

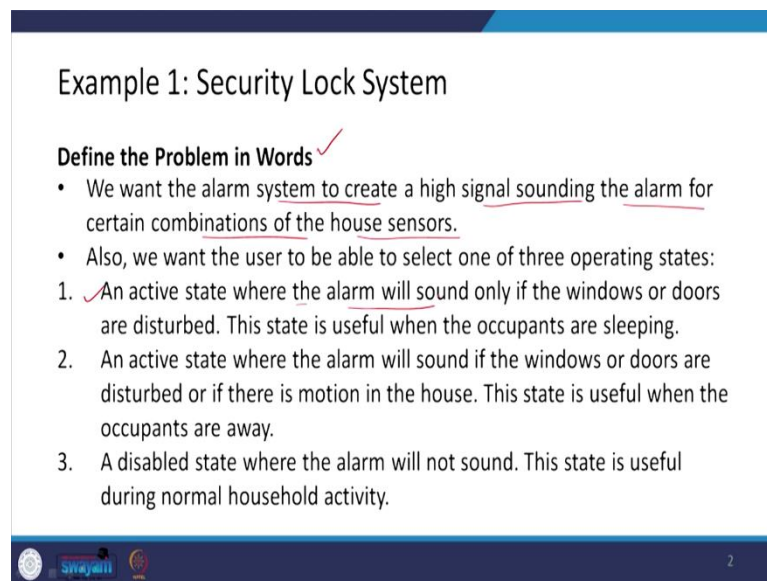
Mechatronics
Prof. Pushparaj Mani Pathak
Department of Mechanical and Industrial Engineering
Indian Institute of Technology, Roorkee

Lecture - 25
Microcontroller Programming Example

I welcome you all to NPTEL online certification course on Mechatronics. Today, we are going to see Microcontroller Programming Examples. In the last two classes, we have discussed the microprocessor and the microcontroller, their features and how do we connect them to the peripherals, all those things we have seen. In this class, I would like to take two examples of programming of the microcontroller.

The first example is on the security system, and the second example is on the snake robot. In the first example, we will try to do with the help of a PIC microcontroller, and in the second example, we will try to do with the help of an Arduino microcontroller.

(Refer Slide Time: 01:33)



Example 1: Security Lock System

Define the Problem in Words ✓

- We want the alarm system to create a high signal sounding the alarm for certain combinations of the house sensors.
- Also, we want the user to be able to select one of three operating states:
 1. ✓ An active state where the alarm will sound only if the windows or doors are disturbed. This state is useful when the occupants are sleeping.
 2. An active state where the alarm will sound if the windows or doors are disturbed or if there is motion in the house. This state is useful when the occupants are away.
 3. A disabled state where the alarm will not sound. This state is useful during normal household activity.

So, let us take the first example, the Security Lock System. This case, I have discussed while taking up the lecture on the logic circuits. In the next few slides, I will be repeating the same that is then I will be stating the problem statement.

The first stage is to define the problem in words as we are trying to solve the alarm system to create a high signal sounding the alarm for certain combinations of house sensors.

So, it is a house security lock system, and I have taken this example from the Mechatronics book. So, we want the user to be able to select one of the three operating states. The first state is an active state when the alarm will sound if the windows or doors are disturbed, and this state is useful when the occupants are sleeping. The second user state could be an active state, where the alarm will sound if the windows or doors are disturbed or if there is motion in the house and this state is useful when the occupants are away, and the third stage is a disabled state, where the alarm will sound, where the alarm will not sound, and this state is useful during the normal household activity.

(Refer Slide Time: 03:36)

• Boolean variables:

- A : state of the door and window sensors
- B : state of the motion detector
- Y : output used to sound the alarm
- CD : 2-bit code set by the user to select the operating state defined by

$$CD = \begin{cases} 01 & \text{operating state 1} \checkmark \\ 10 & \text{operating state 2} \checkmark \\ 00 & \text{operating state 3} \checkmark \end{cases}$$

The Boolean variables could be I define A - state of the door and window center sensors, B - state of the motion detector and C - output used to sound the alarm, and CD be the 2-bit code set by the user to select the operating state, defined by 0 1 is the operating state 1 and 1 0 is the operating state 2 and 0 0 is the operating state 3.

