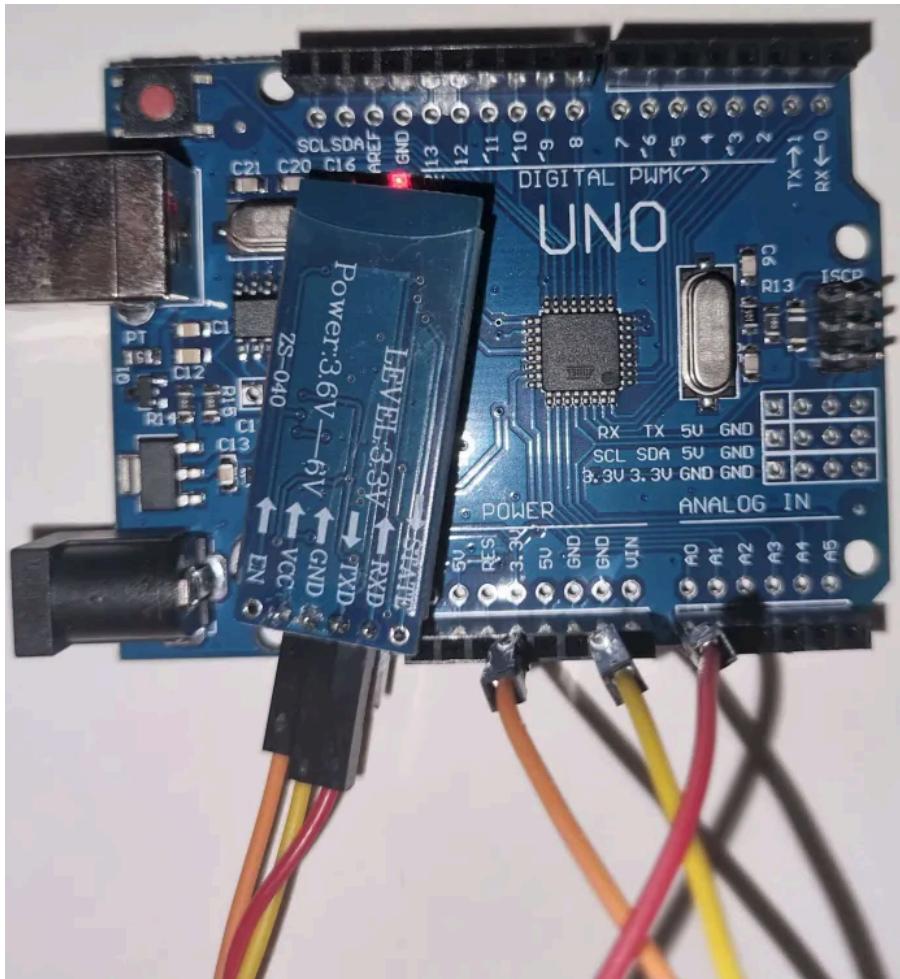


Bluetooth communicating with Arduino

Bluetooth devices are either master or slave devices. Master devices can establish connections to other devices, while slave devices can only accept connections from master devices. Wireless earphones are slave devices as they only broadcast their availability, while the smartphone acts as the master device which establishes the required connection. HC-06 is a slave only bluetooth module, and therefore needs a master device to connect to it.

For our purposes we will have the HC-06 establish a connection with your smartphone. You will send commands to the HC-06 bluetooth module via a wireless connection, then the HC-06 will relay that command over to the arduino via a wired connection.



