Lab no 06: Hand Gesture Recognition Prototype

Bluetooth and Interface with Accelerometer and led Matrix Display

The purpose of this Lab is to learn interfaces with Bluetooth module, accelerometer, and LED matrix display. To do that, we are going to design an Arduino Hand Gesture Recognition project using an accelerometer sensor for movement recognition.

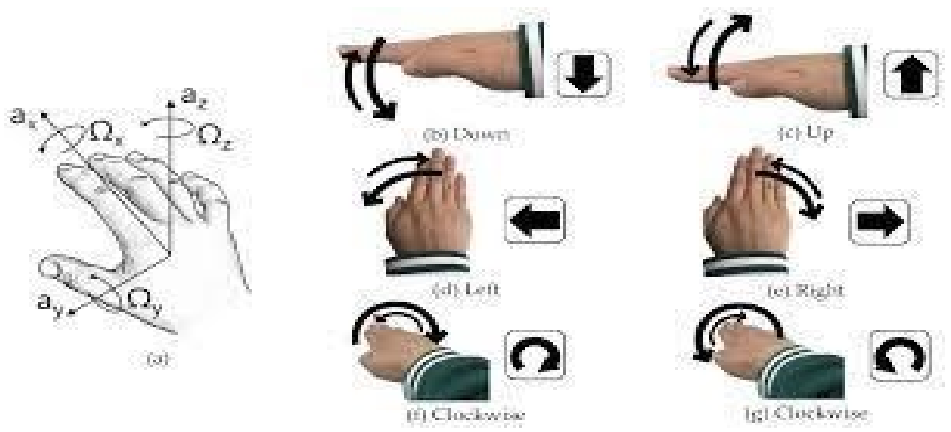
Parts: -

1. Concepts of Hand Gesture Recognition.
2. Understanding the Circuit and its Components
   a) Accelerometer Sensor.
   b) LED Matrix Display.
   c) Bluetooth Module.
3. Hand Gesture Recognition Code.

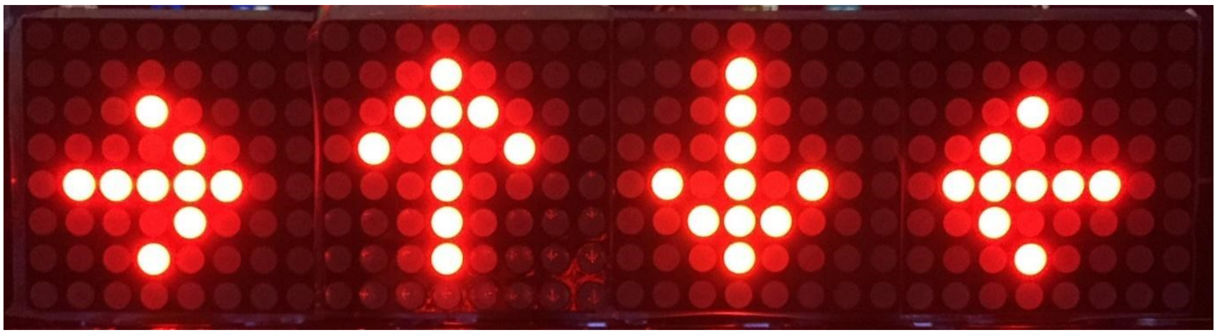
Part 1. Concepts of Hand Gesture Recognition

Hand gesture recognition (HGR) is a natural way of Human-Machine Interaction and has been applied in different areas. In this lab, the prototype of hand gesture recognition is divided into two sets:

Set 1: detects the movement of the hand as shown in the figure below (Up, Down, Right, Left) using an accelerometer. Then it sends the movement notation to **Set 2** through Bluetooth.



Set 2: receives the movements notation through Bluetooth and draws (up, down, right, left) arrow on LED matrix display, as shown in the figure below.

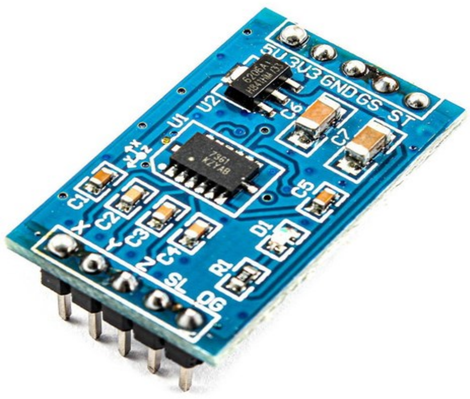


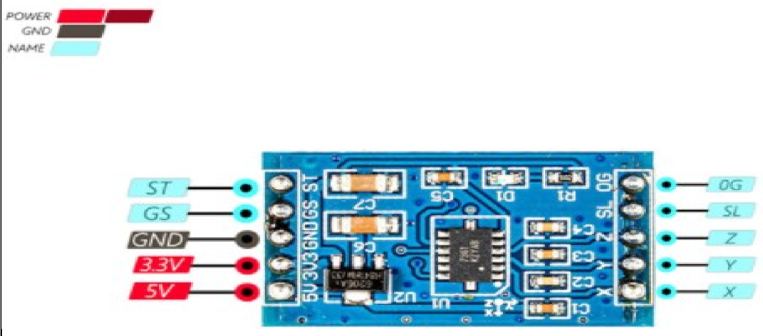
Part 2. Understanding the circuit and its components

Part 2. a) Accelerometer Sensor

The MMA7361 is a 3-axis accelerometer with a ± 1.5g to ± 6g measuring range. You do not need a library and complex code to use the MMA7361 accelerometer.

