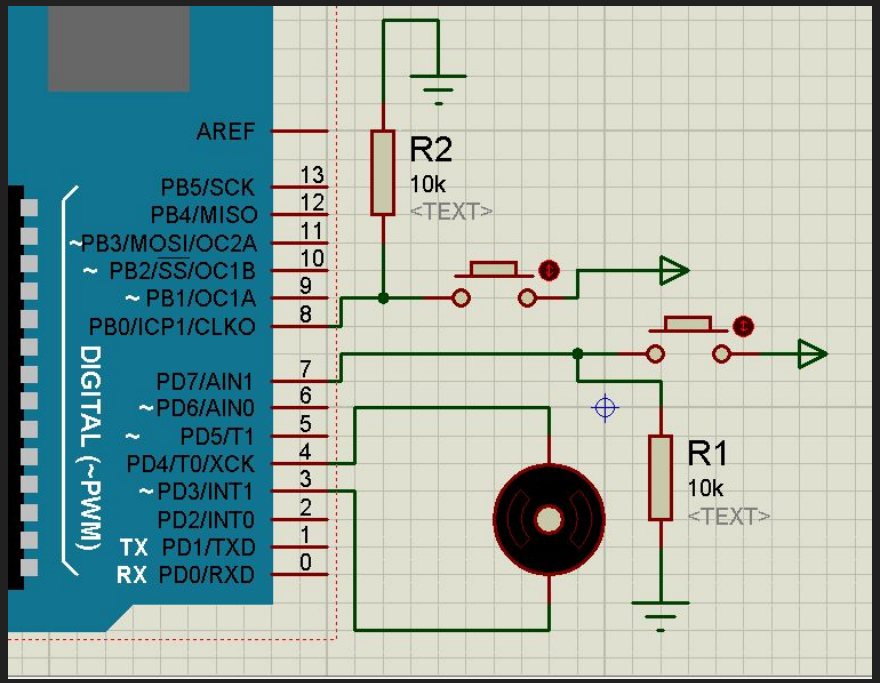
```
void setup()
{
  pinMode(8, OUTPUT);
  digitalWrite(8, LOW);
}

void loop()
{
  digitalWrite(8, HIGH);
  delay(200);
  digitalWrite(8, LOW);
  delay(50);
  digitalWrite(8, HIGH);
  delay(200);
  digitalWrite(8, LOW);
  delay(700);
}
```

Example #3

A program to rotate a motor connected to pin 3 and 4 in a certain direction when a button at pin 7 is pressed, and in opposite direction when a button at pin 8 is pressed.

Connection:



```
void setup()
{
  pinMode(3, OUTPUT); //motor
  pinMode(4, OUTPUT); //motor
  pinMode(7, INPUT); //button 1
  pinMode(8, INPUT); //button 2
}
```

```
void loop()
{
  while((digitalRead(7)==HIGH)and(digitalRead(8)==LOW))
  {
    digitalWrite(3, HIGH);
    digitalWrite(4, LOW);
  }
  while((digitalRead(7)==LOW)and(digitalRead(8)==HIGH)
  {
    digitalWrite(3, LOW);
    digitalWrite(4, HIGH);
  }
}
```