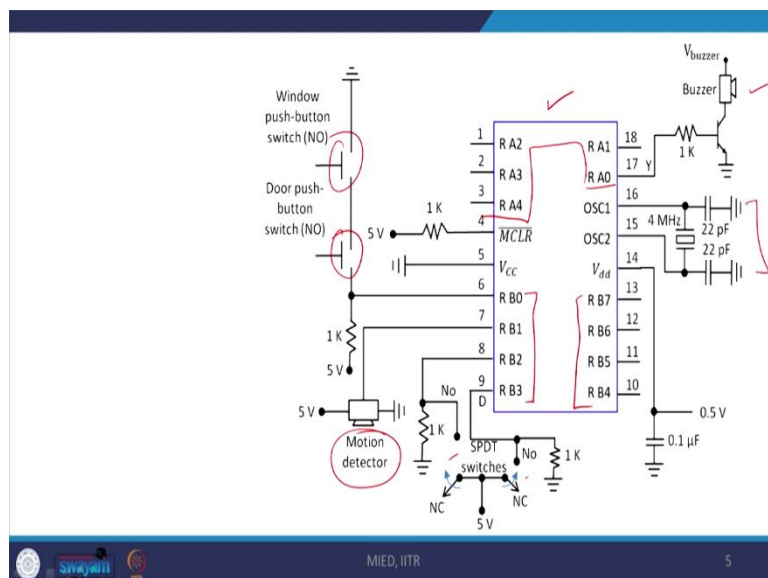
In the first stage, only the door and windows are disturbed, and in the second stage, door windows are disturbed as well as the motion is detected, and in the third space, nothing is happening that is normal state.

(Refer Slide Time: 04:25)

- The door and window sensors are assumed to be normally open switches that are closed when the door and window are closed.
- They are wired in series and connected to 5 V through a pull-up resistor; therefore, if either switch is open, then signal A will be high.
- Both the door and window must be closed for signal A to be low. This is called a wired-AND configuration because it is a hardwired solution providing the functionality of an AND gate.

So, the door and window sensors are assumed to be normally open switches that are closed when the doors and windows are closed. And they are wired in series and connected to 5 volt through a pull-up resistor; therefore, if either switch is open, then signal A will be high. Both the doors and windows must be closed for signal A to be low, and this is called wired-AND configuration because it is a hardwired solution providing the functionality of an AND gate.

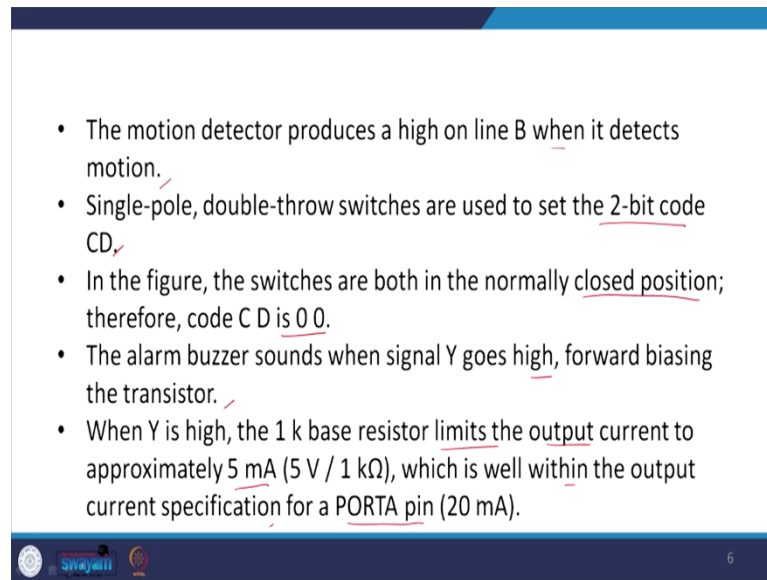
(Refer Slide Time: 05:06)



Here, we can see this is my pin layout for PIC 16F84, and the windows push button switch, this one; door push button switch, this one, and they are normally open. Here, we have the motion detector, and for selecting the operating state, we have an SPDT that is a Single

Pole Double Throw switch, which is being used over here and here, the buzzer is connected through R A0, and we have a crystal connected over here. So, these are the input ports B0 to B7, the eight ports over here, and R A0 to R A4. These are five ports. I have discussed a lot about these things in my previous lecture.