In this module, there is an analog output voltage for each axis acceleration. As a result, you can read and analyze the sensor's outputs using an analog-to-digital converter.





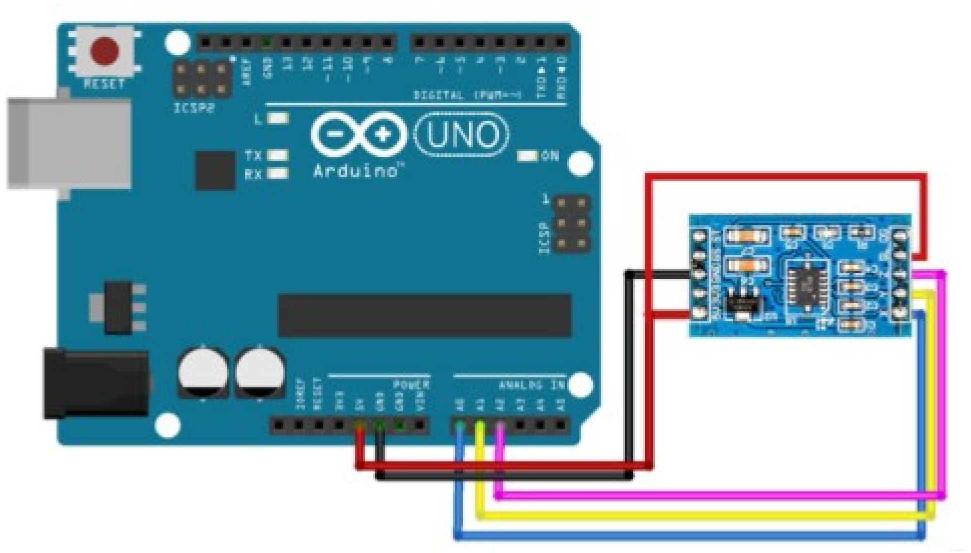
➢ Accelerometer Sensor Pinout

<u>10 Pinouts as follows:</u>

1. **5V:** Module power supply 5 V
2. **3V3:** Module power supply: 3.3V
3. **GND:** Ground
4. **GS:** Specifies the module working mode. If "0", the range is ± 1.5g, and if "1", the range is ±6g.
5. **ST:** To test the module automatically
6. **X:** X-axis acceleration output
7. **Y:** Y-axis acceleration output
8. **Z:** Z-axis acceleration output
9. **SL:** Sleep
10. **0G:** Detects zero acceleration on all axes. This pin is used to detect free fall.

➢ Accelerometer Circuit: -

The following circuit shows how to connect Arduino to the MMA7361 module. Connect wires accordingly.