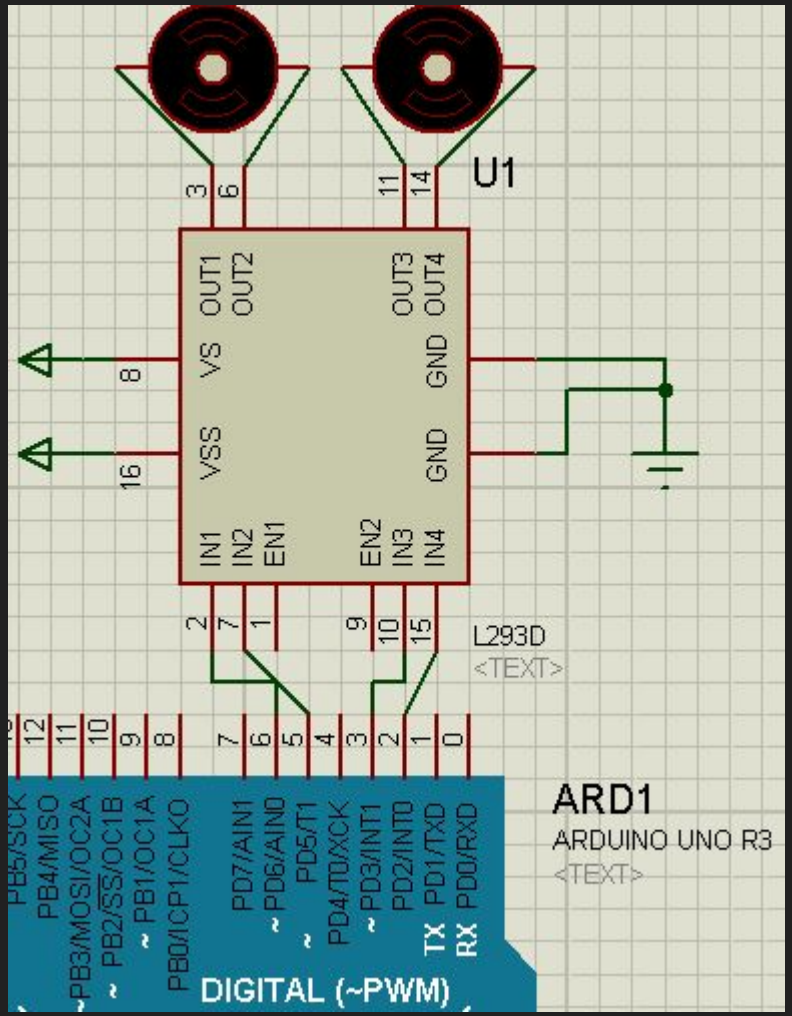
Example #4

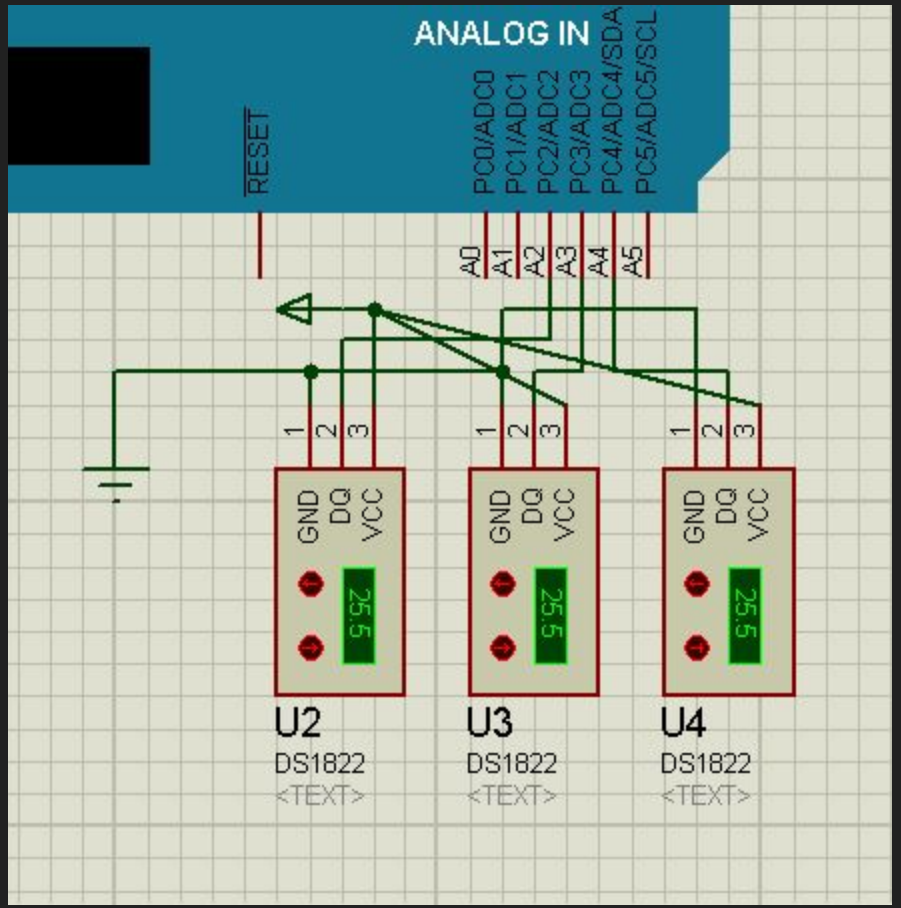
A program to make basic line follower robot using three infrared sensors and two motors.

Line follower robot:

On a white surface, there is a black line. Our robot will follow that line. We use three IR sensors here, if the surface under the IR sensor is black, it will return low analog values. When the left IR sensor show black line under it, we stop the left motor, and keep rotating the right motor, so that the robot takes a turn. For smoother motion, we can also slow down the left motor instead of stopping it using analog output through PWM pins. We perform the same action for right turn and thus, follow the black line.