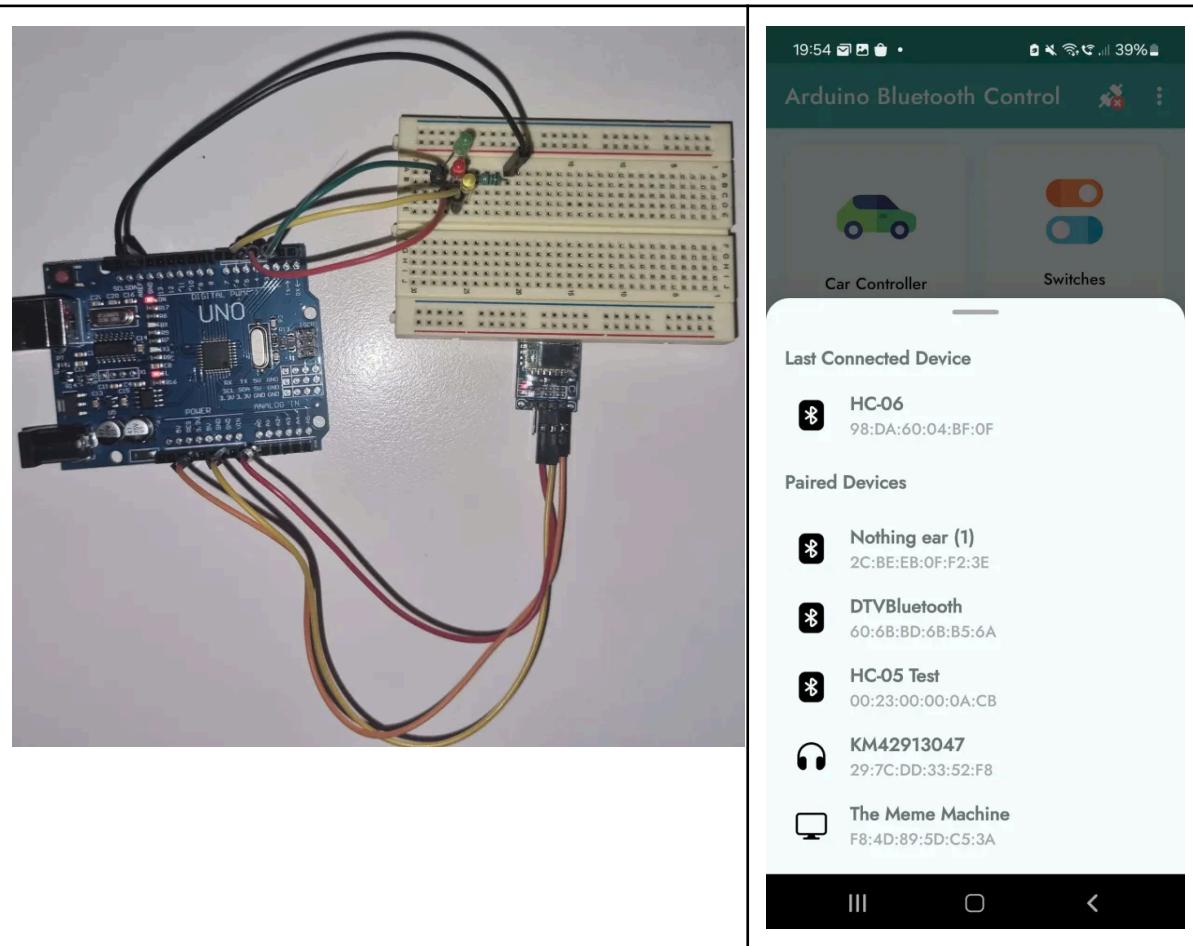
EN	VCC (3.3V)	GND (0V)	TXD	RXD	STATE
Mode - either master or slave. Internally set to slave mode so there is no pin for us to access.	Powers the module.	Powers the module.	Data is transmitted to the arduino from the HC-06. Once the HC-06 receives data from the master device it will relay it to this pin.	Data that the HC-06 receives from the arduino. This pin is not used as the arduino will not talk to the master device.	Flashing LED if not connected to any master device. Stable LED if connection established.