(Refer Slide Time: 06:26)



- The motion detector produces a high on line B when it detects motion.
- Single-pole, double-throw switches are used to set the 2-bit code CD.
- In the figure, the switches are both in the normally closed position; therefore, code CD is 0 0.
- The alarm buzzer sounds when signal Y goes high, forward biasing the transistor.
- When Y is high, the 1 k base resistor limits the output current to approximately 5 mA ($5\text{ V} / 1\text{ k}\Omega$), which is well within the output current specification for a PORTA pin (20 mA).

So, the motion detector produces a high on line B when it detects the motion, as I said this one. And the single-pole, double-throw switches are used to set the 2-bit code that is CD which I talked about. In the figure, the switches are both in the normally closed position; therefore, code CD is 0 0. The alarm buzzer sounds when signal Y goes high, forward biasing the transistor. And when Y is high, 1 k base resistor limits the output current to approximately 5 milli Amps, which is well within the output current specification of PORTA.

(Refer Slide Time: 07:28)

```

PicBasic Pro Program
/security.bas
/PicBasic Pro program to perform the control functions of the
security system
'Define variables for I/O port pins
door_or_window Var PORTB.0 'signal A
Motion VarPORTB.1 'signal B
c VarPORTB.2 'signal C
d VarPORTB.3 'signal D
alarm VarPORTA.0 'signal Y

```

If we look at the PicBasic Pro Program for this case, so, these are the comment statement the file comment statement over here. So, first, we define the variable for the input-output pin. So, you can that, so door or windows PORT 0 motion that is the VarPORT 1 here, the c VarPORTB.2, d is VarPORTB.3 and alarm is A.0.

So, this is how we are connecting this one. So, this is PORTB.0, and this is signal A, and likewise, we have other signals and here, DETECTED Con 1 to indicate that motion is detected.

(Refer Slide Time: 08:27)

```

'Define constants for use in IF comparisons
OPEN Con 1' to indicate that a door OR window is
open
DETECTED Con 1' to indicate that motion is detected
'Make sure the alarm is off to begin
with Low alarm alarm
' Main polling loop always:

```

And here, these are, as I said, the command statements.

(Refer Slide Time: 08:34)

```
If ((c == 0) And (d == 1)) Then 'operating state 1
(occupants sleeping) ✓
  If (door_or_window == OPEN) Then
    High alarm ✓
  Else ✓
    Low alarm ✓
  Endif ✓
Else ✓
```

So, our programming could be like this as I defined the c and d that are the condition states. So, if c is 0 And d is 1, then naturally, it that is the operating state 1, occupants are sleeping, and if door or windows are OPEN, Then it is alarm High else, it is a Low alarm.

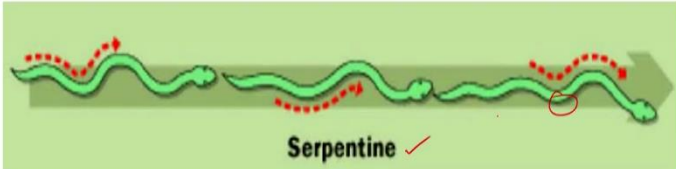
(Refer Slide Time: 09:14)

```
If ((c == 1) And (d == 0)) Then 'operating state 2 (occupants away)
  If ((door_or_window == OPEN) Or (motion == DETECTED)) Then
    High alarm ✓
  Else
    Low alarm ✓
  Endif
Else 'operating state 3 or NA (alarm ' disabled)
  Low alarm ✓
Endif
Endif
Goto always 'continue to poll the inputs
End
```

And so, we have the Endif statement to close the bracket or else, we have the second condition that is the second operating state that is if c is 1 and d is 0 that is the operating state 2 that is occupants are away, then here if door or windows are OPEN or the motion is DETECTED.