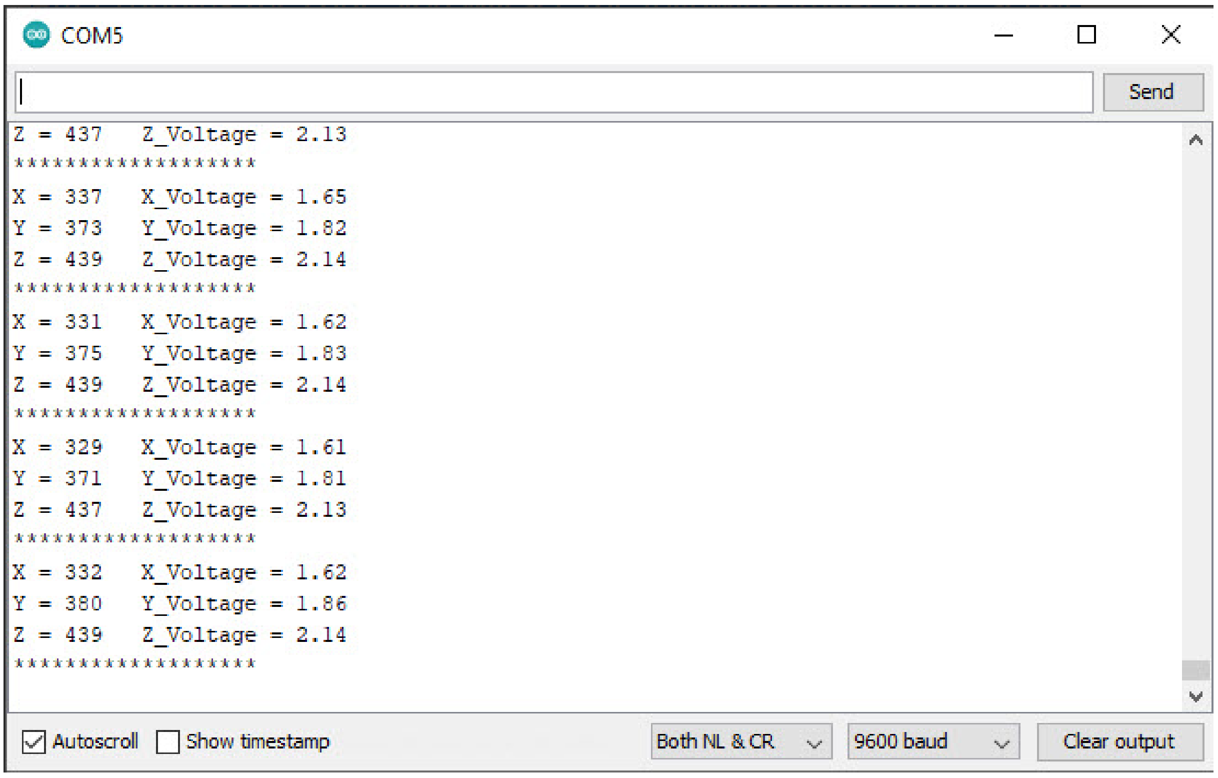


➢ Accelerometer Code: -

Upload the following code to Arduino. This code displays the sensor readings in the serial monitor.

```
int x = 0;
int y = 0;
int z = 0;

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

void loop()
{
  x = analogRead(A0);
  y = analogRead(A1);
  z = analogRead(A2);
  Serial.print("X = ");
  Serial.print(x);
  Serial.print("   X_Voltage = ");
  Serial.println(x*5.0/1024.0);
  Serial.print("Y = ");
  Serial.print(y);
  Serial.print("    Y_Voltage = ");
  Serial.println(float(y)*5.0/1024.0);

  Serial.print("Z = ");
  Serial.print(z);
  Serial.print("    Z_Voltage = ");
  Serial.println(float(z)*5.0/1024.0);

  Serial.println("******************");

  delay(1000);
}
```

After running the code, you will see the following image in the serial output

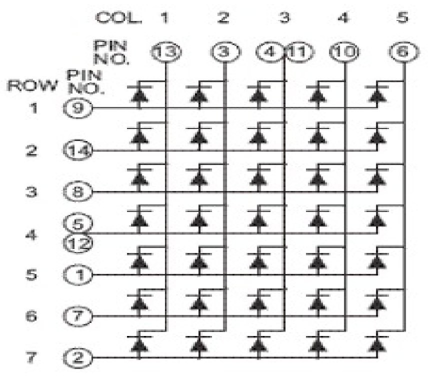


```
Z = 437   Z_Voltage = 2.13
******************
X = 337   X_Voltage = 1.65
Y = 373   Y_Voltage = 1.82
Z = 439   Z_Voltage = 2.14
******************
X = 331   X_Voltage = 1.62
Y = 375   Y_Voltage = 1.83
Z = 439   Z_Voltage = 2.14
******************
X = 329   X_Voltage = 1.61
Y = 371   Y_Voltage = 1.81
Z = 437   Z_Voltage = 2.13
******************
X = 332   X_Voltage = 1.62
Y = 380   Y_Voltage = 1.86
Z = 439   Z_Voltage = 2.14
******************
```

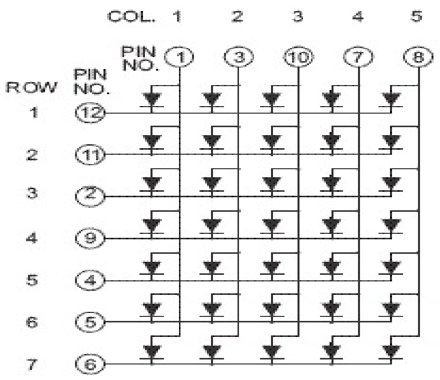
Part 2. b) LED Matrix Display

The LED matrix is an array of multiple LEDs arranged in a row by column order. Arrangement of the LEDs in the matrix pattern is made in either of the **two ways:**
row anode – column cathode or
row cathode – column anode.
The matrix pattern minimizes the number of pins required to drive them.

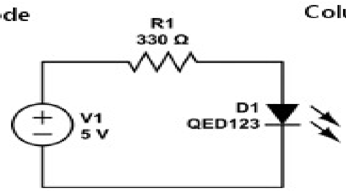
In this sample in the figure above, 35 LEDs are controlled using a combination of 12 pins. Desired character or graphics can be displayed by switching ON/OFF a desired configuration of LEDs, the perfection depends on the number of LEDs on the dot matrix. To light up an LED make the corresponding cathode low and anode high. The quick scanning of rows and columns makes the data visible to our eyes, i.e. persistence of vision.



