In General

the orders is **(3 – 5 - 7 - 9 – 11 – 13 – 15) :** are 4 bits the first bit is the start bit for the communication frame the we send the other 3 bits .

Example : if we want to send 1 its binary is 001

We will add one at the first to start the communication channel

It will be 0011 so in decimal system it is 3

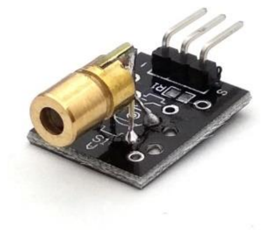
So **WriteMessage (3) ;**

Message	**Function refer to**
3	Move the motor forward
5	Move the motor backward
7	Stop all motors
9	Open the leds
11	close the leds
13	Move servo left
15	Move servo right

The laser system (KY-008)

Transmission circuit :

Laser Transmitter module for Arduino.



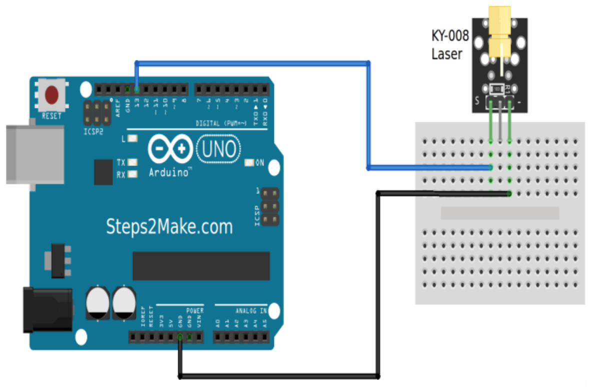
Connections :

- Connect signal (S) to pin 13 on the digital pins Arduino
- ground (-) to GND.

- Middle pin of laser module is not used.

The KY-008 Laser module consists of a 650 nm red laser diode head and a resistor. Handle with caution .

Specifications :



Operating Voltage	5 V
Output Power	5 mW
Wavelength	650 nm
Operating Current	less than 40 mA
Working Temperature	-10 °C ~ 40 °C
Dimensions	18.5 mm x 15 mm

- Dot shaped
- Red laser beam.

Transmission code :

Functions :

- **Void setup**
- **Void loop**
- **WriteMessage**
- **LaserFlash**

1. Void Setup :

we declare which the outputs and inputs pins

There are seven pins

- Laser pin   pin 13

  pinMode (laser , OUTPUT) ;

- Motor 1 : three pins 6 7 8

  2 pins directions and one for speed

  pinMode (in1 , OUTPUT) ;

  pinMode (in2 , OUTPUT) ;

  pinMode (enA , OUTPUT) ;

- Motor 2 : three pins 9 10 11

  2 pins directions and one for speed

  pinMode (in3 , OUTPUT) ;

  pinMode (in4 , OUTPUT) ;

  pinMode (enB , OUTPUT) ;

2. Void loop

To loop the all system .

we call our main function **WriteMessage**

WriteMessage (3) ;

Delay (5000) ;

WriteMessage (5) ;

Delay (5000) ;

WriteMessage (7) ;

Delay (5000) ;

WriteMessage (9) ;

Delay (5000) ;

WriteMessage (11) ;

Delay (5000) ;

WriteMessage (13) ;

Delay (5000) ;

WriteMessage (15) ;

Delay (5000) ;

3. WriteMessage :

This function return void “ nothing ” and take an argument integer refer to the number of order .

void WriteMessage (int num)

Then switch on the integer number and sends the orders bit by bit

switch (num)

true refer to 1

false refer to 0

<u>Write Message Code</u>

```
void WriteMessage ( int num ) {
 switch  ( num ) {
   case  3 :
     //code for 1 is  0011
     LaserFlash ( false ,  false ,  true , true ) ;
     Break ;



   case  5 :
     //code for 2 is 0101
     LaserFlash (  false ,  true ,  false , true ) ;
  Break ;

   case  7 :
     //code for 3 is 0111
     LaserFlash ( false ,  true ,  true , true ) ;
     Break ;

   case  9 :
     //code for 4 is 1001
```

```
          LaserFlash ( true ,  false ,  false , true ) ;
          Break ;

        case 11 :
          // code for 5 is 1011
         LaserFlash ( true ,  false ,  true , true ) ;
          Break ;

        case 13 :
          //code for 6 is 1101
          LaserFlash (  true , true ,  false , true ) ;
          Break ;

        case 15 :
          //code for 7 is  1111
         LaserFlash ( true , true , true , true ) ;
          Break ;

        default :
          Serial.println ( num ) ;
          Serial.println ( " not signal " ) ;
          Break ;  }
    }
```

In any case the application send any wrong message the default behavior is an error serial message will appear and just do nothing.