Next, let us take the example of Generation of a Snake Gait. The snakes usually have different gaits, or snakes follow the different gaits, or they may be using a combination of these gaits. So, these are serpentine or lateral undulation, Concertina, sidewinding, rectilinear progression. So, most snakes are capable of executing all or several of these forms of locomotion and generally switch as per requirement. In some situations, it may even use the combination of more than one gait.

(Refer Slide Time: 11:24)



The diagram shows a green snake moving from left to right, indicated by a large grey arrow. The snake's body is in an S-shape, with red dashed lines and arrows showing the lateral undulation of its body segments. Below the diagram, the word "Serpentine" is written in black with a red checkmark.


Serpentine ✓

- Serpentine:
- This S-shape movement, also known as lateral undulation, is used by most snakes on land and in water.
- On land, a snake usually finds resistance points in the surface - such as rocks, branches or dents and uses its scales to push on the points all at once, thrusting the snake forward.

13

Let us look at the first one that is the Serpentine gait, and here, as you can see, this is an S-shape movement, also known as the lateral undulation, and is used by most snakes on land and water. On land, snakes usually find resistance points in the surface - such as rocks, branches, or dents and use their scale to push on the points all at once, thrusting the snake to move them forward, as you can see over here.

(Refer Slide Time: 11:57)

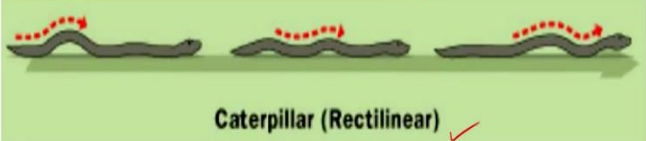


- Sidewinding ✓
- In environments with few resistance points, snakes may use a variation of serpentine motion to get around.
- Contracting their muscles and flinging their bodies, sidewinders create an S-shape that only has two points of contact with the ground; when they push off, they move laterally.
- Much of a sidewinding snake's body is off the ground while it moves. ✓

14

Then, there is another gait called Sidewinding. In the case of sidewinding, in an environment with few resistance points, snakes may use a variation of the serpentine motion to get around. Contracting their muscles and flinging their bodies, sidewinders create an S-shape that only has two points of contact with the ground, and when they push off, they move laterally. So, as you can see over here and much of the sidewinding snake's body is off the ground while it is moving.

(Refer Slide Time: 12:45)




- Caterpillar (rectilinear locomotion) ✓
- A much slower method of movement is caterpillar or rectilinear locomotion.
- This technique also contracts the body into curves, but these waves are much smaller and curve up and down rather than side to side.
- In this gait, the tops of each curve are lifted above the ground as the ventral scales on the bottoms push against the ground, creating a rippling effect. ✓

15

The next gait is the Caterpillar or the rectilinear gait. So, this is a much slower method of movement in Caterpillar or rectilinear locomotion, and I am going to talk about this gait. The example of this gait I am going to take, and this technique also contracts the body into the curves, but these waves are much smaller and curve up and down rather than side to side. And in this gait, the top of each curve is lifted above the ground as the ventral scales on the bottom push against the ground, creating a ripple effect.

(Refer Slide Time: 13:32)



- Concertina
- The snake extends its head and the front of its body along the vertical surface and then finds a place to grip with its ventral scales.
- To get a good hold, it bunches up the middle of its body into tight curves that grip the surface while it pulls its back end up; it then springs forward again to find a new place to grip with its scales.

The last gait which I am going to talk about is Concertina, and this concertina gait, the snake, extends its head and the front of its body along the vertical surface and finds a place to grip with its ventral scales. To get a good hold, it bunches up the middle of its body into the tight curves as you can see over here, that grips the surface while it pulls its back end up; it then springs forward again to find a new place to grip with its scales. So, this is how the concertina gait works.

