

<https://forum.arduino.cc/t/hc-05-bluetooth-module-has-no-password/1304574/18>

check this website and let me know why hc05 password does not set?

Here's what's going on with the **HC-05 Bluetooth module not having a password or not accepting a new password**, based on the Arduino Forum thread you linked and other common reports:

🔍 Why the HC-05 shows “NO KEY” and won’t set a password

1. It may *not actually be an HC-05*

In the forum thread you linked, a user later points out that modules that look like HC-05 **might actually be a different Bluetooth module** (e.g., BLE device like JDY-16) that **doesn't use the same AT commands or default PIN behavior as HC-05**. A 6-digit default PIN (like 123456) is *typical for BLE modules*, not classic HC-05. Because of that, **AT+PSWD won't work and commands behave differently.**

Arduino Forum

2. Firmware on some cheap HC-05 clones doesn't support setting the PIN normally

Many inexpensive HC-05 clones with certain firmware versions (e.g., VERSION:4.0-20190815) will **accept a password change in AT mode (respond OK)** but **not actually save it to permanent memory** unless additional steps are done. In that case the module will revert to the default password after power-cycle.

Byte Blues

3. AT commands in the wrong mode or incomplete AT initialization

— The module must be in *full AT command mode*, with the **KEY/EN pin held high before powering up**.

If it's not truly in AT mode (only partially or after pairing is active), setting PSWD may fail.

Byte Blues

— If it's still paired or active with another device, AT commands may not work properly.

Sathyabama Uni...

4. Some modules require additional steps for settings to “stick”

For certain firmware versions, just issuing **AT+PSWD=xxxx** isn't enough — you may need to change and

then issue another command like `AT+ROLE=0` so the module saves settings to flash memory, otherwise they only update in RAM and reset after power cycle. Byte Blues

5. Incorrect password length or format

Standard HC-05 classic Bluetooth expects a **4-digit numeric PIN**. Sending a different length (e.g., 6 digits) or using quotes incorrectly can cause commands to fail. Some modules reject or ignore incorrect formats. DCC-EX ...

🧪 What “NO KEY” means in your case

In the exact forum report, when the user sends `AT+PSWD?` the module replies:

diff

+PSWD:NO KEY OK

 Copy code

That basically means **this module has no stored PIN at all**, which is unusual for a real HC-05 (it normally has `+PSWD:1234`). On modules that aren’t true HC-05 or have unusual firmware, the *PIN storage area may be blank or non-functional*, so the command returns “NO KEY” and a password cannot be set normally.

Arduino Forum

✓ Practical things to try

If you want to *fix* it, here are reliable next steps:

✓ Verify it’s a real HC-05:

Check the **firmware version** with `AT+VERSION?`. If it’s something unexpected or mentions BLE, it might not be a classic HC-05.

✓ Ensure full AT mode:

Connect KEY/EN high (3.3 V) before power-up so the module enters full AT mode.

✓ Try correct password command formats:

pgsql

AT+PSWD=**1234** (no quotes)

AT+PSWD="**1234**" (with quotes depending on firmware)

 Copy code

Try both forms.

✓ Use additional AT commands to force save:

After password change, try:

pgsql

AT+ROLE=0

 Copy code

This can help force saving settings for some firmwares. Byte Blues

✓ Try connecting once with the new PIN while still in AT mode:

Some people found that pairing from a PC with the new password *while the module is still in AT mode* causes the module to save the new pin. Byte Blues

✓ Otherwise consider using better-supported modules:

If it really *is* a generic or BLE module (not HC-05), many Arduino Bluetooth tutorials won't work, and it's easier to replace it with a known-good HC-05 from a reliable seller.

Problem	Likely Cause
Shows "NO KEY"	Firmware doesn't have a stored PIN or isn't true HC-05
AT+PSWD returns FAIL or OK but then doesn't stick	Module clone firmware issue or requires specific save step
Phone still pairs with default 1234	New PIN not stored permanently
Only works with 6-digit PIN	Might be a BLE module, not HC-05

If you want, I can walk you through **how to confirm the module type and firmware** using AT commands — just tell me what you get when you send `AT+VERSION?` in AT mode.



but some comments are saying it should be 123456?

Ah, yes — that's where it gets confusing. Here's the nuance:

1 Classic HC-05 vs BLE clones

- **Genuine HC-05 (classic Bluetooth):**
 - Default PIN is 1234 (4 digits).
 - Only supports **4-digit numeric PINs**.
 - AT+PSWD sets a new 4-digit PIN.
- **Some cheap clones or BLE modules** (JDY-16, HC-05 lookalikes, "HC-05 BLE" modules):
 - Default PIN is often 123456 (6 digits).

- Some of these modules **ignore the normal AT+PSWD command** or require a different AT command syntax.
- That's why users on forums see 123456 and think "this is normal."

2 Why your module might say "NO KEY"

- It's likely a **clone or a BLE variant**, which doesn't have a 4-digit PIN stored in the classic HC-05 way.
- Even if you try AT+PSWD=1234, the firmware may **ignore it or only temporarily store it in RAM**, reverting to "no key" or the default 6-digit PIN on restart.