It breaks the function and returns to the main to continue the flow of the application code . Now , let’s back to the application code. Its the main function ,that calls the function WriteMessage and pass the number of the order

4. LaserFlash :

this function return void “ nothing “ and take four arguments from type Boolean refer to the 4 bits of the order massage .

then we check on this arguments
if 1 “ true “ then we put one on the laser pin

digitalWrite (laser , HIGH)

if 0 “ false “ then we put one on the laser pin

digitalWrite (laser , LOW)

. we start with the “ d “ bit . it is the start bit . then we wait for 80 millisecond to make everything clear .
Then we send the most bit “ a “ then “ b “ then “ c “
Example :
“ 0011 “ refer to “ abcd “

Laser Flash CODE

```
Void  LaserFlash (  Boolean  a ,  boolean  b ,  boolean  c   , Boolean  d ) {


   if  ( d = = true )  {

    Serial.println ( " start bit " ) ;
    digitalWrite ( laser , HIGH ) ;
    delay ( 80 ) ;  }


  if   ( a = = true ) {
    Serial.println ( " a high " ) ;
    digitalWrite ( laser , HIGH ) ;
     delay ( 80 ) ;


  }
  if  ( a = = false ) {
    Serial.println ( " a low " ) ;
    digitalWrite ( laser , LOW  ) ;
    delay ( 80 ) ;

  }

  if  ( b = = true ) {
    Serial.println ( " b high " ) ;
    digitalWrite ( laser , HIGH ) ;
    delay ( 80 ) ;
  }
   If  ( b = = false ) {
    Serial.println ( " b  low " ) ;
    digitalWrite ( laser ,  LOW ) ;
    delay ( 80 ) ;
  }



if  ( c = = false ) {
  Serial.println ( " c  low " ) ;
    digitalWrite ( laser , LOW ) ;
    delay ( 80 ) ;
  }

  if  ( c = = true ) {
    Serial.println ( " c high " ) ;
    digitalWrite ( l aser , HIGH ) ;
    delay ( 80 ) ;
  }
```

```
digitalWrite ( laser , LOW ) ; }
```

Conclusion of the sequence of the transmission code :

- the main function calls WriteMessage function and sends the message .
- The WriteMessage convert to the representation of the binary by passing the 0 ’s , 1 ’s to the LaserFlash function depended on the message number for just one time for each calling in the main function .
- The LaserFlash function take the binary bits and depended on 1 or 0 it open the led or not .
- After send all bits it waits for 10s before return to the main to allow us to recognize the message .

The receiver system

The Receiver Code Sequence :

- The receiver will wait for the first bit
- and will set a flag and wait (100) milliseconds to shift the read in the
- quarter of the second pulse or the second bit duration .
- The flag enter the while condition to serially save the received bits depended on the LDR states .
- Convert the single bits to binary using the equation .
- check if the incoming message in binary the same as the previous message .
- if it is a different message , calling the function order with the binary message argument .

at first we use the servo motor library

# include < Servo.h >

Functions :

- **void setup**
- **void loop**
- **order**
- **forward_motors**
- **back_motors**
- **stop_motors**
- **open_leds**
- **stop_leds**
- **left_servo**
- **right_servo**

1. void setup :

- **LDR sensor** : pin 13 as input

  pinMode (ldr , INPUT) ;

- **Leds** : pin 13 as output

  pinMode (led , OUTPUT) ;

- **Servo motor** : pin 5 as PWM output

  Servo1.attach (servoPin) ;

- **Motor 1** : three pins 6 7 8

  2 pins directions and one for speed

  pinMode (in1 , OUTPUT) ;

  pinMode (in2 , OUTPUT) ;

  pinMode (enA , OUTPUT) ;

- **Motor 2** : three pins 9 10 11

2 pins directions and one for speed

pinMode (in3 , OUTPUT) ;

pinMode (in4 , OUTPUT) ;

pinMode (enB , OUTPUT) ;

2. **Void loop :**

At first we receive the frame .

