Sensors and Actuators

Dr. Hardik J Pandya
Department of Electronic Systems Engineering
Indian Institute of Science, Bengaluru

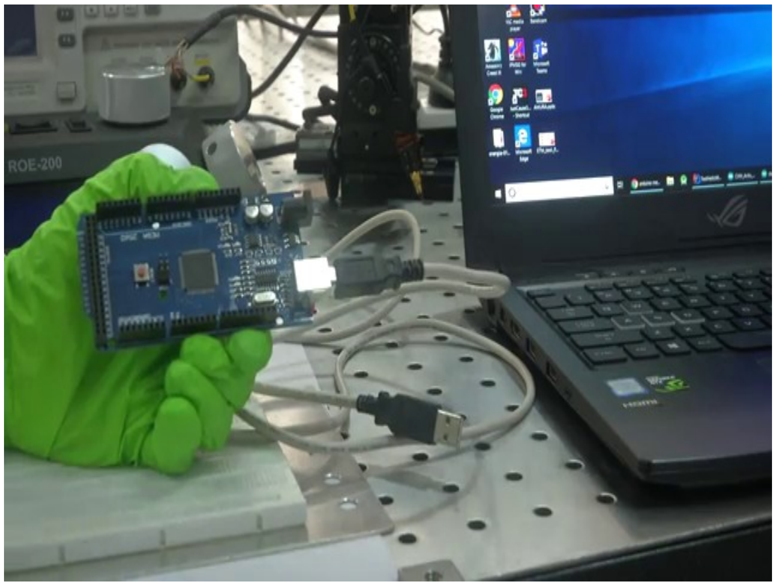
Lecture - 20

In this lab what we will be teaching you is how to interface Sensors and Actuators with Arduino. Now you already have used Arduino, for people who do not know what Arduino is, we will be discussing that particular board and you will be looking at how we can interface actuators and sensors with Arduino board. This is like a practical class. So Arjun, who is my teaching assistant, will be taking this class.

I am sure that in a lot of universities and colleges they teach Arduino as a part of their experimental exercise, particularly for Electronics students. But if you still do not have an idea it is fine. Use this lab as a practice lab for understanding how can you interface several sensors and actuators with the Arduino board. I will see you in the next lab class. Till then you take care bye.

Hi! I am Arjun, TA for the course Sensors and Actuators. We will be having a small tutorial on how to use an Arduino board. It is a microcontroller that can be used to interface our sensors as well as control our actuators that we have discussed earlier. An Arduino board is nothing, but a microcontroller that can be programmed; it is a development board basically. We will be seeing how an Arduino board is used, the features that we can utilize as well as the Arduino IDE, the software platform that we use to control the Arduino.

(Refer Slide Time: 02:21)



In the previous lectures, we discussed different types of Arduino. You may remember Arduino, Arduino mega, Arduino nano and other boards. What you see here is an Arduino mega board. As you can see here, it is written Arduino mega 2 5 6 0. There are different boards and either the capacity changes or the number of pins or size varies according to their applications.

Arduino mega board has 54 digital input-output pins. We can take both inputs as well as output digital pins, besides we have 16 analog pins also. Using these analog pins over there we can take analog inputs and also use them as digital outputs. We have 16 PWM pins among these digital pins which can be utilized to control the speed of the motor and the other things.

You can also see mega 2 5 6 0 IC, which controls the Arduino board. It becomes the brain of the Arduino board. Arduino Mega has a major difference when compared to Arduino Uno, it has for UART ports like for Arduino Uno. We can take TX RX from one set of them; while we have four of them for serial communication. Now, you can see a power jack over there in the mega where we can power it up, you can provide up to 15 volts.

