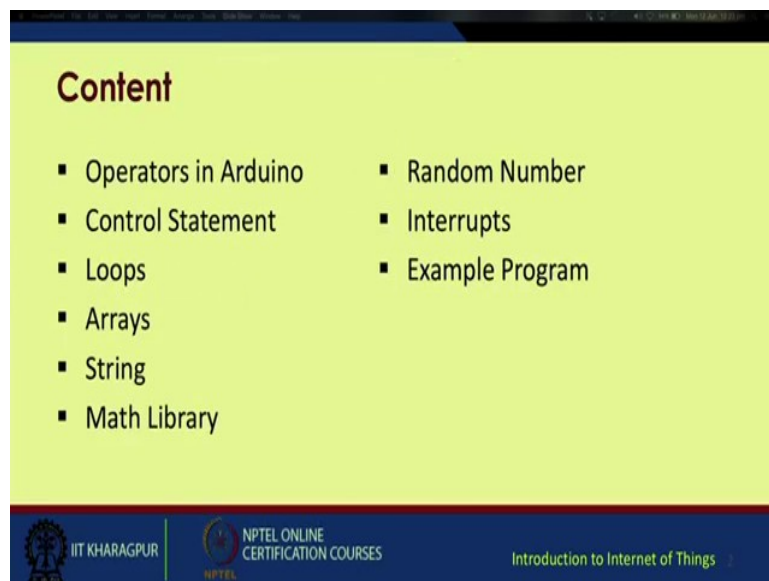


Introduction to Internet of Things
Prof. Sudip Misra
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture - 23
Introduction to Arduino- II

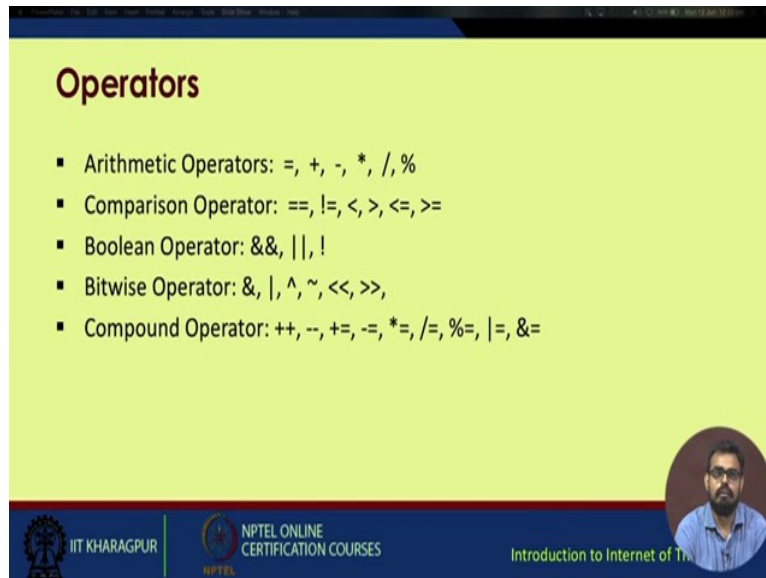
Hi. Now, we will continue with the Introduction to Arduino Programming. This will be the 2nd part and the continuation of the previous one.

(Refer Slide Time: 00:28)



So, we will cover the basic topics, the Operators in Arduino, Control Statements, Loops, Arrays, Strings, The Mathematics Library, Random Number Interrupts and Example Program which will be bit complicated than the previous one.

(Refer Slide Time: 00:44).



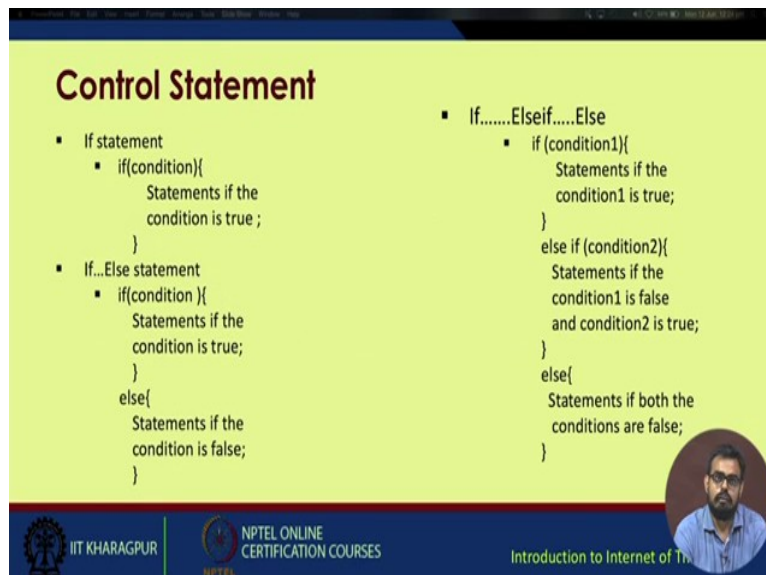
Operators

- Arithmetic Operators: =, +, -, *, /, %
- Comparison Operator: ==, !=, <, >, <=, >=
- Boolean Operator: &&, ||, !
- Bitwise Operator: &, |, ^, ~, <<, >>
- Compound Operator: ++, --, +=, -=, *=, /=, %=, |=, &=

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet of Things

So, basic operators as normal C, C++ or python programming or other languages, you have the basic equal to, plus, minus, multiplication, division module, division operators, then comparison operators, we have equal to, not equal to, less than, greater than and all those operators. Then, we have Boolean operators, bitwise operators and compound operators.

(Refer Slide Time: 01:14)



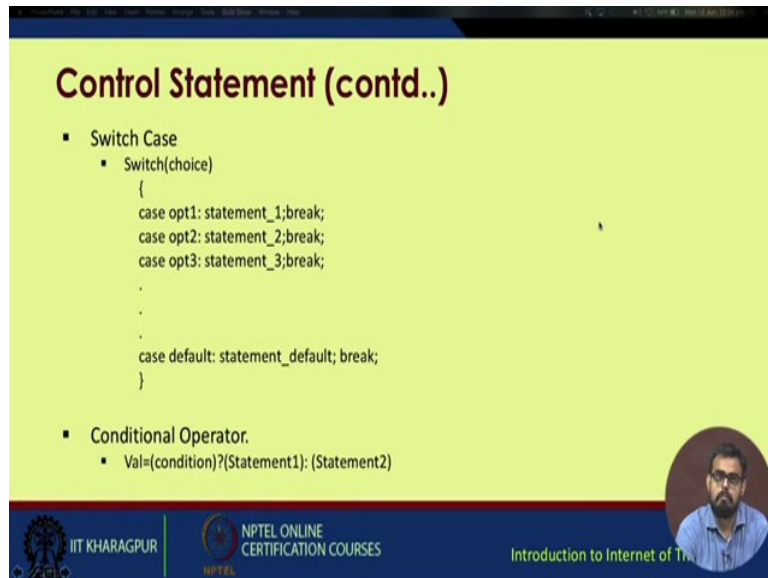
Control Statement

- If statement
 - `if(condition){`
Statements if the condition is true ;
`}`
- If...Else statement
 - `if(condition){`
Statements if the condition is true;
`}`
`else{`
Statements if the condition is false;
`}`
- If.....Elseif.....Else
 - `if (condition1){`
Statements if the condition1 is true;
`}`
`else if (condition2){`
Statements if the condition1 is false and condition2 is true;
`}`
`else{`
Statements if both the conditions are false;
`}`