We wait for the start bit and wait the LDR sensor to read light (0 logic)

(digitalRead (ldr) = = 0) ;

If the LDR sensor read the start bit , the start_bit flag is set to logic 1 .

start_bit = 1 ;

then the code is pooling on the start_bit flag and receive the other bits . bit by bit

the read pulse is being shifted by 40 ms after receive the start bit so we can be sure the we read at the middle of the pulse signal .

then we clear the start bit flag so we can receive another order .

```
if ( ( start_bit = = 0 ) & & ( digitalRead ( ldr ) = = 0 ) )
{ delay ( 120 ) ;
start_bit = 1 ;
}

While ( start_bit ) {

  bit1 = ! ( digitalRead ( ldr ) ) ;
 delay ( 80 ) ;
  bit2 = ! ( digitalRead ( ldr ) ) ;
 delay ( 80 ) ;
  bit3 = ! ( digitalRead ( ldr ) ) ;
  start_bit = 0 ;
  }
```

After this , we collect the data from the received frame and get the result order

And convert it to a decimal shape

result = 100 * bit1 + 10 * bit2 + bit3 ;

as the void loop repeated in small time so we need to write the order just for the first one .

so we define an flag and check if it equal to the result variable or not .

if there are different order the the condition be true and the order send to the function

if not , nothing happen . I just repeated the void loop and check if the start bit come again .

```
if ( temp ! = result ) {
order ( result ) ;
Serial.println ( " result " ) ;
temp = result ;
}
```

3. Order :
   - ➢ This function return void “ nothing “ but it take an argument from the main function as a long type variable
   - ➢ calls the function depended on the received message and the received bits .
   - ➢ There are some functions to test the communication system between the two vehicle.

It switches on the message variable and calls the referred function .

```
void order ( long message ) {
  switch ( message ) {
    case 1 :
      forward_motors ( ) ;
      break ;
      case 10 :
      back_motors ( ) ;
      break ;
case 11 :
stop_motors ( ) ;
break ;
case 100 :
open_leds ( ) ;
break ;
case 101 :
stop_leds ( ) ;
break ;
case 110 :
left_servo ( ) ;
break ;
case 111 :
right_servo ( ) ;
break ;

default :
Serial.println ( message ) ;
Serial.println ( "  wrong signal " ) ;
Break ;  }
}
```

4. Forward motor :
   - This function return void “ nothing “ and take void argument .
   - This function is being called when the tx vehicle sends the 0011 order .
   - This function open the motors in one direction “ forward “ By put logic 1 on the one pin of the direction and 0 logic on the other pin of the direction by the digital function and put PWM on the pin of speed by the analog function.

```
void forward_motors ( void ) {
 Serial.println ( " enter forward motors " ) ;
digitalWrite ( in1 , HIGH ) ;
digitalWrite ( in2 ,  LOW ) ;
analogWrite ( enA , 120 ) ;

digitalWrite ( in3 , HIGH ) ;
digitalWrite ( in4 , LOW ) ;
analogWrite ( enB , 120 ) ;
}
```

5. Backward motors :
    - This function return void “ nothing “ and take void argument .
    - This function is being called when the tx vehicle sends the 0101 order .
    - This function open the motors in one direction “ backward “By put logic 1 on the one pin of the direction and 0 logic on the other pin of the direction by the digital function and put PWM on the pin of speed by the analog function.

```
void backward_motors ( void ) {
 Serial.println ( " enter backward motors " ) ;


digitalWrite ( in1 , LOW ) ;

digitalWrite ( in2 , HIGH ) ;

analogWrite ( enA , 120 ) ;


digitalWrite ( in3 , LOW ) ;

digitalWrite ( in4 , HIGH ) ;

analogWrite ( enB , 120 ) ;

}
```

6. Stop motors :
    - This function return void “ nothing “ and take void argument .
    - This function is being called when the tx vichel send the 0111 order .
    - This function stop the motors By put logic 0 on the two pins of the direction by the digital function and put PWM 0 on the pin of speed by the analog function.

```
void stop_motors ( void ) {

 Serial.println ( " enter stop motors " ) ;



digitalWrite ( in1 , LOW ) ;

digitalWrite ( in2 , LOW ) ;

analogWrite ( enA , 0 ) ;

digitalWrite ( in3 , LOW ) ;

digitalWrite ( in4 , LOW ) ;

analogWrite ( enB , 0 ) ;

}
```

7. Open leds :
    - This function return void “ nothing “ and take void argument .
    - This function is being called when the tx vehicle sends the 1001 order .
    - It open the leds by write logic 1 on the pin of leds using the digital function .

```
void open_leds ( void ) {

Serial.println ( " open leds " ) ;

digitalWrite ( led , HIGH ) ;

}
```

8. stop leds
   - This function return void “ nothing “ and take void argument .
   - This function is being called when the tx vehicle sends the 1011 order .
   - It open the leds by write logic 0 on the pin of leds using the digital function .

```
void stop_leds ( void ) {

Serial.println ( " close  leds " ) ;

digitalWrite ( led , LOW ) ;

}
```

