Using HC05 to Communicate to HC05

Bluestamp Engineering

Written by Justin Miner 2022

Last Modified: 2024

Overview

Here I will overview the entirety of the process to get them connected and to test them. First, you wire up the Bluetooth to the Arduino. For the protection of the Bluetooth module you will need to incorporate a voltage divider on the Arduino's TX pin because the module operates at 3.3 V and you will damage it with a 5V input. The Arduino's RX pin does not need anything because 3.3V is considered HIGH in 5V logic level. Then after plugging the Arduino devices to the boards, you will run a Serial test code where you are able to communicate over the Serial monitor with the Bluetooth in AT mode. After configuring both arduinos, the same code can be used to test that the two devices are communicating by opening two serial monitors and typing on one to see the other.

This module has 3 different blinking cycles: fast blinking- searching, slow blinking- AT Mode, and 2 blinks periodically and synchronously- paired. This will help with debugging.

Initial Parts (Important)

Previously, I was using the HiLetgo HC05s. These did not work because of firmware issues configured for HC06 (I assume). The AT+BIND command was not working which is used to set the pairing address and the AT+UART would not save in memory upon powering it on and off. The current sourcing is available at this link for the https://www.amazon.com/gp/product/B07C1PXM44/ and I have gotten these to work with the current batch.

Definitely, before buying large amounts of whichever, test to make sure you can communicate with them (so follow this guide).