Columns sharing a common-cathode

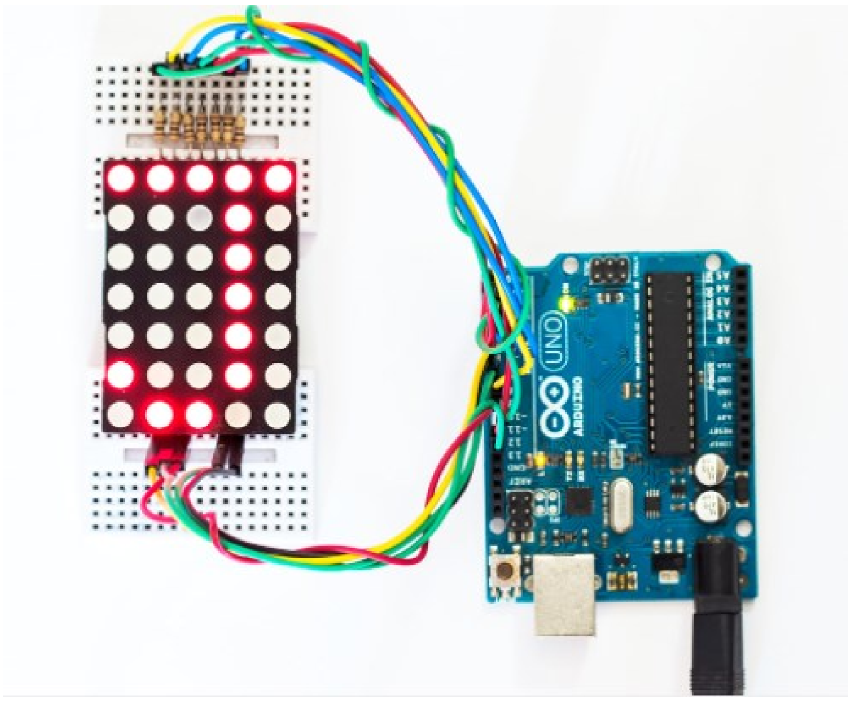


Columns sharing a common-anode



Basic circuit for lighting a led

➢ LED Matrix Display Hardware Connection: -



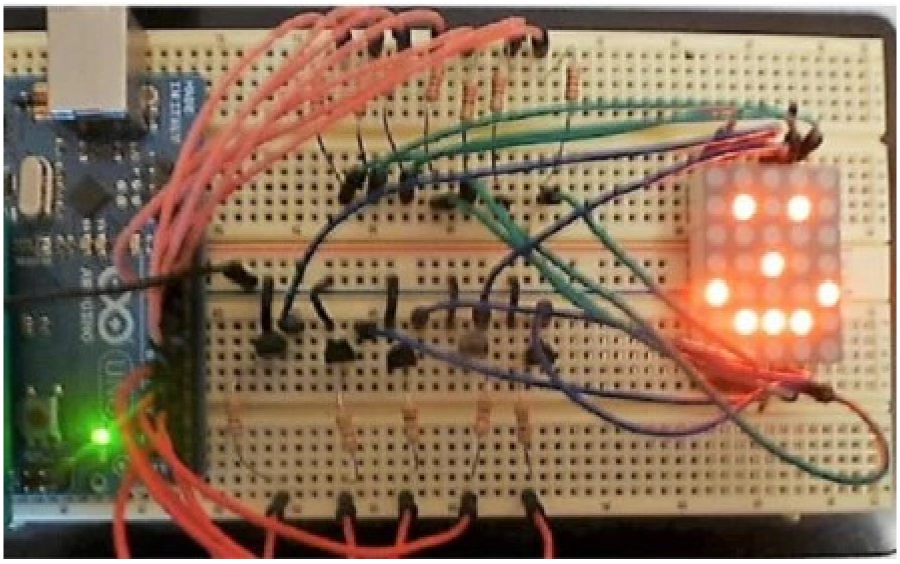
A minimum of 12 pins are required to wire up the dot matrix display with an Arduino board, rows use 7 pins while columns need 5 pins. The rows are driven by pin1 through pin7 and the columns are driven by pin8 through pin12 of Arduino, as illustrated below.

Rows: R0 → 1, R1→2, R2→ 3, R3 → 4, R4 → 5, R5→ 6, R6→ 7
Columns: C0 → 8, C1→9, C2→ 10, C3→11, C4 → 12

➢ LED Matrix Display Example Code:

```
// mBuf  - 2 dimensional array (7 rows, 5 columns)
   byte mBuf[7][5] = {
                     {0, 0, 0, 0, 0},  // 0
                     {0, 1, 0, 1, 0},  // 1
                     {0, 0, 0, 0, 0},  // 2
                     {0, 0, 1, 0, 0},  // 3
                     {1, 0, 0, 0, 1},  // 4
                     {0, 1, 1, 1, 0},  // 5
                     {0, 0, 0, 0, 0}   // 6 };
// array that holds row pin assignments
```

```
int rowPins[7] = {7, 8, 9, 10, 11, 12, 13};
// array that holds column pin assignments
int colPins[5] = {2, 3, 4, 5, 6};
// Set the row pins to mbuff column values
void setRowPins(int c)
  {
    int r;
    for (r = 0; r < 7; r++)
     { digitalWrite(rowPins[r], mBuf[r][c]);}
    return;
}
// send the contents of mbuff to the matrix display
void refreshMatrix()
{
   for (int c = 0; c < 5; c++)
    {
      setRowPins(c);
      digitalWrite(colPins[c], HIGH);
      delay(2);
      digitalWrite(colPins[c], LOW);
    }

  return;
}

void setup() {
// put your setup code here, to run once:
for (int c = 0; c < 5; c++)
  {
    pinMode(colPins[c], OUTPUT);  // initalize column pins
  }
for (int r = 0; r < 7; r++)
// initialize row pins
  {     pinMode(rowPins[r], OUTPUT);  }
}

void loop() {
  // put your main code here, to run repeatedly:
  refreshMatrix();   // updates the Led dot matrix display
}
```