9. Left servo :
   - This function return void “ nothing “ and take void argument .
   - This function is being called when the tx vehicle sends the 1101 order .
   - This function use the servo library to move the servo motor left , by passing a value refer to angle to the servo.write function .

```
void left_servo ( ) {
  Serial.println ( " servo left " ) ;


   Servo1.write ( 20 ) ;
                    }
```

10. Right servo :
    - This function return void “ nothing “ and take void argument .
    - This function is being called when the tx vehicle sends the 1111 order .
    - This function use the servo library to move the servo motor left , by passing a value refer to angle to the servo.write function .

```
void right_servo ( ) {
  Serial.println ( " servo right " ) ;


   Servo1.write ( 110 ) ;
```

}

system of leds

Transmission System Hardware

In this system , we use led to send the communication frame bits . Bit at time . Using the digital function we control on it to be opened and closed . when the code set the pin to 1 “ HIGH “ it is opened . when code clear the pin “ LOW “ it is closed .

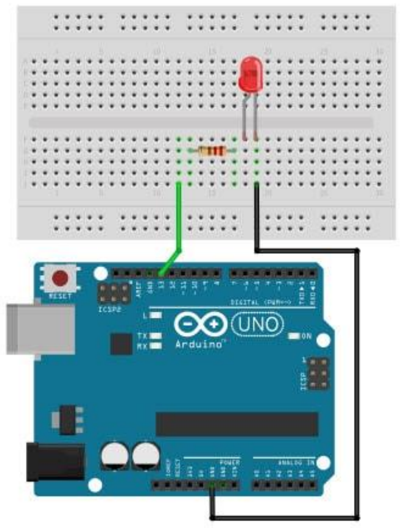
If we want to send 5 decimal , 0101 binary :

- Open the led Wait milliseconds
- Close the led Wait milliseconds

- Open the led Wait milliseconds
- Close the led Wait milliseconds

We should wait for milliseconds as the hardware need some time to response in the both hands , transmitter and receiver . we wait about 200 millisecnds .

We tested many delays but 200 milliseconds was the most sable situations .



Transmission code :

Functions :

- **Void setup**
- **Void loop**
- **WriteMessage**
- **LaserFlash**
- **Motors_Forward**
- **Motors_Backward**
- **Motors_Stop**

1.Void Setup :

we declare which the outputs and inputs pins

There are seven pins

- Led : pin 13

  pinMode (led , OUTPUT) ;

- Motor 1 : three pins 6 7 8

  2 pins directions and one for speed

  pinMode (in1 , OUTPUT) ;

  pinMode (in2 , OUTPUT) ;

  pinMode (enA , OUTPUT) ;

- Motor 2 : three pins 9 10 11

  2 pins directions and one for speed

```
pinMode ( in3 ,  OUTPUT ) ;
pinMode ( in4 ,  OUTPUT ) ;
pinMode ( enB ,  OUTPUT ) ;
```

- 2 Line follower sensor : 2 pins for signal 2 , 3

```
pinMode ( left , INPUT ) ;
pinMode ( right , INPUT ) ;
```

2.Void loop

To loop the all system .

we call our main function WriteMessage , with the master vichel order the slave vichel to open the motors forward or backward the master vichel should also move forward or backward .

```
void loop ( ) {
  WriteMessage ( 3 ) ;
  Motors_Forward ( ) ;
  Delay ( 2000 ) ;

  WriteMessage ( 5 ) ;
 Motors_Backward ( ) ;
  Delay (  2000 ) ;

 WriteMessage  ( 9 ) ;
  Delay ( 2000 ) ;

  WriteMessage ( 11 ) ;
  Delay ( 2000 ) ;

 WriteMessage ( 13 ) ;
  Delay ( 2000 ) ;
  WriteMessage ( 15 ) ;
  Delay ( 5000 ) ;
 }
```

3.WriteMessage :

This function return void “ nothing ” and take an argument integer refer to the number of order .

void WriteMessage (int num)

Then switch on the integer number and send the orders bit by bit

switch (num) ;