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet of Things

Moving on to control statements, these will basically cover the various checking and looping parts. So, a normal if else statement in Arduino, we start off with if statement.

(Refer Slide Time: 01:49)



Control Statement (contd..)

- Switch Case
 - Switch(choice)

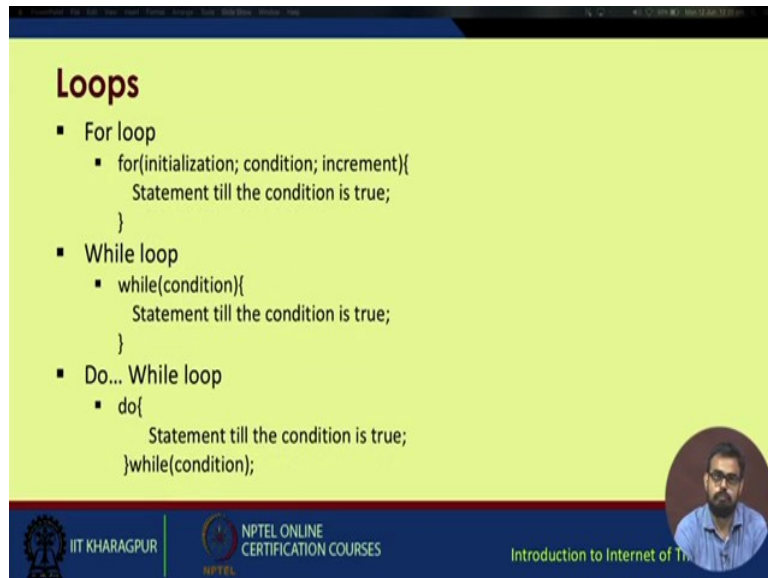
```
{
  case opt1: statement_1;break;
  case opt2: statement_2;break;
  case opt3: statement_3;break;
  .
  .
  .
  case default: statement_default; break;
}
```
- Conditional Operator.
 - Val=(condition)?(Statement1):(Statement2)

The slide features a yellow background with a dark blue header and footer. A small circular inset photo of a man with glasses is in the bottom right corner. The footer contains logos for IIT Kharagpur, NPTEL, and the course title 'Introduction to Internet of Things'.

So, if you have a condition and within this curly braces, if the statement condition is true.

If else, if another statement condition is true or else if none of the above statements are true, then this loop will execute moving on to switch case. You have switch and choices, you have case option 1 and statement and then, a break function after each case. So, case option 2 is statement 2, then again a break and so on and at the end, you have a default case. After that we again have a break function, then you have a conditional operator and we will avoid using conditional operators such as these in Arduino. So, it is condition if it is true, it will execute statement 1, else it will execute statement 2. These kind of statement operators are best avoided during Arduino programming.

(Refer Slide Time: 02:35)



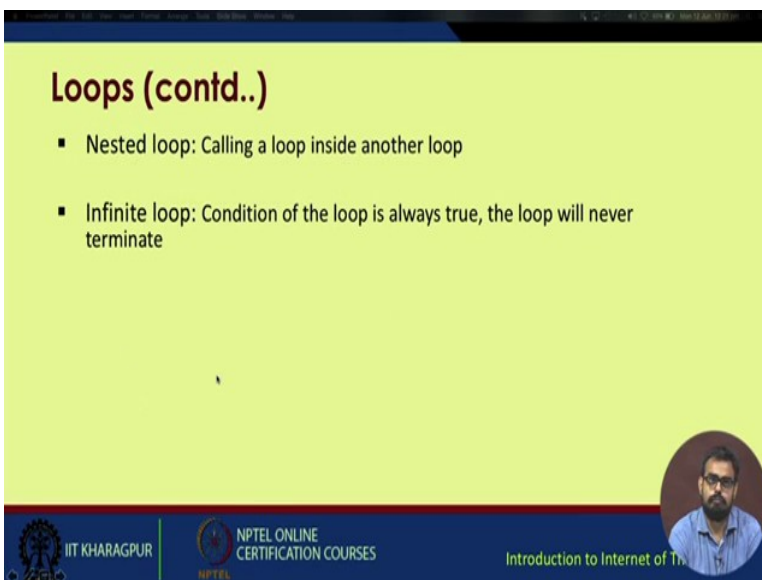
Loops

- For loop
 - `for(initialization; condition; increment){`
Statement till the condition is true;
`}`
- While loop
 - `while(condition){`
Statement till the condition is true;
`}`
- Do... While loop
 - `do{`
Statement till the condition is true;
`}while(condition);`

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet of Things

So, in loops you have the basic for loop, then you have the while loop you have a, do while loop, these are pretty common examples.

(Refer Slide Time: 02:47)



Loops (contd..)