- Since Arduino can not provide sufficient current for operation of motors, we use an IC called L293D, which is a motor driver IC.
- Motor driver IC’s use external power supply source to drive motors, and they take signal from Arduino to determine where and when to supply the voltage to the motor.
- The connections of L293D can be seen in the connection diagram shown in next slide...


U1
OUT1
OUT2
OUT3
OUT4
VS
VSS
GND
GND
IN1
IN2
EN1
EN2
IN3
IN4
L293D
<TEXT>
ARD1
ARDUINO UNO R3
<TEXT>
DIGITAL (~PWM)
PB5/SCK
PB4/MISO
~PB3/MOSI/OC2A
~PB2/SS/OC1B
~PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~PD6/AIN0
~PD5/T1
PD4/T0/XCK
~PD3/INT1
PD2/INT0
TX PD1/TXD
RX PD0/RXD


ANALOG IN
RESET
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
A0
A1
A2
A3
A4
A5
GND
DQ
VCC
25.5
U2
DS1822
<TEXT>
U3
DS1822
<TEXT>
U4
DS1822
<TEXT>

Troubleshooting

Here are the steps to perform troubleshooting in the system:

- Check for loose connections, if any.
- Check for proper shorting and voltage supply to each component.
- Check for connections you actually made and connections that you coded in Arduino.
- Use serial monitor to check code and sensor values:
  - Serial monitor is a special facility in Arduino where you can print anything that goes on with the sensors or the code on your laptop screen.
  - Use Serial.begin(9600); in setup function of the code to start Serial monitor at 9600 baud rate. //Baud rate is the rate of data transmitted per second, it depends on level of our application.
  - We will explore Serial Monitor in upcoming slides...

Serial Monitor

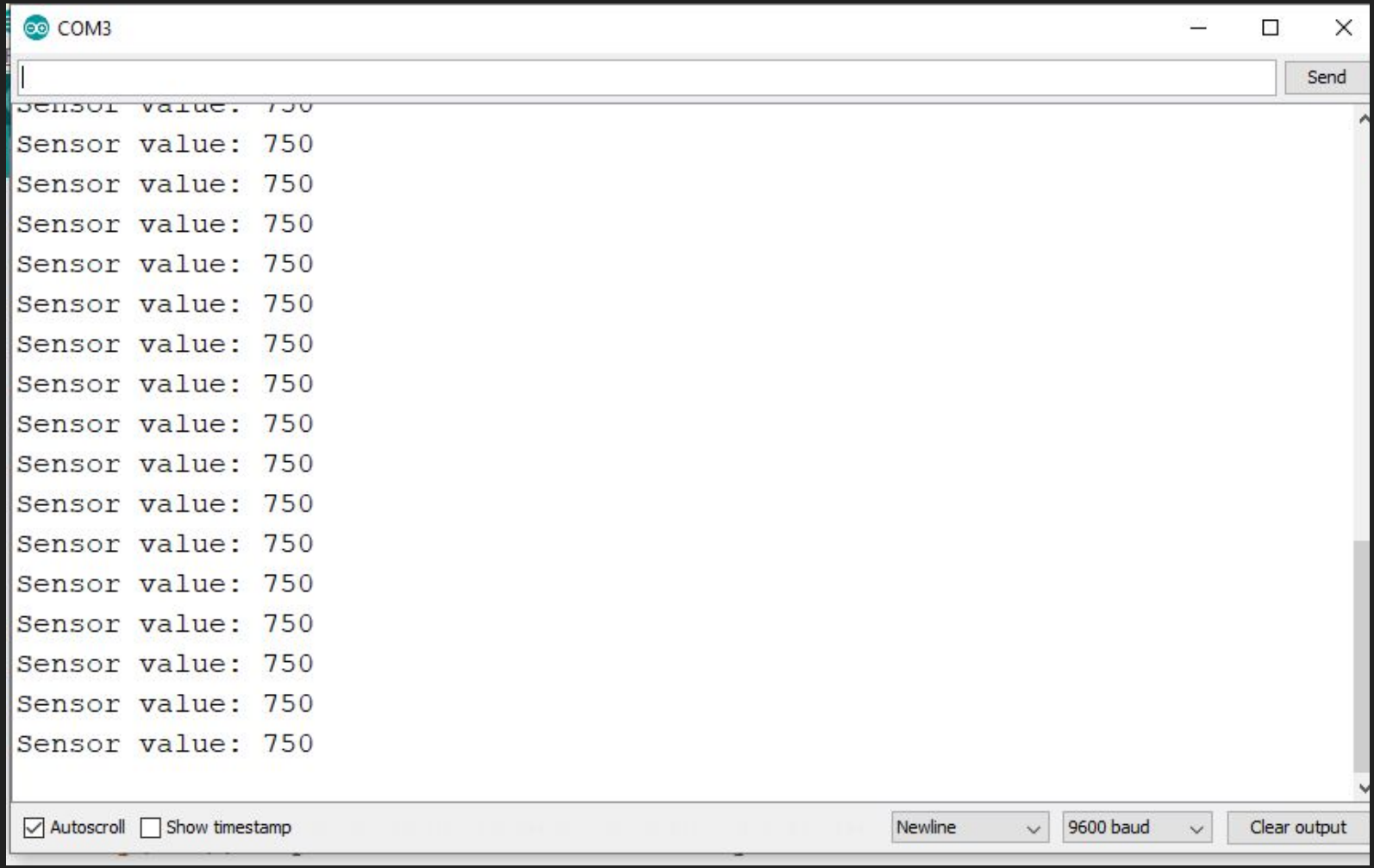
- Serial monitor can be accessed through tools menu of Arduino IDE or by pressing Ctrl+Shift+M.
- Serial.print(variable); command is used to print a variable, generally this is used to check sensor output values or to calibrate sensors.
- Serial.println(variable); prints the value and moves to the next line.
- Serial.print(“Hey!”); can be used to print strings whenever required.
- The upcoming examples show how to use serial monitor...