- true refer to 1
- false refer to 0

<u>Write Message Code</u>

```
void WriteMessage ( int num ) {
 switch  ( num ) {
  case  3 :
   //code for 1 is  0011
   LedFlash ( false ,  false ,  true , true ) ;
   Break ;

  case  5 :
   //code for 2 is 0101
   LedFlash (  false ,  true ,  false , true ) ;
  Break ;

  case  7 :
   //code for 3 is 0111
```

```
      LedFlash ( false ,  true ,  true , true ) ;
      Break ;

    case  9 :
      //code for 4 is 1001
      LedFlash ( true ,  false ,  false , true ) ;
      Break ;

    case 11 :
      // code for 5 is 1011
     LedFlash ( true ,  false ,  true , true ) ;
      Break ;

    case 13 :
      //code for 6 is 1101
      LedFlash (  true , true ,  false , true ) ;
      Break ;

    case 15 :
      //code for 7 is  1111
     LedFlash ( true , true , true , true ) ;
      Break ;

    default :
      Serial.println ( num ) ;
      Serial.println ( " not signal " ) ;
      Break ;  }
  }
```

In any case the application send any wrong message the default behavior is an error serial message will appear and just do nothing.

It breaks the function and returns to the main to continue the flow of the application code . Now , let's back to the application code. Its the main function ,that calls the function WriteMessage and pass the number of the order

4.LedFlash :

this function return void " nothing " and take four arguments from type Boolean refer to the 4 bits of the order massage .

then we check on this arguments
if 1 " true " then we put one on the laser pin

digitalWrite (led , HIGH) ;

if 0 " false " then we put one on the laser pin

digitalWrite (led , LOW) ;

. we start with the " d " bit . it is the start bit . then we wait for 80 millisecond to make everything clear .
Then we send the most bit " a " then " b " then " c "
Example :
" 0011 " refer to " abcd "

<u>Led Flash CODE</u>

```
Void  LedFlash (  Boolean  a ,  boolean  b ,  boolean  c   , Boolean  d ) {

  if  ( d = = true )  {

    Serial.println ( " start bit " ) ;
    digitalWrite ( led , HIGH ) ;
    delay ( 200 ) ;  }


 if   ( a = = true ) {
    Serial.println ( " a high " ) ;
    digitalWrite ( led , HIGH ) ;
     delay ( 200 ) ;
       }

  if  ( a = = false ) {
    Serial.println ( " a low " ) ;
    digitalWrite ( led , LOW  ) ;
    delay (200 )  ;
  }

  if  ( b = = true ) {
    Serial.println ( " b high " ) ;
    digitalWrite ( led , HIGH ) ;
    delay ( 200 ) ;
  }
   If  ( b = = false ) {
    Serial.println ( " b  low " ) ;
    digitalWrite ( led ,  LOW ) ;
    delay ( 200 ) ;
  }

if  ( c = = false ) {
  Serial.println ( " c  low " ) ;
    digitalWrite ( led , LOW ) ;
    delay ( 200 ) ;
  }
   if  ( c = = true ) {
    Serial.println ( " c high " ) ;
    digitalWrite ( led , HIGH ) ;
    delay ( 200 ) ;
  }
```

```
digitalWrite ( led , LOW ) ; }
```

5. **Motors_Forward :**
   - This function return void " nothing " and take void argument .
   - This function we call it just when the master vehicle sends 3 to the slave " 0011 " to move forward
   - This function open the motors of the master vehicle in one direction " forward " By put logic 1 on the one pin of the direction and 0 logic on the other pin of the direction by the digital function and put PWM on the pin of speed by the analog function.
   - This function has counter to terminal it and order the slave to stop .
   - This function use line follower sensors to move in line . by reading the digital value of the left sensor and the right sensor .

```
void Motors_Forward ( void ) {
Serial.println ( " enter forward motors " ) ;
 analogWrite ( enA , 130 ) ;
 analogWrite ( enB , 130 ) ;
while ( wait < = 400 ) {
if ( ( digitalRead ( left ) )  & &  ( digitalRead ( right ) ) ) {
Serial.println ( " forward " ) ;
 digitalWrite ( in1 ,  LOW ) ;
digitalWrite ( in2 ,  HIGH ) ;
digitalWrite ( in3 , LOW ) ;
digitalWrite ( in4 , HIGH ) ; }
 else if ( ( digitalRead ( left ) )  & & ! ( digitalRead ( right ) ) ) {
Serial.println ( " right " ) ;
digitalWrite ( in1 , LOW ) ;
digitalWrite ( in2 ,  HIGH ) ;

digitalWrite ( in3 , LOW ) ;

digitalWrite ( in4 , LOW ) ;

}

else if ( ! ( digitalRead ( left ) ) & & ( digitalRead ( right ) ) ) {

Serial.println ( " left " ) ;

digitalWrite ( in1 ,  LOW ) ;

digitalWrite ( in2 ,  LOW ) ;

//left

digitalWrite ( in3 , LOW ) ;

digitalWrite ( in4 , HIGH ) ;

}

else {

  Motors_Stop ( ) ;

}

  Wait + + ; }

WriteMessage ( 7 ) ;

Motors_Stop ( ) ;

Wait = 0 ; }
```