(Refer Slide Time: 14:21)

Fabrication of Snake Robot



- Economy Standard size 4.5kgcm servo motor with plastic gears. ✓
- 'U' shaped brackets ✓

17

We made a snake robot using economy standards 4.5 kg centimeter servo motors and with plastic gears and we used U shape brackets to connect these motors.

(Refer Slide Time: 14:41)

Microcontroller Arduino

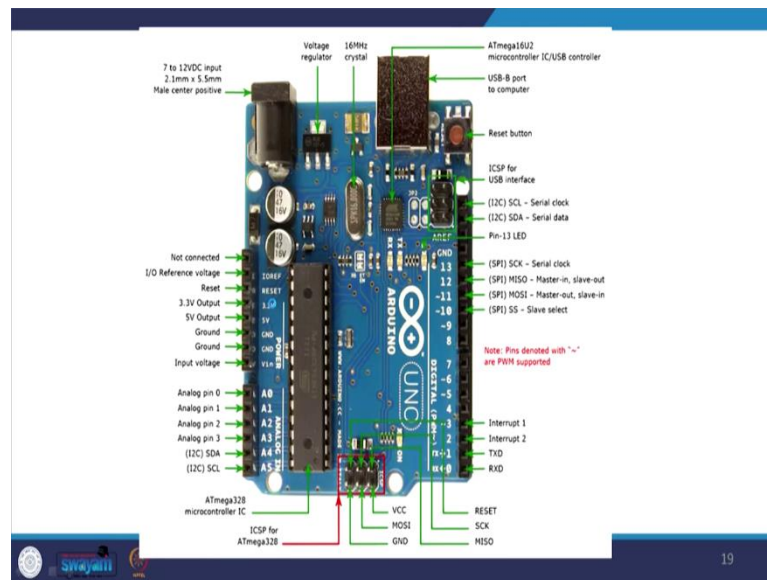
- The Uno is a microcontroller board based on the ATmega328P. ✓



18

The controller used is Arduino Uno, and this is a microcontroller board based on ATmega328P.

(Refer Slide Time: 14:55)



Here, the pin-out connections you can see that here you have A0 to A5; so, we have analog pins. Ports are over here, and here as I said, atmega microcontroller has been used over here.

(Refer Slide Time: 15:24)

- It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.
- With the help of Arduino positions of servomotor shafts are controlled.
- A code is written for the required gait and is initially verified for any errors.
- In absence of errors it is uploaded into the Arduino board and the motion of snake robot is controlled through it.

It has 14 digital input-output pins out of which 6 can be used as PWM output, 6 analog input, a 16 mega Hertz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button. And with the help of Arduino positions of servo motor shafts are controlled.

A code is written for the required gait and is initially verified for any errors. In the absence of error, it is uploaded into the Arduino board, and the motion of the snake robot is controlled through it. So, this is how the code for rectilinear gait is written.

(Refer Slide Time: 16:20)

```

Codes for Rectilinear

#include <Servo.h> // includes the library required to control
servomotor

#define omega 150 // degrees
#define delta 60 // phase difference in degrees

Servo s1; //defines 1st servomotor which sweeps in vertical
plane
//Servo s2; //defines 2nd servomotor which sweeps in
horizontal plane

```

We have to include the command to include library required to control the servomotor, define omega, define delta, some angle, and then, Servo s1; this defines 1st servomotor, which sweeps in the vertical plane, and these are put in command. So, these are not used in this particular gait.