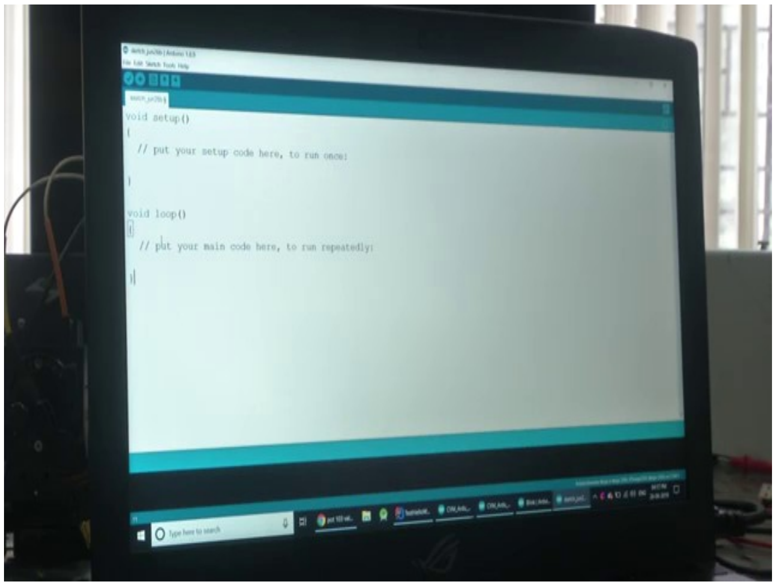
We already have a voltage and current regulator inside it. But safe operating voltage is 9 to 12 volt, so it is better to keep it that way. You can see the Arduino mega Watts at 16 mega Hertz and you can see the crystal oscillator over here. These are the basic constructional things of the

Arduino mega. It also has some 5 volts 3.3 and ground pins which we can utilize for small LEDs and other things that we are going to show.

(Refer Slide Time: 04:45)

Now, I am going to connect the Arduino mega to my laptop. This symbol over here is for the shortcut for Arduino IDE. Arduino IDE is the tool that we will be using to program the Arduino mega or Uno or any of the Arduino boards.

(Refer Slide Time: 05:09)