6. **Motors_Backward :**
   - This function return void “ nothing “ and take void argument .
   - This function we call it just when the master vehicle sends 5 to the slave “ 1011 “ to move backward

- This function open the motors of the master vehicle in one direction " backward" By put logic 1 on the one pin of the direction and 0 logic on the other pin of the direction by the digital function and put PWM on the pin of speed by the analog function.
- This function has counter to terminal it and order the slave to stop .
- This function use line follower sensors to move in line . by reading the digital value of the left sensor and the right sensor .

```
void Motors_Backward ( void ) {
Serial.println ( " enter backward motors " ) ;
 analogWrite ( enA , 130 ) ;
 analogWrite ( enB , 130 ) ;
while ( wait < = 400 ) {
if ( ( digitalRead ( left ) )  & &  ( digitalRead ( right ) ) ) {
Serial.println ( " backward " ) ;
 digitalWrite ( in1 ,  HIGH ) ;
digitalWrite ( in2 ,  LOW ) ;
digitalWrite ( in3 , HIGH ) ;
digitalWrite ( in4 , LOW ) ; }
 else if ( ( digitalRead ( left ) )  & & ! ( digitalRead ( right ) ) ) {
Serial.println ( " right " ) ;
digitalWrite ( in1 , HIGH ) ;
digitalWrite ( in2 ,  LOW ) ;
digitalWrite ( in3 , LOW ) ;
digitalWrite ( in4 , LOW ) ;
}
else if ( ! ( digitalRead ( left ) ) & &  ( digitalRead ( right ) ) ) {
Serial.println ( " left " ) ;
DigitalWrite ( in1 ,  LOW ) ;
DigitalWrite ( in2 , LOW ) ;
//left
DigitalWrite ( in3 , HIGH ) ;
DigitalWrite ( in4 , LOW )  ;
}
else {
  Motors_Stop ( ) ;
}
  Wait + + ; }
WriteMessage ( 7 ) ;
Motors_Stop ( ) ;
Wait = 0 ; }
```

7. **Motors_Stop :**

   This function makes all motors stop .

```
  digitalWrite ( in1 , LOW ) ;
digitalWrite ( in2 ,  LOW ) ;
DigitalWrite ( in3 , LOW ) ;
DigitalWrite  ( in4 , LOW ) ;
```

Conclusion of the sequence of the transmission code :

➢ the main function calls WriteMessage function and sends the message .

- The WriteMessage convert to the representation of the binary by passing the 0 ’s , 1 ’s to the LedFlash function depended on the message number for just one time for each calling in the main function .
- The LedFlash function take the binary bits and depended on 1 or 0 it open the led or not .
- After send each bit it waits for 5s before return to the main to allow us to recognize the message .

Receiver code :

at first we use the servo motor library

# include < Servo.h >

We define some variables

long result = 0 ;

long temp = 0 ;

int start_bit = 0 ;

int bit1= 0 ;

int bit2 = 0 ;

int bit3 = 0 ;

int flag_f = 0 ;

int test = 0 ;

Functions :

- **void setup**
- **void loop**
- **order**
- **forward_motors**
- **back_motors**
- **stop_motors**

- **open_leds**
- **stop_leds**
- **left_servo**
- **right_servo**

1. void setup :

- **LDR sensor** : pin 13 as input

  pinMode (ldr , INPUT) ;

- **Leds** : pin 12 as output

  pinMode (led , OUTPUT) ;

- **Servo motor** : pin 5 as PWM output

  Servo1.attach (servoPin) ;

- **Motor 1** : three pins 6 7 8

  2 pins directions and one for speed

  pinMode (in1 , OUTPUT) ;

  pinMode (in2 , OUTPUT) ;

  pinMode (enA , OUTPUT) ;

- **Motor 2** : three pins 9 10 11

  2 pins directions and one for speed

  pinMode (in3 , OUTPUT) ;

  pinMode (in4 , OUTPUT) ;

  pinMode (enB , OUTPUT) ;

- **Line follower sensor :** two pins 2 3

  pinMode (left , INPUT) ;

  pinMode (right , INPUT) ;

2. Void loop :

At first we receive the frame .