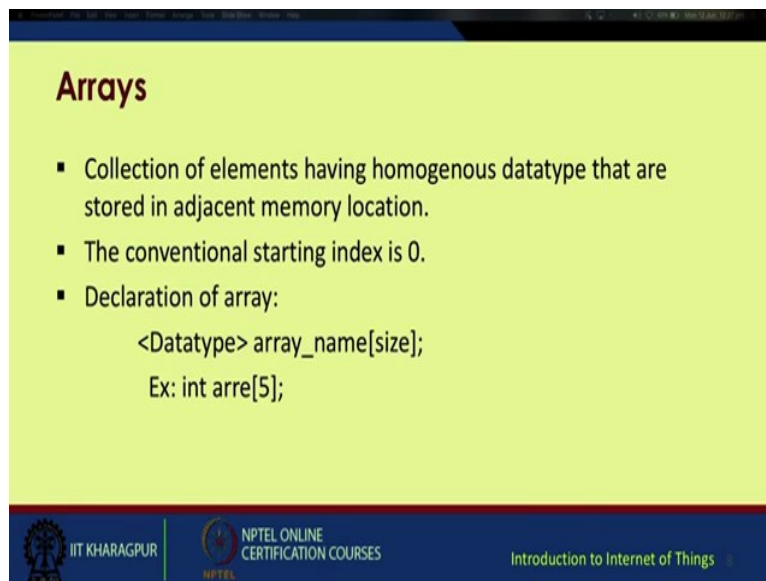
- Nested loop: Calling a loop inside another loop
- Infinite loop: Condition of the loop is always true, the loop will never terminate

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet of Things

You have a nested loop that is a loop inside another loop. You can have many nested loops inside each other. So, you have an infinite loop. So, to run an infinite loop for example, all you need, you develop a system in which you need to turn on and off a light or a LED or any other device infinitely as long as the devices on your system is checking. So, recall from the last lecture which I showed the blinking LED example.

So, you can see if you put it inside an infinite loop as long as the Arduino board is powered, it will keep on blinking. So, your functions can be made more complicated. Instead of LEDs, you can have motors. Instead of motors, you can have actually cameras mounted on the motors and they keep on rotating. You can have a multitude of sensors which are interface with the cameras and the motors. So, for example you can build the security systems which will keep on running as long as your processor board is fine and the power is being supplied, you can always connected to a battery supply to generate power for it.

(Refer Slide Time: 04:08)



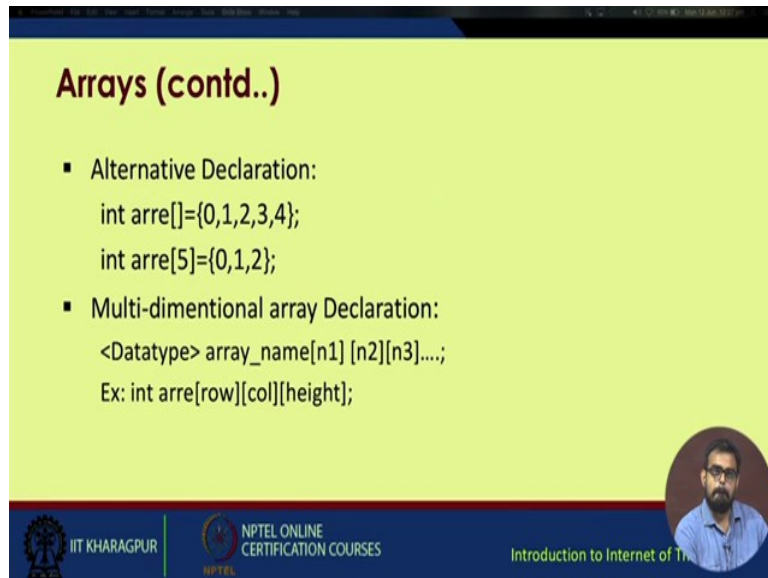
Arrays

- Collection of elements having homogenous datatype that are stored in adjacent memory location.
- The conventional starting index is 0.
- Declaration of array:
`<Datatype> array_name[size];`
Ex: `int arre[5];`

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet of Things

Then, you have arrays. Arrays are collection of elements having homogeneous data type and which are stored in adjacent memory locations. The conventional starting index is 0 in Arduino. So, declaration of array you just start off with the data type. It maybe arrays of integers, so int array name and the size. So, for example, int arre this a variable name are[5], it will allocate five spaces for your array, then you can have an alternate declarations.

(Refer Slide Time: 04:44)



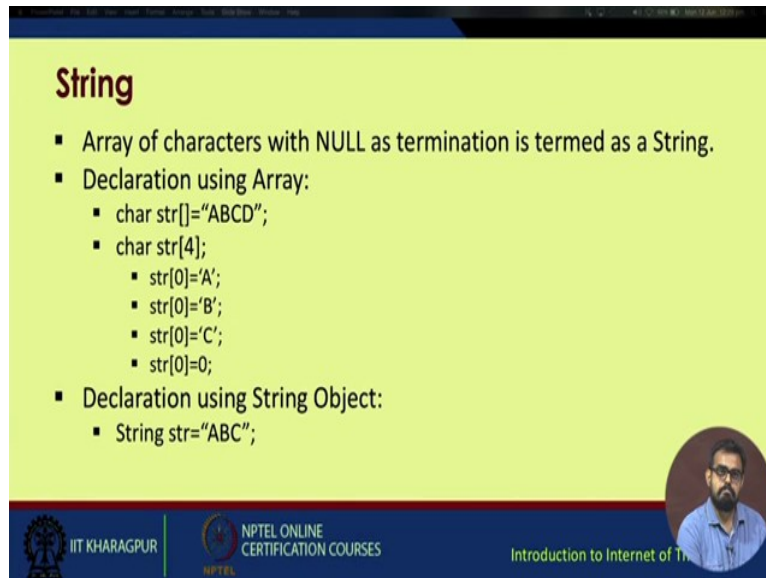
Arrays (contd..)

- Alternative Declaration:
`int arre[]={0,1,2,3,4};`
`int arre[5]={0,1,2};`
- Multi-dimensional array Declaration:
`<Datatype> array_name[n1][n2][n3]....;`
Ex: `int arre[row][col][height];`

The slide features a yellow background with a dark blue header and footer. The footer contains the IIT Kharagpur logo, NPTEL Online Certification Courses logo, and the text 'Introduction to Internet of Things'.

Suppose int array and this blank bracket [] equal to within this curly braces, you have 0 1 2 3 4. So, these will be automatically stored in this array. Then, again you have enter a five, you can just put in three variable, three values inside this array and the remaining will be kept blank maybe for later use. You can fill those also. Then, you have multi-dimensional array declarations same as previous one, you have the data type, then array name, then the dimensions for the first dimension let it be n1 n2 n3. So, for example, if you want to declare an array for an image which normally normal rgb image, so you have three channels red, green and blue. So, each image will have a 2D structures, rows and columns and there will be a depth for each r, g and b. So, maybe for those types of data, we have int array row column height.