Serial monitor example #1

- Suppose you are using analog input pins to get values from an IR sensor to detect black and white lines.
- In a particular lighting condition, the IR sensor returns values below “500” for black surface, and above “500” for white.
- So we code that if(analogRead(3) <= 500){ line = 0; /*0 for black*/ }.
- Now the lighting conditions change, and this time “700” below means black.
- So we need to calibrate the sensor and replace the 500 with 700.
- Now, how do we know about these values that the sensor is returning. This 500, 700, how do we determine these values?
- This is where the Serial monitor comes up!

```
/*Keep the sensor above black surface,
 *Check value of sensor at black and then white
 *Callibrate according to values.
 */
void setup()
{
  Serial.begin(9600);
  //no need to setup analog pins
}

void loop()
{
  int a = 0; //declare variable a
  a = analogRead(A3); //save value of sensor output in a
  Serial.print("Sensor value: ");
  Serial.println(a); //print value of a
  delay(100); //pause code for readability
}
```


COM3
Send
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Sensor value: 750
Autoscroll
Show timestamp
Newline
9600 baud
Clear output

Serial monitor example #2

Suppose you write a conditional code (using if else) in loop function and it doesn’t work! :(

What could have happened! There’s no response from the code, so it might be that the conditional command is false, and hence the code is not going further.

To check the state of the conditional command, we can use Serial monitor here!

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  if(analogRead(A2)>10)
  {
    Serial.println("A2 is > 10");
    if(analogRead(A4)>30)
    {
      Serial.println("A4 is > 30");
      digitalWrite(5, HIGH);
    }
  }
}
```

- Now, we knew that the code wasn’t working since pin 5 was at LOW state. So, to check which if condition if not working, we used the above mentioned code.
- If serial monitor shows that only component at A2 satisfies the condition, we will know that component at A4 is faulty, or if serial monitor doesn’t show anything, then component at A2 may be faulty.
- In this way, the Serial monitor is used to reach the point of error very accurately and eliminates the need of checking each and every component or each and every part of the code.

Back to troubleshooting

- After we are done checking each and every part of the code with Serial monitor, we will be able to find in which part the code goes wrong.
- After debugging the part, we check the functioning of the functions again, till it works perfectly.

Conclusion

- After this basic part of the Arduino tutorial, we have reached the stage from where we can control motors, buzzers, and many more devices using buttons, sensors, or any input devices that we wish to use with Arduino.
- Next up: Using EEPROM memory of Arduino to store data even when it is not working, using advanced Modules like the GPS, GSM, Wifi, etc; Using displays such as OLED, LCD, SSD module, etc; Using Serial communication and much more!
- Have fun exploring the new stuff now!

Get the Arduino IDE at: https://www.arduino.cc/en/Main/Software

SOLDERING IS EASY

HERE'S HOW TO DO IT