We wait for the start bit and wait the LDR sensor to read light (0 logic) or wait the flag to be 1

(digitalRead (ldr) = = 0) | | (flag = = 1)

If the LDR sensor read the start bit , then start_bit is set to logic 1 and the variable flag clear.

start_bit = 1 ;

flag = 0 ;

then the code is pooling on the start_bit flag and receive the other bits . bit by bit
the read pulse is being shifted by 50 ms after receive the start bit so we can be sure the we read at the middle of the pulse signal .
then we clear the start bit flag so we can receive another order .

```
if ( ( start_bit = = 0 ) & & ( ( digitalRead ( ldr ) = = 0 ) | | ( flag = = 1 ) )
{ delay ( 250 ) ;
start_bit = 1 ;
flag = 0 ;
}

While ( start_bit ) {

  bit1 = ! ( digitalRead ( ldr ) ) ;
 delay ( 200 ) ;
  bit2 = ! ( digitalRead ( ldr ) ) ;
 delay ( 200 ) ;
  bit3 = ! ( digitalRead ( ldr ) ) ;
  start_bit = 0 ;
  }
```

After this , we collect the data from the received frame and get the result order
And convert it to a decimal shape

result = 100 * bit1 + 10 * bit2 + bit3 ;

as the void loop repeated in small time so we need to write the order just for the first one . so we define an flag and check if it equal to the result variable or not .

if there are different order the condition be true and the order send to the function

if not , nothing happen . I just repeated the void loop and check if the start bit come again .

```
if ( temp ! = result ) {
order ( result ) ;
Serial.println ( " result " ) ;
temp = result ;
}
```

3. Order :

   - This function return void " nothing " but it take an argument from the main function as a long type variable
   - call the function depended on the received message and the received bits .
   - There are some functions to test the communication system between the two vehicle.

It switch on the message variable and call the referred function .

```
void order ( long message ) {
 switch ( message ) {
  case 1 :
   forward_motors ( ) ;
   break ;
   case 10 :
   back_motors ( ) ;
   break ;
   case 11 :
   stop_motors ( ) ;
   break ;
   case 100 :
   open_leds ( ) ;
break ;
case 101 :
stop_leds ( ) ;
break ;
case 110 :
left_servo ( ) ;
break ;
case 111 :
right_servo ( ) ;
break ;

default :
Serial.println ( message ) ;
Serial.println ( "  wrong signal " ) ;
Break ;  }
}
```

4. Forward motor :
   - This function return " int " 0 or 1 and take void argument .
   - This function is being called when the tx vehicle send the 0011 order .
   - This function open the motors in one direction " forward " By put logic 1 on the one pin of the direction and 0 logic on the other pin of the direction by the digital function and put PWM on the pin of speed by the analog function.
   - This function use line follower sensors to move in line . by reading the digital value of the left sensor and the right sensor .
   - This function check the line and follow it and check if a new start bit come .
   - If new start bit come the function terminated and go back to the main .

There is a problem in this function , the function is faster than the update of the sensor value . so when we call the function the value of the LDR sensor is 0 this equal to the start bit value and this mean the value of the sensor on the registers didn't be updated . It takes some time .

So we wait the first change to 1 and the we wait the 0 signal " the start bit "

```
if ( digitalRead ( ldr ) = = 1 )
{ test_noise = 1 ; }
```

Then we check on the test_noise flag if 1 and the LDR has a new start bit then we terminate the function and set the flag and clear teat_noise

```
If ( ( test_noise = = 1 ) & &  ( digitalRead ( ldr ) = = 0 ) ) {
Test_noise =0;
Serial.println ( " FLAG " ) ;
Flag = 1 ;
return 1 ;
}
```

<u>The all code :</u>

```
Int  forward_motors ( void ) {
 Serial.println ( " enter forward motors " ) ;
  analogWrite ( enA , 90 ) ;
 analogWrite ( enB , 100 ) ;

while ( ( digitalRead ( left ) ) | | ( digitalRead ( right ) ) ) {
 if ( digitalRead ( ldr ) = = 1 )
 { test_noise = 1 ; }
Serial.println ( " forward black " ) ;

If ( ( digitalRead ( left ) ) & & ! ( digitalRead ( right ) ) ) {
Serial.println ( " right " ) ;
digitalWrite ( in1 , LOW ) ;
digitalWrite ( in2 , HIGH ) ;

digitalWrite ( in3 , LOW ) ;
digitalWrite ( in4 , LOW ) ;
}

else if ( ! ( digitalRead ( left ) ) & & ( digitalRead ( right ) ) ) {
Serial.println ( " left " ) ;
digitalWrite ( in1 , LOW ) ;
digitalWrite ( in2 , LOW ) ;

digitalWrite ( in3 , HIGH ) ;
digitalWrite ( in4 , LOW ) ;
}
Else{
digitalWrite ( in1 , HIGH ) ;
digitalWrite ( in2 ,  LOW ) ;

digitalWrite ( in3 , HIGH ) ;
digitalWrite ( in4 , LOW ) ;
}
If ( ( test_noise = = 1 ) & &  ( digitalRead ( ldr ) = = 0 ) ) {
Test_noise =0;
     Serial.println ( " FLAG " ) ;
   Flag = 1 ;
    return 1 ;
  }
}
  return 0 ;
}
```