(Refer Slide Time: 06:03)



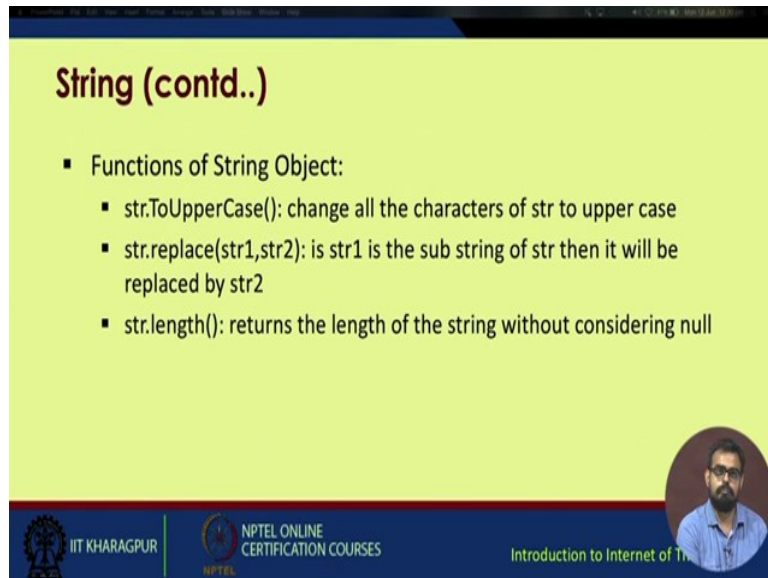
String

- Array of characters with NULL as termination is termed as a String.
- Declaration using Array:
 - `char str[]="ABCD";`
 - `char str[4];`
 - `str[0]='A';`
 - `str[0]='B';`
 - `str[0]='C';`
 - `str[0]=0;`
- Declaration using String Object:
 - `String str="ABC";`

The slide features a yellow background with a dark blue header and footer. The title 'String' is in a bold, dark red font. The content is organized into a bulleted list. In the bottom right corner, there is a circular portrait of a man with a beard and glasses, wearing a blue shirt. The footer contains the IIT Kharagpur logo, the NPTEL logo, and the text 'NPTEL ONLINE CERTIFICATION COURSES' and 'Introduction to Internet of Things'.

So, then moving on to string, string is an array of characters with null as the termination declaration is maybe using `char char string`. Here `str` is the array. So, `abcd`, so this is stored in `str char str[4]` and you can individually access each `ids`, you can store `A B C` or maybe `0`. So, this using the same location if you want to individually store in different locations. So, sorry come in the same location if you keep on storing this. The last one will be last character stored will be updated and other will be overwritten. If you want to store in different locations, you just change it from string it from string `0`, `str[1]`, `str[2]`, `str[3]` and so on. So, you will have consecutive `A B C 0` side by side these locations. Another thing you can also, you also have a data type string. So, `string str = "ABC"` will give you `ABC`. All together you do not have to store in individual locations. So, this is one of the benefits of using Arduino.

(Refer Slide Time: 07:31)



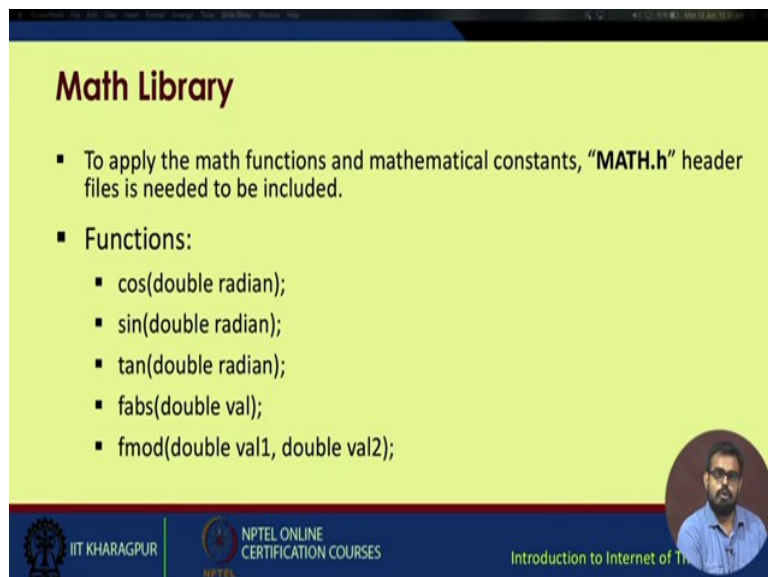
String (contd..)

- Functions of String Object:
 - `str.toUpperCase()`: change all the characters of str to upper case
 - `str.replace(str1, str2)`: if str1 is the sub string of str then it will be replaced by str2
 - `str.length()`: returns the length of the string without considering null

The slide footer includes the IIT Kharagpur logo, NPTEL Online Certification Courses logo, and a small circular portrait of a man.

So, some commonly used functions of string. So, `str.toUpperCase()` point to note is to upper case T U and C are caps. So, this has to be followed strictly since this is part of the syntax. So, it changes all the characters of string to upper case and then, you have string `str.replace(string 1 and string 2)`. So, string 1 if it is sub string of str, then it will be replaced by str 2, then `str.length()`, it returns the length of the string without considering the null character.

(Refer Slide Time: 08:12)



Math Library

- To apply the math functions and mathematical constants, "**MATH.h**" header file is needed to be included.
- Functions:
 - `cos(double radian);`
 - `sin(double radian);`
 - `tan(double radian);`
 - `fabs(double val);`
 - `fmod(double val1, double val2);`

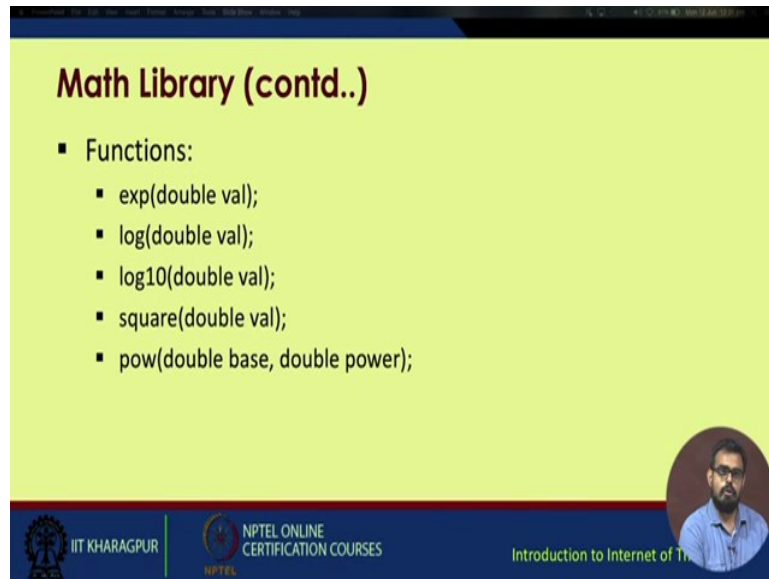
The slide footer includes the IIT Kharagpur logo, NPTEL Online Certification Courses logo, and a small circular portrait of a man.