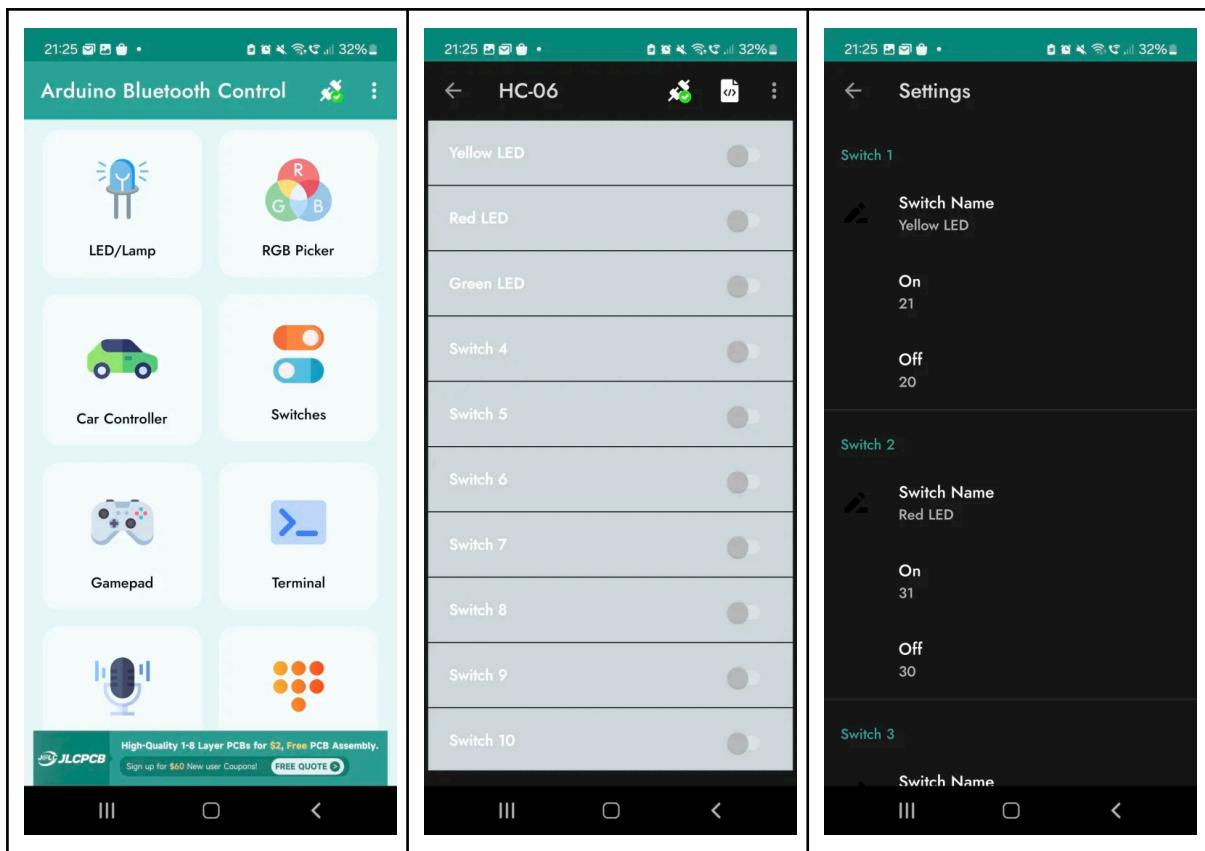
Using your phone as a controller

Android apps to connect to HC-06 (some apps don't work with this module)

- We will use this to play with the terminal and switches function
<https://play.google.com/store/apps/details?id=com.giristudio.hc05.bluetooth.arduino.control>
- Use this one for a virtual controller
<https://play.google.com/store/apps/details?id=com.giristudio.bluetooth.car.controller>

Note down your MAC address, in this case 98:DA:60:04:BF:0F, because you cannot change the name of the HC-06 (other models such as the HM-10 you can). Make sure you add a resistor in series with the LEDs (anything 220Ω or larger is fine) and attach the jumper wire from the arduino to the long leg of the LED, that is the positive terminal.