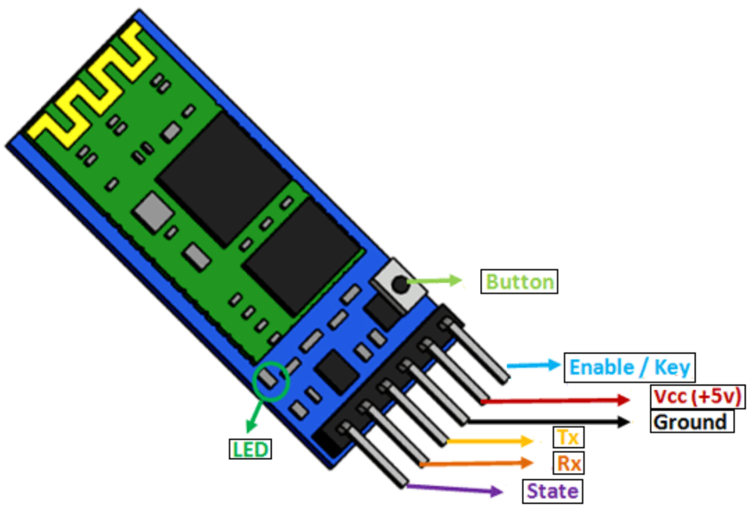
Part 2. c) Bluetooth Module

HC-05 Bluetooth Modules are the go-to Bluetooth modules for any Arduino project! It’s easy to hook up and code in the Arduino IDE. In most projects, we usually connect HC05 to an Arduino and use it to wirelessly communicate with another smart device like a mobile phone. This is fairly simple and we have built many interesting projects with it like Bluetooth Controlled Robot, Bluetooth Voice Control, Bluetooth Home Automation, etc.

HC-05 Bluetooth Module Features are as follows:

- HC-05 Bluetooth module offers two ways of communication for shorter distances with fast speed.
- It has an enable pin which allows toggling between command and data mode.
- The device uses UART (serial communication) which is easy to interface with any microcontroller or system.
- Its range is up to 8 – 10 meters but it signals lower down with any obstacle in its way.

➢ Bluetooth Module Pinout



VCC Pin

GND Pin

TX Pin The HC-05 Bluetooth module interface with the microcontroller through UART communication. The TX is the data transfer pin of the module in UART.

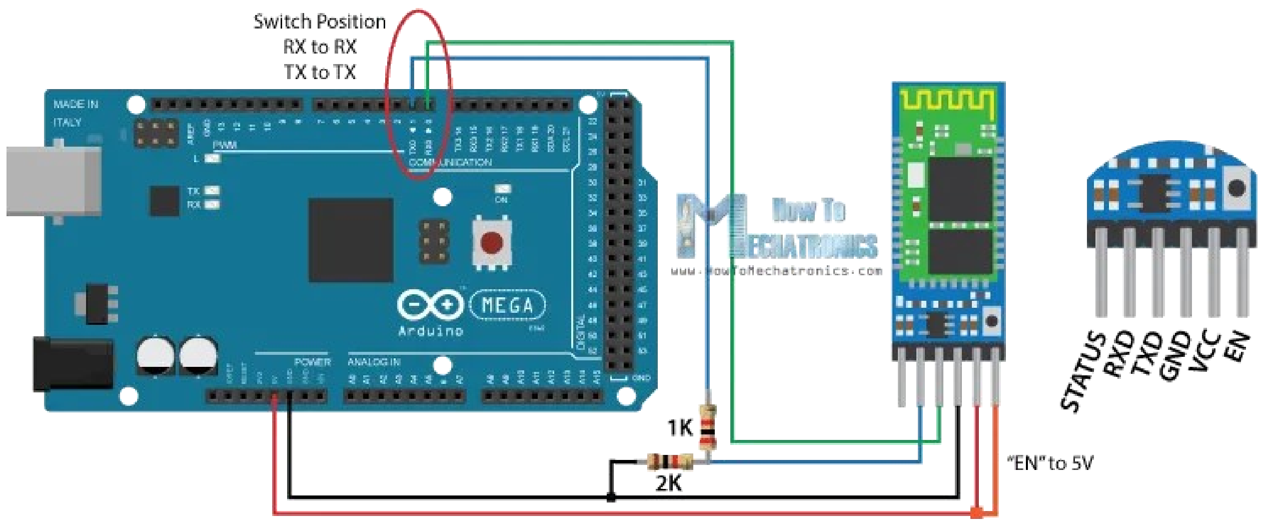
RX Pin This pin is the data receiving the pin in UART communication.

State Pin The state will be to show the current state of the Bluetooth. It gives feedback to the controller about the connectivity of Bluetooth with another device.

Enable/Key Pin Enable/Key pin helps to change the device between data mode and command mode using an external signal. The HIGH logic state will transfer the device in command mode and the LOW logic state will transfer in data mode. The default device state without any signal will be data mode.