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

}

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

}
```

This is how it will come, it is a sketch. This is the window of the Arduino IDE where we will be doing the programming. The programming language over here is based on embedded C. It is not the pure embedded C, but an Arduino variant, but if you have logic in Arduino, the embedded C or C++, it will be very easy for you to catch up with it. Now I will just increase the font.

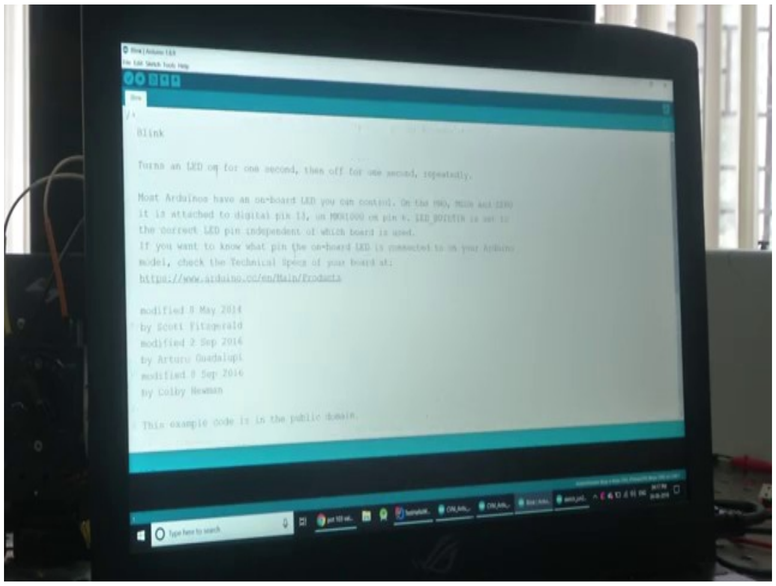
You can see here, by default there are like two functions over here. You must be familiar with functions and other things. There is one void set up here and one void loop here. Why by default there are two functions over here? Consider we are doing some experiment, there will be some initial setup or we have to set up a facility or a lot of equipment or the things that we have to do the experiment with. There will be a setup process before any experiment. This void setup function is something similar to that. This setup or the setting up process will only happen once during an experiment.

Similarly, in Arduino, this void setup runs only once. This setup function will be the place where we will be doing all sorts of initial arrangements and other things. I will let you know what kind of things are those. For example, I have already told you there are digital pins, there are analog pins, there are PWMs. we can generate PWMs and lots of things like that; also we will be able to transfer data serially. All these facilities are there with the Arduino board. In the void setup what we do is, state whether a digital pin is going to be used as an input or as an output.

Similarly, if we are using an analog pin as an input or output, all the setup process comes under the void setup over here, and then there is a void loop. Once we start any experiment, what we do is, we might do multiple iterations with a multiple change of cases and other things. That process repeats over and over; similarly in the Arduino also we have a void loop that repeats over the entire duration of the runtime, as long as the Arduino is powered up. There are two main functions in the Arduino IDE; one is void setup and one is a void loop. Now we will do some sample codes and see how it is implemented in the Arduino.

One of the most common examples is a Blink program. You can see over here I am taking a Blink code.

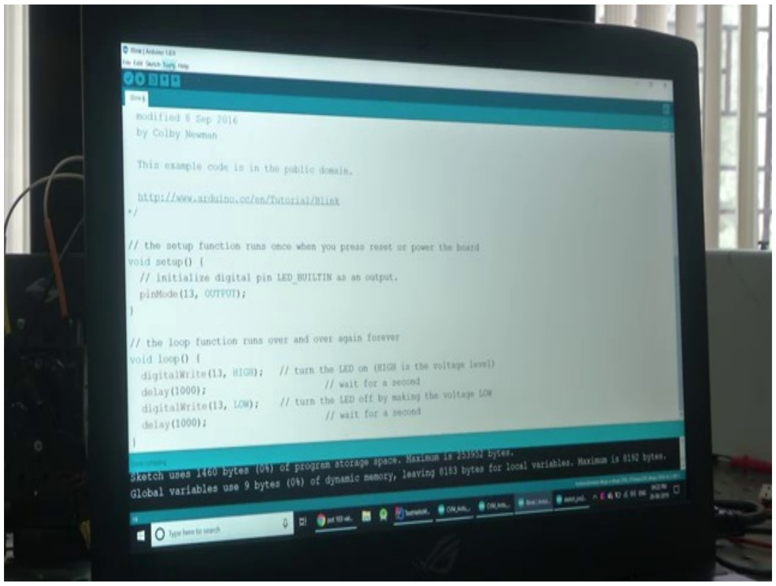
(Refer Slide Time: 08:17)



```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.
```

You can see here in the File menu there are examples, a lot of examples will be there; Arduino has support from a lot of libraries because a lot of developers use Arduino. You can see other drivers and other things are interfaced with Arduino. And once you install the libraries you will get examples for all. I am just taking an example of Basics and Blink. I am going to show you how an LED be blinked using an Arduino program.

(Refer Slide Time: 08:51)



```
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

```
Sketch uses 1460 bytes (0%) of program storage space. Maximum is 253952 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 8183 bytes for local variables. Maximum is 8192 bytes.
```

You can see here in the void setup, we are defining a pin mode LED Built-in, Output. It means that LED Built-in here is a variable. In Arduino mega, it is understood that it is Arduinos, in usual Arduinos LED, Built-in is connected to pin number 13, digital pin number 13. But to avoid confusion, and since you are not aware of that, I am just converting it to 13. This is how it works in a void setup. There is pin mode 13, Output; this means that the 13 pin, the 13 digital pin is an Output pin. If we change this to input it means that the 13 pin is an input pin now.

This is what we set up inside the void setup function. Right now I am putting this 13 pin as Output and what you see here is the void loop. These are one of the functions that you can see digital right 13, High. This is one of the functions that we use to control the digital pins. The digital write command is used to tell the Arduino that, when this command is being called or this function is being called, make the 13 pin high. Since it is a digital pin, it has only two states either high or low.