C/C++

```
#include <SoftwareSerial.h>

#define RXD A0
SoftwareSerial BTSerial (RXD,RXD);
String data = "";

#define YELLOW_LED_PIN 2
#define RED_LED_PIN 3
#define GREEN_LED_PIN 4

void LED(int command){
    switch (command) {

        case 20:
            digitalWrite(YELLOW_LED_PIN, LOW);
            break;

        case 21:
            digitalWrite(YELLOW_LED_PIN, HIGH);
            break;

        case 30:
            digitalWrite(RED_LED_PIN, LOW);
            break;
    }
}
```

```
        break;

    case 31:
        digitalWrite(RED_LED_PIN, HIGH);
        break;

    case 40:
        digitalWrite(GREEN_LED_PIN, LOW);
        break;

    case 41:
        digitalWrite(GREEN_LED_PIN, HIGH);
        break;

    default:
        break;
    }
}

void setup() {
//Setting up serial monitor and bluetooth serial
Serial.begin(9600);
BTSerial.begin(9600);

// Setting up LED pins
pinMode(YELLOW_LED_PIN, OUTPUT);
pinMode(RED_LED_PIN, OUTPUT);
pinMode(GREEN_LED_PIN, OUTPUT);
}

void loop() {
// this is where we will store the command from the bluetooth module
int bt_data = 0;
while (BTSerial.available()){

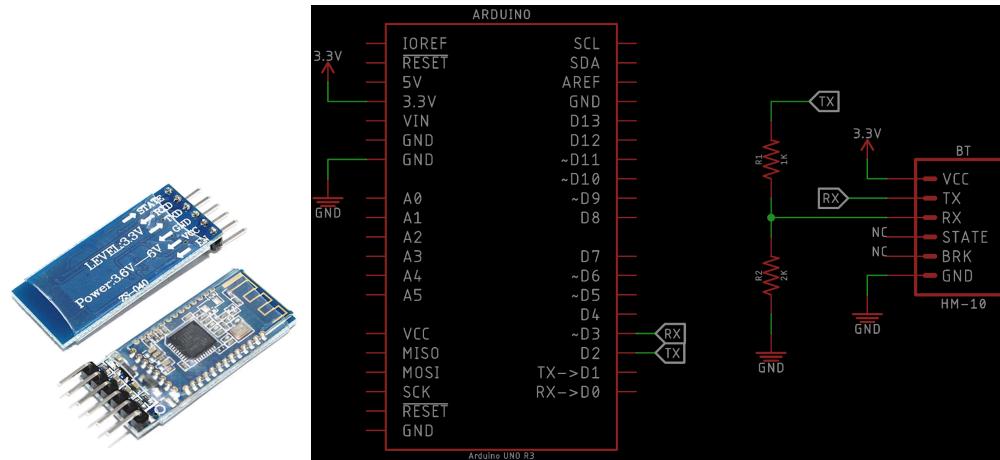
    // BTSerial.read() reads ascii values one by one from the bluetooth
buffer as an integer according to https://www.ascii-code.com/
    char c = BTSerial.read();
    bt_data = bt_data * 10 + (c - '0');
    delay(1); // required for stablity
}
LED(bt_data);
}
```

HM-10 as Master or Slave communication

Establishing a connection

The HM-10 is an alternative which can act as both a master device - one which can transmit data, or a slave device - one which receives the data. Since this module has more functionality AT commands are required for configuration. It is possible to make an arduino based controller, reading inputs from the GPIO pins and transmitting those commands to the HM-10 module to a slave device (Note this will not work between HM-10 and HC-06 as they use different different bluetooth technologies but between two HM-10 modules is possible).

EN	VCC	GND (0V)	TXD	RXD	STATE
Set high to enable module	3.6-6v	Common Ground	Serial UART transmit	Serial UART receive. Ensure that the voltage divider is implemented.	High when connected, Low when not connected



AT commands will be used to communicate and interfere with these modules, so we can connect the bluetooth module to your phone app using Bluetooth connection. Here is a

script that you can run for interfacing with the HM10 bluetooth module via Arduino.