Button Pin The command and data mode states are changeable through a button present on the module.

LED Pin LED will help to visualize the different states of the HC-05 Module.



Reference: [How To Configure and Pair Two HC-05 Bluetooth Modules as Master and Slave | AT Commands - How To Mechatronics]()

➢ Connecting two HC-05s

Connecting two HC-05s together is <u>not as straightforward as connecting an HC05 to a smartphone, there are some additional steps involved:</u>

1. Connect the "Enable" pin of the Bluetooth module to 5 volts
2. Connect the RX pin of the Arduino to the RX pin of the Bluetooth module, <u>**through the voltage divider**</u>, as shown in the figure above, and the TX pin of the Arduino to the TX pin of the Bluetooth module.
3. Hold the small button over the "Enable" pin when powering the module on to enter the command mode. If the Bluetooth module led is flashing every 2 seconds that means that we have successfully entered in the AT command mode.

4. Upload an empty sketch to the Arduino but don't forget to disconnect the RX and TX lines while uploading.

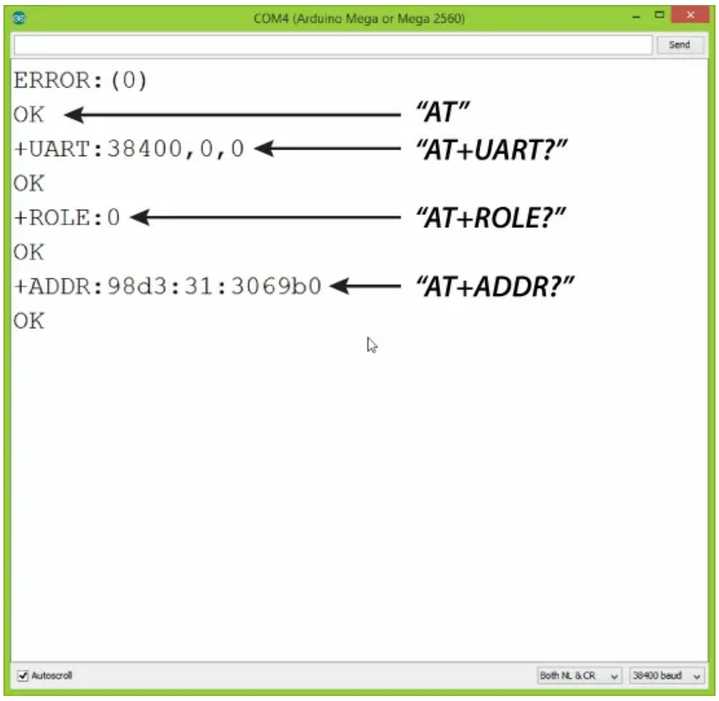
5. Run the Serial Monitor and there select "Both NL and CR", as well as "38400 baud" rate which is the default baud rate of the Bluetooth module. Now we are ready to send commands and their format is as follows.

All commands start with “AT”, followed by the “+” sign, then a <Parameter Name> and they end either with the “?” sign which returns the current value of the parameter or the “=” sign when we want to enter a new value for that parameter.

Slave Configuration

So for example, if we type just “AT” which is a test command we should get back the message “OK”. Then if we type “AT+UART?” we should get back the message that shows the default baud rate which is 38400. Then if we type “AT+ROLE?” we will get back a message “+ROLE=0” which means that the Bluetooth device is in slave mode. If we type “AT+ADDR?” we will get back the address of the Bluetooth module and it should look something like this: 98d3:34:905d3f.

Now we need to write down this address as we will need it when configuring the master device. Actually that’s all we need when configuring the slave device, to get its address, although we can change many different parameters like its name, baud rate, pairing password, and so on, but we won’t do that for this example.



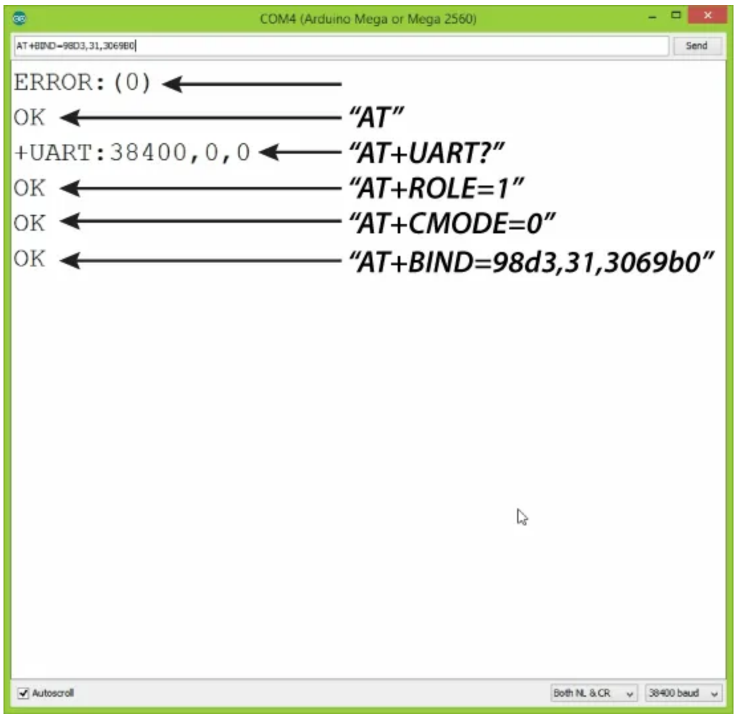
```
ERROR:(0)
OK                      ← “AT”
+UART:38400,0,0         ← “AT+UART?”
OK
+ROLE:0                 ← “AT+ROLE?”
OK
+ADDR:98d3:31:3069b0    ← “AT+ADDR?”
OK
```