When this command or function is being called, it will automatically make the pin state, the pin that is referred here high or low depending upon what comes over here. You can see a similar line here digital write 13, low. this is basically like make the LED that is connected to the 13 pin on, and this is to make the pin LED off. And you can see a delay function here, which is a common function we use. Delay is used to delay the code run for a void. The void loop will be running infinitely like an infinite for loop or void loop. In between if you want to delay or something, you can put a delay like this and whatever number that comes in the brackets is in micro milliseconds.

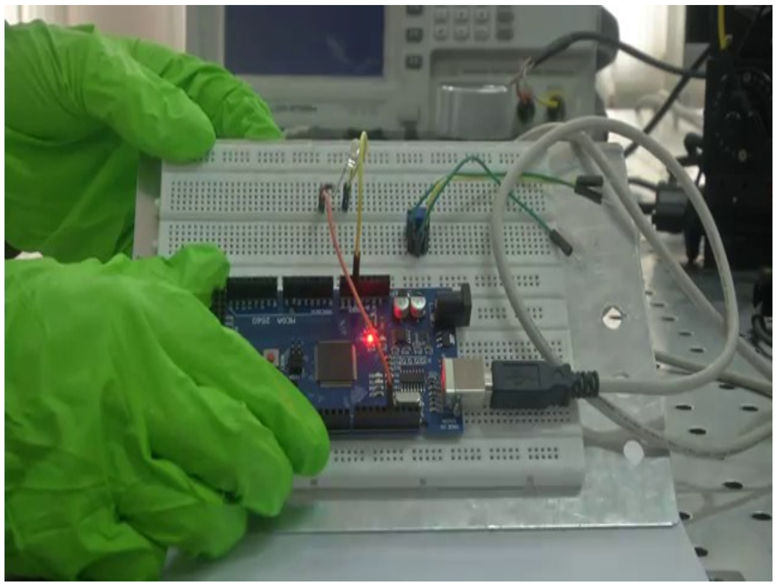
1000 means it is 1 second. What this code will do is, initially it will make say in 13 pin as an output, then after that, it will start running into the void loop infinitely and the first step it will do is it will make the 13 pin high. I am going to connect an LED to the Arduino board and pin number 13. And also in series I am connecting a resistor to prevent it from getting damaged since the high in the high state of the Arduino is 5 volt which is not desirable for the LED for a long time.

What we essentially want after this program is the 13 pin will be high; that means, the LED will glow it will stay like that or it will glow for one second; after that that LED will turn off for one second and again the void loop repeats itself. It will be like an LED blinking at an interval of one second. This is how the code works and you can see multiple buttons over here. You have a lot of options over here like increase their font size and other things, also in the sketch, you can see the commands compile, upload and other things also there is button here which is the compile button.

Once you have done the code, you can just press this tick button and you will show the progress bar here and it will be written done compiling if there are no errors and if you press this button, code will be uploaded to the Arduino board. In the tools, you have to note this one, very important. You have to select the board. Since I have connected an Arduino mega, I have to choose under this board as mega; if it was an Uno you have just select Uno, it is Mega so we are selecting Mega 2 5 6 0 and also you will have to select the comport. Only one Arduino is connected now, it is showing comport 11 only, so I have selected that.

These are the important steps that you have to do before you upload the code. Once I start uploading the code, you will see that the data is being transferred to the Arduino and it will start the RX and TX, the receiver and transmitter LEDs will be starting to blink, showing that the code is being uploaded. After the code is uploaded, the LED attached the pin 13 will also start blinking at the interval of one second. I am going to press this button.