(Refer Slide Time: 16:49)

```

Servo s3;
//Servo s4;
Servo s5;
//Servo s6;
Servo s7;
//Servo s8;
Servo s9;

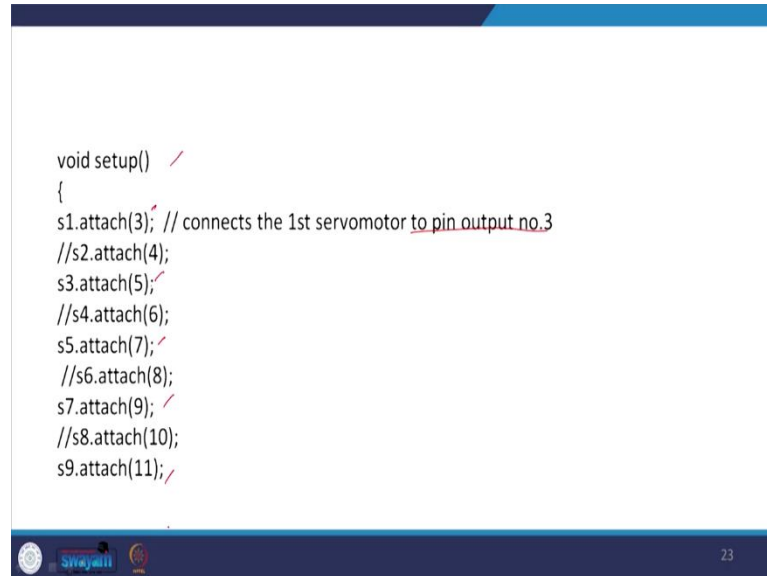
float th1; // position of 1st servo shaft
float th2;
float th3;
float th4;
float th5;
float th6;
float th7;
float th8;
float th9;
float t=-0.4/1000;

```

So, we have Servo s3, s5, s7, s9 over here, and the position of the servo shaft is put theta 1, theta 2, theta 3 to theta 9. A float t is defined over here.

(Refer Slide Time: 17:06)

```
void setup() ✓
{
  s1.attach(3); // connects the 1st servomotor to pin output no.3
  //s2.attach(4);
  s3.attach(5); ✓
  //s4.attach(6);
  s5.attach(7); ✓
  //s6.attach(8);
  s7.attach(9); ✓
  //s8.attach(10);
  s9.attach(11); ✓
}
```

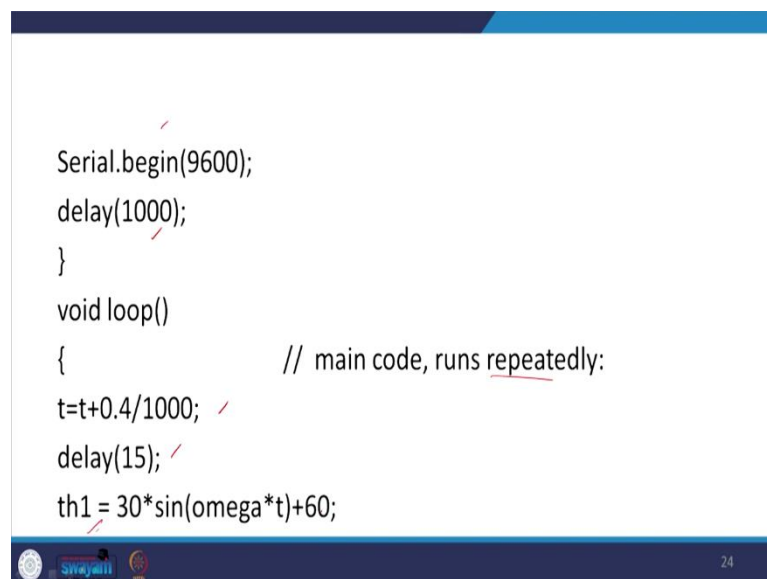


And then, the program starts.

The s1 attach 3, attach 5, attach 7, 9, and 11, and we are assigning 3, 5, 1, 3, 5, 7, and 9 connects 1st servomotor to pin output number 3. Similarly, for third to pin output number 5, 5 to output pin number 7, and so on.

(Refer Slide Time: 17:36)

```
Serial.begin(9600);
delay(1000);
}
void loop()
{
  // main code, runs repeatedly:
  t=t+0.4/1000; ✓
  delay(15); ✓
  th1 = 30*sin(omega*t)+60;
}
```