Then, another commonly used library is the math library. To apply the math functions, the `math.h` header must be initially called, otherwise you will not be able to access these

functions. So, some of the common functions are cos which is in double radian and sin tan floating absolute fabs right floating mod.

So, double value 1 and double value 2, so you have two values and f mod will give you the modular division and the result point will be a floating point number.

(Refer Slide Time: 08:52)



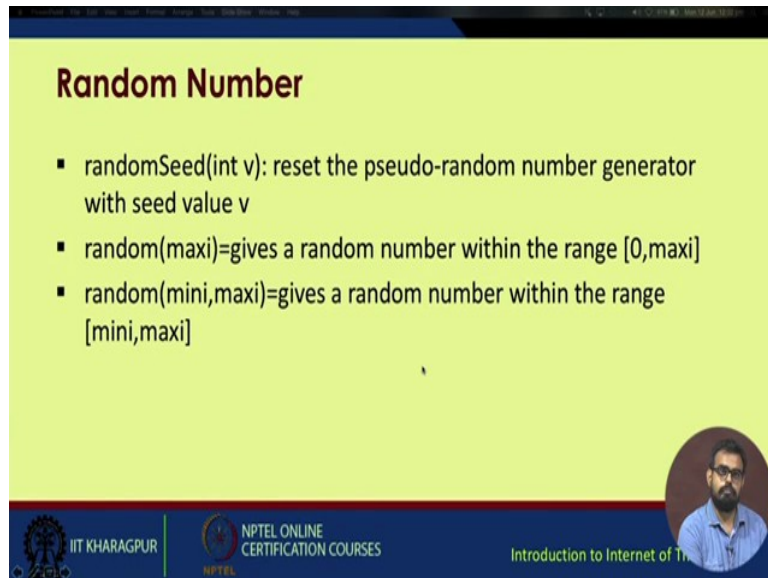
Math Library (contd..)

- Functions:
 - `exp(double val);`
 - `log(double val);`
 - `log10(double val);`
 - `square(double val);`
 - `pow(double base, double power);`

The slide is part of an NPTEL presentation. It features a yellow background with a dark blue header and footer. The footer includes the IIT Kharagpur logo, the NPTEL logo, and the text 'NPTEL ONLINE CERTIFICATION COURSES' and 'Introduction to Internet of Things'.

Then, continuing with the math library again you have exp which signifies the exponential function. You have log function. This will give the natural logarithm of the value. Then, you have log 10, then you have square function, power function. First argument is the base, the second argument signifies the power, then another commonly used example is random number.

(Refer Slide Time: 09:19)



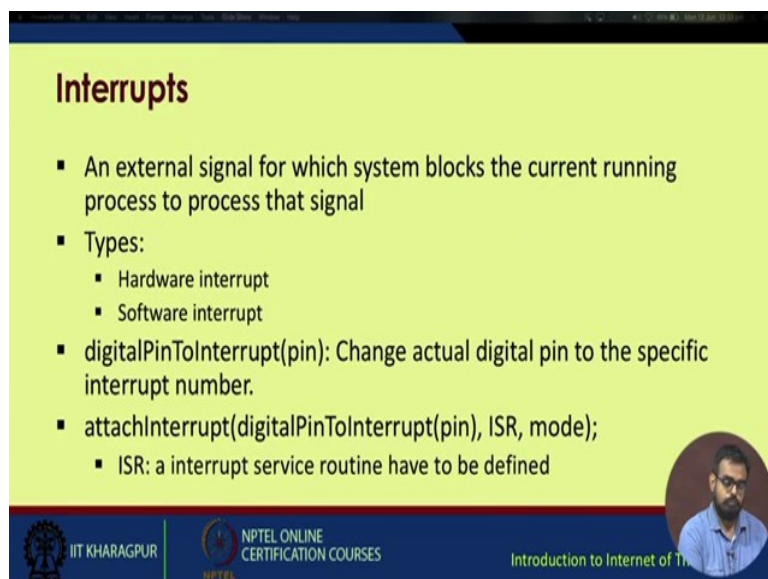
Random Number

- `randomSeed(int v)`: reset the pseudo-random number generator with seed value `v`
- `random(maxi)`=gives a random number within the range `[0,maxi]`
- `random(mini,maxi)`=gives a random number within the range `[mini,maxi]`

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet of Things

So, one of the functions of this random number is random seed. So, the syntax is `randomSeed`, S capital. You need to focus on this one because this is the inbuilt syntax for Arduino. So, `randomSeed(int v)`, it reset the pseudo random number generator with seed value `v`. So, you already the seed value is the starting point from which the random number will initialize its function. So, you gave a starting value, from it the random number will generate, then `random(maxi)` gives the random number within the range 0 to `maxi`, then you have `random(mini ,maxi)` and it gives the random number within the range `mini` and `max`.

(Refer Slide Time: 10:24)



Interrupts

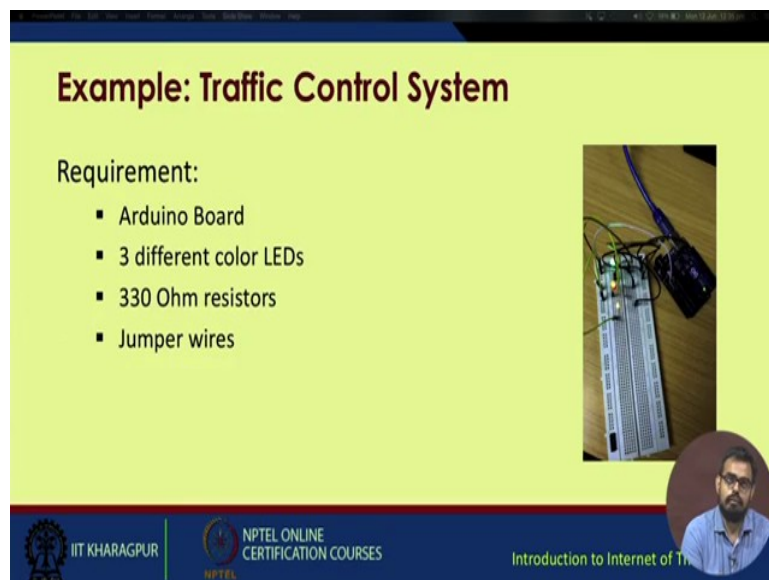
- An external signal for which system blocks the current running process to process that signal
- Types:
 - Hardware interrupt
 - Software interrupt
- `digitalPinToInterrupt(pin)`: Change actual digital pin to the specific interrupt number.
- `attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)`;
 - ISR: a interrupt service routine have to be defined

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet of Things

Then, moving on to interrupts, you have an external signal interrupts. These are basically an external signal for which the system blocks the current running process till receiving that signal.