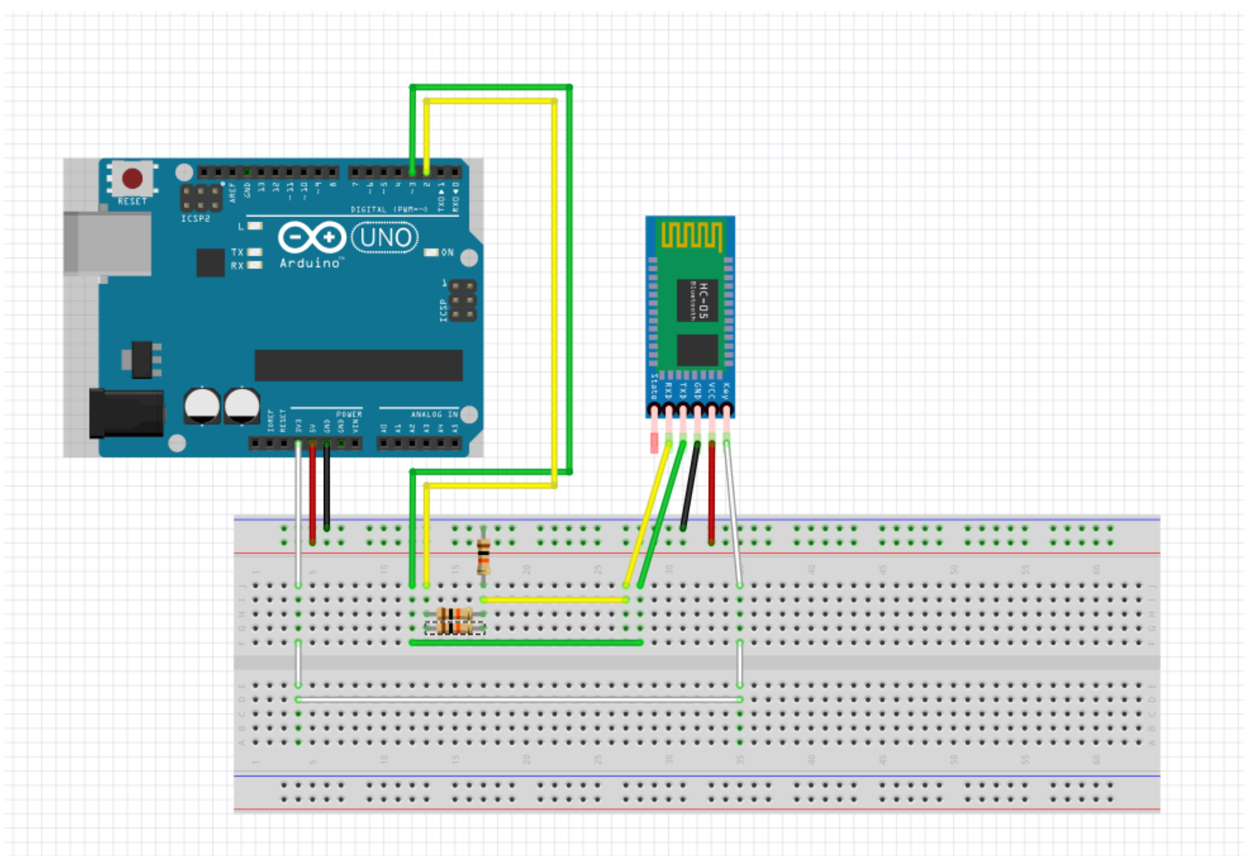
Wiring

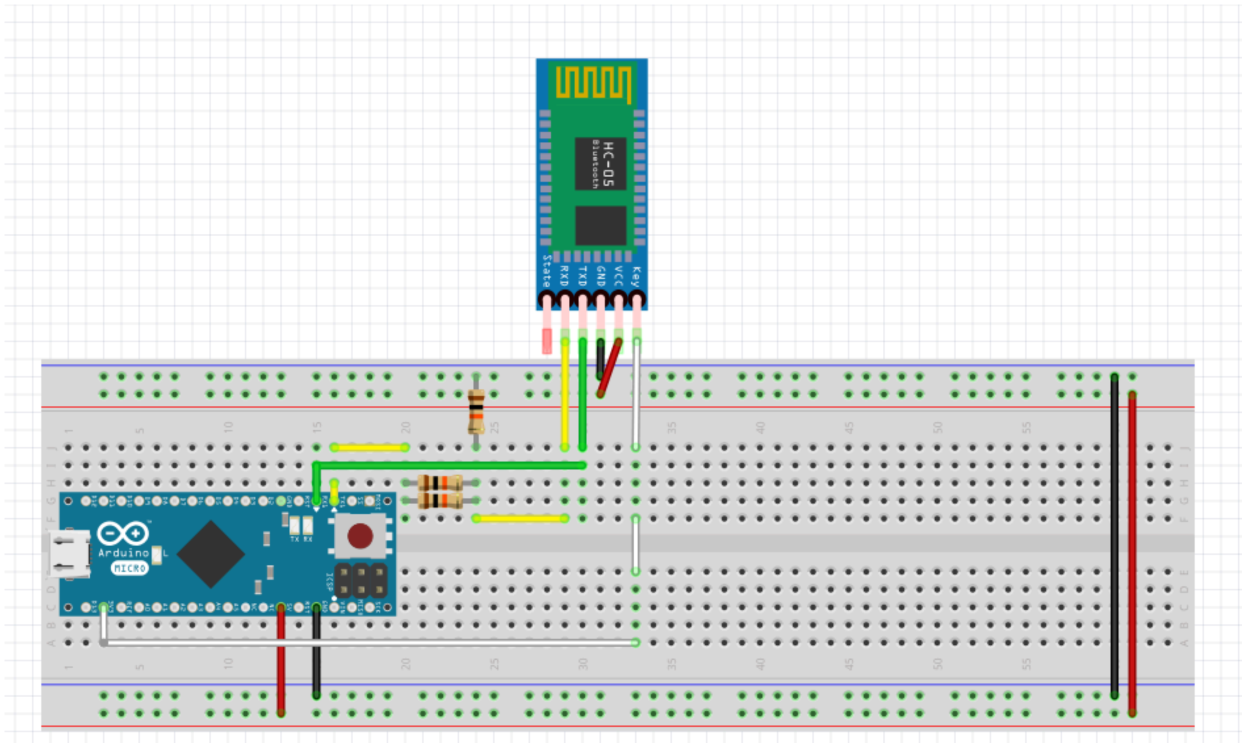
For the connections, it is pretty standard in the way that VCC 5v, GND Ground. The Enable pin should be wired to 3.3 V initially so it can be configured to communicate.