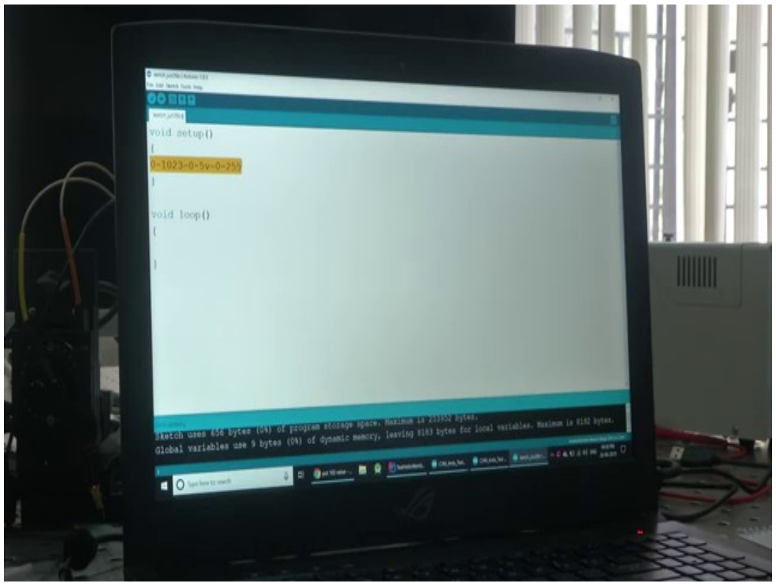
(Refer Slide Time: 13:55)



As you can see, when I uploaded the code the RX and TX pins over there, and the LEDs over there blinked very fast. It means that the code is being uploaded. Now you can see that the white LED that is connected to pin number 13 is blinking at the rate of one second i.e. it is turning on and off at an interval of one second.

Also, you can see a red LED in Arduino blinking at the same rate at the same time. I have already mentioned that the Built-in LED in Arduino Mega is connected to pin number 13 and we have used the same pin. Both LEDs are blinking, that is how through a digital write we will be giving a digital output in Arduino Mega or any of the Arduino boards.

(Refer Slide Time: 14:57)



```
void setup()
{
0-1023-0-5v-0-255
}

void loop()
{

}
```

```
Sketch uses 656 bytes (0%) of program storage space. Maximum is 253952 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 8183 bytes for local variables. Maximum is 8192 bytes.
```

Now, we will see some other functionalities also. Now I am going to show you, another example. we have seen how we send out our digital outputs through Arduino mega. Now, I am going to show another example; how we can take an analog input, along with this I will be showing another functionality called Serial Monitor. In the tools you can see something called a serial monitor. The serial monitor function is used to display some data that Arduino will send to the PC and we can view it.

It is basically used as a method to see some data that we are acquiring from the Arduino. In this experiment, I will be connecting a potentiometer to analog pin A 0. As I told you earlier that there are 16 analog pins, out of which we call the digital pins from 0 to 53, there are 54 pins, so 54 digital pins. They will be named as 0 to 53, 0 1 2 3 like that. We already declared them as pin mode 13 etcetera.

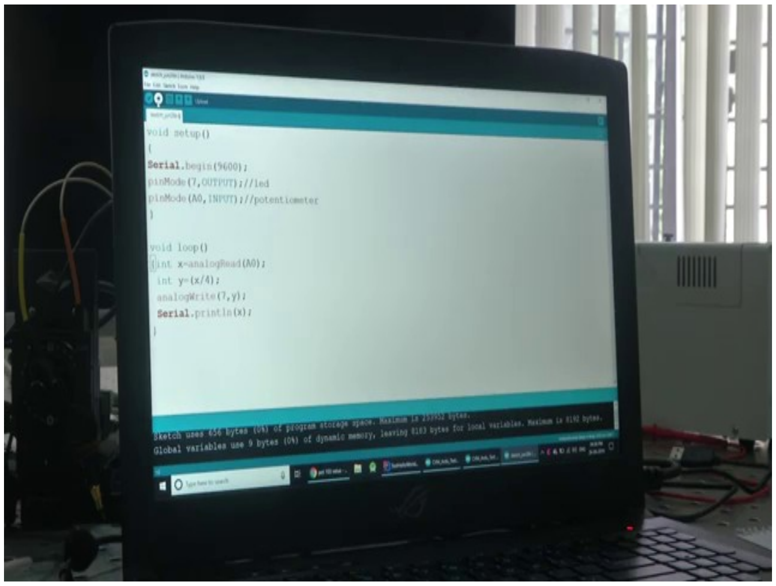
When it comes to calling the name of analog pins we name them as A 0, A 1 up to A 15 there are 16 analog pins. A 0 to A 15 makes 16 pins. I will be taking data from one of the analog pins, I will be using this to control the brightness of an LED that is connected to pin number 7, which is one of the digital pin as well as the PWM pin. We can control the brightness of LED at the same time, I will be using the serial monitor to print the value that we received from the potentiometer.