So, basically you have two types of interrupts. One is hardware and another is software. So, I will give one example. Suppose you are in a loop here waiting for a checking condition whether that checking condition holds true or not and maybe from an external source you are getting that checking condition. For example, you have a button or a digital switch connected to a Arduino board. So, whenever you are pressing that switch, your system will blink an LED, otherwise it will keep the LED off. So, this may be considered as a, partially considered as an interrupt. So, this will be an external interrupt. So, as you can see digital pin to interrupt and then, the pin number it actually changes that digital pin to the specific interrupt number, then attach interrupt digital pin to interrupt, then pin, then ISR, then mode. So, isr is basically known as an Interrupt Service Routine. It has to be defined explicitly. So, these are some of the more complicated function. So, we will not focus on these interruption. Other complications we will just try to keep it as easy as possible.

(Refer Slide Time: 12:07)



Example: Traffic Control System

Requirement:

- Arduino Board
- 3 different color LEDs
- 330 Ohm resistors
- Jumper wires

The slide features a photograph of an Arduino Uno board with three LEDs (red, yellow, and green) connected to its digital pins using jumper wires. A small circular inset in the bottom right corner shows a man with a beard and glasses, likely the lecturer. The slide footer includes the IIT Kharagpur logo, the NPTEL Online Certification Courses logo, and the text 'Introduction to Internet of Things'.

So, we will try to implement within this lecture, we will try to implement a basic rudimentary traffic control system. So, we need Arduino board, three different LEDs, some resistors maybe 220 ohm or 330 ohm and a few connecting jumper wires to connect the various



components on the breadboard as you can see from the previous, as you remember from the previous example.

(Refer Slide Time: 12:31)

Example: Traffic Control System (contd..)

Connection:

- Connect the positive terminals of the LEDs to the respective digital output pins in the board, assigned in the code.
- Connect the negative terminals of the LEDs to the ground



IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet of Things


We connected an LED with the Arduino board. The same process has to be repeated three times using different colored LEDs and at different pins. So, in the last example, we connected the LED to pin 12. So, maybe in this example, we will connect to pin 2 3 and 4 side by side.

(Refer Slide Time: 13:59)

Example: Traffic Control System (contd..)

- Sketch

```
//LED pins
int r = 2;
int g = 3;
int y = 4;
void setup()
{
    Serial.begin(9600);
    pinMode(r, OUTPUT); digitalWrite(r, LOW);
    pinMode(g, OUTPUT); digitalWrite(g, LOW);
    pinMode(y, OUTPUT); digitalWrite(y, LOW);
}
```



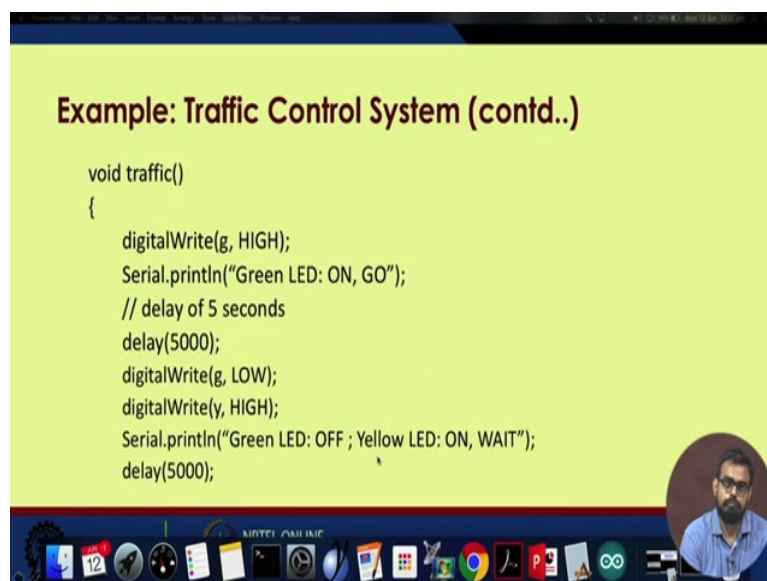
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Introduction to Internet of Things

So, this is the sample sketch. So, as you can see tried to using void setup, we globally define a few value int r = 2. Basically we are taking three colors r g and b, we are not giving b. We are written y.

So, r g and y, red green and yellow, so globally we are defining r as integer equal to 2, g equal to 3, y equal to 4 in within void setup. We initialize the serial port at 9600 baud rate, then pin mode we write r and and output. So, as you can remember the first one was the actual pin on the Arduino device and the second one was the mode either input or output. So, since you are connecting LEDs will be obviously using it as output. So, r was globally assigned as 2. So, this will be 2 and output. So, it translates to pin 2 will work as output and then, digitalWrite r low. So, this function will initially set the a value of pin 2 to 0. So, it will be turned up. The same process is repeated for pin 3 and pin 4 which connected to the green and yellow LEDs.

Now, we define the function traffic. So, data type you have given as void.

(Refer Slide Time: 14:34)

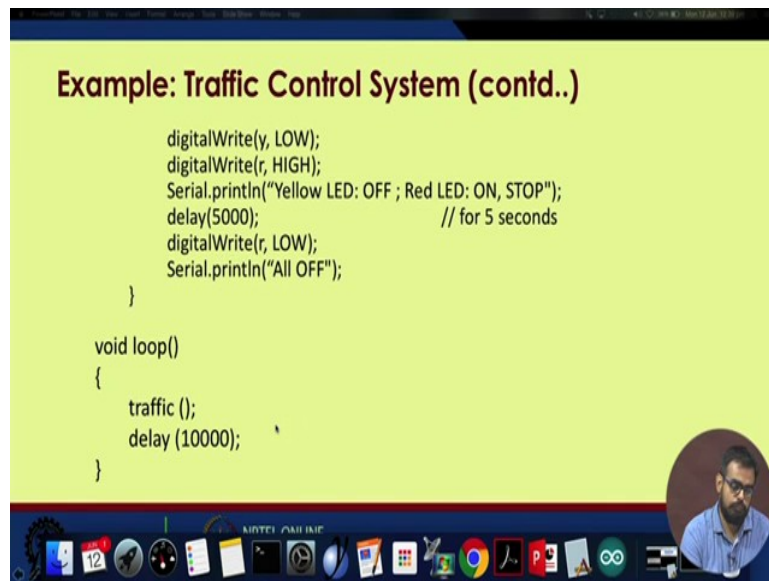


So, void traffic we digit, so digitalWrite g high that was g was defined as pin 3. So, over here pin 3 becomes high, then serial be over the serial port, we print green LED on go. So, since it simulates a traffic signal, you will have a go signal denoted by green, a stop signal denoted by red and a wait signal denoted by yellow. So, green LEDs when it switches on, it signifies go this will be printed on the serial port you can all obviously comment this out.