5. Backward motors :
    - This function return int 0 ,1 and take void argument .
    - This function is being called when the tx vehicle sends the 0101 order .

- This function open the motors in one direction “ backward “By put logic 1 on the one pin of the direction and 0 logic on the other pin of the direction by the digital function and put PWM on the pin of speed by the analog function.

```
Int  backward_motors ( void ) {
 Serial.println ( " enter backward motors " ) ;
  analogWrite ( enA , 90 ) ;
 analogWrite ( enB , 100 ) ;


while ( ( digitalRead ( left ) ) | | ( digitalRead ( right ) ) ) {
 if ( digitalRead ( ldr ) = = 1 )
 { test = 1 ; }
Serial.println ( " backward black " ) ;


If ( ( digitalRead ( left ) ) & & ! ( digitalRead ( right ) ) ) {
Serial.println ( " right " ) ;
digitalWrite ( in1 , HIGH ) ;
digitalWrite ( in2 , LOW ) ;


digitalWrite ( in3 , LOW ) ;
digitalWrite ( in4 , LOW ) ;
}


else if ( ! ( digitalRead ( left ) ) & & ( digitalRead ( right ) ) ) {
Serial.println ( " left " ) ;
digitalWrite ( in1 , LOW ) ;

digitalWrite ( in2 , LOW ) ;


digitalWrite ( in3 , LOW ) ;

digitalWrite ( in4 , HIGH ) ;

}

Else{

digitalWrite ( in1 ,  LOW ) ;

digitalWrite ( in2 ,  HIGH ) ;


digitalWrite ( in3 , LOW ) ;

digitalWrite ( in4 ,HIGH ) ;

}

If ( ( test = = 1 ) & &  ( digitalRead ( ldr ) = = 0 ) ) {

     Serial.println ( " FLAG " ) ;

Test = 0 ;

   Flag = 1 ;

    return 1 ;

  }

}

  return 0 ;

}
```

6. Stop motors :

- This function return void “ nothing “ and take void argument .
- This function is being called when the tx vehicle sends the 0111 order .
- This function stop the motors By put logic 0 on the two pins of the direction by the digital function and put PWM 0 on the pin of speed by the analog function.

```
void stop_motors ( void ) {
 Serial.println ( " enter stop motors " ) ;
digitalWrite ( in1 , LOW ) ;
digitalWrite ( in2 , LOW ) ;
analogWrite ( enA , 0 ) ;
digitalWrite ( in3 , LOW ) ;
digitalWrite ( in4 , LOW ) ;
analogWrite ( enB , 0 ) ;
}
```

7. Open leds :

- This function return void “ nothing “ and take void argument .
- This function is being called when the tx vehicle sends the 1001 order .
- It open the leds by write logic 1 on the pin of leds using the digital function .

```
void open_leds ( void ) {
Serial.println ( " open leds " ) ;
digitalWrite ( led , HIGH ) ;
}
```

8. stop leds

- This function return void “ nothing “ and take void argument .

- This function is being called when the tx vehicle sends the 1011 order .
- It open the leds by write logic 0 on the pin of leds using the digital function .

```
void stop_leds ( void ) {

Serial.println ( " close  leds " ) ;

digitalWrite ( led , LOW ) ;

}
```

9. Left servo :

- This function return void “ nothing “ and take void argument .
- This function is being called when the tx vehicle sends the 1101 order .
- This function use the servo library to move the servo motor left , by passing a value refer to angle to the sevo.write function .

```
void left_servo ( ) {
  Serial.println ( " servo left " ) ;


   Servo1.write ( 20 ) ;
                    }
```

10. Right servo :

- This function return void “ nothing “ and take void argument .
- This function is being called when the tx vehicle sends the 1111 order .
- This function use the servo library to move the servo motor left , by passing a value refer to angle to the servo.write function .

```
void right_servo ( ) {
  Serial.println ( " servo right " ) ;


   Servo1.write ( 110 ) ;
                    }
```