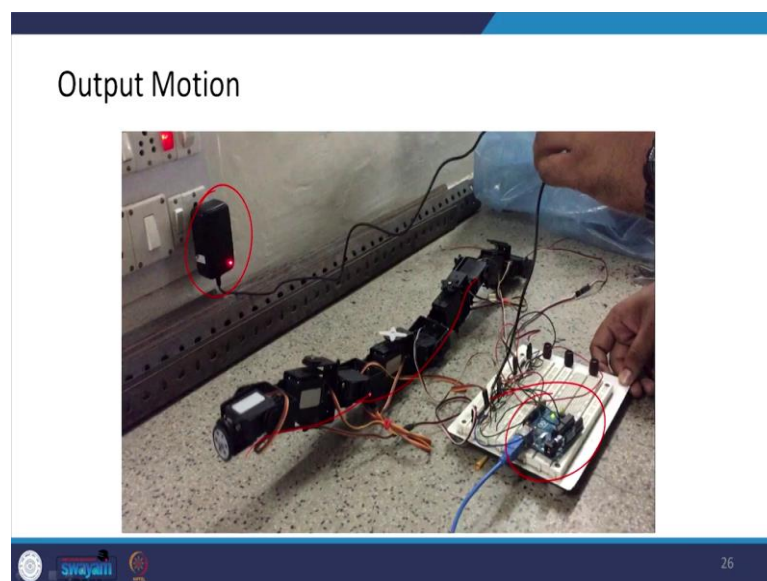
This is how servos are connected to the pins of that and then, serial we begin; then, we give a delay over here and then, the t is defined here, runs repeatedly. The main code runs, there is further a delay, and then theta one is defined like this.

(Refer Slide Time: 18:00)

```
s1.write(th1);  
th3 = 25*sin(omega*t+delta)+90;  
s3.write(th3);  
th5 = 20 *sin(omega*t+2*delta)+90;  
s5.write(th5);  
th7 = 25*sin(omega*t)+90;  
s7.write(th7);  
th9 = 30*sin(omega*t)+95;  
s9.write(th9);  
} }
```

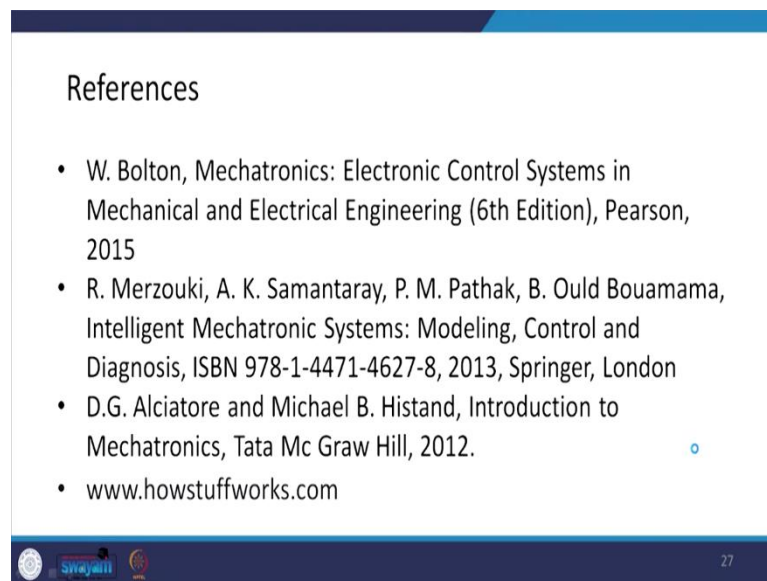
And then, we have s1 theta 3, s3 and theta 5, s5 theta 7, s7, and theta 9 and s9. So, this way, we calculate the angle and then, we give that command to the motor to rotate by that many degrees.

(Refer Slide Time: 18:22)



This is the output motion which you can see which has been obtained over here, and this is for the rectilinear motion which we have got, and this is the Arduino Uno board, and we have used a power supply for this, and of course, as I said, the motors are connected to the Arduino board, and by changing the program, likewise, we can get the other different gaits which I have talked to you either serpentine or sidewinding or Concertina. So, all those gaits can also be achieved.

(Refer Slide Time: 19:16)



These are the references. The first example I have taken from this book and yes and the gaits I have taken, description of the gaits, I have taken from HowStuffWorks dot com. The snake gait problem which I discussed was worked by one of the two of the interns in Robotics and Control Lab, IIT Roorkee.

Thank you.