Then, we induce a delay of 5 seconds using this command `delay 5000`. So, if you remember from the previous example, delay takes an input in the terms of milliseconds. One more thing this double slash, it denotes commenting the character. So, whenever you put `//` in front of a statement, it will not execute. Your compiler will skip the execution.

Next we go to `digitalWrite g low`. So, what is physically happening is at the start of the loop, you have the green signal is glowing and then, after 5 second delay, the green signal is off.

(Refer Slide Time: 16:25)




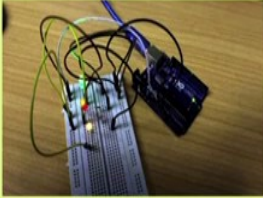
Then, the yellow signal goes high. So, your print again green LED off yellow LED on. So, the status is wait, then again a delay of 5 seconds, then followed by `digitalWrite y` or yellow as low and `digitalWrite r` as high. So, your wait signal will turn off and the stop signal will go high. Same thing is printed on the serial port, then again you have a delay of 5 seconds, then `digitalWrite r` as low. Now, you have all three LEDs turned off and you serially print all off now within this void loop, you call this traffic function. So, this traffic will iteratively run again and again and again till your device is powered and if call a delay of 10000 that is 10 seconds, so your traffic signal loop will run once and then stop for 10 seconds, then it will the whole loop will once again and it will keep on going.

(Refer Slide Time: 17:29)

Example: Traffic Control System (contd..)

Output:

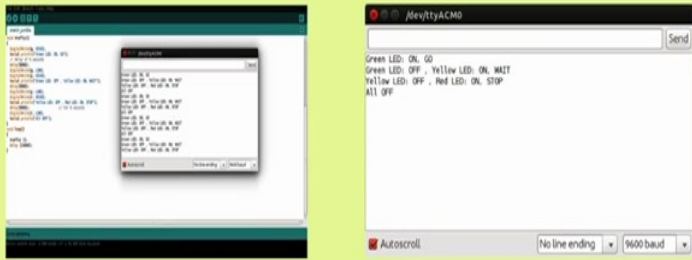
- Initially, all the LEDs are turned off
- The LEDs are turned on one at a time with a delay of 5 seconds
- The message is displayed accordingly
- Figure showing all the LEDs turned on



So, as you can see over here in the image, the LEDs are glowing sequentially. We will come to that on the hands on.

(Refer Slide Time: 17:39)

Output



```
Arduino IDE Code:
// Traffic Control System
// Green LED: ON, GO
// Green LED: OFF, Yellow LED: ON, WAIT
// Yellow LED: OFF, Red LED: ON, STOP
// All OFF

// Pin Definitions
const int GREEN_LED = 2;
const int YELLOW_LED = 3;
const int RED_LED = 4;

// Variables
int delayTime = 5000; // 5 seconds delay

// Setup
void setup() {
  pinMode(GREEN_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  Serial.begin(9600);
}

// Loop
void loop() {
  // Green LED ON, GO
  digitalWrite(GREEN_LED, HIGH);
  Serial.println("Green LED: ON, GO");
  delay(delayTime);

  // Green LED OFF, Yellow LED ON, WAIT
  digitalWrite(GREEN_LED, LOW);
  digitalWrite(YELLOW_LED, HIGH);
  Serial.println("Green LED: OFF, Yellow LED: ON, WAIT");
  delay(delayTime);

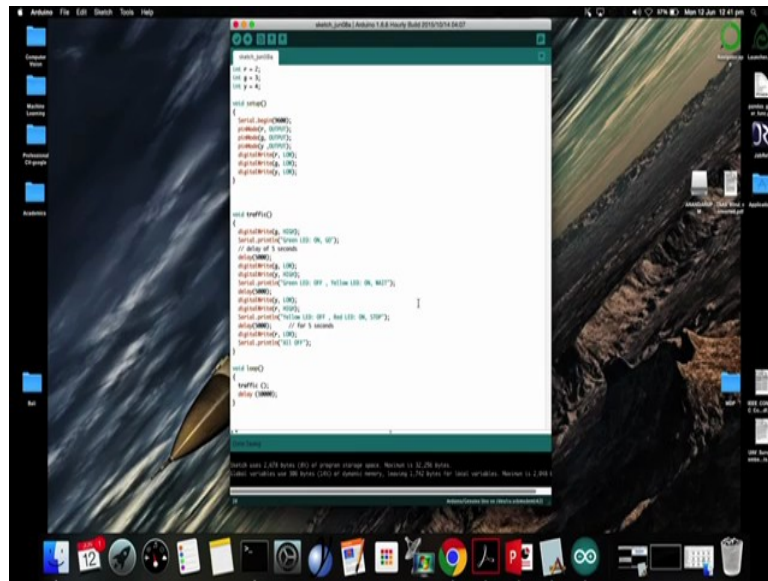
  // Yellow LED OFF, Red LED ON, STOP
  digitalWrite(YELLOW_LED, LOW);
  digitalWrite(RED_LED, HIGH);
  Serial.println("Yellow LED: OFF, Red LED: ON, STOP");
  delay(delayTime);

  // All OFF
  digitalWrite(RED_LED, LOW);
  Serial.println("All OFF");
  delay(delayTime);
}
```

```
Serial Monitor Output:
Green LED: ON, GO
Green LED: OFF, Yellow LED: ON, WAIT
Yellow LED: OFF, Red LED: ON, STOP
All OFF
```