3 How to know which case you have

Run in AT mode:

```
pgsql
```

AT+VERSION?

 Copy code

- If it shows something like "HC-05" and version numbers similar to 2.0-20100601 → classic HC-05, default PIN should be 1234.
- If it shows **JDY, BLE, or unusual firmware** → it's a BLE module or clone; default PIN is usually 123456, and normal AT+PSWD may not work.

 **Key point:** The forum comments about 123456 usually refer to **non-standard HC-05 clones or BLE modules**, not genuine HC-05 modules. That's why you see conflicting info.

If you want, I can give you a **step-by-step method to set or check the password** even on these tricky clones so it actually "sticks." Do you want me to do that?



how to set the class of hc05 for 100m range?

Ah! You want to **increase** the HC-05's Bluetooth range by changing its "class." Let's break this down carefully.

1 What "Class" means in Bluetooth

Bluetooth devices have a **Class of Device (CoD)** setting that defines:

- **Device type** (computer, phone, audio, etc.)
- **Service type** (audio, telephony, etc.)
- **Transmit power / range**

Classes related to range:

Class	Typical Power	Max Range (open space)
Class 1	100 mW	~100 m
Class 2	2.5 mW	~10 m
Class 3	1 mW	~1 m

HC-05 modules by default are **Class 2**, which gives ~10 m range.

2 How to check current class on HC-05

Enter AT mode and send:

objectivec

AT+CLASS?

 Copy code

Typical response for default HC-05:

diff

+CLASS:0x200404

 Copy code

This means **Class 2, device type: desktop/phone, etc.**

3 How to set HC-05 to Class 1 (~100 m)

The AT command is:

objectivec

AT+CLASS=0x1F00

 Copy code

- 0x1F00 is commonly used to indicate **Class 1, “computer” type, generic device.**
- After setting, save the configuration (on some modules):

pgsql

AT+RESET

 Copy code

or power cycle the module.

Tip: You can verify with:

objectivec

 Copy code

AT+CLASS?

4 Practical caveats

1. Hardware limitation:

- Cheap HC-05 modules **may not physically support Class 1 power**, even if AT commands allow it.
- True Class 1 might need **enhanced antenna or power supply** (3.3 V stable, low noise).

2. Range depends on environment:

- 100 m is **line-of-sight**, indoors or with walls, expect **much less (~15–20 m)**.

3. Power consumption:

- Higher class → more transmit power → more current draw (~40–50 mA vs 8–30 mA for Class 2).
-

💡 Summary

- Default HC-05: Class 2 (~10 m)
 - To try 100 m: AT+CLASS=0x1F00 + AT+RESET
 - Success depends on **module quality and environment**
-

If you want, I can give you a **ready AT command sequence** to set HC-05 to Class 1 and maximum baud rate for the best range and speed. It works even on clones that are tricky.

Do you want me to do that?



yes

Perfect! Here's a **step-by-step AT command sequence** to maximize your HC-05 range (Class 1) and speed, including a safe way to make it stick:

Step 1 — Enter AT Mode

1. Disconnect HC-05 from power.
2. Hold the **KEY / EN** pin high (3.3 V) before powering on.
3. Power the module.
4. Open serial monitor in Arduino IDE (or another terminal) with:

pgsql

Baud rate: 38400 (default AT mode for most modules)

Line ending: Both NL & CR

Copy code

5. Test AT mode:

nginx

AT

Copy code

Response should be:

nginx

OK

Copy code

Step 2 — Set Class to Class 1 (max range)

objectivec

AT+CLASS=0x1F00

 Copy code

- 0x1F00 → Class 1 (100 m theoretical), Generic computer/phone device type.
- Response should be:

nginx

OK

 Copy code

Step 3 — Set Maximum Baud Rate (Optional, for faster communication)

AT+UART=115200,0,0

 Copy code

- 115200 → baud rate
- 0,0 → stop bits and parity
- Response: OK

You can keep default 9600 if stability is more important than speed.

Step 4 — Set PIN (Optional)

- Must be **4 digits** for classic HC-05
 - Response: OK
-

Step 5 — Save Settings and Reset

Some modules automatically save, but for reliability:

pgsql

AT+RESET

Copy code

or power cycle the module.

- Verify settings:

objectivec

Copy code

AT+CLASS?

AT+UART?

AT+PSWD?

- Responses should confirm your changes.
-

Step 6 — Test Range

- Keep in mind: **100 m is line-of-sight**. Indoors, expect 15–20 m.
 - Walls, interference, and cheap module quality affect range.
-

Pro Tips

1. Use external antenna if available — some HC-05 boards have pads for soldering a small whip antenna.
 2. Stable 3.3V supply improves transmit power.
 3. Some cheap clones may ignore Class 1; in that case, real 100 m range isn't possible.
-

If you want, I can make a **ready-to-copy AT command script** for Arduino Serial Monitor that sets **Class 1, max baud, and PIN all at once** — so you don't have to type commands manually.

Do you want me to do that?