NAME	SEND	RECEIVE	PARAMETER	DESCRIPTION
Test Command	AT	OK	None	Simple test command to see if a response is received.
Query Module Name	AT+NAME?	OK+NAME:[para1]	para1: Module name, max length is 12.	Queries for the HM-10 module name.
Set Module Name	AT+NAME[para1]	OK+Set:[para1]	para1: Module name, max length is 12. Default: HMSoft	Sets the HM-10 module name.
Query Module (MAC) Address	AT+ADDR?	OK+ADDR:[para1]	para1: MAC address MAC Address format: 0123456789AF	Queries for the HM-10 module MAC address.
Query Baud Rate	AT+BAUD?	OK+Get:[para1]	para1: Baud rate speed. Default: 0 -> 9600	Queries for the HM-10 module's current baud rate.
Set Baud Rate	AT+BAUD[para1]	OK+Set:[para1]	para1: Baud rate speed. AT+BAUD0 -> 9600 AT+BAUD1 -> 19200 AT+BAUD2 -> 38400 AT+BAUD3 -> 57600 AT+BAUD4 -> 115200 AT+BAUD5 -> 4800 AT+BAUD6 -> 2400 AT+BAUD7 -> 1200 AT+BAUD8 -> 230400	Sets the HM-10 module baud rate. If Baud rate set to 7 (1200), module will not support any AT commands, until PIO0 is pressed, then module will change Baud rate to 9600.
Try to connect to a (MAC) address	AT+CON[para1]	OK+CONN:[para2]	para1: MAC address. para2: A, E, F A: Connecting E: Connect error F: Connect fail	Method to pair one HM-10 module to another HM-10 module or device.
Restore all setup to factory setup	AT+RENEW	OK+RENEW	None	Restore HM-10 module to factory settings.
Restart module	AT+RESET	OK+RESET	None	Closes any active connection and reboots the HM-10 module.
Query Software Version	AT+VERR? AT+VERS?	Version information	None	Returns the software version of the HM-10 module.

⚠ Do not include the '[' and ']' when sending the AT command.

🔗 A complete set of AT commands may be found from the datasheet.

C/C++

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(7, 8); // RX, TX
// HM10      Arduino Uno
// TXD      Pin 2
// RXD      Pin 3
void setup() {
    // You may need to change baud rate if it has previously been set to
    // another value
    Serial.begin(9600);
    BTSerial.begin(9600);
}
```