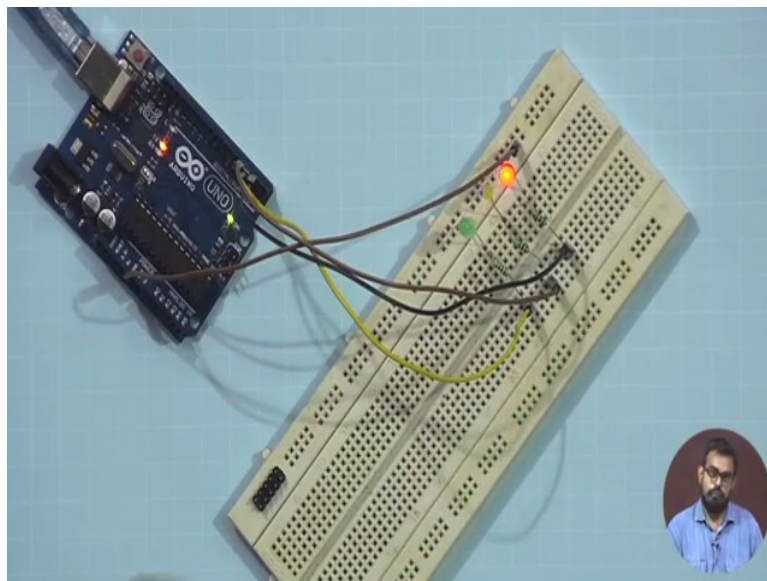
The various outputs are printed on the terminal. Now, if you come back to the circuit part as you can see the code I showed you.

(Refer Slide Time: 17:59)



I have already made the code ready. I have connected the Arduino board.

(Refer Slide Time: 18:14)



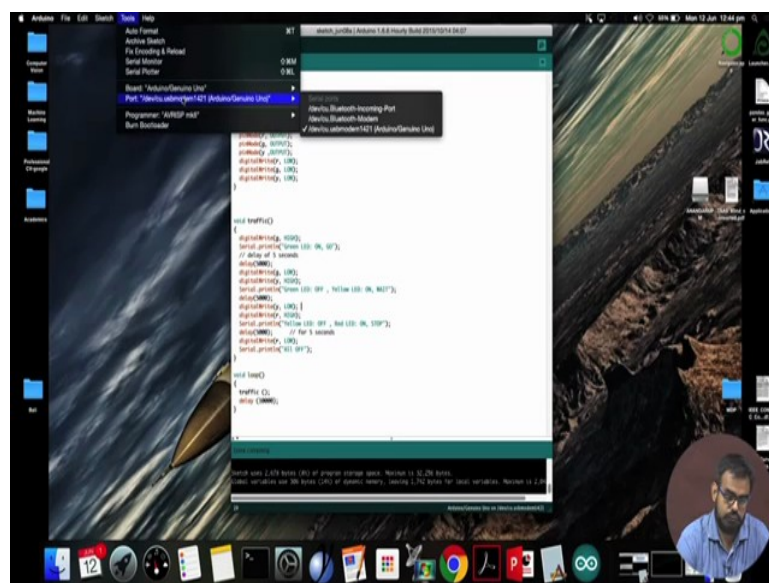
So, I have three differently colored LEDs red, green and yellow. I have three 330 ohm registers, I have Arduino UNO and I have a bread board. I will just connect these sequentially green, yellow, red.

You must always remember this convention that the smaller pin is for ground, the longer pin is for positive signal. So, let us check again this row is supposed to be ground. Since the

yellow is properly connected, green is properly connected. Now, across each LED, we will put up a register, 330 ohm register.

First of all sense in a breadboard, these channels are fused. So, I will only need one signal, single signal to connect to the Arduino. This was negative part will connect to ground. Now, if you remember the green was connected to pin 3, green was assigned a pin 3 in the code. Pin 2 was assigned to red and yellow was assigned pin 4. So, now we have our connections ready. So, three wires for r g and y and one for the ground.

(Refer Slide Time: 20:50)

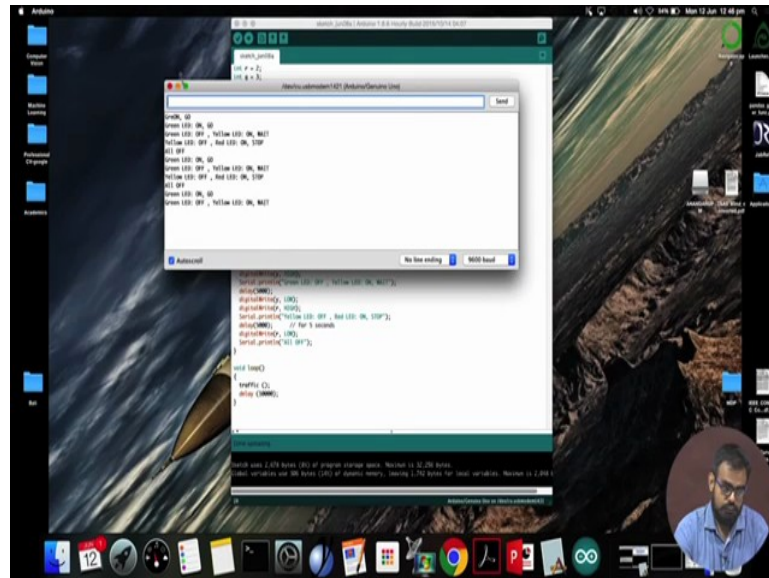


Now, if you go through the code again, so you have globally declared r as pin 2 , g as pin 3 and y as pin 4. So, we have been implemented on the breadboard, then serial port starting baud rate is 9600 pin mode, r g and y as outputs digital write, r g and y as low that is initially at the very beginning everything will be turned off and within void traffic you switch on each LED one at a time starting from green, then yellow, then red. So, this basically what a traffic signal does is, so we will compile this code.

You see the code has been compiled without any errors. I will check whether my board is connected. Yes it is Arduino UNO the port is, so the board and the ports are fine. We will upload the code or the sketch. Now the uploading part is complete. So, if you focus on the Arduino board as you can see you have a reset button. So, it has already entered into a loop. So, we will push the reset button, so that the code execution starts from the very beginning. You see the green LED glows on it will glow for 5 seconds followed by the yellow one which

will again glow for 5 seconds and then, it is followed by the red one which glows for 5 seconds. All three turn off. It will wait for 10 seconds before going again into the loop that void traffic loop.

(Refer Slide Time: 22:44)



Now, same thing if you focus on the serial monitor, you can see green LEDs is on go green off yellow on wait. Now, yellow off red is on stop, right. So, now everything is off. So, it will be wait for 10 seconds.

Then, again the loop starts from the green LED. So, it is actually not required to have a serial communication for this automated traffic control signal, but for the sake of debugging, I actually prefer this thing. So, you can actually when you connected your system to PC, you can actually see which part of the loop your code or the hardware is executing. So, it helps you in debugging your code effectively. So, this is it for now, ok.

Thank you.