Master Configuration

let’s move on and configure the other Bluetooth module as a master device. First, we will check the baud rate to make sure it’s the same 38400 as the slave device. Then by typing “AT+ROLE=1” we will set the Bluetooth module as a master device. After this using the “AT+CMODE=0” we will set the connect mode to “fixed address” and using the “AT+BIND=” command we will set the address of the slave device that we previously wrote down.

Note here that when writing the address we need to use commas instead of colons. Also note that we could have skipped the previous step if we entered “1” instead of “0” at the “AT+CMODE” command, which makes the master to connect to any device in its transmission range but that’s a less secure configuration.

Nevertheless, that’s all we need for a basic configuration of the Bluetooth modules to work as master and slave devices and now if we reconnect them in normal, data mode, and re-power the modules, in a matter of seconds the master will connect to the slave. Both modules will start flashing every 2 seconds indicating a successful connection.



COM4 (Arduino Mega or Mega 2560)

AT+BIND=98D3,31,3069B0

Send

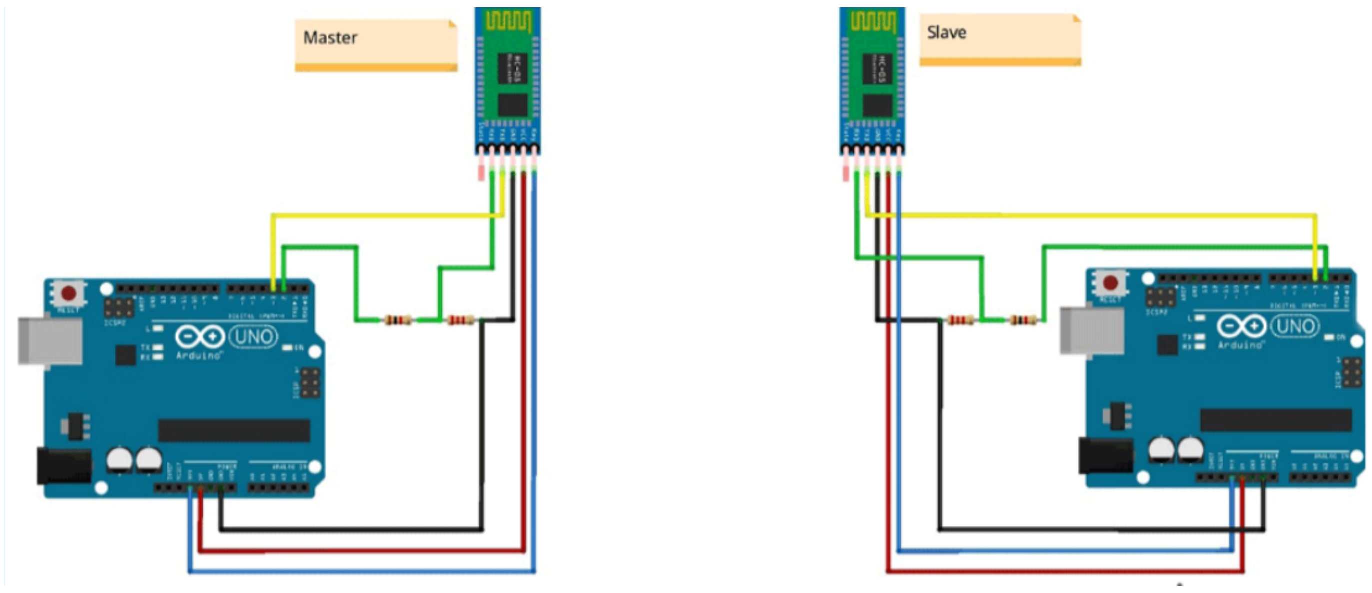
```
ERROR:(0)  <--------------
OK         <-------------- "AT"
+UART:38400,0,0 <--------- "AT+UART?"
OK         <-------------- "AT+ROLE=1"
OK         <-------------- "AT+CMODE=0"
OK         <-------------- "AT+BIND=98d3,31,3069b0"
```

☑ Autoscroll

Both NL & CR

38400 baud

The connection between the Master and slave is as shown below



➢ Bluetooth Module Code Example: -

```
Receiver:

#include <SoftwareSerial.h>
#define tx 2
#define rx 3
SoftwareSerial bt(rx, tx); //RX, TX
void setup()
{
  Serial.begin(9600);
  bt.begin(9600);
  pinMode(tx, OUTPUT);
  pinMode(rx, INPUT);
}
void loop()
{
  if(bt.available()>0)
  {
    Serial.println(bt.read());
  }
}
```

```
Sender:

#include <SoftwareSerial.h>
#define tx 2
#define rx 3
SoftwareSerial bt(rx,tx); //RX, TX
void setup()
{
  Serial.begin(9600);
  bt.begin(9600);
  pinMode(tx, OUTPUT);
  pinMode(rx, INPUT);
}
void loop()
{
  bt.write(123);
}
```