```

void loop() {
  char c;
  // Transmitting to HM-10 via serial communication
  if (Serial.available()) {
    c = Serial.read();
    BTSerial.print(c);
  }
  // Receiving from HM-10
  if (BTSerial.available()) {
    c = BTSerial.read();
    Serial.print(c);
  }
}

```

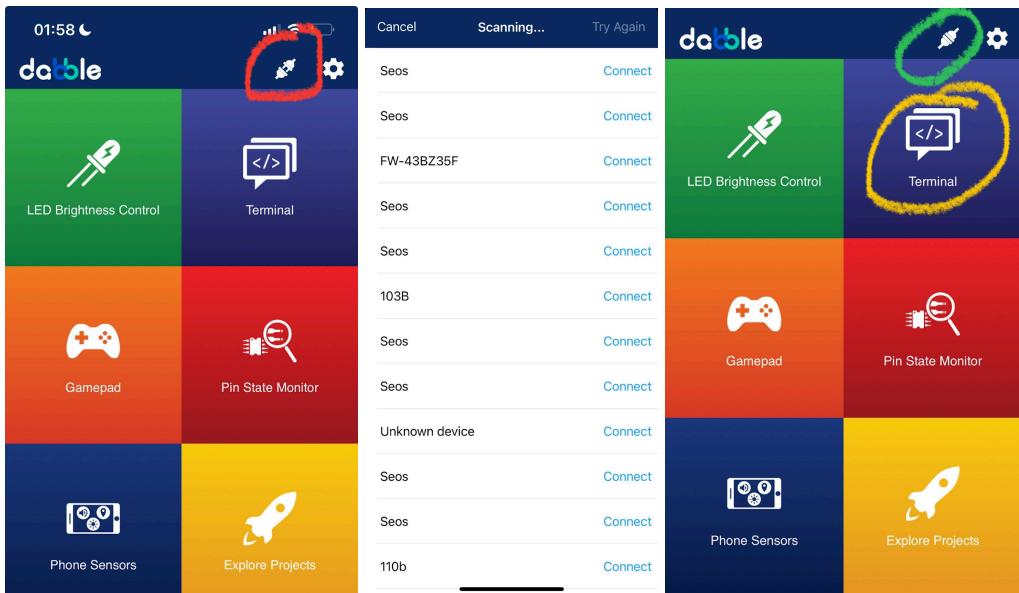
Explore the following query commands in the serial monitor (remember to set the baud rate of the serial monitor to 9600):

- AT
- AT+NAME?
 - device name, by default it is HMSoft.
 - You should change it so we don't have 32 HMSoft to be detected...
 - When you change this name it is saved even if the power is lost.
- AT+IMME0
 - 0 for configuration mode, 1 for control mode.
 - 0 because we still have more things to set
- AT+ROLE0
 - A role of 0 implies a slave (*peripheral*) role, meaning receiving information.
 - A role of 1 implies a master (*central*) role, meaning sending information.
- AT+ADVI5
 - how frequently the sensor advertises its presence
 - you don't really need to care about this, just type it in for setting up Bluetooth
- AT+RESET
 - set the above setting to the Bluetooth

If you get stuck, ask for help !

Apple and android apps to connect to HM-10

- <https://apps.apple.com/au/app/dabble-bluetooth-controller/id1472734455>
- <https://play.google.com/store/apps/details?id=io.dabbleapp&gl=au>



Instructions on how to use the gamepad are in this [link](#).

If you would like to watch a YouTube video about Dabble click on this [link](#).

Activity: Controlling LEDs

Your task is to connect three different LEDs to the arduino that is receiving data and glowing the LED based on the data received.

Say, the number received is "1", LED number 1 should glow, while all others are dimmed.

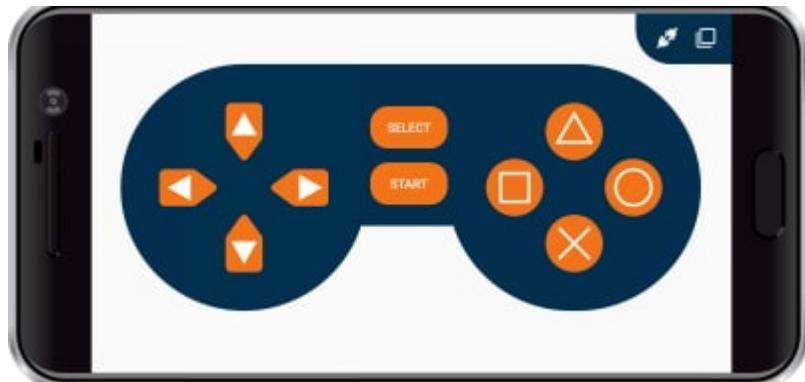
When the number "3" is received, LED number 3 should glow, while all others are dimmed.

That's it ! Try playing around.

Look at the additional resources if you want to delve deeper!

Connect motors to Dabble using HM-10 bluetooth

The GamePad section of the Dabble app will be very useful for the competition. To use it there are 3 functions you will need to use:



Dabble.begin(BAUD, RX, TX)	This initialises the bluetooth to connect to dabble as well as which
----------------------------	--

	Arduino Pins are RX and TX
Dabble.processInput()	This processes what input has been received from the dabble app. For example pressing Forward on the GamePad
GamePad.isXPressed() (replace X with the button you are checking eg: .isRightPressed())	This checks to see if a certain button on the GamePad has been pressed

Visit this [link](#) for a really detailed explanation. See if you can use these functions and your motors to build some basic movement commands.

Troubleshooting

- Check if all the wires are connected to the right pins, right spot.
- Check if all the wires are functional (continuity from multimeter)
- Check if the bluetooth of your phone is turned on.
- Check if the Bluetooth connection is correct
- If you have not used resistors then use it.
- Check the values of your resistor before using them
- If after all these steps, you still don't see Bluetooth coming on your phone, change VCC connection from 3.3V to 5V
- If you have some trouble seeing commands after sending them on the serial monitor, maybe add delay.