When we receive some data or when we acquire some data from an analog pin in Arduino, the value range we will be getting it has voltage only. Arduino works at 0 to 5 volt. When we acquire any signal through an analog pin, what we get is like a range from 0 to 1023; that means, actually there are 1024 values. This 0 to 1023 value is equivalent to 0 to 5 volt when it comes to voltage. That is what we will be converting over here. This value will be the range that we are going to get from the analog input.

And when it comes to a PWM output to control the brightness of the LED, the range is equivalent to 0 to 255. The resolution in input, as well as output, is different. When it comes to PWM output it varies from 0 to 255, when it comes to analog input it is from 0 to 1023 and the voltage corresponding to it all is 0 to 5 volts. This is the basic understanding when you go through some Arduino tutorials, you will be getting more idea about it.

We need not discuss this because our course is basically Sensors and Actuators, I am explaining this all so that, we can interface some sensors and show you how it works.

(Refer Slide Time: 18:37)



```
void setup()
{
Serial.begin(9600);
pinMode(7,OUTPUT);//led
pinMode(A0,INPUT);//potentiometer
}

void loop()
{int x=analogRead(A0);
 int y=(x/4);
 analogWrite(7,y);
 Serial.println(x);
}
```

```
Sketch uses 656 bytes (0%) of program storage space. Maximum is 253952 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 8183 bytes for local variables. Maximum is 8192 bytes.
```

We will show, an example now. You can see that I am typing some function called Serial.begin and in the bracket I have put 9600. This is called the baud rate or at the rate at which the Arduino transfers data through into the laptop. This is one of the minimum baud rate that we can choose that is 4800. You can see over here, when we connect to the serial monitor there are 300 1200 2400 4800 9600. We can select any one of the rates; I am just selecting 9600 only.

I have typed the Serial communication. Once this function is being called, serial data will be sent from Arduino to the PC and we can use it for multiple purposes. That is there, and then I am going to put one more thing. I am going to define one of the pins where we have connected the LED. I am saying a pin mode, I am connecting the LED to the 7 th pin. I am putting it as output. since an LED is an Output, also similarly I am doing pin mode A 0, where I have connected this thing to a potentiometer as input.

Actually, we need not define A 0 as input, because by default the analog pins are input only, but still for clarity, I am putting this as input. Now, since we only need to start serial communication, and then make sure that the 7th pin is an output pin. So that where we have connected the LED, all the commenting function and other things in C++ also works here. I am saying it is LED and this is potentiometer. In the void loop I am going to take the value from the potentiometer and control it, and use it to control the brightness of the LED.

I am defining some integer x and I am going to store the value of the potentiometer into it. I am going to use the function analogRead, the coding in Arduino is Case Sensitive. You have to take care of which all letters to be capitalized and other things; you need to practice to write it by hand. I am going to put this analogRead function. When this function is being called, it will store the value of the potentiometer in the range of 0 to 1023 into this integer x. Now as I have already said this integer value is in the range of 0 to 1023 so, because analog Read has a resolution of from 0 to 1023 and when it comes to PWM output, we only have a range of 0 to 255, which is like one-fourth of the analog Read resolution rate.