Part 3. Hand Gesture Recognition Code.

```
Receiver:
#include <SoftwareSerial.h>
#define tx 2
#define rx 3
int direction = 0;
            byte right[7][5] = {
                                   {1, 1, 1, 1, 1},   // 0
                                   {1, 1, 0, 1, 1},   // 1
                                   {1, 1, 1, 0, 1},   // 2
                                   {0, 0, 0, 0, 0},   // 3
                                   {1, 1, 1, 0, 1},   // 4
                                   {1, 1, 0, 1, 1},   // 5
                                   {1, 1, 1, 1, 1}    // 6  };
            byte up[7][5] = {
                                   {1, 1, 0, 1, 1},   // 0
                                   {1, 0, 0, 0, 1},   // 1
                                   {0, 1, 0, 1, 0},   // 2
                                   {1, 1, 0, 1, 1},   // 3
                                   {1, 1, 0, 1, 1},   // 4
                                   {1, 1, 0, 1, 1},   // 5
                                   {1, 1, 0, 1, 1}    // 6 };
            byte down[7][5] = {
                                   {1, 1, 0, 1, 1},   // 0
                                   {1, 1, 0, 1, 1},   // 1
                                   {1, 1, 0, 1, 1},   // 2
                                   {1, 1, 0, 1, 1},   // 3
                                   {0, 1, 0, 1, 0},   // 4
                                   {1, 0, 0, 0, 1},   // 5
                                   {1, 1, 0, 1, 1}    // 6 };
            byte left[7][5] = {
                                   {1, 1, 1, 1, 1},   // 0
                                   {1, 1, 1, 1, 1},   // 1
                                   {1, 0, 1, 1, 1},   // 2
                                   {0, 0, 0, 0, 0},   // 3
                                   {1, 0, 1, 1, 1},   // 4
                                   {1, 1, 1, 1, 1},   // 5
                                   {1, 1, 1, 1, 1}    // 6 };
```

```
// array that holds row pin assignments
int rowPins[7] = {1, 2, 3, 4, 5, 6, 7};
// array that holds column pin assignments
int colPins[5] = {8, 9, 10, 11, 12};
SoftwareSerial bt(rx, tx); //RX, TX

void setup()
{
  Serial.begin(9600);
  bt.begin(9600);
  pinMode(tx, OUTPUT);
  pinMode(rx, INPUT);
  for (int c = 0; c < 5; c++)
  { pinMode(colPins[c], OUTPUT); // initalize column pins }

for (int r = 0; r < 7; r++)
// initialize row pins
  { pinMode(rowPins[r], OUTPUT);}
}

void setRowPins(int c)
  {
    int r;
  if(direction ==1)
  {
    for (r = 0; r < 7; r++)
     {
       digitalWrite(rowPins[r], left[r][c]);
      }
  }
  else if(direction ==2)
  {
    for (r = 0; r < 7; r++)
     {
       digitalWrite(rowPins[r], up[r][c]);
      }
  }
  else if(direction ==3)
  {
      for (r = 0; r < 7; r++)
     {
       digitalWrite(rowPins[r], down[r][c]);
      }
    }
    else if(direction ==4)
    {
       for (r = 0; r < 7; r++)
     {
       digitalWrite(rowPins[r], right[r][c]);
      }

      }

    return;
}

void refreshMatrix()
{
   for (int c = 0; c < 5; c++)
    {
      setRowPins(c);
      digitalWrite(colPins[c], HIGH);
      delay(2);
      digitalWrite(colPins[c], LOW);
    }


  return;
}
void loop()
{
  if(bt.available()>0)
  {
   Direction= bt.read();
    Serial.println(bt.read());
  }
  refreshMatrix();  // updates the Led dot matrix display
}
```

```
Sender:
#include <SoftwareSerial.h>
#define tx 2
#define rx 3
SoftwareSerial bt(rx,tx); //RX, TX

int x = 0;
int y = 0;
int z = 0;

void setup()
{
  Serial.begin(9600);
  bt.begin(9600);
  pinMode(tx, OUTPUT);
  pinMode(rx, INPUT);
}
void loop()
{
  x = analogRead(A0)/256;
  y = analogRead(A1)/256;
  z = analogRead(A2)/256;

  if(x ==1 && y==0 &&z==1 )
  { for (r = 0; r < 7; r++)
     {   bt.write(1); }
  }
  else if(x ==0 && y==1&&z==1)
  { for (r = 0; r < 7; r++)
     {   bt.write(2);}
  }
  else if(x ==1 && y==1&&z==0)
  { for (r = 0; r < 7; r++)
     {     bt.write(3);   }
  }
  else if(x ==1 && y==2&&z==0)
  { for (r = 0; r < 7; r++)
     {  bt.write(4);    }
  }
}
```