For Micro/Leonardo (boards with a second hardware serial): Wire TX to RX and RX to TX. Use a voltage divider as shown in the image below for the Arduino TX pin due to logic level conversion. The value of the resistors doesn't matter much as long as they're in a 1:2 ratio. I used 10k and 5k which was in the kits.

For all other Arduinos: I usually put RX 2 and TX 3 because software serial is used. TX and RX should not be used because it interferes with programming and it is a pain to take it out and plug it in again. Use the voltage divider as shown below.

With the micro specifically, the most common error students make are putting the wrong direction so RX-RX and TX-TX or they put it in the wrong pins because it is difficult to see it in the breadboard.





Programming

See images of code for micro based boards and uno based boards below (hardware vs. software serial). The code is writing to serial monitor whatever Bluetooth receives and writing from serial monitor to Bluetooth module which allows you to configure it.

```
/*
 * CONNECT TO OUTGOING PORT HC-05 'DEV-B' TO VIEW DATA ON SERIAL MONITOR
 * USE THIS SKETCH ONLY FOR VIEWING SENSOR DATA ON SERIAL MONITOR.....NOT FOR FILE WRITING
 */

void setup()
{
  Serial.begin(38400);
  Serial1.begin(38400);
}

void loop()
{
  if (Serial1.available())
  {
    Serial.print((char)Serial1.read());
  }
  if (Serial.available())
  {
    Serial1.write(Serial.read());
  }
}
```

Code for Micro

```
/*
 * CONNECT TO OUTGOING PORT HC-05 'DEV-B' TO VIEW DATA ON SERIAL MONITOR
 * USE THIS SKETCH ONLY FOR VIEWING SENSOR DATA ON SERIAL MONITOR.....NOT FOR FILE WRITING
 */
#include <SoftwareSerial.h>

#define tx 2
#define rx 3

SoftwareSerial configBt(rx, tx);
long tm,t,d; //variables to record time in seconds

void setup()
{
  Serial.begin(38400);
  configBt.begin(38400);
  pinMode(tx, OUTPUT);
  pinMode(rx, INPUT);
}

void loop()
{
  if (configBt.available())
  {
    Serial.print((char)configBt.read());
  }
  if (Serial.available())
  {
    configBt.write(Serial.read());
  }
}
```

Code for Uno

Configuration

Before doing this, make sure enable is set to 3.3V. This can be confirmed with slow blinking after power cycle. Once the code is uploaded, you can work at one Arduino at a time. Open the Serial Monitor. Set it to 38400 baud rate and set the ending to NL+CR (Newline and Carriage Return). To make sure it is working correctly, type AT which the module should respond back with OK. First you will do the slave configuration, and then bind it to the master. The Arduino Uno is the slave for the gesture controlled car generally, but it doesn't really matter which is which. Following every command you should receive OK.

Slave Configuration

Here are the AT commands you will be sending to the robot line by line.

- AT+ADDR (Note this value down we will use it later)
- AT+UART=38400,0,0 (This is the communication speed between the modules)

Master Configuration

Make sure your slave address is handy. In the AT+BIND command you will put this down with the colons switched to commas and make sure the number of digits are correct (Arduino can remove leading zeros).

- AT+ROLE=1 (This sets it to master mode)
- AT+CMODE=0 (This sets it to connect to one address only)
- AT+BIND=XXXX,XX,XXXXXX (Replace X's with the address of the slave)
- AT+UART=38400,0,0 (This sets the connection speed).

After configuration, unplug the enable pin from 3.3V . It should be left floating. After this, you can power cycle both arduinos and power them on. The Bluetooth modules should do two blinks synchronously every few seconds to show that they are paired together.

Testing

The same code can be used for the testing as the configuration. Open two instances of Arduino and open the serial monitor on both arduinos set to 38400 baud. Then you can type in one serial monitor and it should show up on the other module to indicate that it is working correctly.