I am going to define another value y, and I am going to store this y = x / 4. I am just doing a mathematical function over here I will show you another function also on the later stage. What I am doing is, I am writing analog Write; because now I am going to the y for using the PWM functionality of a digital pin we have to use the analog Write function. I am going to say 7th pin , as y. This means that if this y is 255; that means, it is equivalent to a digitalWrite (7, high) that means, the LED will blink or glow at maximum brightness and if the value y is 0 it is equivalent to low. Since this value y varies from 0 to 255 now, the brightness of the LED will also vary. If I upload this core now, the brightness of the LED will be varying continuously, as we change the value of the potentiometer.

But I am going to put some other function also here. Since I was telling you how to print the value that, we get from the potentiometer directly. I am going to print the value x over here; that you will see how we can display the value that we get also in the serial monitor. I am now going to upload the code here. I will show you, how it will look when we change the value in the potentiometer in the serial monitor as well as how the brightness changes in the LED. I am going to upload this code.

(Refer Slide Time: 24:09)

First, we will compile it to verify. When it is done compiling and there is no error, I am going to upload this code. You can see this green progress bar, Uploading is done. Now, first you can see on the left side the LED is blinking; that means, the potentiometer is sending some value that is not equal to 0. Now I will show you the serial monitor, under tools. You can see some value over here right, this is the value of the potentiometer in the range of 0 to 1023.

When I am turning the knob of the potentiometer in the clockwise direction; that means, I am increasing the value of the port that means the voltage is increasing. I am actually decreasing the resistance in the potentiometer. Another method like here you can see the board rate that we have already chosen. Another method of seeing this since this is a waveform and the tools we have something called serial plotter also.

You can see the scale over here; the problem with the serial plotter is that it will alter just the scale very much, so it is quite confusing. But you can see when I am not turning the potentiometer the value almost remains constant, the value from 8849 to 852 is what it is in between, as you can think of like say 0 to 1023 is approximate to around 1000 values and 5 volts is being divided into 1000 values right, that is around like 0.005, so that much accurately you can measure the voltage or in terms of around 5 millivolts you are able to measure it.

Now if I am turning this potentiometer you can see the value increasing. And also if I turn it the other way around it will start decreasing. I will show another trick to stabilize this one, it is not a good method, but I will show you, at the same time before that you can see that the LED's brightness is also decreasing or increasing. Now, I will start decreasing the brightness. I am turning the potentiometer in an anti-clockwise direction. The value is decreasing here also in the serial plotter, slowly you can see the brightness of the LED is also going down, see the LED is almost dim now and it is off.

Once it reaches the 0 value of the potentiometer, the LED goes low. I will just plot some other value also for you to understand this in the video.

This is not a much-recommended method, but I am just showing that because the scale would not change here. If I turn in the anti-clockwise direction it is going to reduce, this is how we take analog input and print it in a serial monitor or any of the functions. Instead of this y = x /4 I told you earlier, I can even do something else, I am commenting that statement and if I put them into y is equal to map of, this is another function that you can use here. I am putting map. x is in the range of 0 to 1023 and I am going to scale it to the value of 0 to 255. This is another function that we can use alternatively.

You will use this map function a lot of times when you do some complex programs and other things. Now, the value y is mapped with respect to x in this range. x is in the range of 0 to 1023 and you are going to map this value in terms of 0 to 255, it will show you the same results. So this is an alternative method. If you do not know, it is not an easy conversion, then you can just use malfunction. This is how you use; you take analog input from Arduino and maybe use PWM or here PWM to control LED brightness.

PWM is pulse width modulation. You will be learning more about it. This PWM signal can be used to control the brightness of LEDs; as simple as that to even just controlling the speed of motors and other devices and even in many of these sensors and other things, we will know about it later and this is how we use serial monitor. This is the basic intro that you require when you use Arduino. We will be interfacing Arduino with different Sensors from now and we will be using most of these functions and also some actuators and there also you will be using these functions.

Thank you.